

A systematic review of hyperparameter optimization techniques in Convolutional Neural Networks

Mohaimenul Azam Khan Raiaan^a, Sadman Sakib^b, Nur Mohammad Fahad^a, Abdullah Al Mamun^b, Md. Anisur Rahman^b, Swakkhar Shatabda^b, Md. Saddam Hossain Mukta^{c,*}

^a Faculty of Science and Technology, Charles Darwin University, Casuarina, NT 0909, Australia

^b Department of CSE, United International University (UIU), Dhaka 1212, Bangladesh

^c LUT University, LUT School of Engineering Sciences, Lappeenranta-Lahti University of Technology, 53850 Lappeenranta, Finland

ARTICLE INFO

Keywords:

Convolutional Neural Network
Hyperparameter optimization Analysis
Optimizer
Activation function

ABSTRACT

Convolutional Neural Network (CNN) is a prevalent topic in deep learning (DL) research for their architectural advantages. CNN relies heavily on hyperparameter configurations, and manually tuning these hyperparameters can be time-consuming for researchers, therefore we need efficient optimization techniques. In this systematic review, we explore a range of well used algorithms, including metaheuristic, statistical, sequential, and numerical approaches, to fine-tune CNN hyperparameters. Our research offers an exhaustive categorization of these hyperparameter optimization (HPO) algorithms and investigates the fundamental concepts of CNN, explaining the role of hyperparameters and their variants. Furthermore, an exhaustive literature review of HPO algorithms in CNN employing the above mentioned algorithms is undertaken. A comparative analysis is conducted based on their HPO strategies, error evaluation approaches, and accuracy results across various datasets to assess the efficacy of these methods. In addition to addressing current challenges in HPO, our research illuminates unresolved issues in the field. By providing insightful evaluations of the merits and demerits of various HPO algorithms, our objective is to assist researchers in determining a suitable method for a particular problem and dataset. By highlighting future research directions and synthesizing diversified knowledge, our survey contributes significantly to the ongoing development of CNN hyperparameter optimization.

Contents

1.	Introduction	2
2.	Related works.....	3
3.	Methodology	4
3.1.	Literature search	4
3.2.	Literature selection	5
4.	Convolutional neural network.....	5
5.	Hyperparameters of CNN.....	6
5.1.	Hyperparameters of convolutional layers.....	7
5.1.1.	Number of convolutional layer	7
5.1.2.	Number of kernels.....	8
5.1.3.	Size of kernels	8
5.1.4.	Activation function.....	8
5.1.5.	Stride	8
5.1.6.	Padding	8
5.2.	Hyperparameters of fully connected layers	8
5.2.1.	Dropout.....	8
5.2.2.	Connectivity pattern	9

* Corresponding author.

E-mail addresses: mohaimenulazamkhan.raiaan@cdu.edu.au (M.A.K. Raiaan), ssakib191097@bscse.uiu.ac.bd (S. Sakib), nurmohammad.fahad@cdu.edu.au (N.M. Fahad), amamun191104@bscse.uiu.ac.bd (A.A. Mamun), mrahman191191@bscse.uiu.ac.bd (M.A. Rahman), swakkhar@cse.uiu.ac.bd (S. Shatabda), Saddam.Mukta@lut.fi (M.S.H. Mukta).

<https://doi.org/10.1016/j.dajour.2024.100470>

Received 20 September 2023; Received in revised form 24 March 2024; Accepted 19 April 2024

Available online 24 April 2024

2772-6622/© 2024 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY-NC license (<http://creativecommons.org/licenses/by-nc/4.0/>).

5.3.	General hyperparameters	9
5.3.1.	Batch size	9
5.3.2.	Learning rate (LR)	9
5.3.3.	Epochs	10
5.3.4.	Optimizer	11
5.3.5.	Loss function	13
6.	Hyperparameter optimization algorithms	13
6.1.	Tree-structured parzen estimator (TPE)	13
6.2.	Gray wolf optimization	14
6.3.	Harmony search algorithm	15
6.4.	Differential evolution	16
6.5.	Genetic algorithm optimization	16
6.6.	Ant colony optimization	17
6.7.	Particle swarm optimization	17
6.8.	Firefly algorithm (FA)	18
6.9.	Nelder–Mead method	19
6.10.	Bayesian optimization	19
7.	Domain specific application of HPO	20
8.	Discussion	20
9.	Open issues and challenges	22
10.	Critical analysis of future research agendas	22
11.	Conclusion	25
	Declaration of competing interest	25
	Data availability	25
	References	28

1. Introduction

Convolutional neural network (CNN) is a class of deep neural network which has proven its effectiveness in the tasks of computer vision (CV), computer-aided diagnosis (CAD), natural language processing (NLP), and pattern recognition [1,2]. CNN has revolutionized these tasks by using its concise, yet powerful architecture [3]. In recent times, with the core architecture of CNN, several pre-trained models have also been introduced for different purposes [4–7]. Unlike traditional neural networks, CNN can reduce the number of parameters remarkably during feature extraction phase [8]. As in the Support Vector Machine (SVM), three important parameters must be carefully selected, such as the kernel function, the regularization parameter C , and Gamma [9], CNN has number of convolutional layers, kernels, activation function, dropout, etc. which are considered as CNN hyperparameters. One of the major strengths of CNN architecture is that it can apply a reasonable amount of hyperparameters which helps researchers to create different versions of CNN-based models. However, researchers struggle frequently to find the best architectural configuration where they need to determine the optimal number of hyperparameters for a problem setting [10].

CNN hyperparameter optimization (HPO) performs the parameter selection for CNN model which provides the best accuracy for any classification task [11,12]. However, this selection process is challenging since a huge amount of parameters need to be adjusted which is time consuming and requires a substantial amount of computational resources. These hyperparameters are typically set based on heuristic principles and manually fine-tuned, configuring a single CNN parametrization can take several hours. To address the problem, several HPO algorithms [13–16] have been introduced over the years to automate the selection process. By slightly modifying the values of these hyperparameters, performance of these models can be improved and reduce the training time significantly. The procedure of optimization is usually an iterative process where all the parameter values are constantly changed to find the optimal values. The existing survey studies [2,17–20] largely lack of available comprehensive review where no studies have discussed extensively on the variants of HPO algorithms. Fig. 1 depicts the high-level view of how HPO algorithms work.

Recent research has focused on the optimization of hyperparameters in CNN for various applications such as healthcare [21], education [22], industry [23], financial forecasting [24], agriculture [25],

etc. [26,27]. In these studies, tuning hyperparameters substantially enhanced model performance. Furthermore, to optimize architectures for forecasting tasks, HPO algorithms were employed to refine long short-term memory (LSTM), CNN, and multi-layer perceptron (MLP) models [28]. For example, particle swarm optimization (PSO) [28] was utilized to optimize models M-1 (PSO-LSTM), M-2 (PSO-CNN), and M-3 (PSO-MLP), with M-1 producing the most favorable outcomes on Beijing PM2.5 datasets. Numerous such studies [29–31] underscore the significance of optimizing hyperparameters to improve the efficacy of deep learning models. Several HPO approaches have been applied in different solutions which include gradient-based optimization for COVID-19 and colon cancer diagnosis [32], bio-inspired algorithms for detecting breast cancer [33], the fuzzy tree model for image classification, and the application of evolutionary computation methods [34] for detecting network intrusions. The aforementioned methods collectively emphasize the importance of effective hyperparameter optimization in improving the precision and performance of various CNN applications.

Researchers have outlined several aspects of HPO algorithms [2,19, 20] in their review papers, but these studies have several limitations. These studies miss many aspects of HPO algorithms including CNN hyperparameter concepts, related datasets, recent open issues, etc. Those studies largely focus on a single HPO algorithm, where they do not make any comparison with other HPO techniques. Therefore, omission of these key points in a review paper indicates that an extensive investigation is required in the current literature. Therefore, the main objective of this study is to conduct a comprehensive review of CNN, providing e an in-depth analysis of four distinct types of hyperparameter optimization algorithms: Sequential model-based optimization, Metaheuristic optimization, Numerical approach-based optimization, and Statistical modeling-based optimization, in order to assess their effectiveness in enhancing CNN performance. Another objective of our study was to examine datasets utilized in CNN experiments to determine their coverage across different domains. Performance evaluation of CNN models utilizing these hyperparameter optimization algorithms has been conducted to evaluate their accuracy. Moreover, the study highlights open issues and challenges encountered in CNN optimization and proposes a guide for future researchers to navigate potential areas of exploration in their research endeavors. The list of contributions of this paper are as follows:

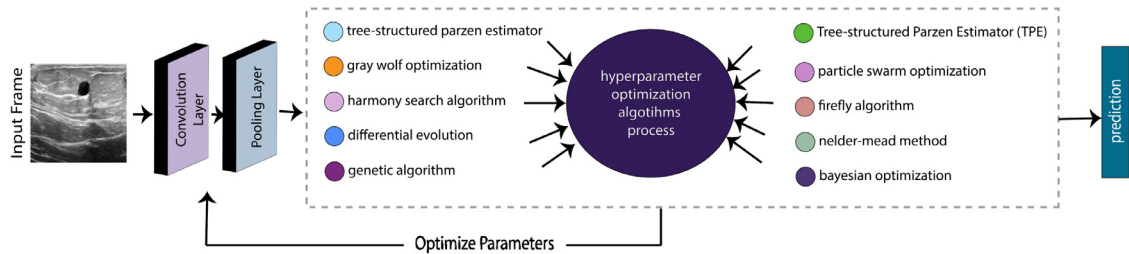


Fig. 1. A standard pipeline how HPO algorithms work on a CNN based system.

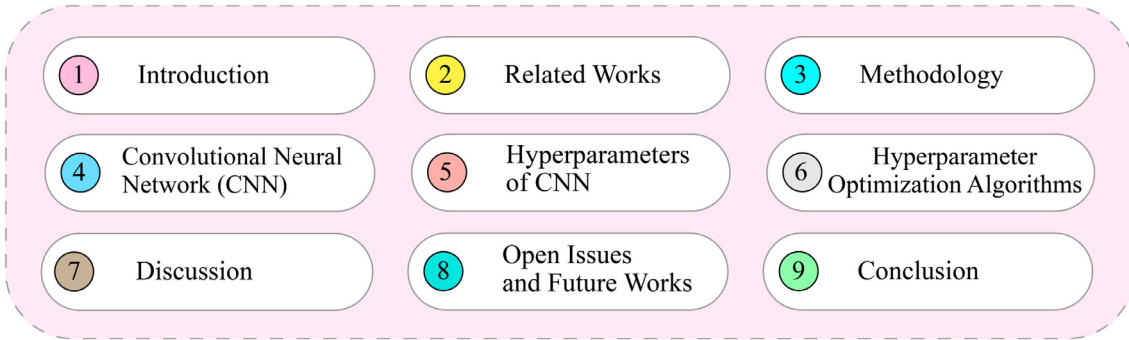


Fig. 2. Organization of the paper.

- A taxonomy of ten HPO algorithms is provided. The HPO algorithms are classified into four distinct categories: *sequential model-based optimization*, *meta-heuristic optimization*, *numerical approach-based optimization*, and *statistical modeling-based optimization*.
- Variants of optimizers are discussed, including their negative and positive attributes, and a visual representation of gradual development has been provided to assist researchers in selecting the best optimizer for their research work.
- Ten HPO algorithms have been meticulously investigated to learn how to optimize CNN hyperparameters using these algorithms.
- Benchmark datasets have been investigated based on the performance of different HPO algorithms, and critical analysis has been made.
- Recent open issues and future work of HPO of CNN using different domain algorithms have been briefly discussed, which will aid future research.

Fig. 2 represents the organization of our paper, including all the contributions. The remainder of the study is structured as follows: Section 2 provides a brief discussion of the state-of-the-art studies, and Section 3 describes the methodology of the entire survey. Sections 4 and 5 comprise the CNN overview and discussion of hyperparameters of CNN, respectively. A detailed discussion of CNN HPO algorithms can be discovered in Section 6. The detailed results of various datasets used for the hyperparameters optimization of CNN are thoroughly discussed in Section 8. Section 9 discusses open issues and future directions in HPO. Section 11 concludes the study in the end.

2. Related works

Optimizing CNNs is important for enhancing model performance, stability, and efficiency. This review examines various HPO techniques for CNNs, including heuristic, meta-heuristic, and statistical-based approaches.

Khalid et al. [17] focused on HPO for smart grid forecasting, favoring nature-inspired heuristic optimization due to its efficiency

with large datasets. They emphasized data preprocessing and feature selection. Han et al. [18] surveyed HPO for single-hidden layer feedforward neural networks, highlighting the importance of meta-heuristic optimization for selecting hyperparameters. The research of Morales et al. [35] reviewed multi-objective HPO algorithms, including metaheuristic-based and metamodel-based approaches, emphasizing trade-offs between performance measures. Nematzadeh et al. [19] proposed hyperparameter tuning by using gray wolf optimization and genetic algorithms for ML algorithms, showing improved training efficacy over grid search. Darwish et al. [20] explored swarm and evolutionary computing techniques for DL, discussing their use in hyperparameter tuning and identifying areas for advancement. Li et al. [2] provided a comprehensive survey of CNN fundamentals, activation functions, loss functions, and optimizers, aiding in function selection based on empirical study. The survey of Elaziz et al. [36] delved into meta-heuristic optimization techniques for enhancing deep neural networks' performance, particularly in handling large-scale data.

The utilization of hyperparameter optimization algorithms in CNN is vital for augmenting the initial explainability of artificial intelligence (AI) systems. By optimizing the parameters of CNN models, these algorithms enhance their interpretability and hold them more responsible for their judgments. Numerous scholarly articles have explored this facet of AI explainability. In this regard, Saranya et al. [37] conducted a systematic literature review of Explainable Artificial Intelligence (XAI) approaches in diverse sectors (including healthcare, manufacturing, transportation, and finance) is presented in this article. From January 2018 to October 2022, the study examined 91 recently published articles. In particular, for high-stakes domains, the review seeks to offer insights and a road map for future research in the field of XAI, emphasizing the need for accountability, transparency, and trustworthiness in AI systems. Another optimization-based research, [38], addresses the vulnerability of IoT-enabled networks to cyber threats and proposes a framework for the intelligent detection of cyber threats. The approach integrates ensemble feature selection based on metaheuristics, employing the Binary Gray Wolf Optimization (BGWO) and Binary

Table 1
Comparison of state-of-the-art survey papers.

Paper	CNN Review	Sequential model based optimization	Meta heuristic optimization	Numerical approach based optimization	Statistical modeling based optimization	Dataset coverage	Performance evaluation	Open issues and challenges	Strengths
Khalid et al. [17] (2020)	✓	X	✓	X	✓	✓	X	✓	Robust analysis and comparison of different algorithms
Han et al. [18] (2018)	X	X	✓	X	X	X	✓	✓	Reduce the complexity of the network and improve the performance.
Morales et al., [35] (2022)	✓	X	✓	X	X	X	X	✓	Explore the potentiality of these algorithms to improve the reproducibility of the solutions.
Nematzadeh et al. [19] (2021)	✓	X	✓	X	X	✓	✓	X	GWO improved the model performance and accelerated the convergence.
Darwish et al. [20] (2019)	✓	X	✓	X	X	X	X	✓	Improve the model classification accuracy and efficiency with a better explainability of the model.
Li et al. [2] (2021)	✓	X	X	X	X	✓	✓	X	Provide insight into the potentiality and advancement of CNN.
Elaziz et al. [36] (2021)	X	X	✓	X	X	✓	✓	X	Perform an experiment and choose the appropriate approach based on the result analysis.
Ours	✓	✓	✓	✓	✓	✓	✓	✓	Various types of optimizers, including their negative and positive sides are discussed, ten hyperparameter optimization strategies were studied to gain insight and explain their use in optimizing CNN hyperparameters, analysis, and comparison hyperparameters optimization techniques on a variety of well-known datasets, evaluate CNN's general hyperparameters, open issues and future work on CNN hyperparameter optimization utilizing various domain algorithms were briefly mentioned.

Gravitational Search Algorithm (BGSA) to optimize feature sets for effective learning.

Our paper comprises a systematic literature review, which undertakes a critical examination and synthesis of extant research within the discipline. Table 1 compares the state-of-the-art studies based on topic coverage, dataset diversity, performance evaluation, and open issues and challenges. Our review encompasses a broader range of optimization techniques and offers a more comprehensive critical analysis, addressing gaps in existing research. Additionally, we conducted experiments on benchmark datasets to assess HPO algorithm performance, further enhancing the dept of our study. Similar to the decision-based review [37–40], our study conducts an exhaustive review that concludes with a conclusive analysis of the insights gleaned from this review. This analysis provides future researchers with a valuable resource that assists them in making well-informed decisions. Moreover, our research extends its scope beyond descriptive and diagnostic analytics by integrating predictive analytics in order to offer future-oriented insights. By adopting this comprehensive approach, the academic and professional community is bolstered with a more reliable basis to make well-informed decisions regarding optimization techniques.

3. Methodology

In our review of hyperparameter optimization algorithms, we follow strict rules called the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) guidelines [41]. A brief discussion related to our methodology is given below.

Table 2
Search queries for hyperparameter optimization algorithms.

Search query	Connector	Algorithm
cnn OR convolutional AND hyperparameter AND optimization	AND	tree AND structured AND parzen gray AND wolf harmony AND search differential AND evolution genetic AND algorithm ant AND colony particle AND swarm firefly AND algorithm nelder AND mead AND method bayesian AND optimization

3.1. Literature search

For the thorough literature search, we conducted searches on two widely used scientific databases: Scopus¹ and Web of Science.² These databases are known for indexing high-quality articles, so we focused solely on them for our search. We used specific search queries outlined in Table 2 to find relevant articles.

¹ <https://www.scopus.com>

² <https://www.webofscience.com>

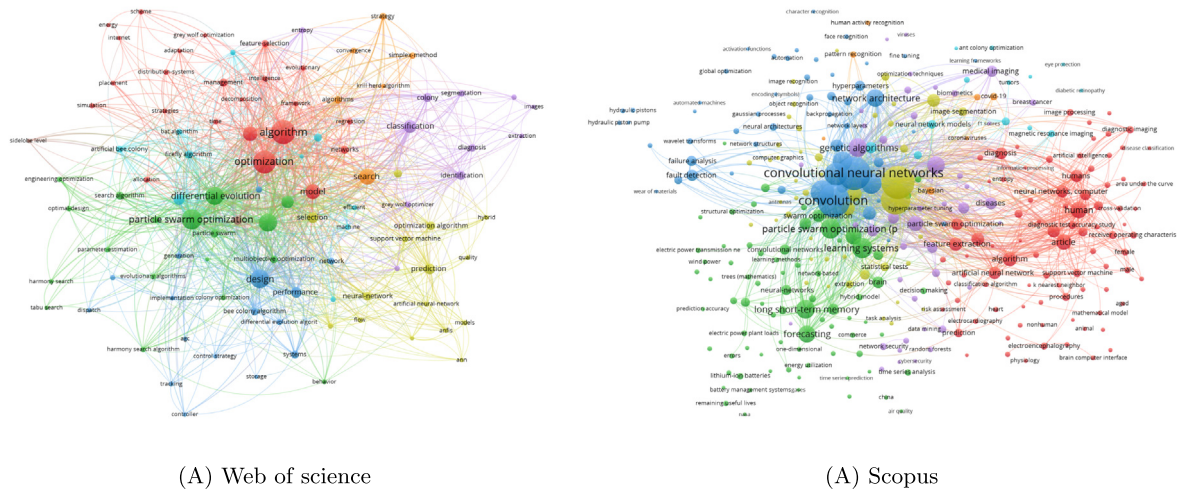


Fig. 3. Visualization of prominent keywords.

Table 3
Inclusion and exclusion criteria.

	Inclusion Criteria(IC)
IC1	Should be listed in one of the chosen databases
IC2	Should be published in the last 8 years (2016–2023)
IC3	Should contain at least one of the keywords
IC4	Should be published in a journal, conference, or magazine
IC5	Title, abstract, and full text should match the study being searched for
	Exclusion Criteria(EC)
EC1	Duplicate items
EC2	Studies not written in English
EC3	The proposed approach does not align with the requirements of hyperparameter optimization algorithms
EC4	Full text cannot be obtained

Our search queries included terms like “CNN” or “convolutional” combined with “hyperparameter” and “optimization”. Additionally, we included ten different names of the algorithms. From Scopus, we initially found 736 articles, while Web of Science yielded 972 articles. This large pool of articles formed the basis of our final article selection. Fig. 3 depicts the keyword variation from the selected 1708 articles. VOSviewer³ created these visualizations, and using this visualization, we eliminated papers that were not relevant to our studies [42,43].

3.2. Literature selection

Following PRISMA guidelines, we follow standard inclusion criteria (IC) and exclusion criteria (EC) when selecting the final research papers. Table 3 is an insightful representation of how we employ the PRISMA guidelines to determine the inclusion and exclusion criteria for the review of our paper. Outlining IC and EC provides a transparent and systematic approach to paper selection. IC requires selected papers to contain relevant keywords, be sourced from designated databases, have a publication date within the last eight years, be published in reputable journals or conferences, and exhibit consistency in title, abstract, and entire text. In contrast, EC allows us to exclude papers not aligning with our research objectives or quality standards, such as redundant articles, incomplete texts, content irrelevant to HPO algorithms, non-English documents, and data quality concerns.

Fig. 4 provides a concise overview of the Prisma diagram, which depicts the systematic review procedure for our review. In the initial

identification stage, $n = 1,708$ papers were initially identified. In the subsequent Screening Stage, we screened abstracts and titles, resulting in a final count of $n = 634$ papers. Moving on to the Eligibility Stage, the third phase consisted of a comprehensive full-text review from which $n = 87$ papers meeting our eligibility criteria were selected. In the final inclusion stage, only 31 papers met our inclusion criteria and have been included in our analysis, and the selected articles are depicted in Table 4.

4. Convolutional neural network

CNN architecture consists of three main layers: (a) Convolutional layers, (b) Pooling layers, and (c) Fully connected layers. There are input and output layers as well [3,74]. All the layers have massive significance for the overall performance of the model. Therefore, in this study, extensively all the descriptions discussed below. Fig. 5 demonstrates the overall architecture of CNN.

