Neural Computing and Applications (2021) 33:14079–14099
https://doi.org/10.1007/s00521-021-05960-5

REVIEW ARTICLE

Advanced metaheuristic optimization techniques in applications of deep neural networks: a review

Mohamed Abd Elaziz[1,2] · Abdelghani Dahou[3] · Laith Abualigah[4] · Liyang Yu[2] · Mohammad Alshinwan[4] · Ahmad M. Khasawneh[4] · Songfeng Lu[5]

Received: 8 September 2020 / Accepted: 25 March 2021 / Published online: 18 April 2021

Abstract
Deep neural networks (DNNs) have evolved as a beneficial machine learning method that has been successfully used in various applications. Currently, DNN is a superior technique of extracting information from massive sets of data in a self-organized method. DNNs have different structures and parameters, which are usually produced for particular applications. Nevertheless, the training procedures of DNNs can be protracted depending on the given application and the size of the training set. Further, determining the most precise and practical structure of a deep learning method in a reasonable time is a possible problem related to this procedure. Meta-heuristics techniques, such as swarm intelligence (SI) and evolutionary computing (EC), represent optimization frames with specific theories and objective functions. These methods are adjustable and have been demonstrated their effectiveness in various applications; hence, they can optimize the DNNs models. This paper presents a comprehensive survey of the recent optimization methods (i.e., SI and EC) employed to enhance DNNs performance on various tasks. This paper also analyzes the importance of optimization methods in generating the optimal hyper-parameters and structures of DNNs in taking into consideration massive-scale data. Finally, several potential directions that still need improvements and open problems in evolutionary DNNs are identified.

Keywords Metaheuristic Optimization · Algorithms · Deep neural networks · Applications · A review

1 Introduction

In 1943, McCilloch and Pitts [1] proposed a new concept and mathematical model of artificial neural network (ANN), which ushered in a new era ANN. The essence of ANN is to provide a simulation and abstract description of biological neuronal structure. In 1958, Rosenblatt proposed a neural network that contains two layers of neural cells and termed this as perceptrons [2]. The perceptron algorithm used the ANN model to classify an input of data

✉ Laith Abualigah
aligah.2020@gmail.com

Mohamed Abd Elaziz
abd_el_aziz_m@yahoo.com

Abdelghani Dahou
dahou.abdghani@univ-adrar.dz

Liyang Yu
1044600735@qq.com

Mohammad Alshinwan
mohmdsh@aau.edu.jo

Ahmad M. Khasawneh
a.khasawneh@aau.edu.jo

Songfeng Lu
lusongfeng@hust.edu.cn

[1] Department of Mathematics, Faculty of Science, Zagazig University, Zagazig 44519, Egypt

[2] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China

[3] Mathematics and Computer Science department, University of Ahmed DRAIA, 01000 Adrar, Algeria

[4] Faculty of Computer Sciences and Informatics, Amman Arab University, 11953 Amman, Jordan

[5] Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong university of Science and Technology, Wuhan 430074, China

represented in multiple dimensions where gradient descent was used to update the weights of ANN. However, ANN is a simple linear model that cannot solve nonlinear problems such as the XOR problem. Meanwhile, Geoffrey et al. [3] proposed backpropagation (BP) algorithm to train multiple layers perceptron (MLP) models and tackle nonlinear classification problems. However, BP was pointed that it may cause a gradient-vanish problem or gradient-explosion problem. Besides, support vector machine (SVM) [4] and other statistical learning methods showed better performance than ANN in that period. Geoffrey Hinton [5, 6] proposed a solution to solve the shortcoming of BP using unsupervised learning to pre-trained deep neural network (DNN) weights and then using supervised learning to train DNN. On account of DNNs better performance, more universities and companies pay considerable attention to DNN research. Microsoft and Google achieved many breakthroughs during that decade. The most noticeable event of DNN is that AlexNet ranked the champion in the 2012 ImageNet [7] competition. Their results outperform the second place model built upon SVM. Different DNNs were designed by human experts since that time (e.g., GoogleNet [8], ResNet [9], DenseNet [10], EfficientNet [11] in image classification problem).

In a traditional machine learning (ML) project, the process of building an ML model can be divided into several steps: modeling the problem into a mathematical problem, collecting data and data cleaning, feature extraction and selection, training, model optimization, and model evaluation. Contrary, DNN belongs to deep end-to-end learning where it can replace several processing steps into a single DNN to reduce the cost of complex feature design, and extraction [12]. DNN only needs enough data and correct hyper-parameters to achieve competitive performance.

The growth of data and computing power can be seen as the most crucial point that pushed DNNs' evolution. With the advent of big-data, DNN got more data to train, effectively reducing the over-fitting problem. Besides, the growth of computing power allows the training of bigger DNN models than before. Although, DNN also has some shortcomings that cannot be ignored. In most cases, the design of DNN architecture needs human intervention. DNN needs a massive amount of labeled data to train on. These labeled data consume significant resources and come with a cost. Moreover, better performance of DNN requires researchers to tune DNNs hyper-parameters iteratively.

This paper reviews existing meta-heuristic optimization techniques and algorithms that have been used in the design and tune their DNN in various research tasks, fields, and applications. These algorithms include genetic algorithm (GA), genetic programming (GP), differential evolution (DE), finite-state machine (FSM), evolution strategies (ESs), particle swarm optimization (PSO), ant colony optimization (ACO), firefly algorithm (FA), and other hybrid methods. This review helps researchers in future research in DNNs using optimization techniques and algorithms by showing the currently published papers and their main proposed methods, advantages, and disadvantages. We found that various published papers design and tune their DNN in different research tasks, fields, and applications. The paper summarizes many nature-inspired meta-heuristic optimization algorithms (MHOA) applied to train DNNs, fine-tuning DNNs, network architecture search (NAS) DNNs hyper-parameters search and optimization. Moreover, discussion, problems, and challenges are presented in the review also.

The main contributions invented in this paper are given as follows:

1. Reviewed the well-known meta-heuristic optimization techniques and algorithms that have been employed to tune the DNN parameters in various applications.
2. Classified the existing optimization techniques that have been used in adjusting DNN parameters.
3. Evaluated several well-known meta-heuristic optimization techniques combined with DNN for image classification.
4. Defined the problems and challenges in the domain of optimization techniques for tuning the DNN.

The rest of this paper is organized as follows: Sect. 2 introduces the background of DL architectures and meta-heuristic techniques. Sections 3 and 4 present the evolutionary algorithms and swarm algorithms used to enhance DNN. Section 5 presents the hybrid meta-heuristic techniques applied to improve the performance of DNN. The challenges and future work are introduced in Sect. 7. The conclusion is given in Sect. 9.