(a) Convolutional layers: CNN is distinguished from other neural networks by its convolutional layer, a significant feature layer comprising sets of convolutional filters [74]. The convolution operation becomes a correlation operation because of a symmetric kernel [75]. It converts the images into compact matrix proportion to kernel size and extracts essential features from images [76]. The equation of convolution operation is given with zero padding [75]:

$$S_{i,j} = (I * K)_{i,j} = \sum_m \sum_n I_{i,j} \cdot K_{i-m,j-n} \quad (1)$$

³ <https://www.vosviewer.com/>

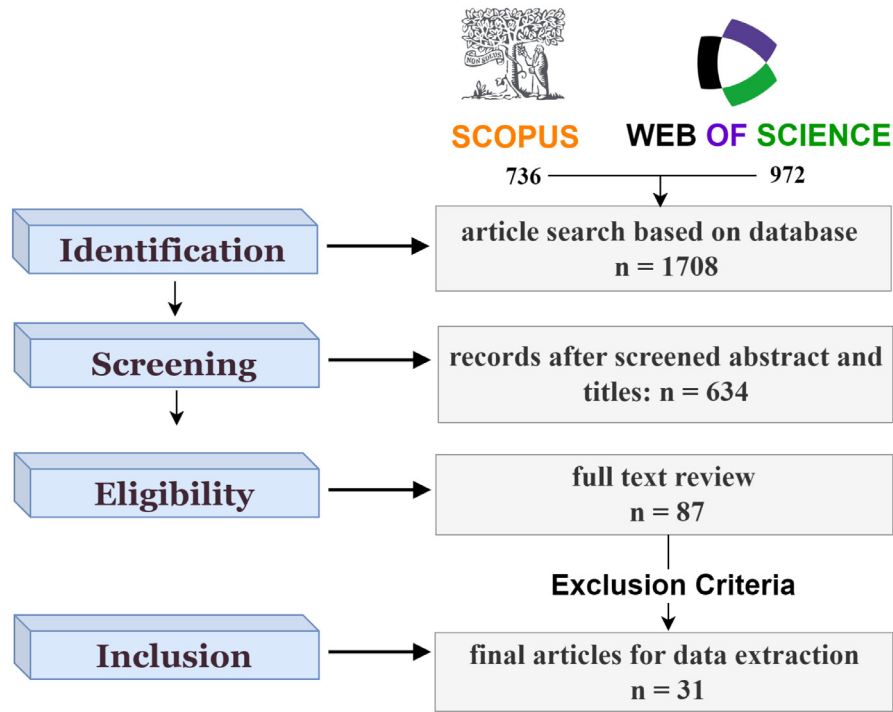


Fig. 4. A PRISMA flowchart illustrating the various phases of the review selection process.

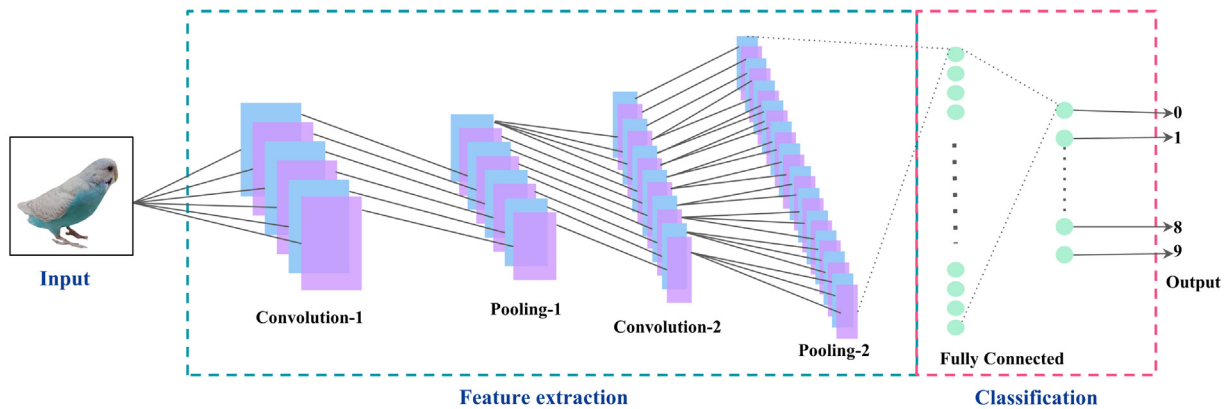


Fig. 5. CNN architecture.

Convolution produces a convolved image by traversing a small filter from top to bottom from left to right and repeats this method to create many output feature maps [77].

(b) Pooling Layer: The pooling is a significant layer in CNN since it reduces the number of connections between convolutional layers as well as reduces computational cost [78]. After convolution, feature maps are produced with crucial features, and the pooling or down-sampling substantially decreases the complexity of the model compiling similar information using dominant features [78,79]. The pooling layers vary by the specific tasks, and Table 5 gives an overview.

Table 5 is a guide for understanding the characteristics, applications, and trade-offs of different pooling layers in CNN. By comparing their strengths and weaknesses, researchers can make informed decisions when selecting the appropriate pooling strategy for their specific tasks. Fig. 6 visualizes the difference in output calculation among all the pooling operations.

(c) Fully connected layers: In CNN, fully connected (FC) layer neurons are all interconnected [91]. It flattens the last convolution layer or pooling layer output into a vector and passes it to the fully connected layer. The FC layer is complex and requires computational

training due to it generating many parameters and it also has different variants [92]. The number of FC layers increases the memory and computational power of the model.

5. Hyperparameters of CNN

Hyperparameters in a CNN are crucial parameters set before model training, governing its learning process and impacting performance. In CNNs, these parameters define aspects like architecture, hidden layers, dropout rates, and learning algorithms, influencing feature extraction, overfitting avoidance, and convergence to optimal solutions. Effective tuning of hyperparameters is essential for optimizing CNN structure and training, improving performance and accuracy. CNN hyperparameters can be divided into three categories:

1. Hyperparameters of convolutional layers (i.e., # of conv. layer, # of kernel, size of kernel, etc.)
2. Hyperparameters of fully connected layers (i.e., dropout, connectivity pattern, etc.)
3. General Hyperparameters (i.e., batch size, learning rate, etc.)

Table 4
List of selected publications.

Author(s)	Journal/conference name	Publication year	Algorithm name
Dong et al. [44]	Powder Technology	2020	Tree-structured Parzen Estimator
Rong et. al. [45]	Remote Sensing	2021	Tree-structured Parzen Estimator
Oyewala et. al. [46]	Applied Science	2022	Tree-structured Parzen Estimator
Shukla et. al. [47]	Neural Computing and Applications	2020	Gray Wolf Optimization
Mohakud et. al. [48]	Journal of King Saud University-Computer and Information Sciences	2022	Gray Wolf Optimization
Lee et al. [12]	Optik	2018	Harmony Search Algorithm
Kim et al. [49]	MDPI Sensors	2020	Harmony Search Algorithm
Huang et al. [50]	Natural Computation, Fuzzy Systems and Knowledge Discovery	2019	Harmony Search Algorithm
Podgorelec et al. [51]	Applied Sciences	2020	Differential evolution
Mahdaddi et al. [52]	Expert Systems with Applications	2021	Differential evolution
Belcuig et al. [53]	Computers in Biology and Medicine	2022	Differential evolution
Ghasemi et al. [54]	Decision Analytics Journal	2023	Differential evolution
Lee et al. [55]	Applied Sciences	2021	Genetic Algorithm
Lopez-Rincon et al. [56]	Applied Soft Computing	2018	Genetic Algorithm
Agrawal et al. [57]	Decision Analytics Journal	2022	Genetic Algorithm
Manna et al. [58]	Decision Analytics Journal	2023	Genetic Algorithm
Byla et al. [59]	Computational Intelligence	2019	Ant Colony Optimization
Lankford et al. [60]	Neural and Evolutionary Computing	2020	Ant Colony Optimization
Tatsuki et al. [61]	Neural and Evolutionary Computing	2020	Particle Swarm Optimization
Singh et al. [62]	Swarm and Evolutionary Computation	2021	Particle Swarm Optimization
Shaikh et al. [63]	Decision Analytics Journal	2023	Particle Swarm Optimization
Zare et al. [64]	Decision Analytics Journal	2023	Particle Swarm Optimization
Bacanin et al. [65]	Algorithms	2020	Firefly Algorithm
Bacanin et al. [66]	Journal of Real-Time Image processing	2021	Firefly Algorithm
Aswanandini et al. [67]	Indian Journal of Science and Technology	2021	Firefly Algorithm
Ghasemi et al. [68]	Decision Analytics Journal	2022	Firefly Algorithm
Albelwi et al. [69]	Entropy	2017	Nelder–Mead method
Loey et al. [70]	Computers in Biology and Medicine	2022	Nelder–Mead method
Sameen et. al. [71]	Catena	2020	Bayesian Optimization
Xu et al. [72]	Decision Analytics Journal	2023	Bayesian Optimization
Lahmiri et al. [73]	Decision Analytics Journal	2023	Bayesian Optimization

Table 5
Overview of different pooling layers.

Pooling type	Definition	Equation	Suitable use case (strength)	Weakness
Max Pooling [80]	Selects the maximum value from a region	$\text{pool} = \text{down}(\max(y_{i,j})), i, j \in p$	- Translation invariant: useful for detecting features regardless of their position in the input data [81,82].	- Loss of spatial information [83].
Average Pooling [84]	Computes the average value in a region	$M_j = \tanh(\beta \sum_{N \times N} M_i^{n \times n} + b)$	- Smoothing effect: helps to reduce noise and overfitting in the data [85,86].	- May not capture important features accurately [87].
Rank-based Average Pooling (RAP) [88]	Calculates the average of sorted values	$S_j = \frac{1}{t} \sum_{i \in R_j, r_i < t} a_i$	- Robust to outliers: less sensitive to extreme values in the input data [89].	- Requires sorting the values which can be computationally expensive, especially for large inputs [90].

5.1. Hyperparameters of convolutional layers

The size and number of kernels are two important hyperparameters that define convolution operations. Other hyperparameters, including stride and padding, also need to be tuned for an optimal solution [93].

5.1.1. Number of convolutional layer

Convolutional layers extract feature information using the convolute operation. The model's overall performance depends on the number of convolutional layers [94]. Sometimes adding more hidden layers can boost the performance, as proven in AlexNet, VGG16, and ResNet

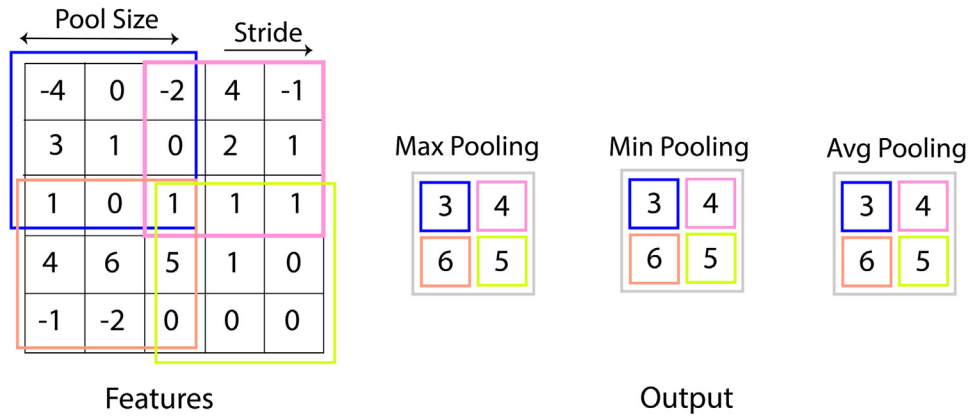


Fig. 6. Illustration of the output calculation of pooling operation.

architecture [95]. These models manually update the number of layers for the best convolutional layer number. It also relies on dataset size. Layers with lower strides achieve higher accuracy in larger datasets, but overfitting occurs in smaller datasets [96].

5.1.2. Number of kernels

The number of convolutional kernels impacts the image's feature extraction. Several kernels, usually 32 or 64, are used in CNN-based models to extract critical features and improve model performance [97]. These kernels create feature maps of the images' bottom edges, corners, diagonal, vertical lines, and angles [98]. The more kernels increase the amount of data subsequently in the model.

5.1.3. Size of kernels

Kernel size is equally important in model performance having different sizes, and the common of them are 1×1 , 2×2 , 3×3 , 5×5 , 7×7 [97]. Some datasets perform well in small kernels, and others obtain the best accuracy in large kernel sizes. Small kernel sizes are preferable since they require less computation cost, whereas large kernels have the drawback of taking a longer time to accomplish the convolution operation [99].

5.1.4. Activation function

The activation function is a decision-making function that aids in the recognition of complex patterns. The learning process can be quickened by selecting the appropriate activation function [79]. There are different types of activation functions, including sigmoid, tanh, and Rectified Linear Unit (ReLU), as well as ReLU variations like leaky ReLU, Exponential Linear Unit (ELU), and Parametric Rectified Linear Unit (PReLU), which are used to implement non-linear feature combinations.

Table 6 illustrates a brief overview of the activation functions of neural networks. The choice of activation function depends on the specific task and network architecture. Sigmoid and Tanh are suitable for specific classification tasks, while ReLU is commonly used in hidden layers. PReLU offers flexibility with its learnable slope, Leaky ReLU prevents dead neurons, ELU provides a smooth output, and Softmax is ideal for multi-class classification. Understanding the strengths and weaknesses of each function is crucial in selecting the most appropriate one for a given neural network application.

5.1.5. Stride

Stride is a method of shifting a certain amount of pixels across the input matrix, which impacts output size and ensures output singularity [94]. In Fig. 7, a detailed visualization is presented.

When a CNN stride is set to 1, the filter travels by one pixel at a time throughout the convolution operation. This results in a more detailed analysis of the input data, capturing features but potentially

leading to a larger computational load due to the higher number of operations [107]. Alternatively, when the stride is set to 2, the filter advances by two pixels in each step, leading to a downsampled output with diminished spatial dimensions. This can reduce the computational complexity and memory requirements of the model while capturing more global features. If a model uses a stride larger than 2, such as three or more, the downsampling effect becomes more pronounced, leading to further reduction in spatial dimensions and an increased focus on capturing high-level features. However, using larger strides can also result in information loss, especially in intricate patterns or small details, potentially impacting the model's ability to learn from fine-grained features. Additionally, larger strides may lead to a coarser representation of the input data, which could affect the model's overall performance on tasks that require detailed information. Therefore, the choice of stride value in a CNN is a trade-off between computational efficiency, feature representation, and model performance based on the specific requirements of the tasks [107,108].

5.1.6. Padding

Padding is a technique for increasing the input matrix size to fit the kernel correctly and can control the behavior of convolutional layers [109]. Three types of padding techniques are available. Table 7 gives an overview for deciding which padding to use in CNN. The selection depends on the model's specific requirements and the task at hand. The same padding is preferred when preserving spatial dimensions, which is crucial, such as in tasks requiring precise object localization or maintaining the resolution of feature maps. It ensures that positional details and fine-grained information are accurately captured by maintaining spatial dimensions throughout the network. Valid padding is suitable when reducing spatial dimensions of feature maps is desired, as it computes only the valid elements without adding new ones, resulting in a smaller output size. Causal padding is specialized for sequence-related tasks like natural language processing and time series analysis, where padding is applied asymmetrically to aid in forecasting early time steps. Each sort of padding has a distinct purpose, depending on whether it aims to preserve spatial information, decrease spatial dimensions, or meet sequence data needs [110].

5.2. Hyperparameters of fully connected layers








The FC has several hyperparameters that are outlined below:

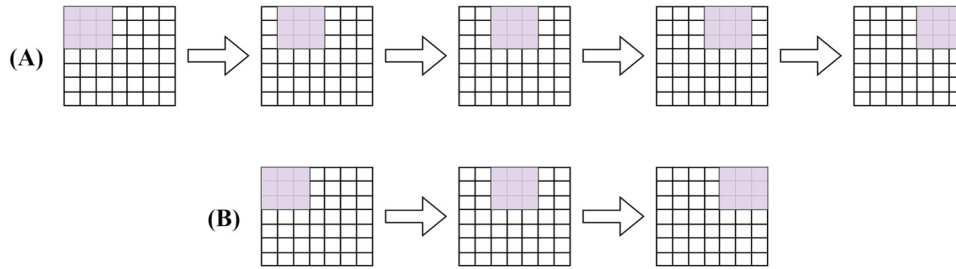
5.2.1. Dropout

In neural networks, working with many data points generates many parameters and causes overfitting during training, and Dropout regularization is introduced to solve this problem by randomly dropping units from networks. [112]. Dropout can be tuned in different values to optimize the hyperparameters. In Fig. 8, a detailed visualization is presented.

Table 6

Overview of different activation functions.

Activation function	Definition	Equation	Range	Suitable use case (strength)	Weakness	Figure
Sigmoid [100]	S-shaped curve, maps inputs to values between 0 and 1	$Sigmoid = \frac{1}{1+e^{-z}}$	(0, 1)	Binary classification, output probability interpretation	Vanishing gradient, not zero-centered	
Tanh (Hyperbolic Tangent) [101]	Maps inputs to values between -1 and 1	$Tanh = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	(-1, 1)	Classification between two classes, zero-centered output	Vanishing gradient at extremes	
ReLU (Rectified Linear Unit) [102]	Outputs zero for negative inputs, linear for positive inputs	$g(x) = \max(0, x);$ $f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}$	$[0, +\infty)$	Hidden layers in deep learning, avoids vanishing gradient	Dying ReLU for negatives	
PReLU (Parametric ReLU) [103]	PReLU with learnable negative slope	$PReLU(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha \cdot x & \text{otherwise} \end{cases}$	$(-\infty, +\infty)$	Improved performance, adaptable slope	Tuning parameter complexity	
Leaky ReLU [104]	Variant of ReLU with small gradient for negatives	$y_i = \begin{cases} x_i & \text{if } x_i \geq 0 \\ \frac{x_i}{a_i} & \text{if } x_i < 0 \end{cases}$	$(-\infty, +\infty)$	Preventing dead neurons, faster convergence	Vanishing gradients at low negatives	
ELU (Exponential Linear Unit) [105]	Exponential Linear Unit allowing negative values	$g(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha (e^x - 1) & \text{if } x \leq 0 \end{cases}$	$(-\infty, +\infty)$	Strong alternative to ReLU, smooth output	Computationally expensive	
Softmax [106]	Converts a vector of real values into a probability distribution	$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$ for $j = 1, \dots, K$	(0,1)	Multi-class classification, interpretable probability distribution	Not suitable for regression tasks	

**Fig. 7.** Visualizations for two scenarios: (a) when the stride is set to 1, and (b) when the stride is set to 2.

5.2.2. Connectivity pattern

Connectivity pattern refers to the connection pattern between a layer and its preceding layers. To solve the information loss [113] this method is introduced, and it decreases the redundancy of the same features.

5.3. General hyperparameters

Apart from the hyperparameters of convolutional layers and fully connected layers, other hyperparameters such as batch size, learning rate, epochs, optimizer, and loss function may be modified to enhance the performance of the CNN architecture. [114,115].

5.3.1. Batch size

Batch size is the number of data processed simultaneously during gradient estimation. A high batch size delays network convergence. Small batch sizes can disrupt networks and yield poor outcomes. Therefore, a model must select the right batch size. According to Kandel et al. [116], small batch sizes yield better results than large ones, ranging between 2 and 32. Batch sizes of 16, 32, and 64 are commonly employed using other hyperparameters.

5.3.2. Learning rate (LR)

The LR determines the frequency of parameter revisions for optimal results. Incorrect learning rates may influence model performance. Georgakopoulos et al. [117] indicate that increasing LR accelerates convergence when two successive steps have the same gradient vector direction. Fig. 9 depicts different LR in different layers.

Table 7
Summary of padding types in CNN.

Padding types	Definition	Suitable use case (strength)	Weakness
Same Padding [111]	Padding technique where additional elements, typically zeros, are added to the input image to maintain the output size the same as the input size.	Preserving spatial dimensions, preventing information loss at edges	Increases computational complexity, potential overfitting
Valid Padding [111]	Padding technique where no additional padding is added, resulting in the output size being smaller than the input size.	Reducing spatial dimensions of feature maps	Loss of information at borders, potential information compression after convolution
Causal Padding [111]	Padding technique used in sequence-to-sequence models and time series forecasting, adding elements to the start of the data to aid in forecasting early time steps.	Sequence-to-sequence models, time series forecasting	Specific to sequence data, may not be suitable for general image processing

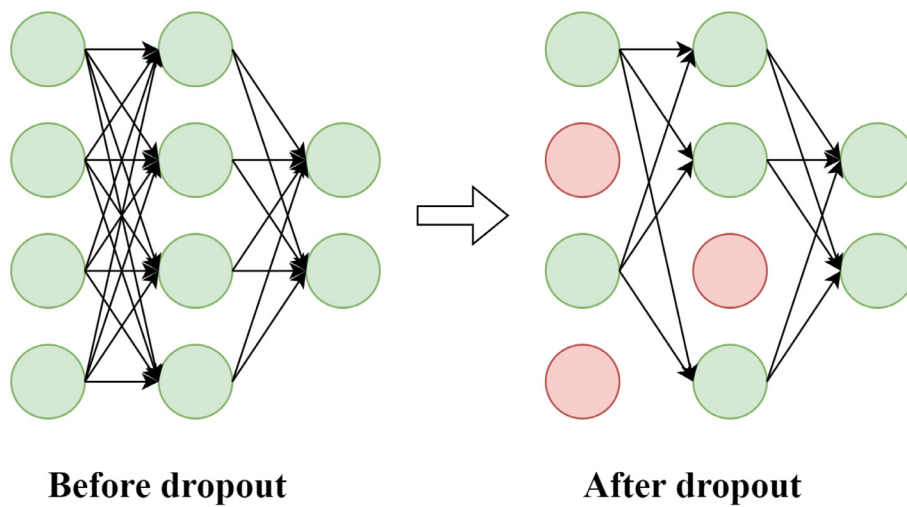


Fig. 8. Visualization before dropout and after dropout (Red neurons are dropped).

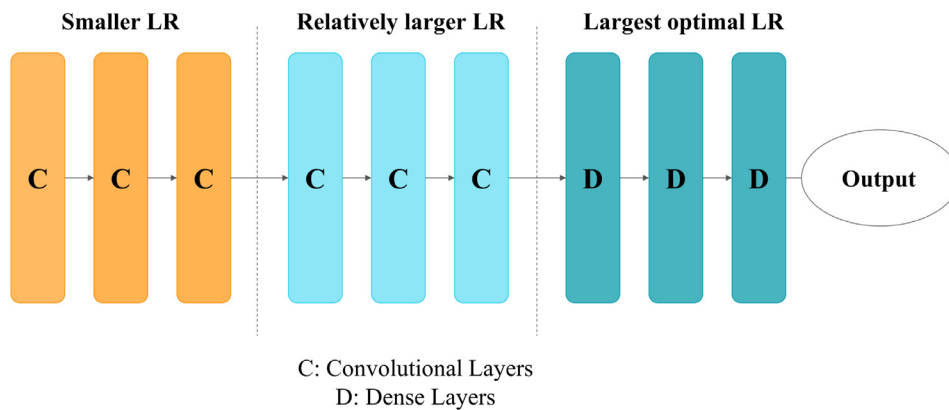


Fig. 9. Different LR in different layers.

Table 8 depicts that fixed learning rates offer simplicity, yet they lack adaptability; adaptive learning rates provide flexibility although they require more complexity, and cyclical learning rates balance exploration and exploitation but demand careful parameter tuning for optimal results in training CNNs. Each learning rate offers better results for different needs and scenarios, emphasizing the importance of selecting the appropriate strategy based on the specific requirements of the neural network being trained.

5.3.3. Epochs

The number of epochs [119] determines the number of full passes (forward and backward) for a neural network during training. Passing all training data once is inadequate to retrieve neural network properties. The dataset may need to be input multiple times to improve generalization, possibly many times. Overfitting is also a major issue in a network, and a model needs the right number of epochs to avoid overfitting. The right epoch size for all datasets is unknown. To determine the optimal number of epochs, Sinha et al. [120] recommended

Table 8
Summary of different types of learning rate in CNN.