2 Background

2.1 Deep learning architectures and techniques

In this section, we will briefly review some of the well-known DL architectures. Besides, we will present various DL techniques which can be investigated using an optimization method. Several parameters need to be tuned in the optimization algorithms and optimization problems. Some of them should be adjusted by the user, and the algorithm itself should fix others.

DL is used to model the complexity and abstractions of data by employing multiple processing layers. Data can be presented in various forms such as text, image, and audio [13, 14]. Thus, DL can be applied in different research areas such as language modeling, pattern recognition,

signal processing, and optimization. The advancements in neuroscience, high-performance computing devices, and big data placed DL methods on the cutting-edge artificial intelligence (AI) research enabling DL methods to learn and perform more complex calculations and representations effectively. Thus, DL methods automatically extract the features of structured and unstructured data and learn their representations using supervised and unsupervised strategies. Although DL has many breakthroughs in different research fields and applications, DL still faces some limitations in its design and hyper-parameters setting.

The demand for robust and innovative DL-based solutions has increased in big data analytic during the past decade. Other areas, including e-commerce, industrial control, bioinformatics [15], robotics [16], and health care [17]. The following facts could characterize DL-based solutions:

1. Extract hidden information and features from noisy data.
2. Learn and train from samples to discover patterns and essential information.
3. Classify structured and unstructured data.
4. Simulate the human brain to solve a given problem using ANN.

DL methods do not rely on human intervention when it comes to feature engineering. It benefits from the hidden layers to build an automatic high-level feature extraction module such in convolutional neural network (CNN) architecture [18]. Thus, in applications such as pattern recognition, DL models can detect objects and translate speech. Although DL provides the best solutions to many problems, DL networks possess various hyper-parameters related to the network architecture, for example, number of hidden layers and the associated numbers of neurons in each layer, number of convolution layers, pooling layer type and size, type of activation function, dropout ratio, optimization algorithm, and so on. The presence of this variety in hyper-parameters alongside the absence of any solid mathematical theory makes it challenging to set the appropriate hyper-parameters, which in most cases are set by trial-and-error process. Thus, EC and SI methods can be used to tackle this problem and evolve the structure and hyper-parameters of a DL network. Finally, it may maximize the network's performance on a specific task without the need for human intervention.

CNN is the most commonly applied DL architecture in various applications such as image and video analysis, natural language processing, anomaly detection, health risk assessment, and time-series forecasting. A CNN architecture can be composed of several convolution layers, pooling layers, and fully connected layers on top of each other. CNN has achieved good results in many applications including: raw audio generation [19], speech synthesis [20], object detection [21], and transfer learning [11]. Besides, CNNs possess relatively fewer parameters to be set, which ease the usage and training of such architecture. CNNs are widely used as feature extractors with image-like data for several reasons, including the successful training process and the network topology, which reduces the number of parameters to be fine-tuned. Moreover, pre-trained CNN architectures have been commonly used, such as VGG [22] which is a simple CNN architecture with increased depth.

ResNet (residual networks) [23] is a network that allows layers (residual blocks) to fit using residual mapping instead of hoping each few stacked layers directly. Compared to a ResNet, ResNeXt [24] uses cardinality (the size of the set of transformations) to perform a set of transformations. EfficientNet [25] uses a compound coefficient to uniformly scales all dimensions of a CNN architecture including depth, width, and resolution. XceptionNet [26, 27] replies on CNN architecture and depthwise separable convolution layers. DenseNet [28] is a CNN based architecture with dense connections between layers (dense blocks). The aforementioned architectures are trained on the ImageNet dataset [7] or other large dataset, and they can run on GPU platforms. Other pre-trained architectures running on CPU platforms include SqueezeNet [29], MobileNet [21, 30], and ShuffleNet [31, 32]. More architectures have been trained for specific tasks such as object detection including You only look once (YOLO) [33, 34], single shot multibox (SSD) [35], and RetinaNet [36].

Recurrent neural network (RNN) [37] and its variants such as gated recurrent unit (GRU) and long short-term memory (LSTM) are widely used in NLP [38] and speech recognition tasks such in language modeling [39], sentiment classification [40], music generation, named entity recognition, and machine translation [41]. Traditional RNNs are a class of ANNs that uses hidden states to allow previous outputs to be used as inputs. Historical information is taken into consideration during the training with shared weights across time. RNNs are suitable for data that can be represented as a sequence to advance or complete information, such as auto-completion. However, RNNs suffer from the vanishing (or exploding) gradient problem. Information can be lost over time, slow computation, hard to reach information from a long time ago. To overcome the vanishing gradient problem and short-term memory of RNN, LSTM, and GRU were proposed with different mechanisms introduced as memory cells with defined gates.

Moreover, each neuron is defined by a memory cell and a specific type of gates where the gates control the information flow. LSTM has input, output, and forge gates. In contrast, GRU has updated and reset gates where in most

cases, they function similarly to LSTM. GRUs are faster and easier to run compared to LSTMs, which requires more computationally resources. Besides, LSTMs can learn complex sequences where the problem of exploding gradient in forwarding propagation can be solved using gradient clipping [42].

In the past few years, many state-of-the-art CNN architectures were introduced in natural language processing (NLP) applications by converting CNN models that work on images to be applied on the text also [43, 44]. Moreover, advanced encoder and decoder architectures known as transformers [45] have been proposed. Transformers mainly rely on an attention mechanism instead of recurrence, which offers more parallelization than methods like RNNs and CNNs. For example, Megatron-LM [46] is a state-of-the-art transformer model in many tasks trained with billions of parameters using an intra-layer model parallel approach. BERT [47] (bidirectional encoder representations from transformers) is an improved version of the standard transformer architecture, which employs masked language model (MLM) pretraining objective. RoBERTa [48] is an extent ion of BERT that has been trained for more extended time on more data consisting of longer sequences. Dilbert [49] is a distilled version of BERT with reduced size of 40% for fast and light training. BART [50] is a denoising autoencoder that uses arbitrary noising function and tries to reconstruct the original input (test) for pretraining sequence-to-sequence models. T5 [51] is also known as the text-to-text transfer transformer, which uses the text-to-text method which extends its application across diverse tasks such as translation, question answering, and classification. Reformer [52] uses locality-sensitive hashing instead of dot-product attention and employs reversible residual layers to improve the efficiency of the standard transformer. GPT-3 is an autoregressive transformer that uses dense and locally banded sparse attention patterns [53]. These architectures have been applied in different fields including bioinformatics [54], text ranking [55], machine translation [56], and question answering [57].