Types	Definition	Strength and use case	Weakness
Fixed Learning Rate [118]	A constant learning rate that remains the same throughout training	Simple to implement, suitable for stable datasets and architectures	May converge slowly or get stuck in suboptimal solutions
Adaptive Learning Rate [118]	Adjusts the learning rate during training based on the model's performance	Adapts to different parts of the model, speeds up convergence, and handles sparse data well	Complexity in implementation and tuning, may require more computational resources
Cyclical Learning Rate [118]	Varied learning rates that cyclically change during training to explore different learning rates	Helps escape local minima, improves generalization, and fine-tunes models effectively	Requires careful tuning of cycle lengths and learning rate ranges

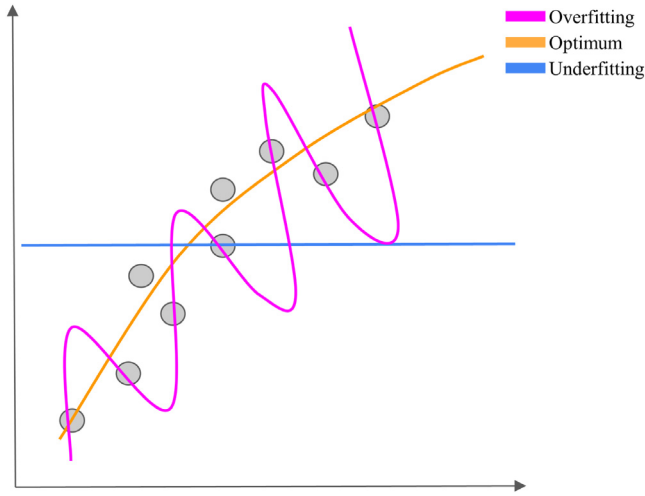


Fig. 10. Overfitting, optimum, and underfitting.

utilizing a self-organized map (SOM) technique. SOM helps choose data for network training and testing to avoid overfitting and optimal epoch size.

Fig. 10 illustrates the overfitting, optimum, and underfitting issues.

5.3.4. Optimizer

Optimizers are crucial for training neural networks by adjusting their weights and learning rates to minimize the loss function. In the CNN context, various optimizers are available, each with advantages and suitability for different datasets. A detailed discussion of mainly used optimizers is given below :

(a) Gradient Descent (GD): Gradient Descent is a fundamental optimization technique in ML and DL. It updates weights iteratively to minimize the cost function. Three key variations of GD are often used: Batch Gradient Descent (BGD), Stochastic Gradient Descent (SGD), and Mini-Batch Gradient Descent. The method is expressed as follows [121]:

$$w_{t+1} = w_t - \gamma \nabla_{w_t} J(w_t) \quad (2)$$

Eq. (2) represents the update rule for Gradient Descent (GD), where w are the weights, γ is the learning rate, J is the cost function, and t is the iteration index.

GD has three common variations: Batch Gradient Descent (BGD), Stochastic Gradient Descent (SGD), and Mini-Batch Gradient Descent (Mini-batch GD).

(i) Batch Gradient Descent (BGD): BGD processes the entire training dataset in each iteration, making it suitable for convex problems

but computationally expensive for large datasets [122]. The formula for BGD is:

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta) \quad (3)$$

Here, θ is the entire training dataset.

(ii) Stochastic Gradient Descent (SGD): SGD selects a single random training instance in each iteration, making it faster but less stable in convergence compared to BGD [123].

$$w_{t+1} = w_t - \gamma \nabla_{w_t} J(x^t, y^t; w_t) \quad (4)$$

Here, y is the target variable, x denotes a single observation, (w_t) is the current parameter value at t time step and (w_{t+1}) is the updated parameter value and γ is the learning rate.

(iii) Mini Batch Stochastic Gradient Descent (Mini-batch GD): Mini-batch GD combines features of both BGD and SGD by processing a batch of training data in each iteration. It balances computation cost and convergence speed [124].

$$w_{t+1} = w_t - \gamma \nabla_{w_t} J(x^{t:t+b}, y^{t:t+b}; w_t) \quad (5)$$

Here, b denotes the size of a single batch, y stands for the target variable, γ is the learning rate, x is a single observation, (w_t) is the current parameter value at t time step, (w_{t+1}) is the updated parameter value, $J(x^{t:t+b}, y^{t:t+b})$ is the loss function.

(b) Nesterov Accelerated Gradient (NAG): SGD's drawbacks negotiating ravines near minima can cause ascending slopes and slow progress. Momentum [125] accelerates SGD but may exceed the minimum. Using NAG [126], movement intelligence is improved by calculating momentum and identifying direction, addressing momentum overshooting. The momentum equation:

$$v_t = \gamma v_{t-1} + \eta \nabla_{\theta} J(\theta) \quad (6)$$

$$\theta = \theta - v_t \quad (7)$$

The momentum optimization approach utilizes the parameter γ , usually set at 0.9, to determine the momentum accumulation rate. When momentum exceeds the minimum, a positive slope and deeper steps back to the minimum can occur. NAG introduces a look-ahead step to save steps. This is the NAG equation:

$$v_t = \gamma v_{t-1} + \eta \nabla_{\theta} J(\theta - \gamma v_{t-1})$$

To avoid overshooting and improve convergence, the gradient at $\theta - \gamma v_{t-1}$ is calculated before updating v_t .

(c) Adam : The Adam optimizer implements stochastic gradient descent with memory-efficient replacement optimization. Kingma et al. proposed it in 2015 [127], an effective algorithm to deal with a large amount of data or parameters. It combines RMSProp, Nesterov momentum, and AdaGrad optimizer features [128,129] to accelerate convergence in the early stages of training while gradually decreasing the learning rate throughout the training period [130]. Adam optimizer is chosen over conventional optimizers due to its memory efficiency,

capacity to handle noisy data, and faster convergence time [128]. The mathematical expression of Adam optimizer is given by [131]:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (8)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (9)$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (10)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (11)$$

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \quad (12)$$

Where, At time step t , η represents the step size, m_t is the gradient moving average, v_t denotes the squared gradient moving average, β_1 and β_2 indicate hyperparameters, and g_t denotes the current mini-batch gradient. At timestep t , \hat{m}_t reflects the bias-corrected estimators for the first moment, and \hat{v}_t is the second moment. At timestep t , the model weight is w_t , and at timestep $t-1$, it is w_{t-1} . The constant ϵ ensures numerical stability.

(d) Nadam Nesterov-accelerated Adaptive Moment Estimation (Nadam) [132] an extended version of Adam momentum, combines it with Nesterov momentum gradient descent [129] and utilizes NAG insights [133]. For Nadam, The Nesterov momentum approach accelerates model training with exponential decay based on the gradient moving average. Compared to the Adam optimizer, the Nadam optimizer converges faster and is better for pre-training [134]. Avoiding local optimums and achieving high-accuracy solutions with rapid convergence make the Nadam optimizer preferable to other optimizers. To minimize overfitting, The Nadam can collaborate with the early-stop category during training [135]. The mathematical expression of Nadam [129]:

$$w_t^i = w_{t-1}^i - \frac{\eta}{\sqrt{v_t} + \epsilon} \cdot \tilde{m}_t \quad (13)$$

$$\tilde{m}_t = \beta_1^{t+1} \hat{m}_t + (1 - \beta_1^t) \hat{g}_t \quad (14)$$

$$\hat{m}_t = \frac{m_t}{1 - \prod_{i=1}^t \beta_1^i} \quad (15)$$

$$\hat{g}_t = \frac{g_t}{1 - \prod_{i=1}^t \beta_1^i} \quad (16)$$

Where, η is the step size, β_1 is the hyperparameter, at time step t , w_t is the weight, m_t is the first moment, g_t is the gradient.

(e) Adamax: The Adamax [127] optimizer is an extended version of the Adam optimizer. It employs gradient descent optimization and outperforms Adam optimizer [136]. Adamax optimization updates weight parameters using the moment's infinity norm, not the second-order moment estimate. In contrast to the Adam optimizer, AdaMax's parameter update is easy. Adamax optimizer has more accurate weight update rules [134]. AdaMax's mathematical expression is [137]:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (17)$$

$$u_t = \max(\beta_2 u_{t-1}, |g_t|) \quad (18)$$

$$\theta_t = \theta_{t-1} - \left(\frac{\alpha}{1 - \beta_1^t} \right) \frac{m_t}{u_t} \quad (19)$$

Where, at time step t , m_t is the first moment vector, g_t is the gradient, β_1 , β_2 both are exponential decay rates, u_t is the exponentially weighted infinity norm, and α is the learning rate.

(f) RMSprop: RMSprop introduced by Geoff Hinton, an adaptive learning rate optimization method [138]. It utilizes gradient descent optimization to provide a solution to the problem of excessive reduction of the learning coefficient [136]. New learning coefficients are utilized

at each phase to address excessive reduction in learning coefficients. Similar to Adadelta, it divides the learning rate by the squared gradient average, which decays exponentially [130]. The RMSprop optimization algorithm mathematical expression is [136]:

$$E[g^2]_t = 0.9 E[g^2]_{t-1} + 0.1 g_t^2 \quad (20)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \quad (21)$$

$$g_t = \nabla_{\theta_t} \mathfrak{F}(\theta_t) \quad (22)$$

Where, η is the step size, at time step t , g_t is the stochastic objective, E is the expected value, ϵ is a constant for numerical stability, and \mathfrak{F} denotes the objective function that needs to be minimized.

(g) Adaptive Gradient (AdaGrad): Adaptive Gradient (AdaGrad) is a gradient-based optimization method that outperforms other optimizers for sparse gradients [139]. It is introduced to scale the learning rate for each weight by updating different weights [129]. The algorithm automatically adjusts the learning rate based on parameters, with frequent parameters receiving small updates and infrequent parameters receiving large updates [129,140]. Hence, it performs better with sparse data. The mathematical expression for the AdaGrad optimization is:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\epsilon I + \text{diag}(G_t)}} g_t \quad (23)$$

$$g_t = \frac{1}{n} \sum_{i=1}^n \nabla f(x^{(i)}, y^{(i)}, \theta_t) \quad (24)$$

$$G_t = \sum_{\tau=1}^t g_{\tau} g_{\tau}^T \quad (25)$$

Where, η is the learning rate, at time step t , θ is the updated parameter, ϵ is a constant for avoiding the division by zero, g_t is the gradient, I is the identity matrix.

(h) AdaDelta: Adaptive Moment Estimation(AdaDelta) optimization [141] proposed by Zeiler is a stochastic gradient descent method and an extension of the Adagrad optimizer reduces Adagrad's aggressive and monotonically declining learning rate automatically [142]. Like AdaGrad, It also performs minor updates for frequent parameters and significant updates for infrequent parameters [130]. Although it is regarded as being faster and outperforming other optimizers, its performance may vary depending on the input attributes and activated activation techniques [140]. Zeiler suggests it is less computationally intensive and resistant to large gradients than gradient descent [141]. The mathematical expression for the AdaDelta optimization is:

$$g_t = \frac{\partial J(\theta)}{\partial(\theta)}, \quad \Delta \theta_t = \theta_t - \theta_{t-1} \quad (26)$$

$$n_t = m n_{t-1} + (1 - m) g_t^2 \quad (27)$$

$$\Delta \theta_t = - \frac{\alpha}{\sqrt{n_t} + \epsilon} g_t \quad (28)$$

$$E[g^2]_t = 0.5 E[g^2]_{t-1} + 0.5 g_t^2 \quad (29)$$

$$\Delta \theta_t = - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} g_t \quad (30)$$

Where m is the coefficient, α is the learning rate, ϵ is a constant to stop the denominator from being zero, g_t is the gradient, θ_t is the updated parameter, and $E[g^2]_t$ is the anticipated gradient's absolute squared value.

Optimizer methods reduce losses and provide precise outcomes. Finding the right network optimizer is crucial. Fig. 11 in our study explores how various optimizers overcome disadvantages.

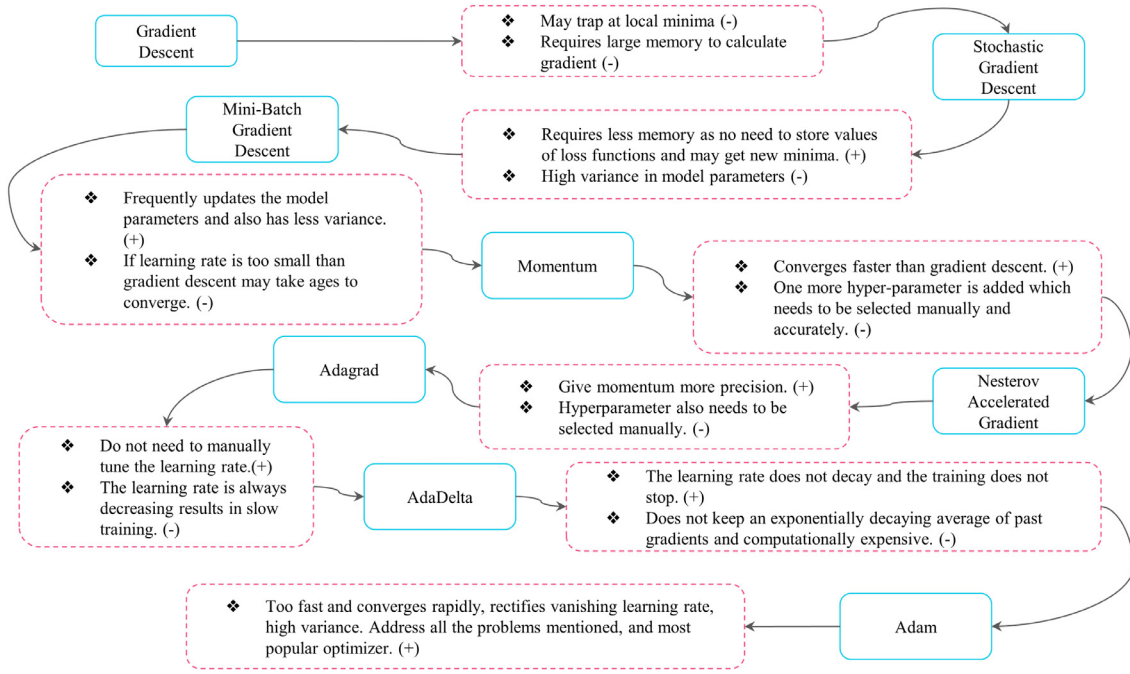


Fig. 11. Variants of optimizer and its negative and positive sides.

5.3.5. Loss function

The loss function is the difference between the predicted and generated results by the ML model [143]. The loss function can drive gradients to change network weights. A mathematical function evaluates model performance. High-loss functions yield poor model accuracy, while low-loss functions produce correct results. Fine-tuning parameters reduces loss and improves model accuracy [144].

(a) Categorical Crossentropy: The categorical cross-entropy loss function solves the multiclass classification problem [145]. In categorical cross-entropy, softmax is used as an activation function and the model's output is scaled using this function to give it the appropriate properties. As it combines cross-entropy loss and softmax activation, it is also known as softmax loss. Additionally, the categorical cross-entropy loss function determines the probability distribution difference. The equation for Categorical Crossentropy is given [146]:

$$CE_{general} = - \sum_i^C t_i \log s_i \quad (31)$$

Where, for the i th class in classes C , t_i is the true value, and s_i is the predicted value.

(b) Sparse Categorical Crossentropy:

There is only one distinction between categorical cross-entropy and sparse categorical cross-entropy [128], and that is the format of true labels. In contrast to categorical cross-entropy, which generates a one-hot array containing the likely match for each category, sparse categorical cross-entropy generates a category index of the most likely matching category. The equation for Sparse Categorical Crossentropy is given by [147]:

$$Loss = - \sum_{i=1}^M y_i \log(p_i) \quad (32)$$

Where y_i is the true value, and p_i is the predicted value, M is the total number of data.

(c) Binary Crossentropy:

Binary cross-entropy is a loss function that solves problems involving binary class classification. When each classification is reduced to a binary choice, binary cross-entropy can solve multiple classification problems simultaneously. In binary cross-entropy, the sigmoid is used as an activation function. As it combines cross-entropy loss and sigmoid

activation, it is also known as sigmoid cross-entropy loss [148]. In contrast to categorical cross-entropy loss, it is not dependent on each vector component which means that the values of other components have no impact on the loss computed for each vector component of the CNN output. The equation for Binary Crossentropy is given by [149]:

$$L(y, \hat{y}) = - \frac{1}{N} \sum_{i=0}^N (y * \log(\hat{y}_i) + (1 - y) * (1 - \hat{y}_i)) \quad (33)$$

Where, y is the true value, and \hat{y}_i is the predicted value, N is the total number of data.

(d) Cosine similarity: The similarity of two objects is measured using cosine similarity by calculating the cosine of the angle between the feature vectors [150]. Cosine similarity measures vector similarity in the direction of vectors but ignores the vector magnitude differences. As the range for cosine is $-1 \leq \cos \alpha \leq 1$, if the value is -1 , the two objects are entirely dissimilar, but if the value is 1 , the two objects are identical. When the value is near -1 , it indicates higher dissimilarity; when it is near 1 , it indicates higher similarity. As a result, it can be used as a loss function in situations where the similarity between predictions and targets is maximized. Cosine similarity is calculated using the formula given below [150]:

$$\cos \alpha = \frac{A \times B}{|A| \times |B|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}} \quad (34)$$

Where A and B are the weight of each feature of vectors A , and B , respectively.

6. Hyperparameter optimization algorithms

Several HPO algorithms have been introduced in different taxonomies over the years to optimize the network architecture of CNN. In this study, we briefly review ten different most popular algorithms from four different taxonomy, which is described in this section. A detailed taxonomy of HPO algorithms is provided in Fig. 12.

6.1. Tree-structured parzen estimator (TPE)

Tree-structured Parzen Estimator (TPE) is an effective method to optimize the hyperparameters of the CNN model. The method is based

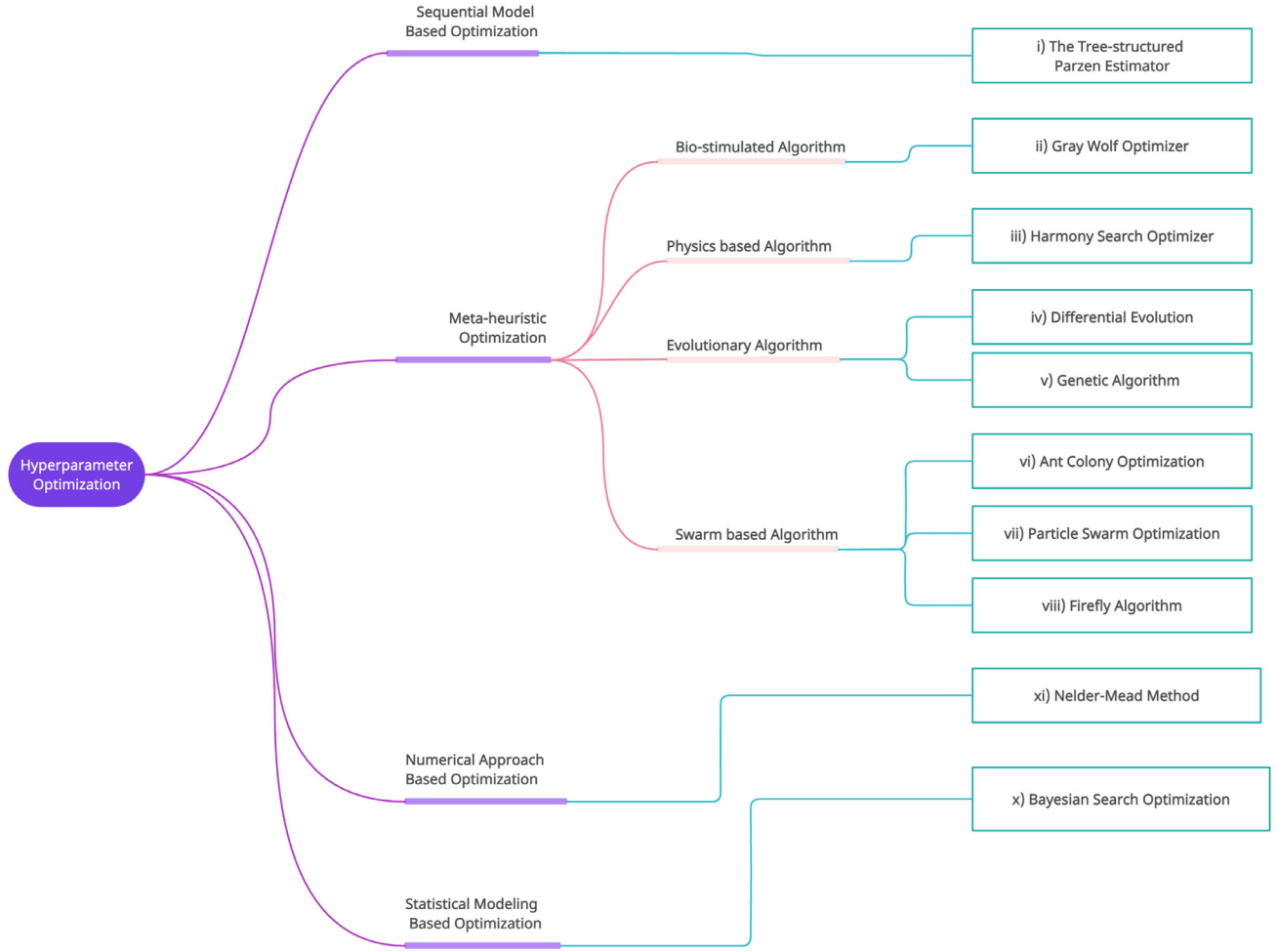


Fig. 12. Taxonomy of hyperparameter optimization algorithms.

on the Bayesian approach [151]. TPE provides simple solutions for the model and tunes the hyperparameters significantly. The first step of this approach is to construct an initial configuration of hyperparameter sets. The sets are then split into good and bad groups. The groups are updated, and the group's quality is estimated in the next step. EI denotes the improvement of splitting, and the process is ongoing until the maximum EI value is attained. The equation for TPE is followed below [152]:

$$P_s(a/b) = \begin{cases} l(b), & \text{if } a < a^* \\ g(b), & \text{if } a \geq a^* \end{cases} \quad (35)$$

$$EI(a) = \int_{-\infty}^{a_{\min}} \max(a_{\min} - a(b), 0) P_s(a/b) \quad (36)$$

In Eq. (48), $l(x)$ is the density estimate formed by the observations, and $g(x)$ is the density formed by using the remaining observations. The current minimum loss is a_{\min} , and b refers to the local optimal hyperparameter settings. Guangzhi et al. [45] introduce the TPE optimization technique in three neural network models for assessing landslide susceptibility. In their experiment's early stages, they collected landslide data and used multiple preprocessing techniques to prepare the data. Then, they use Deep Neural Network (DNN), Recurrent Neural Network (RNN), and CNN to train the model and evaluate the performance. In the next stage, they used TPE-optimized models (TPE-DNN, TPE-RNN, and TPE-CNN) to improve the accuracy. They observed the TPE optimization impact on these models. For these models, they have tuned the dropout, batch size, epoch number, and optimizer. To facilitate the comparison, they conduct multiple performance measurement techniques. The result reflects that the overall accuracy of those models is increased efficiently by using the TPE optimization algorithm.

David et al. [46] design deep learning (DL) applications to classify supply chain pricing datasets of health medications using TPE methods. They conducted the experiment by developing a novel model combining 1D CNN with Long Short-Term Memory (LSTM). To enhance the model performance, they utilized repeated k-fold cross-validation in their study. The TPE method was introduced to configure the model and determine specific parameters. After using the optimization techniques, the parameter increases substantially and contributes to improving the accuracy. The algorithm optimized the model's filter number, loss function, and dropout value.