Deep autoencoders [58] and generative adversarial networks (GANs) [59] can be classified as generative models where they tend to model data generatively. These models are commonly used in computer vision to approximate the probability distribution of specific data. Generative models such as deep autoencoders and GANs can be applied to various applications, including denoising, dimensionality reduction, image modeling [60], adversarial training [60], and image generation [61]. Briefly, an autoencoder uses an encoder to transfer a high-dimensional input into a latent low-dimensional representation and a decoder to inverse the operation and reconstruct the input using the low-dimensional representation. Meanwhile, GAN trains two models, which are the generative model and the discriminative model, simultaneously. The generative model learns to imitate the data distribution.

In contrast, the discriminative model estimates the probability that a sample came from the training data rather than the generative model. Recently, some state-of-the-art architectures have been introduced, including LeakGAN [62] which is a GAN-based model for long text generation, which proved to improve the performance in short text generation applications. StyleGAN [60, 63] uses style transfer approaches to propose an alternative generator for unsupervised separation of high-level attributes and image generation. U-Net GAN [64] uses U-Net [65] architecture as a discriminator by improving U-Net training with a per-pixel consistency regularization technique for image generation.

2.2 Meta-heuristics techniques

Within this section, meta-heuristic (MH) techniques are discussed. In general, the process of use MH techniques to solve different applications has grown exponentially. They are free gradient methods and can solve highly complex optimization problems with results better than the traditional methods [66]. In addition, they are simple in the implementation and fast than classical optimization methods [67, 68].

There are variant inspirations for MH techniques to be classified into different groups according to these inspirations. These groups are evolutionary algorithms (EAs), swarm intelligence (SI) methods, natural phenomena approaches, and human inspiration algorithms [69, 70]. Figure 1 depicts these groups.

In the first group that called EAs, the inspiration of the algorithms depends on simulating the natural genetic concepts such as crossover, mutation, and selection. Several MH methods belong to this group such as Arithmetic Optimization Algorithm (AOA) [71], evolutionary programming [72], GA [73, 74], ES [75] DE [76], and GP [77].

The second group, named SI, simulates the behavior of swarm in nature during searching for food. The most popular of this group is PSO [78], salp swarm algorithm (SSA) [79], marine predators algorithm (MPA) [80], and whale optimization algorithm (WOA) [81].

The third group aims to emulate the natural phenomena such as the rain, the spiral, the wind, and the light. This group includes water cycle algorithm (WCA) [82], spiral optimization (SO) [83], and wind-driven optimization (WDO) [84]. Moreover, other methods belong to this group but emulate physical laws. For example, field of force (FOF) [85], electromagnetism algorithm [86], charged system search (CSS) [87], simulated annealing [88],

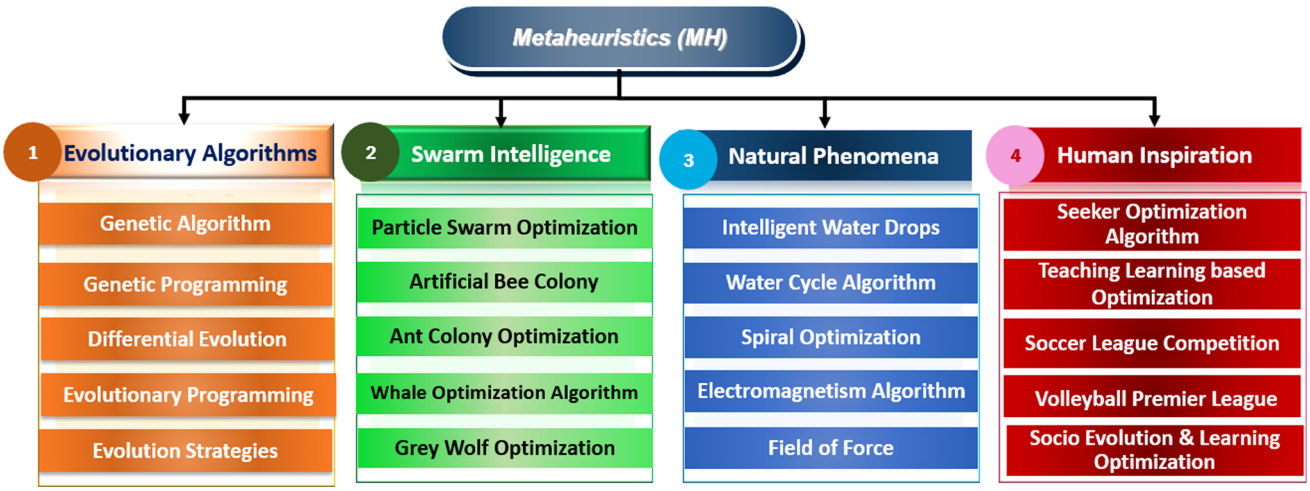

Fig. 1 The different categories of meta-heuristics

gravitational search algorithm (GSA) [89], Aquila Optimizer (AO) [90], low regime algorithm (fra), [91], electromagnetism-like mechanism [85], charged system search [87], optics inspired optimization (oio) [92], chemical-reaction-inspired metaheuristic [93, 94], and ant lion optimizer [94].

Also, the fourth group depends on inspiration the behaviors of the humans [95]. The following algorithms are sample belonging to this group: teaching learning-based optimization (TLBO) [96], volleyball premier league algorithm (VPL) [97], like seeker optimization algorithm (SOA) [98], soccer league competition (SLC) [99], and league championship algorithm (LCA) [100].

3 Evolutionary computing for DNNs

Algorithms employed to solve, search, optimize, and find optimal solutions for problems are called evolutionary computing (EC) [101]. The principles of natural genetics inspire common EC algorithms. This section will discuss the recent progress of research related to the evolution of deep learning paradigms, including their topology, hyper-parameters, and learning rules using EC algorithms. Non-convex and black-box optimization problems tend to benefit most from solutions that use EC algorithms where they can be used in various disciplines such as robotics [102], optimal control [103], control servo systems [104], robot path planning [105], body-worn sensors for human activity [106], games [107], and medicine [108]. The variation and selection operations are essential to differentiate the EC method from the other. For instance, evolution strategies (ESs), genetic algorithms (GAs), genetic programming (GP), differential evolution (DE), and estimation of distribution (EDA) algorithms are sub-families under EC.

3.1 Evolution using GA

Al-Hyari and Areibi [109] used GA to evolve CNNs by designing the following space exploration approach. The covered search space includes the number of convolutional layers, the number of filters for each convolutional layer, the number of fully connected layers, and each layer's size. Also, filter size, dropout rate, pooling size, and type were also included in the proposed chromosome structure. The proposed framework was evaluated using the MNIST dataset for handwritten digit classification, and they obtained an accuracy equal to 96.66%.

Lingxi et al. [110] proposed a genetic DCNN (GDCNN) that can automatically build deep CNN architectures. GDCNN model the network architectures as a hyper-parameters search and optimization problem involving an evolutionary-based approach. The authors proposed an encoding scheme to generate randomized network structure embeddings (individuals) using a fixed-length binary string (binary bit) to initialize the GA algorithm. An individual encodes only convolutional and pooling operations where the size of convolutional filters is fixed within each stage. The elimination of weak individuals is defined by the generated network classification accuracy on different datasets such as MNIST, CIFAR10, and ILSVRC2012. Such algorithms are computationally time-consuming since each individual's evaluation is performed via training-from-scratch on the selected dataset. Moreover, a limited number of layers, fixed filter size, and several filters per filter size can be seen as a limitation of this method. Felbinger [111] proposed an automated approach for CNN to detect the ball in the NAO robotic. The GA is used to find a network and enhance classification. The method showed superior performance and was implemented on the NAO RoboCup 2017.

Yanan et al. [112] used the skip connections concept introduced in ResNet to proposed GA-based system (CNN-GA) to build CNN architectures automatically. Their goal was to create deeper architectures that maximize the performance accuracy on image classification tasks by discovering the best depth instead of limiting the network's depth. The combination of skip layers (two convolutional layers and a skip connection) and pooling layers is encoded into each individual to represent a network architecture. Encoded individuals allow the network to deal with complex data by avoiding the gradient vanishing (GV) problems, reducing the search space, and minimizing the parameters. CIFAR10 and CIFAR100 were used to evaluate the performance of CNN-GA. Although CNN-GA achieved remarkable results on the well-known benchmark, CNN-GA cannot generate the arbitrary graph structure's CNN architectures.

Yanan et al. [113] developed an automatic design architecture of CNN using GA according to DenseNet and ResNet's blocks. This algorithm does not require pre-processing or post-processing in the process of designing CNN. To evaluate this algorithm, two datasets, namely CIFAR10 and CIFAR100, were used. Also, a set of 18 state-of-the-art methods are compared against it. Experimental results illustrate the high performance of the developed method overall other state-of-the-art hand-crafted CNNs according to various performance metrics, including classification accuracy and time complexity.

Baldominos et al. [114] developed a method for automatically designing CNNs according to GA and grammatical evolution. The developed model has been evaluated using the MNIST dataset for handwritten digit recognition. The results are compared with other models. The results showed the high ability of this model without using data augmentation.

Yanan et al. [115] developed the evolving deep CNN (evoCNN) method to overcome the limitations of HyperNEAT [116]. The authors used GA to search for the optimal CNN architecture and connection weight initialization values for image classification. The authors designed a variable-length chromosome encoding to encode CNN building blocks and depth. Also, they developed a representation scheme for CNN connection weights initialization to avoid networks getting stuck into local minima. EvoCNN has been compared with 22 existing algorithms on nine benchmark datasets (Fashion, the Rectangle, the Rectangle Images (RI), the Convex Sets (CS), the MNIST Basic (MB), the MNIST with Background Images (MBI), Random Background (MRB), Rotated Digits (MRD), and with RD plus Background Images (MRDBI)) in terms of classification error rate and CNN parameters.

Yang et al. [117] proposed an alternative pruning CNN's method using GA and multi-objective trade-offs among error, sparsity, and computation. The developed method was used to prune a pre-trained LeNet model. The evaluation has been performed using the MNIST dataset, which shows faster performance by 16 times and decreases 95.42% parameter size. Besides, by comparing the results of GA with state-of-the-art compression methods, GA outperforms most of them according to various performance measures.

David et al. [118] describe their experiment using deep learning to identify galaxies in the zone of avoidance (ZOA), a binary classification problem. They aimed to use an evolutionary algorithm to evolve the topology and configuration of CNN to identify galaxies in the Zone of Avoidance automatically. They use EA, similar to a DeepNEAT implementation, to search the topology and parameters of a CNN. The selected search space including the number of convolution, pooling, fully connected layers, kernel size and stride, pooling size and stride, fully connected layer size, and the learning rate.

Bohrer et al. [119] propose an adapted implementation of GA-based evolutionary technique (CoDeepNEAT) [120] to automate the tasks of topology and hyper-parameter selection. The authors used two types of layers to build their networks which are convolutional and dense layers. For the convolutional layer, hyper-parameter such as Filters, Kernel size, and Dropout was investigated. Besides, the number of units was investigated in dense layers. The evaluation has been conducted on two image classification datasets which are MNIST and CIFAR-10.

3.2 Evolution using GP

Agapitos et al. [121] used the greedy layer-wise training protocol, which is a GP-based system for multi-layer hierarchical feature construction and classification. The authors used two successive runs of GP to evolve two cascaded transformation layers rather than a single evolution stage.

Suganuma et al. [122] tried to solve the classification of the image by designing a CNN using the GP of kind Cartesian. Convolutional blocks and tensor concatenation represent the nodes of the network. He checked the proposed method performance using the dataset of CIFAR-10. The experimental result showed that the proposed method could automatically reach the competitive CNN architecture compared with the state-of-the-art models like VGG and ResNet.

Wong et al. [123] proposed adaptive grammar-based deep neuroevolution (ADAG-DNE) for the ventricular tachycardia signals classification. ADAG-DNE uses a grammar-based GP with a Bayesian network (BGBGP) to model probabilistic dependencies among the network modules. Authors compared using ADAG-DNE against

two neural network techniques named AlexNet, ResNet, and other seven popular classifiers, including Bayesian ANN classifier, least square SVM, ANN, pruned, and simple KNN, SVM, Random Forest, and Boosted-CART classifier. In terms of accuracy, ADAG-DNE is slightly better than AlexNet-A, AlexNet-B, and ResNet18. ADAG-DNE shows high efficiency in reducing the number of deaths caused by ventricular tachycardia and various heart diseases.

3.3 Evolution using DE

Zhang et al. in [124] proposed a multi-objective deep belief network (DBN) ensemble (MODBNE) method which employs a competitive multi-objective evolutionary algorithm, called MOEA/D [125] and a single-objective DE [126]. Several scalar optimizations have been composed as subproblems from a multiobjective optimization problem using MOEA/D. MOEA/D controls the number of hidden neurons per hidden layer, the weight cost, and learning rates used by the conventional DBN. The single-objective DE uses the mean training error to optimize the combination weights of an RBMs stacking ensemble.

Choi et al. [127] introduce a DE algorithm that controls each evolutionary process. Then, the algorithm assigns proper monitor parameters based on evolved of the individual. The CEC 2014 benchmark problems were used to demonstrate the performance; the proposed approach shows superior performance than the conventional DE algorithm.