In another study, Haipai et al. [44] applied the TPE method as an HPO technique to improve the accuracy of the SMOTE-XGBoost-based model for copper flotation method classification. XGBoost was chosen because it outperforms other decision tree ensemble models on this problem, and the synthetic minority oversampling (SMOTE) technique oversampled the minority class data in the training set. The TPE method precisely tunes the hyperparameters of XGBoost. They observed that the TPE method is more flexible as well as effective than the traditional approach for obtaining the highest accuracy.

6.2. Gray wolf optimization

The gray wolf optimization (GWO) algorithm is a bio-simulated metaheuristic algorithm. Mirjalili et al. [153] proposed GWO in 2014 by mimicking the gray wolves' social behavior, leadership hierarchy, and hunting in group property. As suggested by the name, the algorithm was influenced by gray wolves. To illustrate the GWO mathematical model, the best candidate is considered as α wolf, followed by β wolf,

δ wolf, and the other candidates by ω wolf. Gray wolves' behavior tells that they initially circle their prey. This encircling behavior is defined by the mathematical formulae below:

$$\bar{D} = |\bar{C} \cdot \bar{X}_p(t) - \bar{X}(t)| \quad (37)$$

$$\bar{X}(t+1) = \bar{X}_p(t) = \bar{A} \cdot \bar{D} \quad (38)$$

Where \bar{X}_p represents the position vector of the prey, \bar{X} is the gray wolf's position vector, t is the current iteration, \bar{A} and \bar{C} are coefficient vectors.

$$\bar{A} = 2\bar{a} \cdot \bar{r}_1 - \bar{a} \quad (39)$$

$$\bar{C} = 2\bar{r}_2 \quad (40)$$

Where r_1 and r_2 are arbitrary values ranging from 0 to 1, and \bar{a} is a decreasing parameter from 2 to 0. To imitate hunting, α , β , and δ are supposed to have the most in-depth knowledge of the prey. The following equations are used to update the values of these three categories of wolves first and subsequently the positions of the other search agents (ω wolves).

$$\bar{D}_\alpha = |\bar{C}_1 \cdot \bar{X}_\alpha - \bar{X}|, \bar{D}_\beta = |\bar{C}_2 \cdot \bar{X}_\beta - \bar{X}|, \bar{D}_\delta = |\bar{C}_3 \cdot \bar{X}_\delta - \bar{X}| \quad (41)$$

$$\bar{X}_1 = \bar{X}_\alpha - \bar{A}_1 \cdot (\bar{D}_\alpha), \bar{X}_2 = \bar{X}_\beta - \bar{A}_2 \cdot (\bar{D}_\beta), \bar{X}_3 = \bar{X}_\delta - \bar{A}_3 \cdot (\bar{D}_\delta) \quad (42)$$

$$\bar{X}_{(t+1)} = \frac{\bar{X}_1 + \bar{X}_2 + \bar{X}_3}{3} \quad (43)$$

Gray wolves attack their victim after it stops moving to complete the hunt. Since \bar{A} is a random variable within the range $[-a, a]$, the value of \bar{a} drops over the iteration. As a result, $|\bar{A}| < 1$ forces the wolves to assault the prey. Another element of GWO that encourages exploration is \bar{C} , which works to prevent local optimum and encourage exploration. It only comprises random values between $[0, 2]$ at the beginning of the process and at the end, which strengthens the exploration idea without bias. Lastly, The fulfillment of an end criterion finally brings an end to the GWO algorithm.

By following these processes, Shukla et al. [47] employed GWO to identify the ideal CNN tuning parameter values for transmission line protection. They simulated the power system model to collect real-time voltage and current signals at relaying buses for various fault scenarios. Following that, they defined the number of gray wolves, iterations, and maximum-maximum constraints corresponding to each CNN hyperparameter, after which the starting population corresponding to each design variable was generated randomly inside the search space. Then, they trained CNN using the setting for the individual population of hyperparameters. Finally, the terminating criteria cause the search process to end, and the alpha wolf position indicates the best CNN hyperparameter value for the particular fault classification task at hand.

Mohakud et al. [48] optimized hyperparameters using GWO in four phases to identify skin cancer. The steps are as follows: parameter encoding, population initialization, fitness evaluation, and construction of the next-generation population. Using the algorithm, they optimized several parameters, such as the kernel size and number of kernels associated with the convolution layer, the kernel size in the pooling layer, and the dropout rate. These parameters' values are set to arbitrary numbers within a predetermined range. They trained the model using a lightweight model with less randomness to save time, and they eliminated the unneeded iterations by determining if the change in accuracy was minor or not. Following this strategy, they achieved the optimal result.

6.3. Harmony search algorithm

The harmony search (HS) algorithm is a metaheuristic optimization technique. It is influenced by musical phenomena, primarily jazz improvisation [154]. The way a musician finds a better harmonic sound is imitated by the HS algorithm in order to find the best solution with less number of iterations. Similar to how a musician performs a note from memory or at random, a harmony search algorithm derives value from memory or at random [12]. The HS algorithm mainly consists of five operations, i.e., value initialization, harmony memory (HM) initialization, creation of new harmony, Update HM, and repetition operations [49].

- **Value initialization:** The HS algorithm for HPO initializes values first. Harmony memory size (HMS), harmony memory considering ratio (HMCR), and pitch adjusting ratio (PAR) must be initialized in the HS algorithm.
- **HM initialization:** To identify the optimal answer and update the process, the HS algorithm uses HM to save excellent harmony while excluding bad harmonies.
- **Creation of new harmony:** This process excludes the worst harmonies and adds the better ones. HMCR is utilized in this process to decide whether to make a new harmony or one by gently altering the already-existing harmonies.
- **Update HM:** HMCR and PAR's new harmonies are compared to HM's worst. New harmonies that outperform previous ones are added to the HM, while the worst ones are eliminated.
- **Repetition operations:** Creation of new harmony and update HM steps are repeated for the specified number of iterations and HPO is completed by selecting the best harmony of the final HM.

Lee et al. [12] proposed a method utilizing the Parameter-setting-free HS algorithm to optimize the hyperparameter by modifying the size of the feature map of the layer in the CNN's feature extraction step. Their approach starts by initializing the PAR, HMCR, and HMS. Following that, they create harmony by setting the hyperparameters of CNN. The initial HM is built by calculating the CNN loss for each harmony. To create an initial HM, this procedure is repeated as many times as HMS. They use a gradient descent algorithm to train the CNN and use a cross-entropy function as a loss function. They use the sigmoid function to modify the equations for HMCR and PAR. As harmonies, they specify the kernel size, stride, zero padding, total number of channels in the convolutional layer, and kernel size and stride in the pooling layer. To evaluate the performance of their proposed method, they use the MNIST and the CIFAR-10 dataset.

In another study, Kim et al. [49] proposed a method using an HS algorithm to optimize the hyperparameters to improve the respiration pattern recognition accuracy in 1D CNN. In their proposed method, they set the hyperparameters like the kernel size (KS), the kernel count of each convolution layer (KC), and the neuron count of each dense layer (DLNC) as target variables to optimize the hyperparameters of 1D CNN. In this method, when HM has initialized, they set a large number for HMS to reduce the execution time significantly to reach the global maxima in less iteration. HMCR, PAR, and MPAP are the hyperparameters that hugely influence the HS algorithm's performance in the proposed method. These are used to optimize the KS, KC, and DLNC hyperparameters. This approach can optimize the hyperparameters quickly compared to the existing methods.

In another study, Huang et al. [50] proposed a self-adaptive HS algorithm to optimize CNN architecture. In their method, they divide the system architecture into two phases. In phase 1, they look for the best CNN layer length and use a self-adaptive harmony search (SAHS) algorithm to generate CNN candidates. The SAHS algorithm's termination condition is specified at the maximum number of iterations. The system moves to phase 2 when it has gone through all possible iterations. Using the SAHS algorithm, they generate various hyperparameter values in phase 2. The algorithm creates an optimized

CNN after reaching the maximum number of iterations. To evaluate the performance of their proposed method, they used popular datasets like MNIST, CIFAR-10, and Caltech-101. HMCR, PAR, and *hms* are parameters of the SAHS algorithm that are tuned for all three datasets to get optimal results.

6.4. Differential evolution

Differential evolution (DE) is a population-based search meta-heuristic algorithm introduced by Storn and Prince in 1997 [155]. DE algorithm is straightforward to implement, flexible, and versatile [155,156]. Like the genetic algorithm, the DE algorithm also uses the initial generation of population, mutation, crossover, and selection operators. The basic difference is that the genetic algorithm uses crossover, whereas differential evolution uses mutation [157]. The DE algorithm's main steps are as follows [51]:

The DE algorithm comprises N_p real-coded vectors, mutation, crossover, and selection operators. The N_p real-coded vectors represent the candidate solutions. Each solution can be described as shown in the following equation, where each element of the solution falls within the interval $x_{i,1}^{(t)} \in [x_i^{(L)}, x_i^{(U)}]$, where $x_i^{(L)}$ denotes the lower and $x_i^{(U)}$ denotes the upper bound of the i th variable.

$$x_i^{(t)} = (x_{i,1}^{(t)}, \dots, x_{i,n}^{(t)}), \text{ for } i = 1, \dots, N_p \quad (44)$$

The next step is mutation, where for each target vector, a mutation vector is created. The mutation operation is represented by the equation below:

$$u_i^{(t)} = x_{r_1}^{(t)} + F \cdot (x_{r_2}^{(t)} - x_{r_3}^{(t)}), \text{ for } i = 1, \dots, N_p \quad (45)$$

Where F denotes the scaling factor that scales the rate of modification, and in the interval $(1 \dots N_p)$ r_1 , r_2 , and r_3 are randomly selected values.

The following step is crossover. It was introduced to improve the variety of parameter vectors. It is represented by the following equation:

$$w_{i,j}^{(t+1)} = \begin{cases} u_{i,j}^{(t)} & \text{rand}_j(0,1) \leq CR \vee j = j_{rand}, \\ x_{i,j}^{(t)} & \text{otherwise}, \end{cases} \quad (46)$$

Where the fraction of parameters copied to the trial solution is controlled by $CR \in [0.0, 1.0]$, which is the crossover probability rate.

The final stage is selection, which determines whether or not the produced vector should be included in the generation using the greedy criterion. The selection can be expressed as follows:

$$x_i^{(t+1)} = \begin{cases} w_i^{(t)} & \text{if } f(w_i^{(t)}) \leq f(x_i^{(t)}), \\ x_i^{(t)} & \text{otherwise}, \end{cases} \quad (47)$$

Where $f(x_i^{(t)})$ is the fitness function. The parameters N_p , F , and CR must be set correctly for the DE algorithm to operate well. Here, N_p represents the population size. F represents the differential weight that specifies the distance the offspring should be generated from point x_i [158]. CR means the crossover probability rate that regulates how many variables in expectation are modified in a population member [159].

Vili et al. [51] developed a method for classifying different sports categories from images using the transfer learning of CNN with HPO based on DE. They compared their proposed and conventional CNN models and determined the best result. DE algorithm optimized multiple hyperparameters of the model. Various parameters like batch size, epochs, dimension of the problem, population size, scale factor, crossover rate, number of function evaluations, learning rate, optimizer, and dropout are tuned to achieve the optimal result.

Mahdaddi et al. [52] introduced an evolutionary algorithm-based novel framework for effective drug-target interaction prediction. The proposed CNN-AbiLSTM framework combines a CNN with attention-based biLSTM, where the DE algorithm precisely tunes the model's

parameters for better performance. For the DE algorithm, they tuned the Population size, scale factor, crossover rate, number of generations, dropout value, LSTM dimension, kernel size to get the optimal result. The result of the proposed DE-based CNN-AbiLSTM model outperforms the baseline CNN model.

Another Study [53] initialized the DE algorithm in their model to determine the best network architecture to improve the accuracy. For the performance comparison assessment, they evaluate their results with multiple TL models. After following all the steps of the DE method, it configures the parameters and chooses the optimal value. The results of the experiments on different datasets prove the robustness and stability of their model.

In another recent study, Ghasemi et al. [54] developed a self-competitive DE algorithm to improve the exploration ability of the standard DE technique for optimization problems. The paper shows that by including a competitive control parameter and a new mutation method, DE algorithms can outperform their conventional counterparts when addressing real-world optimization issues and optimizing the tuning of a PID controller within an AVR system.

6.5. Genetic algorithm optimization

Genetic algorithm (GA) [160] is a computational model that simulates natural selection and the genetic mechanisms of Darwin's theory of biological evolution. A group of candidate solutions is kept in each iteration by simulating the natural processes of reproduction, crossover, and mutation [161]. A general framework for addressing complicated system optimization issues is provided by genetic algorithms that do not depend on the field and type of problems. The main components of the genetic algorithm can be described as the coding mechanism, the fitness function, the genetic operator (selection, crossover, and variation), and control parameters. The steps of the genetic algorithm can be stated as follows [161]:

1. Choose the appropriate coding approach for the given problem and create a population of N chromosomes with a predetermined length at random.

$$pop_i(t), t = 1, i = 1, 2, \dots, N \quad (48)$$

2. Calculate the fitness value of each chromosome $tpop(t)$ in the $pop(t)$ population.

$$f_i = fitness(pop_i(t)) \quad (49)$$

3. Evaluate if the convergence requirement is fulfilled. If it is satisfied with the output search results, proceed to the next set of steps.
4. Select operation is another significant part of genetic algorithms, and the selection probability is measured based on the fitness of each individual.

$$P_i = \frac{f_i}{\sum_{i=1}^N f_i}, i = 1, 2, 3, \dots, N \quad (50)$$

The above-mentioned distribution of probabilities is randomly selected from the current TPOP (t) population and forms a new population in the next generation of the population.

$$newpop(t+1) = \{pop_j(t) | (j = 1, 2, \dots, N)\} \quad (51)$$

5. After the selection in crossover operation, a group of N chromosomes $tcrosspop(t+1)$ was obtained by matching probability P_c .

6. And lastly, in mutation operation to drive chromosomal genes to mutate, use a smaller probability P_m . It develops a new population, $tmutpop(t+1)$. $Pop(t) = tmutpop(t+1)$ is the term given to the latest population that arises from a genetic operation. It also goes back to 2 and serves as the genetic operation's father at the same time.

Genetic algorithm is widely used for optimizing the hyperparameters of CNN in a very efficient manner. In a recent study [55], to find the optimal network architecture for a given input, both network

configurations, and hyperparameters as search space and use GA to search for the ideal structure have been chosen. As a hyperparameter in their study, layers of the model and the activation function are also considered for optimization. In order to develop an algorithm, a binary representation is utilized for encoding connections between convolutional layers. They encode the activation function and optimization algorithm into bits and separate them by bars. The number of bits needed for the n -hyperparameters can be calculated using the following:

$$\text{number of bits} = \lfloor \log_2(n) \rfloor + 1 \quad (52)$$

In the initialization phase, a population of CNN models is generated in encoded binary form. In this process, each bit of a chromosome is independently analyzed using a Bernoulli distribution. The optimal fitness score is set in the algorithm and repeated throughout the procedure. After selection, the remaining chromosomes produce the next generation of individuals through a uniform crossover process. Finally, the mutation applies to survivors of chromosomes. After constructing the optimal network architectures, an average classification accuracy of 81.74% was achieved.

Given the intricacy of the medical image, classification or segmentation of medical images by neural network is a tedious and time-consuming task. To address these shortcomings of neural networks, CNN plays a significant role in efficiently performing complicated image processing tasks. A study [56] investigated microRNA cancer classification using a convolution neural network and employed a genetic algorithm to optimize the hyperparameters of CNN. As an initial step in decoding the CNN parameters to the genetic algorithm for a CNN with three layers, wk, wdk, and hk output channels, convolution window, and max-pooling window of the k th layer, respectively. The first population is generated by selecting hyperparameters at random from a preset range. The population's fitness is determined by the average accuracy of CNN using these hyperparameters for a given epoch. After building an optimal architecture, their model got a small error rate of 0.14.

Addressing a problematic form of the Vehicle Routing Problem, Agrawal et al. [57] concentrate on maximizing the delivery of time-sensitive commodities subject to quality restrictions. To address the computational complexity encountered in practice, it presents a mathematical model as a Mixed Integer Non-Linear Programming problem and suggests a heuristic based on the Genetic Algorithm. The approach improves distribution efficiency for perishable goods through numerical scenarios and sensitivity analysis.

Using a novel mathematical model and a multiparent-based memetic genetic algorithm, Manna et al. [58] provide a ground-breaking solution to the 4D Clustered Traveling Salesman Problem (CTSP) in a paper. The research tackles security and risk issues by developing new heuristics for longitude and latitude-based cluster management and risk assessment. This novel strategy makes important contributions to the sector by efficiently addressing issues facing the tourism industry today.

6.6. Ant colony optimization

Ant colony optimization (ACO) uses ant search patterns to determine optimal routes for food gathering [15]. Ants use a short way from their nest to acquire food. They track the shortest route by leaving pheromones on the ground. Ants can detect and interpret pheromones, which dissipate and lose density. The high-density pheromone helps other ants discover food through density. Pheromone density rises when more ants deposit pheromones on shorter pathways. However, the other path's pheromone fades and disappears. The method focuses on ant m pheromone value updates. In every iteration, all the m ants who have built a solution can update the value of the pheromone as follows [15]:

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \sum_{k=1}^m \Delta \tau_{ij}^k \quad (53)$$

Here τ_{ij} indicates the value of pheromones, and i and j indicate the source and destination, respectively. ρ implies the evaporation rate, m indicates the number of ants, and $\Delta \tau_{ij}^k$ implies the number of pheromones on the path i, j by the ant K .

$$\Delta \tau_{ij}^k = \begin{cases} Q/L_k & \text{if ant } k \text{ used edge } i, j \\ 0 & \text{otherwise} \end{cases} \quad (54)$$

Here, Q is a constant, and L_k indicates the length of the touring path. A stochastic mechanism decides the movement of the ant. If the ant is currently in position i and the probability of movement to j is as follows:

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^k \cdot \eta_{ij}^\beta}{\sum_{c_{il} \in N(s^p)} \tau_{il}^\alpha \cdot \eta_{il}^\beta} & \text{if } c_{il} \in N(s^p), \\ 0 & \text{otherwise} \end{cases} \quad (55)$$

Here, $N(s^p)$ indicates the set of feasible components; that is, edge (i, l) , where l is still not visited by the ant K . α and β handles the importance of the pheromones and the heuristic information η_{ij} .

$$\eta_{ij} = \frac{1}{d_{ij}} \quad (56)$$

Here, d_{ij} indicates the distance between i and j .

In a study, Byla et al. [59] use ACO to find an efficient neural architecture. According to the algorithm, firstly, the ant is placed in a node and searches for possible transitions. After moving to the next node, it traverses all the attributes, updates the value of pheromones, and generates a new ant population. The parameter is used for the number of convolution layers, kernel size, filter count, dropout rate, strides, pooling size, and activation function. The local and global pheromones have been updated to ensure the exploration and exploitation. The algorithm's robustness and performance are evaluated using three different datasets.

In another study, ACO is used by Lankford et al. [60] to choose the right neural network and the number of hyperparameters. The method creates two configurations with various ant counts and epoch counts. Config. 1 is designed with 8 ants and 30 epochs, where config. 2 is defined with 16 ants and 15 epochs. The other tuned hyperparameters are kernel size, dropout rate, and maximum and minimum layers. From the top of the model, fully connected layers were initially removed to perform fine-tuning. Finally, they evaluated the model among two different datasets and found a promising result.

6.7. Particle swarm optimization

A particle Swarm is a method for the optimization of continuous nonlinear functions. Kennedy et al. [162] introduced this algorithm by inspiring social behavior. Particle swarm optimization (PSO) is a metaheuristic-based stochastic population evolutionary optimization technique. Each Particle in PSO represents an individual solution. First, it initializes the main parameters. Then, generate a random position value. Based on the termination criteria, the algorithm stopped and provided the best result. If not, the particle fitness values are evaluated, the local and global best results and positions are recorded, and the particle velocities and positions are updated. Two mathematical equations update the positions of each particle in global search after every iteration in the PSO technique. The equations are followed below [163]:

$$v_i^{t+1} = w * v_i^t + c1r1(xBest_i^t - x_i^t) + c2r2(gBest_i^t - x_i^t) \quad (57)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (58)$$

In Eq. (57), v_i^{t+1} is the velocity of each article. The term w refers the initial inertia, v_i^t is the previous velocity of the particle, $r1$ and $r2$ are the random numbers and $c1$ and $c2$ are the corresponding accelerations, $xBest_i^t$ denotes the best fitness value of local particle and $gBest_i^t$ is the global particle best fitness value. The previous positions of swarm i at

time t denoted by x_i^t . The updated positions of the swarm are calculated using Eq. (58).

In a study, Tatsuki et al. [61] approached a novel PSO method, namely Linearly Decreasing Weighted PSO (LDWPSO), to improve the accuracy of CNN-based models. He efficiently determines the parameters of the CNN model by performing HPO using LDWPSO. At the experiment's beginning, each particle's position and velocity are set. In each CNN execution, these parameters are updated and report the best results. Multiple parameters are optimized through this technique such as the number of convolutional layers, the neurons for fully connected layers, the kernel size, activation functions, learning rate, and batch size. Both swarm size and weight decrease linearly in this study. Thus, it determines the optimum parameters and significantly improves the model's performance.

In another study, Pratibha et al. [62] introduced a Hybrid Multi-level PSO technique to find the architectures and hyperparameters of the CNN simultaneously. They conduct the experiments on two levels: first, they optimize the architectures and then optimize the hyperparameters. In this method, multiple swarm is utilized in order to find the optimal configuration of the model. In the first step of the experiment, they initialized a swarm to a random value for convolutional, pooling, and fully connected layers. For each particle of the swarm at level-1, another corresponding swarm is initialized at level-2. Consequently, it results in multiple swarms at level-2 which denotes the number of filters, filter size, stride size, padding, and the number of output neurons. This procedure is repeated until the stopped condition is fulfilled. The fitness function is calculated and recorded as the best one. For every swarm of level-1, the procedure is followed by the swarm of level-2 also. Thus the algorithms provide the maximum accuracy for the CNN model.

Shaikh et al. [63] present an innovative approach in their study to approximating transmission line parameters that are vital for ensuring efficient power distribution and high-quality service. It addresses the shortcomings of prior systems by introducing a hybrid algorithm that combines moth-flame optimization (MFO) and PSO methodologies. The hybrid algorithm shows promise as a method for precise transmission line parameter estimation thanks to its higher performance in solution quality and convergence speed compared to conventional PSO and MFO approaches, as shown by several simulations and mathematical validations.

Inspired by Hummingbird Flight (HBF) patterns, Zare et al. [64] present a new PSO method to improve search quality and population variety. Extensive evaluations of benchmark suites and practical applications, such as economic dispatch and dynamic economic dispatch, show the higher performance of the proposed PSO-HBF algorithm. It demonstrates superior performance in optimization tasks than other modified PSO algorithms by outperforming them on various benchmarks and making considerable improvements in handling complicated real-world issues.

6.8. Firefly algorithm (FA)

The firefly algorithm (FA) was first developed by Yang in 2008 and has been used for numerous optimization problems [164] including CNN HPO [65]. The algorithm is based on three rules and is primarily inspired by the flashing behavior of fireflies in nature [164]:

- As fireflies are unisex, so one firefly can be attracted by any other firefly.
- The attraction is dependent on the brightness of the fireflies. Attraction increases if the distance between the fireflies decreases and vice versa.
- The fitness value is determined by the brightness of a firefly.