Wang et al. [128] introduced crossover operator which is suggested in [128] to develop length CNN architectures, which is called DECNN. The suggested DECNN approach integrates three concepts. First, an established successful encoding method is optimized to accommodate for CNN architectures of variable length; Second, to optimize the hyper-parameters of CNNs, the novel mutation and crossover operators are established for variable-length DE; finally, the latest second crossover is implemented to improve the depth of CNN architectures. The proposed method is evaluated on six commonly used benchmark datasets. The findings are compared to 12 state-of-the-art techniques, which indicates that the proposed method is significantly efficient compared with state-of-the-art algorithms.

Peng et al. [129] presented DE-LSTM to evolve LSTM using DE algorithm. DE-LSTM is used to perform electricity price forecasting for balancing electricity generation and consumption. The method used to select and identify optimal hyper-parameters for LSTM where results indicate that DE outperforms PSO and GA in forecasting accuracy.

3.4 Evolution using FSM

Alejandro et al. [130] presented an automated way based on evolutionary approaches named EvoDeep for evolving both the architecture and the parameters of DNN. Authors used finite-state machine (FSM) as their evolutionary-based model design to generate valid networks to improve the level of classification accuracy and maintain a valid layers sequence. Several tuning parameters and network structure were included in the search space to generate DNN, including optimizer, several epochs, batch size, layer type, initialization function, activation function, and pooling other parameters. During their experiments, the MNIST dataset was used to evaluate EvoDeep. The model achieved a mean accuracy of 98.42% showing that the proposed algorithm improves the DNN architecture.

3.5 Evolution using evolution strategies (ESs)

Tirumala et al. [131] proposed a new approach for optimizing the DNN and improve the training process using EC. The proposed approach compared with various methods, including support vector machine (SVM), reinforcement, neuroevolution (NE), and GAs. However, this approach aims to decrease the DNN learning time by optimizing the DNNs. The system examines using the MNIST dataset (images dataset) and shows an improvement in the classification process's accuracy.

Ororbia et al. [132] expand EXALT [133] algorithm to propose a new algorithm EXAMM (Evolutionary eXploration of Augmenting Memory Models), which can evolve recurrent neural networks (RNNs). In particular, EXAMM refines EXALT's mutation operations to reduce hyper-parameters, and Δ-RNN, GRU, LSTM, MGU, and UGRNN cells were implemented to encode different memory structures rather than using simple neurons or LSTM cells only. EXAMM has been evaluated on large-scale, accurate world time-series data prediction from the aviation and power industries.

Badan and Sekanina [134] proposed EA4CNN, a framework employed to evolve CNNs using an evolutionary algorithm (EA) TinyDNN library for image classification concerning the classification error and CNN complexity. EA4CNN uses the following operations to build the EA: two-member tournament selection, crossover, and mutation. Also, a replacement algorithm that uses a simple speciation mechanism based on the CNN age. EA4CNN was used to design and optimize CNNs where each CNN is represented in the chromosome, including a variable-length list of layers, the age, and the learning rate. Moreover, two types of layers were studied: the Convolutional layer represented by the kernel size, number of

filters, stride size, and padding. Furthermore, the pooling layer is encoded with the stride size, subsampling type, and subsampling size. EA4CNN has been evaluated on two image classification datasets which are MNIST and CIFAR-10.

Irwin-Harris et al. [135] suggested an encoding strategy based on a directed acyclic graph representation and use this encoding to implement an algorithm for random generation of CNN architectures. Unlike previous research, the proposed encoding method is more specific, allowing representation of CNNs of unspecified connectional structure and infinite depth. It demonstrated the effectiveness through a random search, in which it evaluates 200 randomly generated CNN architectures. The 200 CNNs are trained by using 10% of CIFAR-10 training data to enhance computational efficiency; the three best-performing CNNs are then retrained on the entire training set. The findings demonstrate that given the random search method's simplicity and the reduced data set, the proposed representation and setup approach can obtain impressive efficiency compared to manually designed architectures.

Bakhshi et al. [136] proposed a genetic algorithm that can efficiently investigate a specified space of potential targets CNN architectures and optimize their hyper-parameters simultaneously for a given image processing function. Authors called this fast, automatic optimization model fast-CNN and used it to find efficient CNN image classification architectures on CIFAR10. In a set of simulation tests, it has been able to show that the network built by fast-CNN has obtained fair accuracy comparing with some other best network work existing models, but fast-CNN has taken much less time. Also, the trained model of the fast-CNN network generalized well to CIFAR100.

Baldominos et al. [137] enhanced previous research to optimize the topology of DNNs that can be employed to tackle the handwritten character recognition issue. Also, it takes advantage of evolutionary algorithms to optimize the population of candidate solutions by integrating a collection of the best-developed techniques resulting in a convolutional neural network committee. This method is strengthened by the use of unique strategies to protect population diversity. Consequently, it discusses one of the drawbacks of neuroevolution in this paper: The mechanism is very costly in terms of computational time. To tackle this problem, authors explore topology transfer learning efficiency: Whether the optimal topology achieved for a given domain utilizing neuroevolution could be adapted successfully to a different domain. In doing so, the costly neuroevolution method could be recycled to address various issues, transforming it into a more promising method to optimizing topology design for neural networks. Findings confirm that both the use of neuroevolved committees and topology transfer learning are successful: Convolutional neural network committees can optimize classification outcomes compared to a single model, and topologies learned for one issue could be repeated for another issue and data with good efficiency.

Benteng et al. [138] proposed the genetic DCNN (deep convolutional neural networks) designed to generate DCNN architectures for image classification automatically. The proposed framework uses refined evolutionary operations, including selection, mutation, and crossover, to search for the optimal DCNN architectures. Each individual in the DCNN architectures population encodes the following network parameters convolution, pooling, fully connection, batch normalization, activation, and dropout ratio as an integer vector. With the proposed DCNN architecture encoding scheme inspired by the representation of locus on a chromosome, DCNN architectures can be decomposed into a meta convolution block (p-arm) and a metadata-connected block (q-arm) where the VGG-19 model has been taken as the case study. The framework has been evaluated using MNIST, Fashion-MNIST, EMNIST-Digit, EMNIST-Letter, CIFAR10, and CIFAR100 datasets.

Fan et al. [108] proposed a novel method that implements neural architecture search (NAS) to improve a retinal vessel segmentation encoder–decoder architecture. An enhanced evolutionary technique is employed to develop encoder–decoder frame architectures with limited computational resources. The developed model achieved by the introduced approach achieved the best performance on the three datasets among all the comparative approaches, called DRIVE, STARE, and CHASE DB1, with selected criteria. Also, cross-training findings demonstrated that the proposed model improves scalability, which implies great potential for diagnosis of clinical disease. We conclude the evolutionary algorithms used to improve DNNs in Table 1, which illustrates the advantages and limitations of each model.

4 Swarm intelligence for DNNs

Natural or artificial entities represented in groups that can emerge collective intelligent behaviors are referred to as swarm intelligence (SI) [139]. The interaction of these entities with small groups or the environment can create global intelligent behavior. In the last few decades, many algorithms have been regularly proposed to simulate these intelligent behaviors, for instance, cuckoo search [140], firefly algorithms [140], grey wolf optimization (GWO) [141], particle swarm optimization (PSO) [142], ant colony optimization (ACO) [143], whale optimization algorithm [144], and squirrel search algorithm [145]. The following subsections summarize some of the key approaches of evolving DNNs by using SI algorithms.

Table 1 Evolutionary computing for DNNs summary

The EC algorithm	The EC algorithm mechanism	Advantages/disadvantages
General deep neural networks		
EvoDeeP finite-state machine (FSM) [130]	The search performed using EvoDeeP over the parameters space FSM used to determine the possible transitions between layers (multiple types)	*Pros* good accuracies has been achieved using the generated DNNs architectures *Cons* the generated DNNs needed high computational resources and training time
Two different types of co-evolutionary processes [131]	The competitive strategy used on population P1 to identify the fittest individual The cooperative strategy used on population P2 to construct an optimal solution and decrease the DNN learning time	*Pros* optimized DNNs were employed to reduce learning time and improve classification accuracy *Cons* hard to select the initial topology, weights, and biases to start with
Convolutional neural network		
GA [109]	A chromosome structure with two sub-parts for the convolutional layer parameters and the fully connected layer parameters, respectively. GA had its usual operators	*Pros* high accuracy on the MNIST dataset *Cons* limited set of CNN parameters and components were explored
GA [110]	A fixed-length binary string (binary bit) encoding scheme is used. GA had its usual operators	*Pros* good generalization of the generated structures on other tasks *Cons* computation time-consuming A limited and fixed number of explored parameters The network training process is performed separately
GA based on ResNet blocks and DenseNet blocks [113]	A variable-length encoding scheme is employed for the unpredictably optimal depth of the CNN A new crossover and mutation operators were used for local and global search Another encoding strategy was designed based on the ResNet and DenseNet blocks	*Pros* no users with the domain, problem, or GA knowledge are required It outperforms other state-of-the-art CNN with less time consumed *Cons* relative slow speed of the fitness evaluation
GA [112]	A new variable-length encoding strategy and crossover operator The incorporation of skip connections to avoid the vanishing gradient (VG) problems An asynchronous computational component was developed	*Pros* outperform the existing automatic CNN architecture with fewer parameter numbers and computational resources *Cons* slow speed for the fitness evaluation of CNNs
GA and grammatical evolution [114]	A new encoding scheme of the most relevant parameters of CNN Approximating the quality function with a reduced sample of the training A niching scheme was proposed to preserve the diversity of the population	*Pros* competitive results with the state-of-the-art without relying on data augmentation or preprocessing *Cons* fixed search space—memory and computation bottleneck
GA [115]	A new variable-length encoding scheme A new representation for weight initialization A slacked binary tournament selection	*Pros* outperform the state-of-the-art algorithms in terms of the classification error rate with less number of parameters (weights) *Cons* high computational resource and long time training on large-scale data
Darwinian biological evolution [118]	An EA similar to DeepNEAT algorithm	*Pros* highly specific CNNs to the task of galaxies classification and identifying Larger input images and all JHK passband *Cons* less resolution information was needed and human is still needed to verify the results
GA [119]	CoDeepNEAT algorithm	*Pros* only small population sizes and few generations were needed *Cons* computation time-consuming Network evaluations are narrowed to smaller search spaces
Cartesian genetic programming (CGP) [122]	Node functions modules employs convolutional blocks and tensor concatenation	*Pros* competitive CNN architectures compared with the state-of-the-art *Cons* high computational cost with the existence of redundant or less effective layers
Probabilistic Model Building Genetic Programming (PMBGP) [123]	A set of rules in Probabilistic Context-Sensitive Grammar (PCSG) used as a grammar to encode structural dependencies within DNN components	*Pros* better accuracy than AlexNet, ResNet, and seven non-neural network classifiers New forms of regularities were discovered and new traits were extracted *Cons* long training time of CNN Difficult to trace a prediction result back CNN model cannot handle missing feature values
DE [128]	IP-Based Encoding Strategy with new mutation and crossover operators for variable-length DE	*Pros* very competitive accuracy on the MBI and MRB datasets *Cons* not tested on large and more complex datasets

Table 1 (continued)

The EC algorithm	The EC algorithm mechanism	Advantages/disadvantages
EA [134]	TinyDNN library	*Pros* good trade-offs between the classification error and CNN complexity *Cons* limited search quality and computing resources
EA and Random search [135]	A directed acyclic graph representation encoding strategy	*Pros* proposed representation and initialization method can achieve promising accuracy with less computation time *Cons* large search space and limited training data
GA [136]	Elite selection, random selection, and breeding as genetic operations	*Pros* outperformed all manually designed models in terms of classification accuracy with fewer GPU days *Cons* lower performance compared to some semi-automatic and automatically designed models
GA and grammatical evolution [137]	Same NE algorithm from [114] Niching strategies, historical set, and hall-of-fame were used.	*Pros* ensembles of CNNs can outperform individual models and transferred to different problems *Cons* less adaptation to non-similar domains
Deep convolutional neural network		
GA [138]	An encoding scheme inspired by the representation of locus on a chromosome Redefined selection, crossover, and mutation operators	*Pros* comparable performance to the state of the art *Cons* very high computational and space complexity
Deep belief networks		
Multi-objective EA and DE [124]	MODBNE used to evolve multiple DBNs simultaneously Combination weights optimized using a single-objective DE	*Pros* good performance of MODBNE on accuracy and diversity *Cons* not tested on more or different conflicting objectives Low computational speed
*Recurrent neural networ*k		
DE [129]	DE had its usual operators	*Pros* outperform existing forecasting models *Cons* less generalization on new datasets
EXAMM [132]	EXALT algorithm with either simple neurons or LSTM cells Refined EXALT's mutation operations EXAMM uses islands [151] instead of a single steady-state population	*Pros* performant architectures *Cons* less reliability

4.1 Evolution using PSO

Khalifa et al. [146] proposed a method to optimize the connection weights of a CNN architecture for handwritten digit classification. PSO was used to optimize the last classification layer weights of a CNN with seven layers. Authors reported that the proposed method showed improved accuracy compared to a generic CNN that uses stochastic gradient descent (SGD) only.

Fei Ye [147] proposes an approach for automatically find the best network configuration such as hyper-parameters and network structure for the DNN. There is the combination of the gradient descent approach and PSO. The PSO algorithm is used to explore optimal network configurations; on the other hand, the gradient descent is employed to train the DNN classifier. Throughout the search process, the PSO algorithm is applied to explore optimal network configurations. The steepest gradient descent scheme is utilized for training the DNN classifier.