In the very beginning of the algorithm, fireflies (solutions) are generated randomly with the lower and upper bound by the following equation [66]:

$$x_{i,j} = lb_j + rand(ub_j - lb_j), \quad (59)$$

here $x_{i,j}$ describes the i th solution and the respective j th element; the lower bound is symbolized by lb_j and the upper bound is symbolized as ub_j ; $rand$ is a random number.

The firefly updates its location through the following formula:

$$x_i^{t+1} = x_i^t + \beta_0 r^{-\gamma r_{i,j}} (x_j - x_i) + \alpha(rand - 0.5), \quad (60)$$

here the new location is symbolized by x_i^{t+1} , current location as x_i^t . β_0 denotes the attractiveness at the distance 0. The brighter solution is x_j and the current solution moves to the brighter one. Alpha is the randomization parameter. The distance between two fireflies x_i and x_j is the Euclidean distance:

$$r_{i,j} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^d (x_{i,k} - x_{j,k})^2}, \quad (61)$$

where $r_{i,j}$ denotes the distance between the solutions x_i and x_j .

Once, distance is calculated and we need to calculate the new attractiveness value as follows:

$$\beta(r) = \frac{\beta_0}{1 + \gamma r^2}, \quad (62)$$

where $\beta(r)$ is the new attractiveness value.

The light intensity is calculated as follows:

$$I(r) = \frac{I_0}{1 + \gamma r^2}, \quad (63)$$

where $I(r)$ is the light intensity at distance r and γ denotes the light absorption coefficient.

Bacanin et al. [66] use the firefly algorithm for CNN HPO with a modified technique to balance exploration and exploitation. There is a randomized parameter that decides whether the position of the firefly will be updated by the standard firefly search equation or the new position will be updated with their newly defined equation. With the purpose of HPO, a feasible search space is defined within a boundary. They consider the number of convolutional layers, size of the kernel, pooling layer, activation function, dropout rate, number of fully connected layers, number of hidden layers, type of optimizer, and the learning rate. In the meantime, the maximum iteration number and other control parameters are defined and all the values of the parameters are kept within a range. In every iteration of the algorithm, a set of parameters is passed, and gradually find the best set of hyperparameters.

Nebojsa et al. [65] use an enhanced FA algorithm for HPO. The following CNN hyperparameters are optimized in the study: the number of convolutional layers, the number of kernel, kernel size, the number of fully connected layers, and hidden units in the fully connected layer. The value for randomization parameter α , and light absorption coefficient γ are kept at 0.5, 1.0 respectively, and the other parameters that are not to the optimization subject are kept fixed, i.e., they use ReLU as an activation function, Adam optimizer with the learning rate 0.0001, and pooling layer of fixed size 2×2 . In one epoch the model is trained with a batch size of 54. Thus, they get the perfect set of hyperparameters.

In another study, Aswanandini et al. [67] use the FA for hyperparameter tuning to find such parameters with a lower classification error rate. For this purpose, they define two heuristics called low-level and high-level strategy where low-level makes the rule to select the solutions (fireflies), and high-level picks the best solution from the set of solutions. The hyperparameters that are subject to optimization are the number of convolutional layers, the number of pooling layers, the number of fully connected layers with hidden units, and the kernel size. For proper exploitation, the value of β has been kept to 0.02 and γ between 0.01 and 100. In this study, some parameters like learning rate, dropout rate, etc. have not been optimized as they mostly have real values. Finally, they test their model on the NSL-KDD dataset and pick the configuration with the lowest classification error rate.

In order to improve global exploration and convergence when optimizing complex engineering problems, Ghasemi et al. [68] present

FA 1 to 3. FA 1 to 3 outperforms the standard FA and other state-of-the-art FA algorithms on the CEC2014 benchmark functions. The study also applies FA levels 1 to 3 to practical engineering challenges, demonstrating the method's viability and efficacy. Based on simulation results, FA 1 to 3 is a robust and straightforward method for addressing a wide variety of complicated engineering problems, outperforming both older and more modern algorithms.

6.9. Nelder–Mead method

Nelder–Mead (NM) method is an optimization method that is simplex-based and was proposed by John Nelder and Roger Mead [165]. The NM method is also known as a simplex method [166]. It is a direct search method to solve an optimization problem based on the values of the objective function without using derivatives [69,167]. It belongs to a simplex group of algorithms. The simplex concept is a geometric shape with $n + 1$ vertices that can be used to optimize n hyperparameters [167]. Initially, the initial simplex is generated randomly by the NM method with $[Z_1, Z_2, \dots, Z_{n+1}]$ simplex vertices, where each vertex represents one CNN architecture. Based on the value of the objective functions $f(Z_1) \leq f(Z_2) \leq \dots \leq f(Z_{n+1})$, the vertices are ordered ascendingly, where the best vertex is Z_1 , which produces the best CNN architecture, and the worst vertex is Z_{n+1} , which produces the worst CNN architecture. When constructing a CNN architecture, the NM method looks for the optimum hyperparameters λ^* to minimize the objective function as follows:

$$\min_{\lambda^*} f(\lambda^*)$$

where $f(\lambda^*)$ is the objective function to be minimized [69].

$$\lambda^* = \arg \min f(\lambda), \text{ where } \lambda \in \psi \quad (64)$$

Based on the application of four fundamental operations—reflection, expansion, contraction, and shrinkage—the search in the space solution is performed. These operations are each accompanied by a scalar coefficient of α (reflection), β (expansion), γ (contraction), and δ (shrinkage). To create a new simplex in every iteration, a current simplex is updated in the NM method, which decreases the value of the objective function. In order to generate a new simplex in each iteration, the NM method attempts to update an existing simplex, which decreases the value of the objective function. The method replaces the worst vertex with the best one found by reflecting, expanding, or contracting vertices. Otherwise, except the best one, all other vertices of simplex will shrink around the best vertex. This process will be repeated until the best vertex is discovered and the stop criterion is met. The vertex that produces the lowest objective function is the best vertex [69].

Albelwi et al. [69] use the Nelder–Mead Method to optimize the hyperparameters of CNN. By minimizing the proposed objective function, they apply the NM method to automatically explore a huge search space for a high-performing CNN architecture. They use the NM method as it is quicker than other derivative-free optimization algorithms because a small number of vertices are evaluated in each iteration in the NM method. To ensure that the output values are positive integers, some constraints are imposed during the calculation of each vertex. The value of the correlation results is normalized in each iteration of the NM method. They provided a new objective function that makes use of error rate information. They use the CIFAR-10, CIFAR-100, and MNIST datasets to assess the performance of the framework. They use SGD optimizer to train the CNN. To achieve the optimal result they tuned various hyperparameters like epoch number, learning rate, batch, weight decay, and dropout. Using Xavier initialization technique [168] weights of all layers are initialized. For NM settings to define the total number of hyperparameters n , they initialize 80 random CNN architectures. The value of the correlation coefficient parameter = 0.20 is set for all the datasets.

In another study, Albelwi et al. [169] use the NM method to optimize the hyperparameters of CNN architecture. In their proposed

method they propose a new objective function. The NM method maximizes the objective function by modifying the hyperparameters of CNN architecture. During the optimization process, the objective function combines accuracy and visualization to determine which architecture is superior to the other. They use the CIFAR-10, and Caltech-100 datasets to assess the performance of the framework. They used Theano [170] to train CNNs. For the initial simplex vertices of the NM method different hyperparameters such as the number of fully connected layers, depth, number of feature maps, kernel sizes, number of pooling layers, pooling windows sizes, and number of units in fully connected layers are set in a specific range. For NMA settings to define the total number of hyperparameters n , they initialize 100 random CNN architectures. The value of $\beta = 0.25$, and the total number of iterations for the NM method is 25. They use stochastic gradient descent and also tune batch size, weight decay, and dropout rate to get the optimal result.

6.10. Bayesian optimization

Finding the ideal hyperparameter combination is one of the hardest parts of DL systems like CNNs. Hyperparameter combinations can reduce or improve fitness function. Finally, it becomes challenging and computationally expensive to determine the global optimal solution without using the right set of hyperparameters. Grid search, random search, and Bayesian optimization are the three methods that are utilized to overcome this issue.

The grid search [171] method has the drawback that, in order to choose the optimum set of algorithm parameters, it must optimize every conceivable parameter setting. This process takes time, especially when the search space and hyperparameters are both substantial.

Using a random selection of hyperparameter values, the random search [172] technique assesses the model's precision. The evaluation is iterative, and once all iterations have been completed, an assessment metric selects the best hyperparameter. There is no assurance in this method that the subsequent assessment will be superior to the previous setting. The disregard for past data in optimization methods based on grid and random search is a key issue.

Bayesian optimization [173] builds a probabilistic model called "Surrogate" by applying a systematic approach based on Bayes' theorem that maps hyperparameters to the probability of performing well on the objective function. The approach focuses on problem solving [174]:

$$\max_{x \in A} f(x) \quad (65)$$

where, $x \in \mathbb{R}^d$ for bounding values for d and Gaussian Process regression is required to model f . Bayesian optimization follows these steps:

1. Create a surrogate probability model for the objective function.
2. Determine the best hyperparameters for the surrogate.
3. Change these hyper-parameters to the objective function.
4. Update the surrogate model with new data.
5. Once the maximum number of iterations has been reached, repeat steps 2–4 once more.

Loey et al. [70] found hyperparameters like LR, SGD with momentum, Depth of the network stage, and L2 regularization via Bayesian Optimization to generate optimal architecture.

Conversely, Sameen et al. [71] found several hyperparameters, e.g., no. of filters Sequence length, Batch size, Activation function, Optimization method, Neurons in the hidden layers, the Dropout rate to train their CNN model for landslide susceptibility assessment.

Using Gaussian process regression models and various kernels and basis functions, Xu et al. [72] estimate retail property price indices in ten major Chinese cities between 2005 and 2021. There are ten reliable forecasting models created using Bayesian optimization and cross-validation, with relative root mean square errors ranging from

0.0113% to 0.4835%. These models are helpful for analyzing and forecasting retail property prices in China's ever-changing real estate market.

Using artificial intelligence (AI) and ML methods, Lahmiri et al. [73] look into accurate forecasting systems for valuing Taiwanese homes. The study compares various methods, such as artificial neural networks and multivariate regression, and concludes that boosting ensemble regression trees, support vector regression, and Gaussian process regression with Bayesian optimization get the best results. These results demonstrate the promise of enhancing ensemble regression trees for precise, practical house price prediction, which would be of excellent service to real estate market stakeholders like investors, creditors, and governments.

After reviewing all the HPO algorithms, we offered a clear visualization of all the algorithms in Table 9.

Table 10 presents an exhaustive synopsis of different HPO algorithms, elaborating on their methodology and technical intricacies. Scholars and professionals specializing in machine learning and optimization may derive significant knowledge from these algorithms, empowering them to discern the most appropriate HPO approach for their particular undertakings. The fundamental operations of each algorithm, including its exploration of the hyperparameter space, evaluation of solution quality, and optimization of hyperparameter configurations, are delineated in the algorithm description. The table is a valuable reference guide by providing a succinct overview of these HPO techniques. This aids researchers in making well-informed decisions regarding selecting the most effective algorithm to improve the performance of machine learning models.

7. Domain specific application of HPO

CNN is widely used across several domains for its remarkable performance. However, HPO techniques are also adopted in these domains to attain optimal performance. Table 11 provides a concise summary of the utilization of CNN HPO techniques for different domains, including healthcare, financial forecasting, social media, security, industry and education. It is observed that the efficient optimization of these hyperparameters is substantially required to enhance model accuracy and robustness. We briefly highlight the application of CNN HPO techniques in popular research domains.

Overall, it is underlined that HPO techniques play a significant role in upgrading CNN applications, therefore increasing their applicability for real-world cases. The combination of CNN and HPO algorithms surpasses the state of art and shows promise for widespread application in practical circumstances as well.

8. Discussion

This study contributes to the current discussion on finding optimal CNN model parameters through the utilization of HPO methods. The fundamental goals of this research were to evaluate the performance of these different types of optimization algorithms, understand their internal mechanisms, and emphasize their performance on benchmark datasets. We observed that many algorithms displayed distinct traits regarding their convergence speed, their exploration of search space, and their effectiveness in optimization. To facilitate a thorough understanding of the effectiveness of these algorithms in the realm of CNN-based models, we have incorporated two tables (Table 12 and Table 13) that present a detailed overview of the outcomes derived from existing literatures.

Table 12 presents a concise overview of the performance of different HPO algorithms on different benchmark datasets, such as Modified National Institute of Standards and Technology (MNIST) and Canadian Institute for Advanced Research (Cifar-10), with 60,000 images each, Fashion-MNIST, CIFAR-100, and the Caltech-101 dataset, respectively. Accuracy (A) and error rate (E), and Mean Squared Error (MSE) are

utilized as performance indicators throughout the table. Almost every algorithm is tested on the MNIST dataset which is a relatively simpler dataset where the focus is to classify the handwritten digits from images. The Harmony Search (HS) Algorithm obtains the highest accuracy of 99.25% on the MNIST dataset compared to other algorithms. HS is particularly effective in this case due to its effectiveness in finding the optimal parameters for the CNN-based model that executes well in simpler datasets. Fashion-MNIST, a challenging variant of MNIST dataset and each of the algorithms were evaluated in this dataset. Table 12 reports that Harmony Search Algorithm which was previously mentioned that performed well in MNIST dataset, also demonstrated strong performance in this dataset also by attaining an accuracy of 97.96%. Additionally, Particle swarm optimization also performed well in this dataset by effectively minimizing the classification error rate in comparison to the error rates of other algorithms. Cifar-10 had extensive use of all algorithms and it is a common dataset for validation. The firefly method demonstrates superior performance, achieving an accuracy of 96.7% substantially. Moreover, HS and PSO also show decent performance obtaining an accuracy of 95.03% and 89.5%, respectively. The firefly algorithm's exploratory capabilities and adaptability make it ideal for classifying 10 different classes. Cifar-100 is the extended version of Cifar-10, encompassing a collection of 100 distinct categories of images. Due to the complexity of this dataset only a few algorithms were tested and among them Ant Colony Optimization (ACO) achieves the best results of 89.79% accuracy. The algorithm's ability in dealing with a complex classification task is the reason for its satisfactory performance. Regarding the Caltech-101 dataset, we can observe from the Table 12 that the HS method achieves the highest accuracy of 92.88% and an MSE of 0.1695. This findings suggests that the approach is effective for this specific dataset.

Furthermore, this study also investigates different types of datasets from different domains to test the effectiveness of these algorithms. Table 13 provided a complete comparison assessment of algorithm performance for a particular dataset. For medical image datasets, Bayesian Optimization, Gray Wolf Differential Evolution, Genetic Algorithm and Firefly Algorithm shows commendable performance compared to other approaches. The flexibility and efficiency of these algorithms in exploring hyperparameter combinations make them suitable for optimizing models' parameters on medical images. From the Table 13, it is also seen that the Tree-structured Parzen Estimator technique shows excellent performance when applied to datasets relating to industrial products and processes. The adaptability and search capabilities of this algorithm can be advantageous for these datasets in complicated industrial scenarios. Image classification datasets such as MNIST, CIFAR-10, and CIFAR-100 exhibit impressive results when using several algorithms, including Harmony Search Algorithm, Ant Colony Optimization, Particle Swarm Optimization, and Nelder-Mead Method.

Through this study, we can gain sufficient knowledge in the selection of algorithms for the suitable datasets. The findings obtained from this extensive examination of HPO methods across many datasets offer valuable direction for future research efforts in model optimization. Furthermore, researchers can utilize this knowledge to adapt their optimization strategy, fine-tuning hyperparameters to get boosted outcomes across many domains.

In certain domain such as healthcare, industry, security etc., real world applications are developed to make the process more effective and reliable and the performance of these applications are crucial to determine the acceptability of the whole system. Nowadays, CNN architectures are extensively employed in several real-world applications, including image classification problems, object detection, medical imaging, facial recognition, sentiment analysis, etc. However, these investigations' performance might have dropped due to the inappropriate hyperparameters' size or shape. In this regard, HPO contributes to reliable performance by tuning the optimal parameters according to the domain and architecture. As a result, the acceptance of these automated practical applications increased substantially. Some of the

Table 9

Overview of CNN hyperparameter optimization algorithms.

Algorithm	Starting	Steps	End
Tree-structured Parzen Estimator	Define a tentative configuration of the hyperparameters in random manner.	(i) Formulate an initial configuration of hyperparameter sets randomly. (ii) Split the sets into good and bad groups. (iii) Checks the estimated group quality and updates the parameters set. (iv) Calculate the expected improvement.	Terminate the process until the EI achieves the maximum value.
Harmony Search Algorithm	The method starts by initialize algorithm parameters: HMS, initial HMCR, initial PAR.	(i) Set the hyperparameter (kernel size, stride, padding, number of channels in the convolutional layer, and kernel size and stride in the pooling layer) as harmony. (ii) Create a harmony vector and calculate the loss function. (iii) Initialize the HM and configure the initial HM. (iv) Calculate new HMCR and PAR. (v) Create a new solution. (vi) Compare the new solution with the worst one in the HM. (vii) Replace the worst solution with the new one if the new one is better.	When met the set termination condition/ iteration is completed, return the best solution in HM and determine of hyperparameters.
Differential Evolution	Set the population weight matrices to zero.	(i) Initialize the population weight vectors. (ii) Generate a mutant vector. (iii) Perform crossover. (iv) Selection operation. (v) Calculate fitness function.	Repeat the steps until the best fitness functions have been generated.
Genetic Algorithm	Mapping the solution into the algorithm's search space.	(i) Generate CNN model as population. (ii) Calculate the accuracy in the CNN model. (iii) Calculate fitness function (f) in the model. (iv) If acceptable f is found, then end the process; Otherwise: Select Rank-based s (chromosome). (v)Crossover: Uniform c (selected). (vi) Apply crossover & mutation.	Repeat the procedure till the best fitness functions are formed.
Ant Colony Optimization	Generate ants and initialize pheromone trails and other parameters.	(i) The ant is assigned to a starting spot and searches for possible transitions in the search space of CNN hyperparameters. (ii) It then selects the next destination in the CNN structure's search space using the Ant Colony Search rule. (iii) It then selects hyperparameter attributes such as kernel size, activation function, etc. (iv) The pheromone is updated by the end of the ant journey. (v) The maximum depth of exploration has been increased, and a new ant population has been created.	According to the pheromone value returns the best ant.
Particle Swarm Optimization	Establish a number of Particles as a population.	(i) Create a population of particles. (ii) Calculate the best and global position. (iii) Evaluate each particle position and update the best position. (iv) Find the best particle. (v) Update the velocities and move to new positions. (vi) Update the velocities and move to new positions.	Repeat the step until stopping criteria are satisfied
Firefly Algorithm	Initially, randomly set the population of solutions(CNN structure) between a range as a firefly.	(i) Set related FA parameters. (ii) Calculate the fitness value of each solution. (iii) Find the current best solution. (iv) Complete the evaluation task of N population of solutions. (v) Sort the solutions based on fitness. (vi) Determine the current best solution. (vii) Perform exploitation. (vii) Perform exploration. (ix) Finish if it reaches maximum iteration.	Return the best solution.
Nelder–Mead Method	The method starts by generating an initial simplex (total number of hyperparameters is required to construct the initial simplex).	(i) Set the value of the correlation coefficient parameter. (ii) Calculate the new objective function for all vertices. (iii) Sort the vertices in ascending order. (iv) Calculate the centroid of vertices without considering the worst vertex. (v) Calculate reflected, expanded, and contracted vertices. (vi) Replace the worst vertices with the best one if any best vertex is found. (vii) Except for the best one, shrink all the vertices (shrink toward the best vertex direction).	The optimal solution returned at the end of this process is the vertex with the lowest objective function value.
Bayesian Optimization	Randomly select set of hyperparameters before starting the optimization.	(i) Create model surrogate functions. (ii) Determine which hyperparameters the surrogate responds to the best. (iii) Update the true objective function with the best of these hyperparameters. (iv) Update surrogate functions for new hyperparameters. (v) Repeat steps (ii) to (iv) until the maximum number of repetitions are done.	The best collection of hyperparameters is the one that corresponds to the the least value across all observation points.

(continued on next page)

Table 9 (continued).

Algorithm	Starting	Steps	End
Gray Wolf Optimization	GWO starts with randomly initialized number of gray wolves, iterations, and upper and lower bounds for each of the hyperparameters.	(i) Using the specific set of hyperparameters setup, CNN is trained on the provided dataset. (ii) Calculate the effectiveness of each population member's fitness level. (iii) Alpha, beta, and delta (best 3 solutions) are obtained based on the fitness value. (iv) Then update the positions of all wolves and repeat from CNN training steps onwards till the maximum number of iterations is reached.	Finally, the position of alpha wolves are found, which represents the optimum hyperparameter setting.

empirical evidence of HPO in CNN is highlighted in this section for a precise understanding of the contribution of HPO.

Image classification is a common phenomenon of CNN-based model utilization. Different HPO techniques were successfully applied to enhance the performance of the model. Hong et al. [197] introduced a novel opposite-based particle swarm optimization technique in the CNN model for effective defect classification from semiconductor photomasks. This proposed metaheuristic technique improved the selection of optimal hyperparameters for the model and attained an accuracy of 100%. Singh et al. [198] integrated a fast-forward optimization algorithm with CNN for digital image classification. This algorithm searches for the optimal weights and minimizes the classification error. Additionally, HPO provides significant advancement in the field of medical imaging. Rajesh et al. [199] designed an automated medical image denoising technique integrating DE and CNN. The evolutionary process of DE helps to generate the parameters for the model and improve the optimization rate. Kumar et al. [200] proposed a nature-inspired ResNet 152 model for efficient brain tumor detection. The proposed algorithm outperformed the baseline model, attaining an accuracy of 99.57%. Furthermore, many HPO approaches enhance the efficiency of the detection and recognition model. Gutierrez et al. [201] enhanced the Mask-RCNN architecture by incorporating an improved weighted regularization technique with the colliding bodies optimization (CBO) algorithm to improve ship detection accuracy. Raziani et al. [182] utilized metaheuristic methods to choose the hyperparameter of deep CNN. The utilization of this model aids in reducing the number of layers and epochs, hence minimizing computational costs and enhancing the performance of human activity recognition.

9. Open issues and challenges

The review of existing literature highlights the continuous nature of research in HPO. Future directions and challenges in HPO have been identified within the literature, particularly concerning DL, which undergoes frequent changes with new models and techniques emerging. Developing new models and application-based techniques is imperative, requiring efficient optimization mechanisms to fine-tune their performance. This section briefly discusses the open issues and challenges of HPO for CNN using various domain algorithms.

- **Dataset Imbalance:** Dataset imbalance poses a significant challenge in HPO as model performance is heavily influenced by dataset characteristics such as size, noise, class distribution, and quality. Imbalanced datasets, where majority and minority class numbers are disproportionate, are prevalent across different domains, leading to suboptimal model performance [202]. Traditional HPO strategies often do not address this issue, and overfitting further reduces model accuracy, necessitating specialized attention from HPO algorithms.
- **Dataset Size:** Dataset size greatly impacts DL model performance, especially in small datasets, where DL models struggle due to their complex architectures generating numerous parameters [203]. Augmentation techniques, such as generative adversarial networks (GAN), geometric augmentation, elastic deformation, and photometric augmentation, offer solutions by

- artificially increasing the dataset size [204–207]. However, HPO alone cannot address this issue as it focuses solely on optimizing parameter values without expanding the dataset size, highlighting the necessity of dataset augmentation alongside HPO.
- **Selection of Suitable HPO Algorithm:** Choosing the appropriate HPO algorithm is crucial, given the plethora of available options, including Meta-heuristic optimization, Sequential Model-Based optimization, and Statistical Modeling-Based optimization [151, 154,161,162,173]. Different CNN models and datasets require tailored HPO techniques, making determining the most suitable approach challenging. While automated HPO methods often yield optimal results, their efficacy varies depending on the model and dataset characteristics, emphasizing the need for careful selection.
 - **Time and Space Complexity:** HPO experiments are typically iterative, with algorithms like genetic algorithms and particle swarm optimization requiring repeated iterations until termination criteria are met [51]. Consequently, model execution times are prolonged, and architectural designs become more intricate, increasing time and space complexity. Balancing computational resources with optimization efficacy poses a significant challenge in HPO.

The challenges outlined in this section underscore the significant impact of HPO algorithms on overall model performance. As HPO evolves, addressing emerging issues and optimizing efficiency will be effective in future research endeavors.

10. Critical analysis of future research agendas

This section emphasizes investigating and critically analyzing potential avenues for future research. In contrast to the previous section's examination of research challenges in HPO algorithms, this section significantly underscores the importance of developing a critical perspective on future research directions. This is accomplished by analyzing existing knowledge and exploring opportunities for improvement. Thus, we delve into understanding existing comprehension regarding the methodological, conceptual, and developmental aspects of HPO algorithms while also exploring new domains. Table 14 presents a comprehensive analysis of future research directions based on existing knowledge.

RQ1: How do considerations of interpretability and explainability influence the selection and application of HPO algorithms in complex artificial intelligence models? This question focuses on understanding how interpretability and explainability requirements impact the decision-making process regarding the selection and implementation of hyperparameter optimization algorithms in complex machine learning models [208, 209]. It aims to explain the influence of these considerations on the decision-making process concerning HPO methodologies.

RQ2: How does establishing a threshold for the overfitting ratio before initializing HPO algorithms enhance model accuracy?

This question explores the significance of defining an overfitting threshold to address model accuracy issues [210]. It seeks to understand how proactively managing overfitting can improve performance during training and testing. Research addressing this question could guide the selection of appropriate thresholds based on dataset complexity and model architecture [211]. Furthermore, investigating adaptive

Table 10
Technical details of HPO algorithm.

HPO algorithm	Technical details
Tree-structured Parzen Estimator	<ol style="list-style-type: none"> 1. Employ probabilistic models to systematically navigate the search space with efficiency. 2. TPE utilizes Parzen estimation to create two distinct probability density functions for hyperparameters, one for good and one for bad observations. 3. To analyze hyperparameter space, TPE employs a binary tree in which nodes indicate hyperparameters and edges reflect a comparison between observed values and a threshold. 4. TPE employs an acquisition function to decide which node to divide next.
Harmony Search Algorithm	<ol style="list-style-type: none"> 1. The objective function quantifies the effectiveness of ML models based on the specific combination of hyperparameters and analyzes new harmonies. 2. Every harmony corresponds to a potential solution for the optimization problem. 3. Pitch adjustment optimizes the hyperparameters by adjusting them within a specific range to efficiently analyze the search space. 4. The memory consideration method is employed to maintain a balance between exploration and exploitation.
Differential Evolution	<ol style="list-style-type: none"> 1. The mutation operation is employed to create new candidate solutions. 2. The crossover operation is utilized to merge the mutant solution with the initial solution. 3. A crossover probability is utilized to generate the trial solution.
Genetic Algorithm	<ol style="list-style-type: none"> 1. Objective function is employed to assess the suitability of every possible configuration. 2. The fitness function measures the effectiveness of the ML models. 3. Selection procedures such as roulette wheel selection, tournament selection, or rank-based selection can be employed. 4. Crossover operations such as one-point crossover, two-point crossover, or uniform crossover are conducted to generate offspring. 5. Maintain a stable population using methods such as generational replacement and steady-state replacement.
Ant Colony Optimization	<ol style="list-style-type: none"> 1. Assess the standard of solutions by utilizing the objective function. 2. Each ant employs a probabilistic strategy to select hyperparameters by considering pheromone levels and heuristic information. 3. Based on how well the ants built the solutions, the algorithm updates pheromone trails. 4. A local search is utilized to enhance solution quality.
Particle Swarm Optimization	<ol style="list-style-type: none"> 1. An objective function evaluates how well the ML model works with its hyperparameters. 2. Every particle retains the record of its most optimal position discovered thus far. 3. The objective function values determine the optimal particle within the total population.
Firefly Algorithm	<ol style="list-style-type: none"> 1. Fireflies are drawn to one another by the intensity of their brightness. 2. The brightness of every firefly determines its attractiveness, which is calculated using the objective function. 3. A distance metric is established to quantify the proximity between fireflies. 4. The attractiveness of the fireflies is contingent upon their brightness and proximity to other fireflies. 5. The Euclidean distance is employed to quantify the spatial separation between fireflies within the solution space.
Nelder–Mead Method	<ol style="list-style-type: none"> 1. Reflections and expansions are used to enlarge and broaden the search space. 2. Contractions and shrinkages are used to narrow in on the best solutions. 3. The objective function directs the simplex transformations that NMM utilizes to progress toward the best possible hyperparameter settings. 4. The objective function determines the placements of the points in the simplex.
Bayesian Optimization	<ol style="list-style-type: none"> 1. It uses a probabilistic surrogate model which estimates the unknown objective function. 2. The search is directed by an acquisition function quantifying the potential of different hyperparameter configurations. 3. Exploitation is concentrated on areas where the objective function is expected to be at its best according to the surrogate model. 4. The surrogate model is continuously updated by picking new hyperparameter settings based on the acquisition function.
Gray Wolf Optimization	<ol style="list-style-type: none"> 1. The wolves' hierarchy directly impacts their exploration and exploitation capabilities in the optimization process. 2. The positions of wolves are assessed by an objective function. 3. Assess the performance of wolf's hyperparameter setup by using the objective function. 4. The hierarchy guides the wolves' movement inside the search space.

Table 11

Domain-specific application of HPO algorithms.

Domain name	Author	Algorithm type	Parameter optimization	Findings
Healthcare	Amou et al. [175]	Bayesian Optimization	Activation function, batch size, dropout, dense nodes and optimizer.	Experimenting with brain MRI images to classify brain tumors. The optimization technique helps the model outperform existing models and improve the feasibility of the model as well.
	Murugan et al. [176]	Gray wolf optimizer	Learning rate, epoch, momentum, regularization coefficient.	Optimize the architecture of CNN to effectively classify Covid-19 and pneumonia from X-ray images. Achieved an optimal performance with less computational cost.
	Lee et al. [55]	Genetic Algorithm	Activation function, batch size, epoch, learning rate, and optimizer.	The implementation of genetic algorithms helps to build a robust CNN model that effectively diagnoses Alzheimer's disease.
Financial	Chung et al. [177]	Genetic Algorithm	Filter size, epoch, activation function, optimizer.	Developed a multi-channel convolutional network and optimized the parameters for accurate stock price prediction.
	Xie et al. [178]	Gray wolf optimizer	Conv layer size, pooling layer size, lstm units, dense units and dropout.	Optimize the parameters of a CNN-LSTM architecture for financial forecasting on time series benchmark data.
	Ehsan et al. [179]	Differential evolution.	Batch size, hidden unit, dropout, epoch.	Developed a LSTM based architecture and optimize the hyperparameters using evolutionary algorithms for stock price prediction.
Social media	Kumari et al. [180]	Particle Swarm Optimization	Activation function, epoch, dropout, optimizer.	Developed a CNN-based optimized architecture for aggression prediction from social media posts.
	Cabada et al. [181]	Genetic algorithm	Filter size, batch size, epoch, optimizer.	Implemented a CNN architecture for emotion recognition from facial expressions. Improve the performance of the model using hyperparameter optimization.
	Raziani et al. [182]	Particle Swarm Optimization, Gray wolf optimization.	Batch size, epoch, num of filters, kernel size and pooling size.	Implemented several meta-heuristic algorithms to determine the optimal parameters of deep CNN for human activity recognition.
Security Applications	Li et al. [183]	Bayesian optimization	Number of layers, number of filters, kernel size, activation function, number of FC layers, dropout, optimizer	Modify the architecture of 1D CNN using optimization techniques and effectively perform the anomaly detection.
	Mitra et al. [184]	Particle swarm optimization	Number of kernel, kernel size, learning rate, epoch	Developed a computationally efficient 1D CNN model where the hyperparameters are optimized using PSO. The model outperforms other algorithms in classifying transmission line fault detection.
	Kilchev et al. [34]	Genetic algorithm, Particle swarm optimization.	Number of layers, Number of filters, Filter size, Activation function, optimizers	Extensively experimented with two hyperparameters optimization algorithms to set the optimal parameter and attain the best performance for network intrusion detection.
Industry	Kolar et al. [185]	Bayesian optimization	Number of kernels, Kernel size, Activation function, momentum, learning rate	Implemented an effective technique for rotary machine fault diagnosis. Optimized the architecture of CNN using bayesian optimization and maximized the performance.
	Liu et al. [186]	Genetic algorithm, Firefly algorithm.	LSTM unit, FC neurons, Dropout, batch size, learning rate	Modified the hyperparameters of CNN-LSTM architectures and attained the highest classification accuracy.
	Darwish et al. [187]	Particle swarm optimization.	Batch size, learning rate, dropout.	Optimized the parameters of every model from a CNN based ensemble model. The final outcome highlights the efficiency of the approaches by obtaining a satisfactory performance.

Table 12
Evaluation of the HPO algorithms on benchmark datasets.

Datasets	MNIST	Fashion-MNIST	CIFAR-10	CIFAR-100	Caltech-101 dataset
Harmony Search Algorithm	A: 99.25% [12] A: 99.37% [50]	A: 97.96% [188]	A: 74.76% [12] A: 95.03% [50]		A: 92.88% [50] MSE: 0.1695 [189]
Ant Colony Optimization	E: 0.46% [59]	A: 93.4% [60] E: 6.75% [59]	A: 84.8% [60] E: 12.70% [60]	A: 89.79% [190]	
Particle swarm optimization	A: 98.95% [61] A: 99.89% [62]	E: 5.53% [191] A: 92.67% [193]	A: 69.37% [61] A: 89.5% [61]	A: 71.55% [62]	A: 79.8% [192] MSE: 0.1691 [189]
Firefly Algorithm	A: 99.16% [65]	E: 5.52% [194]	A: 96.7% [195]	A: 77.75% [195]	MSE: 0.1589 [189]
Nelder–Mead Method	E: 0.42% [69]		E: 10.05% [61] A: 83.63% [169]	E: 35.46% [69]	A: 55.7% [169]

thresholding methods could enhance model accuracy by dynamically adjusting thresholds during optimization.

RQ3: *What are the implications of the time-consuming nature of HPO algorithms on model optimization, and how can existing frameworks expedite this process?*

This question examines the challenges posed by the iterative nature of HPO algorithms in terms of time efficiency [212]. By exploring these implications and evaluating existing frameworks like Optuna [213], Hyperband [214], and Scikit-Optimize [215], researchers can propose strategies to expedite the optimization process without compromising performance. Future work may focus on developing parallelization techniques or hybrid algorithms to utilize computational resources more effectively.

RQ4: *How do different optimization techniques compare in terms of efficiency and effectiveness for HPO tasks?*

This question aims to compare the performance of various optimization techniques, such as Bayesian and evolutionary algorithms, in hyperparameter optimization. Comparative studies on optimization techniques provide valuable insights into their strengths and weaknesses. Addressing this question enables researchers to identify the most suitable algorithms for specific optimization tasks and model architectures. Future research could explore hybrid approaches that combine multiple techniques to leverage their complementary strengths for enhanced performance.

RQ5: *What role does feature selection play alongside HPO algorithms, and how does it affect model generalization and performance?*

This question investigates the synergy between feature selection and hyperparameter optimization algorithms [216]. It explores how strategically selecting features alongside HPO methodologies influences model generalization, performance, and the overall predictive capabilities of machine learning models. By integrating feature selection with HPO methodologies, researchers can optimize model architecture and input features simultaneously. Future research may explore automated feature selection methods tailored to HPO frameworks, facilitating more effective optimization in high-dimensional datasets.

RQ6: *Can transfer learning be integrated with HPO methodologies to improve the efficiency of hyperparameter tuning across various models and datasets?*

This question explores the potential benefits of combining transfer learning techniques with hyperparameter optimization strategies. This combination has the potential to accelerate the optimization process and enhance model performance across diverse datasets. Addressing this question can lead to developing transfer learning-aware HPO frameworks, enabling efficient knowledge transfer between related tasks and domains. Future research may explore transfer learning strategies synergizing with HPO methodologies to achieve optimal model adaptation.

RQ7: *What are the trade-offs between computational resources, such as GPU utilization, and the performance gains achieved through advanced HPO algorithms?*

This question investigates the trade-offs between computational resources, particularly GPU utilization, and the performance improvements derived from advanced hyperparameter optimization techniques [217]. Addressing this question can inform researchers and practitioners about the scalability and efficiency of advanced HPO techniques. Future research may focus on developing resource-aware optimization strategies that dynamically adapt to available computational resources.

RQ8: *How can ensemble methods be combined with HPO strategies to enhance model robustness and predictive accuracy across diverse datasets?* This question explores the potential synergies between ensemble methods and hyperparameter optimization strategies to improve model robustness and predictive accuracy across various datasets [218]. It aims to investigate how combining ensemble learning with HPO can enhance model performance and reliability in machine learning applications.

11. Conclusion

This review provides a concise and comprehensive overview of optimization approaches used for optimizing hyperparameters of the CNN model and the proper architectural guidelines. From the deep analysis of the existing literature, which is presented in this study, researchers can find the four different types of fields, and the ten algorithms of the corresponding fields give optimal solutions based on the different hyperparameters and different datasets. We observe that no algorithm is perfect for automated HPO, but it gives a great solution to create the optimal network architecture. Metaheuristic algorithms are the most common algorithms for the given task and provide great accuracy in the maximum problem, although statistical, sequential, and numerical algorithms also compete perfectly, considering their optimal accuracy. This study also highlighted open hyperparameter challenges, which may aid in providing valuable insights into the application of currently proposed algorithms to manage the various unraveled elements of CNN hyperparameter adjustment in other application areas.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Table 13
Performance of different HPO algorithms in different datasets.

Optimization algorithm	Paper	Dataset	Accuracy/ error
Tree-structured Parzen Estimator	[45]	Supply chain shipping pricing dataset	Acc: 63.33%
	[44]	Copper ore flotation dataset	Preferential flotation=0.849 Partial flotation=0.831 Mixed flotation=0.922
Harmony Search Algorithm	[12]	MNIST dataset	Acc: MNIST=99.25%,
		CIFAR-10 dataset	CIFAR-10= 74.76%
	[49]	UWB respiration signal dataset	Acc: 96.7%
	[50]	MNIST, CIFAR-10, Caltech-101 dataset	Acc: MNIST=99.37%, CIFAR-10= 95.03%, Caltech-101=92.88%
Differential Evolution	[51]	Sports image dataset	Acc: 81.31%
	[52]	Drug-target interaction dataset	MSE:
		1. Kiba	Kiba Dataset=0.179,
		2. Davis	Davis Dataset=0.235
	[53]	Brain cancer-Bc, Lc-lung cancer, Cc-colon cancer, Maternal-fetal ultrasound-FP, Brain planes- FB	Accuracy: Lc=99.05%, Cc=99.50%, FP=96.29%, Bc=90.04%, FB=78.73%
Genetic Algorithm	[55]	18F-Florbetaben AmyloidPET/CT image dataset	Acc: 81.74%
	[56]	Cancer Genome Atlas	Error: 0.14
Ant Colony Optimization	[60]	CIFAR10	Acc: CIFAR10 = 84.8%
	[59]	Fashion Mnist	Fashion Mnist = 93.4%
		MNIST, Fashion-MNIST, CIFAR-10	Error rate in MNIST-0.46% Fashion-MNIST - 6.75% CIFAR-10 -12.70%
Particle swarm optimization	[61]	MNIST, CIFAR-10	Acc: MNIST =98.95, CIFAR-10 =69.37
	[62]	MNIST, CIFAR-10, CIFAR-100, Convexset, MDRBI	Acc: MNIST= 99.89, CIFAR-10=89.56, CIFAR-100=71.55, Convexset=92.73, MDRBI=68.1
Firefly Algorithm	[66]	IXI dataset, REMBRANDT dataset, TCGA-GBM dataset, TCGA-LGG dataset	Acc: Dataset 1 = 93.3%, Dataset 2 = 96.5%
	[65]	MNIST benchmark dataset, MNIST dataset	Acc: 99.16%
	[67]	NSL-KDD	Acc: NSL-KDD = 96.66%
		ISCX-IDS	ISCX-IDS = 93.33%
Nelder–Mead Method	[69]	CIFAR-10, CIFAR-100, and MNIST dataset	Error rate: MNIST= 0.42%, CIFAR-10= 10.05%, CIFAR-100=35.46%
	[169]	CIFAR-10, Caltech-101	Acc: CIFAR-10=83.63%, Caltech-101=55.7%
Bayesian Optimization	[70]	Radiography dataset, Chest X-ray Images	Acc: 96.00%
	[71]	Landslide Inventories Factor	Acc: 83.11%
Gray Wolf Optimization	[196]	International Skin Imaging Collaboration (ISIC)	Acc: 98.33%
	[47]	Custom Dataset	Acc: 99.47%

Table 14

Future research agenda.

Research Question (RQ)	Issues/topics/research gaps	Research paths
RQ1	Interpretability and explainability	<ul style="list-style-type: none"> → Develop methods to incorporate interpretability measures directly into the hyperparameter optimization (HPO) process → Design algorithms that prioritize transparent models or provide explanations alongside model predictions → Explore novel techniques for explainable hyperparameter tuning, such as surrogate models or interpretable search spaces
RQ2	Overfitting threshold	<ul style="list-style-type: none"> → Determine optimal overfitting thresholds based on dataset and model characteristics → Explore adaptive thresholding methods → Investigate the impact of threshold selection on model performance and generalization
RQ3	Time-consuming nature of HPO	<ul style="list-style-type: none"> → Implement parallelization techniques to distribute HPO computations across multiple processors or machines → Develop algorithms that dynamically adjust optimization strategies based on available computational resources → Investigate the use of surrogate models or meta-learning approaches to accelerate the HPO process while maintaining performance
RQ4	Efficiency of optimization	<ul style="list-style-type: none"> → Benchmark different optimization techniques across various machine learning tasks and datasets → Develop hybrid optimization approaches that leverage the strengths of multiple techniques to improve overall efficiency and effectiveness → Investigate transferability of optimized hyperparameters across different model architectures and datasets to generalize optimization performance
RQ5	Feature selection with HPO	<ul style="list-style-type: none"> → Integrate feature selection techniques directly into the HPO process to jointly optimize model architecture and feature subsets → Develop automated feature selection algorithms that consider both predictive performance and computational efficiency within HPO frameworks → Investigate the interplay between feature selection methods and HPO strategies to maximize model generalization and predictive capabilities
RQ6	Transfer learning with HPO	<ul style="list-style-type: none"> → Develop transfer learning-aware HPO frameworks that leverage pre-trained models and transfer learning techniques to accelerate the optimization process → Investigate the use of transfer learning for initializing hyperparameters or guiding the search space exploration in HPO → Explore techniques for transferring knowledge between related tasks or domains to improve optimization efficiency and final model performance
RQ7	Computational resource trade-offs	<ul style="list-style-type: none"> → Investigate the resource requirements of different HPO algorithms and techniques to understand their trade-offs in terms of computational resources → Develop resource-aware optimization strategies that dynamically adjust optimization settings based on available resources → Explore techniques for efficient knowledge transfer between tasks or domains to optimize the use of computational resources in HPO

(continued on next page)

Table 14 (continued).