Qolomany et al. [148] investigated the usage of the PSO algorithm to optimize parameter settings of a DNN and predict the number of occupants and their locations from a Wi-Fi campus network. PSO is used to optimize the number of hidden layers and the number of neurons in each layer compared to the grid search method.

Wang et al. [149] proposed an alternative method to design the architecture of CNNs automatically using improved PSO. In this algorithm, the PSO is modified using three strategies; the first strategy applies an encoding technique inspired by computer networks that aim to encode the CNN layers is simple. In the second strategy, a disabled layer is prepared to hide some information about the particle's dimensions to learn the variable-length architecture of CNN. The third strategy aims to speed up learning the model on extensive data by randomly selecting partial datasets. The developed algorithm is evaluated by comparing it with 12 state-of-the-art image classification methods. According to the results, the proposed algorithm outperforms other algorithms based on the results of classification error.

Gustavo et al. [150] proposed a technique to overcome CNN's overfitting problem using PSO for image

classification. PSO was used to select the proper regularization parameter, which is the dropout ratio in CNNs. The loss function guided the dropout ratio selection as a fitness function over the validation set. Moreover, experiments were conducted and compared to other meta-heuristic optimization techniques, including bat algorithm (BA), Cuckoo Search (CS), and firefly algorithm (FA) as well as a standard dropout ratio and a dropout-less CNN. The proposed technique was evaluated using four benchmarks, including MNIST, CIFAR-10, Semeion Handwritten Digits dataset, and USPS dataset.

Wang et al. [152] suggested an efficient PSO scheme called EPSOCNN to develop CNN architectures influenced by the concept of transfer learning. EPSOCNN effectively reduces the computation cost by reducing the search space to a single block and using a tiny number of the training set to analyze CNNs during development. Moreover, EPSOCNN still keeps the accuracy rate stable by stacking the developed block several times to match the entire training dataset. The suggested EPSOCNN technique is tested on the CIFAR10 dataset and compared with 13 peer competitors like hand-crafted deep CNNs, trained by deep learning techniques developed by evolutionary approaches. It demonstrates promising results in terms of classification precision, number of parameters, and cost. Besides, the developed transferable CIFAR-10 block is transferred and tested on several datasets—CIFAR-100 and SVHN. It showed promising findings on both datasets, showing the transferability of the developed block. All the tests were conducted several times. From a statistical point of view, Student's t-test is employed to compare the developed technique with peer competitors.

Junior et al. [153] proposed a new scheme based on PSO, capable of rapid convergence compared to other evolutionary techniques, to automatically search for relevant deeply CNNs architectures for image classification tasks, called psoCNN. A novel strategy for direct encoding and a velocity operator was developed, enabling PSO use with CNNs to be optimized. Experimental results demonstrated that psoCNN could easily find successful CNN architectures that obtain comparable quality performance with state-of-the-art architectures.

Pinheiro et al. [154] proposed a new approach that evolves CNN (U-Net) using the SI algorithm for biomedical image segmentation (pulmonary carcinogenic nodules detection) in cancer diagnosis. U-net imaging process layers are maintained during the training phase, whereas the last classification layer was remodeled. Moreover, U-Net parameters were included in the search space of several test SI algorithms, including PSO. The study shows a competitive performance of the swarm-trained DNN with 25% faster training than the back-propagation model. PSO achieved the top performance among other SI algorithms on various tested metrics.

4.2 Evolution using ACO

Desell et al. [155] applied ACO to optimize deep RNN structure for general aviation flight data prediction (airspeed, altitude, and pitch). A forward information flow path is selected by the artificial ants over neurons to generate a fully connected topology having the input, hidden, and output layers. An evaluation process is performed after generating the neural network by integrating the paths selected by a pre-specified number of ants.

ElSaid et al. [156] used ACO to evolve an LSTM network for aircraft engine vibrations. The authors improved the analytical calculations prediction of engine vibration by developing the LSTM cell structure using ACO. In a recent work on the same problem, ElSaid et al. [157] used ACO to evolve LSTMs cells using artificial ants to produce the path through the input and the hidden layer connections, taking into consideration the previous cell output.

Edvinas and Wei [158] proposed a new neural architecture search (NAS) method named Deepswarm based on swarm intelligence. Deepswarm uses ACO to search for the optimal neural architecture. Furthermore, authors use local and global pheromone update rules to guarantee the balance between exploitation and exploration. In addition, they combined advanced neural architecture search with weight reusability to evolve CNNs. Deepswarm was tested on MNIST, Fashion-MNIST, and CIFAR-10 dataset and achieved 1.68%, 10.28%, and 11.31% error rates.

ElSaid et al. [159] proposed a novel neuroevolution scheme based on ACO, named ant swarm neuroevolution (ASNE), for specific optimization of RNN topologies. The technique chooses from various typical recurrent forms of cells such as Δ-RNN, GRU, LSTM, MGU, and UGRNN, as well as from recurrent connections that can cover several layers/or stages of time. It also investigates variants of the core algorithm to add an inductive bias that promotes the creation of sparser synaptic communication patterns. ASNE formulates various functions that guide the evolutionary structure of pheromone stimulation (which mimics L1 and L2 regularization in ML) and implement ant agents with specific tasks such as explorer ants that create the initial feed-forward network and social ants that choose nodes from the network connections. Moreover, the number of training backpropagation epochs has been reduced by using the Lamarckian weight initialization strategy. Findings show that ASNE's evolved sparser RNNs outperform conventional one- and two-layer architectures built using modern memory cells and other well-known methods such including NEAT and EXAMM [132].

4.3 Evolution using firefly algorithm

Sharaf et al. [160] proposed an automated tool to enable non-experts to develop their CNN framework without prior expertise in this area. The suggested technique encoded the skip connections, which reflect the standard CNN block to produce the problem's search space. An improved firefly algorithm was introduced by exploring the search space to find optimal CNN architecture. The suggested firefly was developed to reduce the algorithm's complexity based on a neighborhood attraction model. The CIFAR-10 and CIFAR-100 had been utilized for training and testing the proposed algorithm. The new scheme obtained high results and improved accuracy compared with the cutting-edge CNN architecture.

5 Hybrid MH methods

We summarized the advantages and disadvantages of the swarm techniques that have been introduced in this study to enhance DNNs in Table 2. Garrett et al. [161] explored novel evolved activation functions using a tree-based search space that outperforms rectified linear unit (ReLU). Exhaustive search, mutation, and crossover have been employed on a set of candidate activation functions. Each activation function was defined as a tree structure (unary and binary functions) and grouped in layers. These activation functions were evaluated using a wide residual network with depth 28 and widening factor 10 (WRN-28-10) trained on two well-known image classification benchmarks, CIFAR-10 and CIFAR-100. WRN-28-10 with ReLU activation function got 94% accuracy on CIFAR-10 and 73.3% on CIFAR-100, respectively. WRN-28-10 uses the activation function searched by their algorithm got 94.1% on CIFAR-10 and 73.9% on CIFAR-100.