Research Question (RQ)	Issues/topics/research gaps	Research paths
RQ8	Ensemble methods with HPO	<p>→ Investigate the integration of ensemble learning techniques directly into the HPO process to optimize ensemble model architectures and hyperparameters simultaneously</p> <p>→ Develop adaptive HPO methodologies that dynamically adjust optimization strategies based on the ensemble's performance and computational resources</p> <p>→ Explore techniques for efficient ensemble model evaluation and selection within the HPO framework to optimize overall predictive performance</p>

References

- [1] Saad Albawi, Tareq Abed Mohammed, Saad Al-Zawi, Understanding of a convolutional neural network, in: 2017 International Conference on Engineering and Technology, ICET, Ieee, 2017, pp. 1–6.
- [2] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, Jun Zhou, A survey of convolutional neural networks: Analysis, applications, and prospects, IEEE Trans. Neural Netw. Learn. Syst. (2021).
- [3] Andrew Mann, Surya R. Kalidindi, Development of a robust cnn model for capturing microstructure–property linkages and building property closures supporting material design, Front. Mater. 9 (2022) 851085.
- [4] Anis Shazia, Tan Zi Xuan, Joon Huang Chuah, Juliana Usman, Pengjiang Qian, Khin Wee Lai, A comparative study of multiple neural network for detection of Covid-19 on chest x-ray, EURASIP J. Adv. Signal Process. 2021 (1) (2021) 1–16.
- [5] Monika Bansal, Munish Kumar, Monika Sachdeva, Ajay Mittal, Transfer learning for image classification using vgg19: Caltech-101 image data set, J. Ambient Intell. Humaniz. Comput. (2021) 1–12.
- [6] Aayush Jaiswal, Neha Gianchandani, Dilbag Singh, Vijay Kumar, Manjit Kaur, Classification of the Covid-19 infected patients using densenet201 based deep transfer learning, J. Biomol. Struct. Dyn. 39 (15) (2021) 5682–5689.
- [7] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, Chunfang Liu, A survey on deep transfer learning, in: International Conference on Artificial Neural Networks, Springer, 2018, pp. 270–279.
- [8] Divya Arora, Mehak Garg, Megha Gupta, Diving deep in deep convolutional neural network, in: 2020 2nd International Conference on Advances in Computing, Communication Control and Networking, ICACCCN, IEEE, 2020, pp. 749–751.
- [9] Ashis Pradhan, Support vector machine-a survey, Int. J. Emerg. Technol. Adv. Eng. 2 (8) (2012) 82–85.
- [10] Gonzalo I. Diaz, Achille Fokoue-Nkoutche, Giacomo Nannicini, Horst Samulowitz, An effective algorithm for hyperparameter optimization of neural networks, IBM J. Res. Dev. 61 (4/5) (2017) 1–9.
- [11] Özkan İnık, Cnn hyper-parameter optimization for environmental sound classification, Appl. Acoust. 202 (2023) 109168.
- [12] Woo-Young Lee, Seung-Min Park, Kwee-Bo Sim, Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm, Optik 172 (2018) 359–367.
- [13] Seyedali Mirjalili, Genetic algorithm, in: Evolutionary Algorithms and Neural Networks, Springer, 2019, pp. 43–55.
- [14] Dongshu Wang, Dapei Tan, Lei Liu, Particle swarm optimization algorithm: An overview, Soft Comput. 22 (2) (2018) 387–408.
- [15] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization-artificial ants as a computational intelligence technique, IEEE Comput. Intell. Mag. (2006).
- [16] Martin Pelikan, David E. Goldberg, Erick Cantú-Paz, et al., Boa: The Bayesian optimization algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, vol. 1, Citeseer, 1999, pp. 525–532.
- [17] Rabiya Khalid, Nadeem Javid, A survey on hyperparameters optimization algorithms of forecasting models in smart grid, Sustainable Cities Soc. 61 (2020) 102275.
- [18] Fei Han, Jing Jiang, Qing-Hua Ling, Ben-Yue Su, A survey on metaheuristic optimization for random single-hidden layer feedforward neural network, Neurocomputing 335 (2019) 261–273.
- [19] Sajjad Nematzadeh, Farzad Kiani, Mahsa Torkamanian-Afshar, Nizamettin Aydin, Tuning hyperparameters of machine learning algorithms and deep neural networks using metaheuristics: A bioinformatics study on biomedical and biological cases, Comput. Biol. Chem. 97 (2022) 107619.
- [20] Ashraf Darwish, Aboul Ella Hassanien, Swagatam Das, A survey of swarm and evolutionary computing approaches for deep learning, Artif. Intell. Rev. 53 (2020) 1767–1812.
- [21] İlhan Firat Kilincer, Fatih Ertam, Abdulkadir Sengur, Ru-San Tan, U. Rajendra Acharya, Automated detection of cybersecurity attacks in healthcare systems with recursive feature elimination and multilayer perceptron optimization, Biocybern. Biomed. Eng. 43 (1) (2023) 30–41.
- [22] Daud Muhajir, Muhammad Akbar, Affindi Bagaskara, Retno Vinarti, Improving classification algorithm on education dataset using hyperparameter tuning, Procedia Comput. Sci. 197 (2022) 538–544.
- [23] Stephanie Holly, Thomas Hiessl, Safoura Rezapour Lakani, Daniel Schall, Clemens Heitzinger, Jana Kemnitz, Evaluation of hyperparameter-optimization approaches in an industrial federated learning system, in: IDSC2021, Springer, 2022, pp. 6–13.
- [24] Kazi Ekramul Hoque, Hamoud Aljamaan, Impact of hyperparameter tuning on machine learning models in stock price forecasting, IEEE Access 9 (2021) 163815–163830.
- [25] Ugur Erkan, Abdurrahim Toktas, Deniz Ustun, Hyperparameter optimization of deep cnn classifier for plant species identification using artificial bee colony algorithm, J. Ambient Intell. Humaniz. Comput. 14 (7) (2023) 8827–8838.
- [26] Xiangdong Leng, Eghbal Amidi, K.M. Shihab Uddin, William C. Chapman Jr., Hongbo Luo, Sitai Kou, Steve Hunt, Matthew Mutch, Quing Zhu, Assessing rectal cancer treatment response using photoacoustic microscopy: Deep learning cnn outperforms supervised machine learning model, in: Photons Plus Ultrasound: Imaging and Sensing 2021, vol. 11642, SPIE, 2021, p. 116420S.
- [27] Rifai Chai, Dadang Gunawan, et al., Optimizing cnn hyperparameters for blastocyst quality assessment in small datasets, IEEE Access 10 (2022) 88621–88631.
- [28] Andri Pranolo, Yingchi Mao, Aji Prasetya Wibawa, Agung Bella Putra Utama, Felix Andika Dwiyanto, Optimized three deep learning models based-pso hyperparameters for beijing pm2. 5 prediction, 2023, arXiv preprint arXiv: 2306.07296.
- [29] Xianping Du, Hongyi Xu, Feng Zhu, Understanding the effect of hyperparameter optimization on machine learning models for structure design problems, Comput. Aided Des. 135 (2021) 103013.
- [30] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, Frank Hutter, Smac3: A versatile bayesian optimization package for hyperparameter optimization, J. Mach. Learn. Res. 23 (54) (2022) 1–9.
- [31] Daniel Mesafint Belete, Manjaiah D. Huchaiah, Grid search in hyperparameter optimization of machine learning models for prediction of hiv/aids test results, Int. J. Comput. Appl. 44 (9) (2022) 875–886.
- [32] Soner Kiziloluk, Eser Sert, Covid-ccd-net: Covid-19 and colon cancer diagnosis system with optimized cnn hyperparameters using gradient-based optimizer, Med. Biol. Eng. Comput. 60 (6) (2022) 1595–1612.
- [33] Caroline Barcelos Gonçalves, Jefferson R. Souza, Henrique Fernandes, Cnn architecture optimization using bio-inspired algorithms for breast cancer detection in infrared images, Comput. Biol. Med. 142 (2022) 105205.
- [34] Dushmanth Kilichev, Woosong Kim, Hyperparameter optimization for 1d-cnn-based network intrusion detection using ga and pso, Mathematics 11 (17) (2023) 3724.
- [35] Alejandro Morales-Hernández, Inneke Van Nieuwenhuysse, Sebastian Rojas Gonzalez, A survey on multi-objective hyperparameter optimization algorithms for machine learning, Artif. Intell. Rev. 56 (8) (2023) 8043–8093.
- [36] Mohamed Abd Elaziz, Abdelghani Dahou, Laith Abualigah, Liyang Yu, Mohammad Alshinwan, Ahmad M. Khasawneh, Songfeng Lu, Advanced metaheuristic optimization techniques in applications of deep neural networks: A review, Neural Comput. Appl. (2021) 1–21.
- [37] A. Saranya, R. Subhashini, A systematic review of explainable artificial intelligence models and applications: Recent developments and future trends, Decis. Anal. J. (2023) 100230.
- [38] Arun Kumar Dey, Govind P. Gupta, Satya Prakash Sahu, A metaheuristic-based ensemble feature selection framework for cyber threat detection in iot-enabled networks, Decis. Anal. J. 7 (2023) 100206.

- [39] Carlos Francisco Moreno-Garcia, Chrisina Jayne, Eyad Elyan, Magaly Aceves-Martins, A novel application of machine learning and zero-shot classification methods for automated abstract screening in systematic reviews, *Decis. Anal. J.* (2023) 100162.
- [40] Koppiahraj Karuppiah, Bathrinath Sankaranarayanan, Syed Mithun Ali, A systematic review of sustainable business models: Opportunities, challenges, and future research directions, *Decis. Anal. J.* 8 (100272) (2023) 100272.
- [41] Matthew J. Page, Joanne E. McKenzie, Patrick M. Bossuyt, Isabelle Boutron, Tammy C. Hoffmann, Cynthia D. Mulrow, Larissa Shamseer, Jennifer M. Tetzlaff, David Moher, Updating guidance for reporting systematic reviews: Development of the prisma 2020 statement, *J. Clin. Epidemiol.* 134 (2021) 103–112.
- [42] Tianji Huang, Huayong Wu, Shengdong Yang, Bao Su, Ke Tang, Zhengxue Quan, Weiyang Zhong, Xiaojie Luo, Global trends of researches on sacral fracture surgery: A bibliometric study based on vosviewer, *Spine* 45 (12) (2020) E721–E728.
- [43] Yuetian Yu, Yujie Li, Zhongheng Zhang, Zhichun Gu, Han Zhong, Qiongfang Zha, Luyu Yang, Cheng Zhu, Erzhen Chen, A bibliometric analysis using vosviewer of publications on Covid-19, *Ann. Transl. Med.* 8 (13) (2020).
- [44] Haipai Dong, Dakuo He, Fuli Wang, Smote-xgboost using tree Parzen estimator optimization for copper flotation method classification, *Powder Technol.* 375 (2020) 174–181.
- [45] Guangzhi Rong, Kaiwei Li, Yulin Su, Zhijun Tong, Xingpeng Liu, Jiquan Zhang, Yichen Zhang, Tiantao Li, Comparison of tree-structured Parzen estimator optimization in three typical neural network models for landslide susceptibility assessment, *Remote Sens.* 13 (22) (2021) 4694.
- [46] David Opeoluwa Oyewola, Emmanuel Gbenga Dada, Temidayo Oluwatosin Omotehinwa, Onyeka Emebo, Olugbenga Oluseun Oluwagbemi, Application of deep learning techniques and bayesian optimization with tree Parzen estimator in the classification of supply chain pricing datasets of health medications, *Appl. Sci.* 12 (19) (2022) 10166.
- [47] Sunil K. Shukla, Ebha Koley, Subhojit Ghosh, Grey wolf optimization-tuned convolutional neural network for transmission line protection with immunity against symmetrical and asymmetrical power swing, *Neural Comput. Appl.* 32 (22) (2020) 17059–17076.
- [48] Rasmiranjan Mohakud, Rajashree Dash, Designing a grey wolf optimization based hyper-parameter optimized convolutional neural network classifier for skin cancer detection, *J. King Saud Univ.-Comput. Inform. Sci.* 34 (8) (2022) 6280–6291.
- [49] Seong-Hoon Kim, Zong Woo Geem, Gi-Tae Han, Hyperparameter optimization method based on harmony search algorithm to improve performance of 1d cnn human respiration pattern recognition system, *Sensors* 20 (13) (2020) 3697.
- [50] Yin-Fu Huang, Jung-Sheng Liu, Optimizing convolutional neural network architecture using a self-adaptive harmony search algorithm, in: *the International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, Springer, 2019, pp. 3–12.
- [51] Vili Podgorelec, Špela Pečnik, Grega Vrbančič, Classification of similar sports images using convolutional neural network with hyper-parameter optimization, *Appl. Sci.* 10 (23) (2020) 8494.
- [52] Abba Mahdaddi, Souham Meshoul, Meriem Belguidoum, Ea-based hyperparameter optimization of hybrid deep learning models for effective drug-target interactions prediction, *Expert Syst. Appl.* 185 (2021) 115525.
- [53] Smaranda Belciug, Learning deep neural networks' architectures using differential evolution. case study: Medical imaging processing, *Comput. Biol. Med.* (2022) 105623.
- [54] Mojtaba Ghasemi, Abolfazl Rahimnejad, Milad Gil, Ebrahim Akbari, S. Andrew Gadsden, A self-competitive mutation strategy for differential evolution algorithms with applications to proportional-integral-derivative controllers and automatic voltage regulator systems, *Decis. Anal. J.* 7 (2023) 100205.
- [55] Sanghyeop Lee, Junyeob Kim, Hyeon Kang, Do-Young Kang, Jangsik Park, Genetic algorithm based deep learning neural network structure and hyperparameter optimization, *Appl. Sci.* 11 (2) (2021) 744.
- [56] Alejandro Lopez-Rincon, Alberto Tonda, Mohamed Elati, Olivier Schwander, Benjamin Piwowarski, Patrick Gallinari, Evolutionary optimization of convolutional neural networks for cancer Mirna biomarkers classification, *Appl. Soft Comput.* 65 (2018) 91–100.
- [57] Anil Kumar Agrawal, Sushel Yadav, Amit Ambar Gupta, Suchit Pandey, A genetic algorithm model for optimizing vehicle routing problems with perishable products under time-window and quality requirements, *Decis. Anal. J.* 5 (2022) 100139.
- [58] Apurba Manna, Arindam Roy, Samir Maity, Sukumar Mondal, Izabela Ewa Nielsen, A multi-parent genetic algorithm for solving longitude-latitude-based 4d Traveling Salesman problems under uncertainty, *Decis. Anal. J.* 8 (2023) 100287.
- [59] Edvinas Byla, Wei Pang, Deepswarm: Optimising convolutional neural networks using swarm intelligence, in: *UK Workshop on Computational Intelligence*, 2019, pp. 119–130.
- [60] Séamus Lankford, Diarmuid Grimes, Neural architecture search using particle swarm and ant colony optimization, in: *AICS*, 2020, pp. 229–240.
- [61] Tatsuki Serizawa, Hamido Fujita, Optimization of convolutional neural network using the linearly decreasing weight particle swarm optimization, 2020, arXiv preprint arXiv:2001.05670.
- [62] Pratibha Singh, Santanu Chaudhury, Bijaya Ketan Panigrahi, Hybrid mpso-cnn: Multi-level particle swarm optimized hyperparameters of convolutional neural network, *Swarm Evol. Comput.* 63 (2021) 100863.
- [63] Muhammad Suhail Shaikh, Saurav Raj, Rohit Babu, Shubash Kumar, Kapil Sagrolkar, A hybrid Moth-Flame algorithm with particle swarm optimization with application in power transmission and distribution, *Decis. Anal. J.* 6 (2023) 100182.
- [64] Mohsen Zare, Mohammad-Amin Akbari, Rasoul Azizpanah-Abarghoee, Mostafa Malekpour, Seyedali Mirjalili, Laith Abualigah, A modified particle swarm optimization algorithm with enhanced search quality and population using hummingbird flight patterns, *Decis. Anal. J.* 7 (2023) 100251.
- [65] Nebojsa Bacanin, Timea Bezdan, Eva Tuba, Ivana Strumberger, Milan Tuba, Optimizing convolutional neural network hyperparameters by enhanced swarm intelligence metaheuristics, *Algorithms* 13 (3) (2020) 67.
- [66] Nebojsa Bacanin, Timea Bezdan, K. Venkatachalam, Fadi Al-Turjman, Optimized convolutional neural network by firefly algorithm for magnetic resonance image classification of glioma brain tumor grade, *J. Real-Time Image Process.* 18 (4) (2021) 1085–1098.
- [67] R. Aswanandini, C. Deepa, Hyper-heuristic firefly algorithm based convolutional neural networks for big data cyber security, *Indian J. Sci. Technol.* 14 (38) (2021) 2934–2945.
- [68] Mojtaba Ghasemi, Soleiman kadhoda Mohammadi, Mohsen Zare, Seyedali Mirjalili, Milad Gil, Rasul Hemmati, A new firefly algorithm with improved global exploration and convergence with application to engineering optimization, *Decis. Anal. J.* 5 (2022) 100125.
- [69] Saleh Albelwi, Ausif Mahmood, A framework for designing the architectures of deep convolutional neural networks, *Entropy* 19 (6) (2017) 242.
- [70] Mohamed Loey, Shaker El-Sappagh, Seyedali Mirjalili, Bayesian-based optimized deep learning model to detect Covid-19 patients using chest x-ray image data, *Comput. Biol. Med.* 142 (2022) 105213.
- [71] Maher Ibrahim Sameen, Biswajeet Pradhan, Saro Lee, Application of convolutional neural networks featuring Bayesian optimization for landslide susceptibility assessment, *Catena* 186 (2020) 104249.
- [72] Xiaojie Xu, Yun Zhang, A Gaussian process regression machine learning model for forecasting retail property prices with Bayesian optimizations and cross-validation, *Decis. Anal. J.* (2023) 100267.
- [73] Salim Lahmiri, Stelios Bekiros, Christos Avdoulas, A comparative assessment of machine learning methods for predicting housing prices using Bayesian optimization, *Decis. Anal. J.* 6 (2023) 100166.
- [74] Ameer Hamza, Muhammad Attique Khan, Shui-Hua Wang, Abdullah Alqahtani, Shtwai Alsubai, Adel Binbusayis, Hany S. Hussein, Thomas Markus Martinetz, Hammam Alshazly, Covid-19 classification using chest x-ray images: A framework of cnn-lstm and improved max value moth flame optimization, *Front. Public Health* 10 (2022) 948205.
- [75] Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep Learning*, MIT Press, 2016.
- [76] Jake Bouvrie, Notes on convolutional neural networks, 2006.
- [77] Vincent Dumoulin, Francesco Visin, A guide to convolution arithmetic for deep learning, 2016, arXiv preprint arXiv:1603.07285.
- [78] Nagisa Masuda, Ikuko Eguchi Yairi, Multi-input cnn-lstm deep learning model for fear level classification based on eeg and peripheral physiological signals, *Front. Psychol.* 14 (2023) 1141801.
- [79] Asifullah Khan, Anabia Sohail, Ummeh Zahoor, Aqsa Saeed Qureshi, A survey of the recent architectures of deep convolutional neural networks, *Artif. Intell. Rev.* 53 (8) (2020) 5455–5516.
- [80] Guan Wang, Jun Gong, Facial expression recognition based on improved lenet-5 cnn, in: *2019 Chinese Control and Decision Conference, CCDC, IEEE*, 2019, pp. 5655–5660.
- [81] Hongfeng You, Long Yu, Shengwei Tian, Xiang Ma, Yan Xing, Ning Xin, Weiwei Cai, Mc-net: Multiple max-pooling integration module and cross multi-scale deconvolution network, *Knowl.-Based Syst.* 231 (2021) 107456.
- [82] Chenyang Lu, Gijs Dubbelman, Semantic foreground inpainting from weak supervision, *IEEE Robot. Autom. Lett.* 5 (2) (2020) 1334–1341.
- [83] André de Souza Brito, Marcelo Bernardes Vieira, Mauren Louise Sguario Coelho De Andrade, Raul Queiroz Feitosa, Gilson Antonio Giraldo, Combining max-pooling and wavelet pooling strategies for semantic image segmentation, *Expert Syst. Appl.* 183 (2021) 115403.
- [84] Shuihua Wang, Yongyan Jiang, Xiaoxia Hou, Hong Cheng, Sidan Du, Cerebral micro-bleed detection based on the convolution neural network with rank based average pooling, *IEEE Access* 5 (2017) 16576–16583.
- [85] Samuel Kumaresan, K.S. Jai Aultrín, S.S. Kumar, M. Dev Anand, Transfer learning with cnn for classification of weld defect, *Ieee Access* 9 (2021) 95097–95108.
- [86] Yaxin Li, Kesheng Wang, Modified convolutional neural network with global average pooling for intelligent fault diagnosis of industrial gearbox, *Eksploatacja i Niezawodność* 22 (1) (2020) 63–72.
- [87] Yongliang Zhang, Shengyi Pan, Xiaosi Zhan, Zhiwei Li, Minghua Gao, Chenhao Gao, Fldnet: Light dense cnn for fingerprint liveness detection, *IEEE Access* 8 (2020) 84141–84152.

- [88] Zenglin Shi, Yangdong Ye, Yunpeng Wu, Rank-based pooling for deep convolutional neural networks, *Neural Netw.* 83 (2016) 21–31.
- [89] Yu-Dong Zhang, Suresh Chandra Satapathy, Di Wu, David S. Guttery, Juan Manuel Górriz, Shui-Hua Wang, Improving ductal carcinoma in situ classification by convolutional neural network with exponential linear unit and rank-based weighted pooling, *Complex Intell. Syst.* 7 (2021) 1295–1310.
- [90] Nadeem Akhtar, U Ragavendran, Interpretation of intelligence in cnn-pooling processes: A methodological survey, *Neural Comput. Appl.* 32 (3) (2020) 879–898.
- [91] Hatem Sindi, Majid Nour, Muhyaddin Rawa, Şaban Öztürk, Kemal Polat, Random fully connected layered 1d cnn for solving the z-bus loss allocation problem, *Measurement* 171 (2021) 108794.
- [92] M. Sornam, Kavitha Muthusubash, V. Vanitha, A survey on image classification and activity recognition using deep convolutional neural network architecture, in: 2017 Ninth International Conference on Advanced Computing, ICoAC, IEEE, 2017, pp. 121–126.
- [93] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, Kaori Togashi, Convolutional neural networks: An overview and application in radiology, *Insights into Imaging* 9 (4) (2018) 611–629.
- [94] Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie, Laith Farhan, Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions, *J. Big Data* 8 (1) (2021) 1–74.
- [95] Jinzhu Lu, Lijuan Tan, Huanyu Jiang, Review on convolutional neural network (cnn) applied to plant leaf disease classification, *Agriculture* 11 (8) (2021) 707.
- [96] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, *IEEE Trans. Neural Netw.* 5 (2) (1994) 157–166.
- [97] Sanjit Maitra, Rahul Kumar Ojha, Kuntal Ghosh, Impact of convolutional neural network input parameters on classification performance, in: 2018 4th International Conference for Convergence in Technology, I2CT, IEEE, 2018, pp. 1–5.
- [98] Lixing Huang, Jietao Diao, Hongshan Nie, Wei Wang, Zhiwei Li, Qingjiang Li, Haijun Liu, Memristor based binary convolutional neural network architecture with configurable neurons, *Front. Neurosci.* 15 (2021) 639526.
- [99] Zongmei Gao, Zhongwei Luo, Wen Zhang, Zhenzhen Lv, Yanlei Xu, Deep learning application in plant stress imaging: A review, *AgriEngineering* 2 (3) (2020) 29.
- [100] Heny Pratiwi, Agus Perdana Windarto, S. Susliansyah, Rin Restu Aria, Susi Susilowati, Luci Kanti Rahayu, Yuni Fitriani, Agustien Merdekawati, Indra Riyana Rahadjeng, Sigmoid activation function in selecting the best model of artificial neural networks, *J. Phys.: Conf. Ser.* 1471 (2020) 012010.
- [101] Bin Ding, Huimin Qian, Jun Zhou, Activation functions and their characteristics in deep neural networks, in: 2018 Chinese Control and Decision Conference, CCDC, IEEE, 2018, pp. 1836–1841.
- [102] Srigiri Krishnapriya, Yepuganti Karuna, Pre-trained deep learning models for brain mri image classification, *Front. Hum. Neurosci.* 17 (2023) 1150120.
- [103] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.
- [104] Bing Xu, Naiyan Wang, Tianqi Chen, Mu Li, Empirical evaluation of rectified activations in convolutional network, 2015, arXiv preprint [arXiv:1505.00853](https://arxiv.org/abs/1505.00853).
- [105] Djork-Arné Clevert, Thomas Unterthiner, Sepp Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), 2015, arXiv preprint [arXiv:1511.07289](https://arxiv.org/abs/1511.07289).
- [106] Sagar Sharma, Simone Sharma, Anidhya Athaiya, Activation functions in neural networks, *Towards Data Sci.* 6 (12) (2017) 310–316.
- [107] HamidReza Naseri, Vahid Mehrdad, Novel cnn with investigation on accuracy by modifying stride, padding, kernel size and filter numbers, *Multimedia Tools Appl.* 82 (15) (2023) 23673–23691.
- [108] Chen Yang, Yizhou Wang, Xiaoli Wang, Li Geng, A stride-based convolution decomposition method to stretch cnn acceleration algorithms for efficient and flexible hardware implementation, *IEEE Trans. Circuits Syst. I. Regul. Pap.* 67 (9) (2020) 3007–3020.
- [109] Joseph D. Prusa, Taghi M. Khoshgoftaar, Improving deep neural network design with new text data representations, *J. Big Data* 4 (1) (2017) 1–16.
- [110] Maite Gimenez, Javier Palanca, Vicent Botti, Semantic-based padding in convolutional neural networks for improving the performance in natural language processing: a case of study in sentiment analysis, *Neurocomputing* 378 (2020) 315–323.
- [111] Mahdi Hashemi, Enlarging smaller images before inputting into convolutional neural network: Zero-padding vs. interpolation, *J. Big Data* 6 (1) (2019) 1–13.
- [112] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, *J. Mach. Learn. Res.* 15 (1) (2014) 1929–1958.
- [113] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, Kilian Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.
- [114] Blanca Dalila Pérez-Pérez, Juan Pablo García Vázquez, Ricardo Salomón-Torres, Evaluation of convolutional neural networks' hyperparameters with transfer learning to determine sorting of ripe medjool dates, *Agriculture* 11 (2) (2021) 115.
- [115] Giuseppe Pezzano, Vicent Ribas Ripoll, Petia Radeva, Cole-cnn: Context-learning convolutional neural network with adaptive loss function for lung nodule segmentation, *Comput. Methods Programs Biomed.* 198 (2021) 105792.
- [116] Ibrahim Kandel, Mauro Castelli, The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset, *ICT Express* 6 (4) (2020) 312–315.
- [117] Spiros V. Georgakopoulos, Vassilis P. Plagianakos, A novel adaptive learning rate algorithm for convolutional neural network training, in: International Conference on Engineering Applications of Neural Networks, Springer, 2017, pp. 327–336.
- [118] Leslie N. Smith, Cyclical learning rates for training neural networks, in: 2017 IEEE Winter Conference on Applications of Computer Vision, WACV, IEEE, 2017, pp. 464–472.
- [119] Sandhya Sharma, Shefali Gupta, Deepali Gupta, Junaid Rashid, Sapna Juneja, Jungeun Kim, Mahmoud M. Elarabawy, Performance evaluation of the deep learning based convolutional neural network approach for the recognition of chest x-ray images, *Front. Oncol.* 12 (2022) 932496.
- [120] Shivam Sinha, T.N. Singh, V.K. Singh, A.K. Verma, Epoch determination for neural network by self-organized map (som), *Comput. Geosci.* 14 (1) (2010) 199–206.
- [121] Aatila Mustapha, Lachgar Mohamed, Kartit Ali, An overview of gradient descent algorithm optimization in machine learning: Application in the ophthalmology field, in: International Conference on Smart Applications and Data Analysis, Springer, 2020, pp. 349–359.
- [122] D. Randall Wilson, Tony R. Martinez, The general inefficiency of batch training for gradient descent learning, *Neural Netw.* 16 (10) (2003) 1429–1451.
- [123] Léon Bottou, Stochastic gradient descent tricks, in: *Neural Networks: Tricks of the Trade*, Springer, 2012, pp. 421–436.
- [124] Sarit Khirirat, Hamid Reza Feyzmahdavian, Mikael Johansson, Mini-batch gradient descent: Faster convergence under data sparsity, in: 2017 IEEE 56th Annual Conference on Decision and Control, CDC, IEEE, 2017, pp. 2880–2887.
- [125] Ning Qian, On the momentum term in gradient descent learning algorithms, *Neural Netw.* 12 (1) (1999) 145–151.
- [126] Ilya Sutskever, James Martens, George Dahl, Geoffrey Hinton, On the importance of initialization and momentum in deep learning, in: International Conference on Machine Learning, PMLR, 2013, pp. 1139–1147.
- [127] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- [128] Akhilesh Kumar Sharma, Gaurav Aggarwal, Sachit Bhardwaj, Prasun Chakrabarti, Tulika Chakrabarti, Jemal H. Abawajy, Siddhartha Bhattacharyya, Richa Mishra, Anirban Das, Hairulnizam Mahdin, Classification of indian classical music with time-series matching deep learning approach, *IEEE Access* 9 (2021) 102041–102052.
- [129] Ibrahim Kandel, Mauro Castelli, Aleš Popović, Comparative study of first order optimizers for image classification using convolutional neural networks on histopathology images, *J. Imaging* 6 (9) (2020) 92.
- [130] Yixiang Wang, Jiqiang Liu, Jelena Mišić, Vojislav B. Mišić, Shaohua Lv, Xiaolin Chang, Assessing optimizer impact on dnn model sensitivity to adversarial examples, *IEEE Access* 7 (2019) 152766–152776.
- [131] Md Nasim Khan, Mohamed M. Ahmed, Trajectory-level fog detection based on in-vehicle video camera with tensorflow deep learning utilizing shrp2 naturalistic driving data, *Accid. Anal. Prev.* 142 (2020) 105521.
- [132] Timothy Dozat, Incorporating nesterov momentum into adam, 2016.
- [133] Yohan Muliono, Hanry Ham, Dion Darmawan, Keystroke dynamic classification using machine learning for password authorization, *Procedia Comput. Sci.* 135 (2018) 564–569.
- [134] Bin Xiao, Yonggui Liu, Bing Xiao, Accurate state-of-charge estimation approach for lithium-ion batteries by gated recurrent unit with ensemble optimizer, *IEEE Access* 7 (2019) 54192–54202.
- [135] Shouxiang Wang, Haiwen Chen, A novel deep learning method for the classification of power quality disturbances using deep convolutional neural network, *Appl. Energy* 235 (2019) 1126–1140.
- [136] Ali Sezer, Aytac Altan, Detection of solder paste defects with an optimization-based deep learning model using image processing techniques, *Soldering Surface Mount Technol.* (2021).
- [137] Abdullah Emir Cil, Kazim Yildiz, Ali Buldu, Detection of ddos attacks with feed forward based deep neural network model, *Expert Syst. Appl.* 169 (2021) 114520.
- [138] Wei-Min Chu, Endah Kristiani, Yu-Chieh Wang, Yen-Ru Lin, Shih-Yi Lin, Wei-Cheng Chan, Chao-Tung Yang, Yu-Tse Tsan, A model for predicting fall risks of hospitalized elderly in Taiwan-a machine learning approach based on both electronic health records and comprehensive geriatric assessment, *Front. Med.* 9 (2022) 937216.
- [139] Sebastian Ruder, An overview of gradient descent optimization algorithms, 2016, arXiv preprint [arXiv:1609.04747](https://arxiv.org/abs/1609.04747).

- [140] Filippos Giannakas, Christos Troussas, Ioannis Voyiatzis, Cleo Sgouropoulou, A deep learning classification framework for early prediction of team-based academic performance, *Appl. Soft Comput.* 106 (2021) 107355.
- [141] Matthew D. Zeiler, Adadelta: An adaptive learning rate method, 2012, arXiv preprint arXiv:1212.5701.
- [142] Zhijian Qu, Shengao Yuan, Rui Chi, Liuchen Chang, Liang Zhao, Genetic optimization method of pantograph and catenary comprehensive monitor status prediction model based on adadelta deep neural network, *IEEE Access* 7 (2019) 23210–23221.
- [143] Zhiqiang Hao, Zhigang Wang, Dongxu Bai, Bo Tao, Xiliang Tong, Baojia Chen, Intelligent detection of steel defects based on improved split attention networks, *Front. Bioeng. Biotechnol.* 9 (2022) 810876.
- [144] Qi Wang, Yue Ma, Kun Zhao, Yingjie Tian, A comprehensive survey of loss functions in machine learning, *Ann. Data Sci.* 9 (2) (2022) 187–212.
- [145] Yaoshiang Ho, Samuel Wooley, The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling, *IEEE Access* 8 (2019) 4806–4813.
- [146] Mutegeki Ronald, Alwin Poulse, Dong Seog Han, Isplinception: An inception-resnet deep learning architecture for human activity recognition, *IEEE Access* 9 (2021) 68985–69001.
- [147] Jinhyun Park, Seung-Eun Lee, Hak-Joon Kim, Sung-Jin Song, Sung-Sik Kang, System invariant method for ultrasonic flaw classification in weldments using residual neural network, *Appl. Sci.* 12 (3) (2022) 1477.
- [148] Usha Ruby, Vamsidhar Yendapalli, Binary cross entropy with deep learning technique for image classification, *Int. J. Adv. Trends Comput. Sci. Eng.* 9 (10) (2020).
- [149] Prashant Brahmabhatt, Siddhi Nath Rajan, Skin lesion segmentation using segnet with binary crossentropy, in: *Proceedings of the International Conference on Artificial Intelligence and Speech Technology, AIST2019, Delhi, India, 2019*, pp. 14–15.
- [150] Alfrina Rizqi Lahitani, Adhistya Erna Permasari, Noor Akhmad Setiawan, Cosine similarity to determine similarity measure: Study case in online essay assessment, in: *2016 4th International Conference on Cyber and IT Service Management, IEEE, 2016*, pp. 1–6.
- [151] Keyan Shen, Hui Qin, Jianzhong Zhou, Guanjun Liu, Runoff probability prediction model based on natural gradient boosting with tree-structured Parzen estimator optimization, *Water* 14 (4) (2022) 545.
- [152] Hoang-Phuong Nguyen, Jie Liu, Enrico Zio, A long-term prediction approach based on long short-term memory neural networks with automatic parameter optimization by tree-structured Parzen estimator and applied to time-series data of npp steam generators, *Appl. Soft Comput.* 89 (2020) 106116.
- [153] Seyedali Mirjalili, Seyed Mohammad Mirjalili, Andrew Lewis, Grey wolf optimizer, *Adv. Eng. Softw.* 69 (2014) 46–61.
- [154] Jin Hee Yoon, Zong Woo Geem, Empirical convergence theory of harmony search algorithm for box-constrained discrete optimization of convex function, *Mathematics* 9 (5) (2021) 545.
- [155] Rainer Storn, Kenneth Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, *J. Glob. Optim.* 11 (4) (1997) 341–359.
- [156] B.V. Babu, M. Mathew Leenus Jehan, Differential evolution for multi-objective optimization, *CEC'03*, in: *The 2003 Congress on Evolutionary Computation, 2003*, vol. 4, IEEE, 2003, pp. 2696–2703.
- [157] Eslam Mohammed Abdelkader, Abobakr Al-Sakkaf, Ghasan Alfalah, Nehal Elshaboury, Hybrid differential evolution-based regression tree model for predicting downstream dam hazard potential, *Sustainability* 14 (5) (2022) 3013.
- [158] Ernesto Mininno, Ferrante Neri, A memetic differential evolution approach in noisy optimization, *Memet. Comput.* 2 (2) (2010) 111–135.
- [159] Swagatam Das, Ponnuthurai Nagarathnam Suganthan, Differential evolution: A survey of the state-of-the-art, *IEEE Trans. Evol. Comput.* 15 (1) (2010) 4–31.
- [160] Sourabh Katoch, Sumit Sing Chauhan, Vijay Kumar, A review on genetic algorithm: Past, present, and future, *Multimedia Tools Appl.* 80 (5) (2021) 8091–8126.
- [161] Hui Zhi, Sanyang Liu, Face recognition based on genetic algorithm, *J. Vis. Commun. Image Represent.* 58 (2019) 495–502.
- [162] Riccardo Poli, James Kennedy, Tim Blackwell, Particle swarm optimization, *Swarm Intell.* 1 (1) (2007) 33–57.
- [163] Jhansi Rani Challapalli, Nagaraju Devarakonda, A novel approach for optimization of convolution neural network with hybrid particle swarm and grey wolf algorithm for classification of Indian classical dances, *Knowl. Inf. Syst.* 64 (9) (2022) 2411–2434.
- [164] Xin-She Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, 2010.
- [165] Yoshihiko Ozaki, Masaki Yano, Masaki Onishi, Effective hyperparameter optimization using nelder-mead method in deep learning, *IPSJ Trans. Comput. Vis. Appl.* 9 (1) (2017) 1–12.
- [166] Hui Zhang, Deep neural network for object classification and optimization algorithms for 3d positioning in ultrasonic sensor array, 2021.
- [167] Rafael Marconi Ramos, Célia Ghedini Ralha, Tahsin M. Kurc, Joel H. Saltz, George Teodoro, Increasing accuracy of medical cnn applying optimization algorithms: An image classification case, in: *2019 8th Brazilian Conference on Intelligent Systems, BRACIS, IEEE, 2019*, pp. 233–238.
- [168] Xavier Glorot, Yoshua Bengio, Understanding the difficulty of training deep feedforward neural networks, in: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010*, pp. 249–256, JMLR Workshop and Conference Proceedings.
- [169] Saleh Albelwi, Ausif Mahmood, Automated optimal architecture of deep convolutional neural networks for image recognition, in: *2016 15th IEEE International Conference on Machine Learning and Applications, ICMLA, IEEE, 2016*, pp. 53–60.
- [170] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, Yoshua Bengio, Theano: A cpu and gpu math compiler in Python, in: *Proc. 9th Python in Science Conf*, vol. 1, 2010, pp. 3–10.
- [171] Petro Liashchynskyi, Pavlo Liashchynskyi, Grid search, random search, genetic algorithm: A big comparison for nas, 2019, arXiv preprint arXiv:1912.06059.
- [172] WL1551847 Price, Global optimization by controlled random search, *J. Optim. Theory Appl.* 40 (3) (1983) 333–348.
- [173] Jonas Mockus, Bayesian approach to global optimization: theory and applications, vol. 37, Springer Science & Business Media, 2012.
- [174] Peter I. Frazier, Bayesian optimization, in: *Recent Advances in Optimization and Modeling of Contemporary Problems, Informatics*, 2018, pp. 255–278.
- [175] Mohamed Ait Amou, Kewen Xia, Souha Kamhi, Mohamed Mouhafid, A novel mri diagnosis method for brain tumor classification based on cnn and Bayesian optimization, in: *Healthcare*, 10, MDPI, 2022, p. 494.
- [176] Ghada Atteia, Nagwan Abdel Samee, El-Sayed M. El-Kenawy, Abdelhameed Ibrahim, Cnn-hyperparameter optimization for diabetic maculopathy diagnosis in optical coherence tomography and fundus retinography, *Mathematics* 10 (18) (2022) 3274.
- [177] Hyejung Chung, Kyung-shik Shin, Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction, *Neural Comput. Appl.* 32 (12) (2020) 7897–7914.
- [178] Hailun Xie, Li Zhang, Chee Peng Lim, Evolving cnn-lstm models for time series prediction using enhanced grey wolf optimizer, *IEEE Access* 8 (2020) 161519–161541.
- [179] Ehsan Rokhsatyazdi, Shahryar Rahnamayan, Hossein Amirinia, Sakib Ahmed, Optimizing lstm based network for forecasting stock market, in: *2020 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2020*, pp. 1–7.
- [180] Kirti Kumari, Jyoti Prakash Singh, Yogesh K. Dwivedi, Nripendra P. Rana, Multi-modal aggression identification using convolutional neural network and binary particle swarm optimization, *Future Gener. Comput. Syst.* 118 (2021) 187–197.
- [181] Ramon Zatarain Cabada, Hector Rodriguez Rangel, Maria Lucia Barron Estrada, Hector Manuel Cardenas Lopez, Hyperparameter optimization in cnn for learning-centered emotion recognition for intelligent tutoring systems, *Soft Comput.* 24 (10) (2020) 7593–7602.
- [182] Saied Raziani, Mehran Azimbagirad, Deep cnn hyperparameter optimization algorithms for sensor-based human activity recognition, *Neurosci. Inform.* 2 (3) (2022) 100078.
- [183] Xiaofei Li, Hainan Guo, Langxing Xu, Zezheng Xing, Bayesian-based hyperparameter optimization of 1d-cnn for structural anomaly detection, *Sensors* 23 (11) (2023) 5058.
- [184] Subhrajit Mitra, Rajarshi Mukhopadhyay, Paramita Chattopadhyay, Pso driven designing of robust and computation efficient 1d-cnn architecture for transmission line fault detection, *Expert Syst. Appl.* 210 (2022) 118178.
- [185] Davor Kolar, Dragutin Lisjak, Michal Pajak, Mihael Gudlin, Intelligent fault diagnosis of rotary machinery by convolutional neural network with automatic hyper-parameters tuning using bayesian optimization, *Sensors* 21 (7) (2021) 2411.
- [186] Xiaolu Liu, Li Jia, Yang Li, A genetic-firefly algorithm based cnn-lstm for lithium-ion battery fault diagnosis, in: *2023 6th International Conference on Robotics, Control and Automation Engineering, RCAE, IEEE, 2023*, pp. 377–382.
- [187] Ashraf Darwish, Dalia Ezzat, Aboul Ella Hassanien, An optimized model based on convolutional neural networks and orthogonal learning particle swarm optimization algorithm for plant diseases diagnosis, *Swarm Evol. Comput.* 52 (2020) 100616.
- [188] Dongmei Liu, Haibin Ouyang, Steven Li, Chunliang Zhang, Zhi-Hui Zhan, Hyperparameters optimization of convolutional neural network based on local autonomous competition harmony search algorithm, *J. Comput. Des. Eng.* (2023) qwad050.
- [189] Gustavo Rosa, João Papa, Kelton Costa, Leandro Passos, Clayton Pereira, Xin-She Yang, Learning parameters in deep belief networks through firefly algorithm, in: *Artificial Neural Networks in Pattern Recognition: 7th IAPR TC3 Workshop, ANNPR 2016, Ulm, Germany, September (2016) 28–30, Proceedings 7, Springer, 2016*, pp. 138–149.
- [190] Gustavo Rosa, Joao Papa, Aparecido Marana, Walter Scheirer, David Cox, Fine-tuning convolutional neural networks using harmony search, in: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 20th Iberoamerican Congress, CIARP 2015, Montevideo, Uruguay, November (2015) 9–12, Proceedings 20, Springer, 2015*, pp. 683–690.
- [191] Francisco Erivaldo Fernandes Junior, Gary G Yen, Particle swarm optimization of deep neural networks architectures for image classification, *Swarm Evol. Comput.* 49 (2019) 62–74.

- [192] Yanan Sun, Bing Xue, Mengjie Zhang, Gary G. Yen, A particle swarm optimization-based flexible convolutional autoencoder for image classification, *IEEE Trans. Neural Netw. Learn. Syst.* 30 (8) (2018) 2295–2309.
- [193] Wei-Chang Yeh, Yi-Ping Lin, Yun-Chia Liang, Chyh-Ming Lai, Chia-Ling Huang, Simplified swarm optimization for hyperparameters of convolutional neural networks, *Comput. Ind. Eng.* 177 (2023) 109076.
- [194] Junhao Huang, Bing Xue, Yanan Sun, Mengjie Zhang, A flexible variable-length particle swarm optimization approach to convolutional neural network architecture design, in: 2021 IEEE Congress on Evolutionary Computation, CEC, IEEE, 2021, pp. 934–941.
- [195] Ahmed I Sharaaf, El-Sayed F Radwan, An automated approach for developing a convolutional neural network using a modified firefly algorithm for image classification, in: *Applications of Firefly Algorithm and Its Variants: Case Studies and New Developments*, Springer, 2019, pp. 99–118.
- [196] Cristian Muro, R. Escobedo, L. Spector, R.P. Coppinger, Wolf-pack (canis lupus) hunting strategies emerge from simple rules in computational simulations, *Behav. Processes* 88 (3) (2011) 192–197.
- [197] Tzu-Yen Hong, Chin-Chien Chen, Hyperparameter optimization for convolutional neural network by opposite-based particle swarm optimization and an empirical study of photomask defect classification, *Appl. Soft Comput.* (2023) 110904.
- [198] Pritpal Singh, Monoj Kumar Muchahari, Solving multi-objective optimization problem of convolutional neural network using fast forward quantum optimization algorithm: Application in digital image classification, *Adv. Eng. Softw.* 176 (2023) 103370.
- [199] Chilukamari Rajesh, Sushil Kumar, An evolutionary block based network for medical image denoising using differential evolution, *Appl. Soft Comput.* 121 (2022) 108776.
- [200] KS. Ananda Kumar, A.Y. Prasad, Jyoti Metan, A hybrid deep cnn-cov-19-res-net transfer learning archetype for an enhanced brain tumor detection and classification scheme in medical image processing, *Biomed. Signal Process. Control* 76 (2022) 103631.
- [201] José Escorcia-Gutierrez, Margarita Gamarra, Kelvin Beleño, Carlos Soto, Roman F. Mansour, Intelligent deep learning-enabled autonomous small ship detection and classification model, *Comput. Electr. Eng.* 100 (2022) 107871.
- [202] Mirza Amaad Ul Haq Tahir, Sohail Asghar, Awais Manzoor, Muhammad Asim Noor, A classification model for class imbalance dataset using genetic programming, *IEEE Access* 7 (2019) 71013–71037.
- [203] Kitsuchart Pasupa, Wisuwat Sunhem, A comparison between shallow and deep architecture classifiers on small dataset, in: 2016 8th International Conference on Information Technology and Electrical Engineering, ICITEE, IEEE, 2016, pp. 1–6.
- [204] Sagar Kora Venu, Sridhar Ravula, Evaluation of deep convolutional generative adversarial networks for data augmentation of chest x-ray images, *Future Internet* 13 (1) (2020) 8.
- [205] Mohamed Elgendi, Muhammad Umer Nasir, Qunfeng Tang, David Smith, John-Paul Grenier, Catherine Batte, Bradley Spieler, William Donald Leslie, Carlo Menon, Richard Ribbon Fletcher, et al., The effectiveness of image augmentation in deep learning networks for detecting covid-19: A geometric transformation perspective, *Front. Med.* 8 (2021) 629134.
- [206] Eduardo Castro, Jaime S. Cardoso, Jose Costa Pereira, Elastic deformations for data augmentation in breast cancer mass detection, in: 2018 IEEE EMBS International Conference on Biomedical & Health Informatics, BHI, IEEE, 2018, pp. 230–234.
- [207] Luke Taylor, Geoff Nitschke, Improving deep learning with generic data augmentation, in: 2018 IEEE Symposium Series on Computational Intelligence, SSCI, IEEE, 2018, pp. 1542–1547.
- [208] Julia Moosbauer, Julia Herbringer, Giuseppe Casalicchio, Marius Lindauer, Bernd Bischl, Explaining hyperparameter optimization via partial dependence plots, *Adv. Neural Inf. Process. Syst.* 34 (2021) 2280–2291.
- [209] Paul Novello, Gaël Poëtte, David Lugato, Pietro M. Congedo, Explainable hyperparameters optimization using hilbert-schmidt independence criterion, 2021.
- [210] Tong Yu, Hong Zhu, Hyper-parameter optimization: A review of algorithms and applications, 2020, arXiv preprint arXiv:2003.05689.
- [211] Anastasia Makarova, Huibin Shen, Valerio Perrone, Aaron Klein, Jean Baptiste Faddoul, Andreas Krause, Matthias Seeger, Cedric Archambeau, Automatic termination for hyperparameter optimization, in: *International Conference on Automated Machine Learning*, PMLR, 2022, pp. 1–7.
- [212] Yasser A. Ali, Emad Mahrous Awwad, Muna Al-Razgan, Ali Maarouf, Hyperparameter search for machine learning algorithms for optimizing the computational complexity, *Processes* 11 (2) (2023) 349.
- [213] Polipreddy Srinivas, Rahul Katarya, Hyoptxg: Optuna hyper-parameter optimization framework for predicting cardiovascular disease using xgboost, *Biomed. Signal Process. Control* 73 (2022) 103456.
- [214] Noor Awad, Neeratyoy Mallik, Frank Hutter, Dehb: Evolutionary hyperband for scalable, robust and efficient hyperparameter optimization, 2021, arXiv preprint arXiv:2105.09821.
- [215] Thomas Bartz-Beielstein, Hyperparameter tuning cookbook: A guide for scikit-learn, pytorch, river, and spotpytho, 2023, arXiv preprint arXiv:2307.10262.
- [216] Muammer Türkoğlu, Hüseyin Polat, Cemal Koçak, Onur Polat, Recognition of ddos attacks on sd-vanet based on combination of hyperparameter optimization and feature selection, *Expert Syst. Appl.* 203 (2022) 117500.
- [217] Malliga Subramanian, L.V. Narasimha Prasad, V.E. Sathishkumar, Hyperparameter optimization for transfer learning of vgg16 for disease identification in corn leaves using bayesian optimization, *Big Data* 10 (3) (2022) 215–229.
- [218] Vinod Jagannath Kadam, Shivajirao Manikrao Jadhav, Performance analysis of hyperparameter optimization methods for ensemble learning with small and medium sized medical datasets, *J. Discrete Math. Sci. Cryptogr.* 23 (1) (2020) 115–123.