Bin et al. [162] proposed a hybrid evolutionary computation (EC) method that evolves CNNs architecture automatically for the image classification task. Instead of traditionally connecting layers with each other, authors integrated shortcut connections in their method by proposing a new encoding strategy. CNN (DynamicNet[1]) architecture, the shortcut connections are encoded separately. The authors combined PSO and GA as their primary EC method (HGAPSO) to search and generate optimal CNNs to maximize network classification accuracy. HGAPSO has been evaluated on three CONVEX benchmark datasets, MB, and MDRBI, for 30 runs and one run on CIFAR-10. HGAPSO has been compared with 12 peer non-EC-based competitors and one EC-based competitor that uses a differential evolution approach [128].

Yanan et al. [163] proposed an approach named EUDNN (evolving unsupervised deep neural networks) for learning meaningful representation through evolving unsupervised DNN using EA (heuristically searches) to overcome the lack of labeled data for training which is often expensive to acquire. EUDNN aims to find the optimal architecture and the initial parameter values for a classification task. During the evolution search stage, a small set of labeled data ensures the learned representations are meaningful. Also, the authors proposed a gene encoding scheme addressing high-dimensional data with limited computational resources. EUDNN starts by searching for the optimal architecture, initial connection weights, and activation functions and then fine-tuning all parameter values in connection weights. The authors used MNIST and CIFAR-10 as benchmarks to assess the performance of EUDNN, where an error classification rate of 1.15% has been reported on the MNIST dataset.

Verbancsics and Harguess [164] investigated the construction of deeper architectures of DNNs applying HyperNEAT to an image classification task. HyperNEAT is used to generate the connectivity for a defined CNN. Then, the CNN is used to extract image features trained on a specific dataset. After that, extracted features are given to another machine-learning algorithm to calculate a fitness score given as feedback to hyperNEAT for tuning the connectivity. The authors conducted experiments on MNIST and BCCT200 dataset and compared results to other networks trained by using BP directly.

Adarsh et al. [165] proposed a hybrid evolutionary approach to recognize handwritten Devanagari numerals using CNN. GA was employed to develop the deep learning model (CNN). Firstly, they use L-BFGS to train an auto-encoder to extract image features where they fixed the auto-encoder parameters. Lastly, they train a fully connected layer by the features extracted by the auto-encoder where GA was integrated to get the best weights for the softmax classifier. The proposed approach has achieved around 96.09% recognition accuracy for handwritten Devanagari numerals.

Liu et al. [166] have been developed an alternative approach to find the optimal structure of CNNs. This approach is named advanced neural architecture search, and it depends on sequential model-based optimization (SMBO) strategy. According to [166], this model has some advantages over other models, such as its simple structure, so it trains faster, which will learn the surrogate quickly. The quality of the structures is predicted using a surrogate that was not used before. The authors also factorized the search domain into a product of smaller search domains, which leads to improving the searching process with more

[1] DynamicNet is implemented as one of the Python library models: https://pypi.org/project/convtt/.

Table 2 SI approaches summary

The EC algorithm	The EC algorithm mechanism	Advantages/disadvantages
General deep neural networks		
PSO and steepest gradient descent algorithm [147]	A real-number vector as the individuals of PSO A flexible method to select the number of classifiers (ensemble model)	*Pros* outperform the random approach in terms of the generalization performance *Cons* limited search space
PSO [148]	PSO had its usual operators	*Pros* efficient approach for tuning DNN parameters when compared to grid search *Cons* limited search space
Convolutional neural network		
PSO [149]	The IP address as a particle encoding scheme Disabled layer to attain a variable-length particle Evolved and evaluated on randomly picked partial datasets	*Pros* strong competitor to the state-of-art algorithms in terms of classification error Lower computational cost *Cons* poor generalization on other tasks
PSO [150]	PSO had its usual operators	Pros overcome overfitting problem Cons small search space and the DNN architecture was not evolved
PSO [152]	A single dense block of hyper-parameters was encoded A training subset used to learn a transferable block A proposed automatic and progressive process	*Pros* lower computation cost and competitive classification accuracy The ability to transfer the evolved block *Cons* not evaluated on large datasets and CNN block was not evolved
PSO [153]	A virtually no size limitation variable-length particles A novel velocity operator	*Pros* fast convergence when compared with other evolutionary approaches A good balance between speed and accuracy *Cons* high computational power
ACO [158]	Local and global pheromone update rules Heuristic information Progressive NAS with weight reusability	*Pros* balance between exploitation and exploration Competitive performance *Cons* evaluated on fewer well-known datasets
FA [160]	The skip connections encoded for CNN blocks A neighborhood attraction model	*Pros* significant results and higher accuracy *Cons* Block-QNN-S model performs better than the proposed model
PSO [154]	PSO had its usual operators	*Pros* competitive results with 25% faster than the back-propagation model and classical learning algorithms *Cons* CNN architecture was not evolved
Recurrent neural network		
ACO [155]	Ants select a forward propagating path through neurons randomly based on the pheromone on each path	*Pros* easily parallelizable and scalable approach and trained in fewer iterations lower computational cost *Cons* limited network depth and fewer training epochs
ACO [159]	Schemes for dynamically modifying the pheromone traces Ant agents with specialized roles Lamarckian strategy for weight initialization	*Pros* outperform traditional one, two-layer architectures, and well-known NE algorithms *Cons* poor performance of explorer ants on the mean and median cases - poor performance of combined explorer and social ants on the best cases
ACO [156]	Message passing interface (MPI) version of ACO	*Pros* lower prediction error *Cons* only "M1" LSTM cells were studied

blocks. The results showed that the developed model is efficient five times more than others, including reinforcement learning (RL) [167] in terms of the number of models evaluated, and eight times faster total compute. In addition, the model outperformed other state-of-the-art classification methods using two datasets, namely CIFAR-10 and ImageNet.

Martín et al. [168] proposed a hybrid procedure for optimizing the inference block of the CNN by utilizing and developing a recent meta-heuristic known as statistically driven coral reef optimization (SCRO) and then hybridizing it with a backpropagation algorithm to make a final fine-grained weights update. There is a need to implement a set of fully connected layers, which leads to an enormous size of the network. Hence, the authors considered the hybrid between the two algorithms. They tested the performance of the proposed method (HSCRO) on two datasets, CIFAR-10 and CINIC-10 datasets; HSCRO could optimize the reasoning block of the VGG-16 CNN architecture with a reduction of around 90% of the weights of the connections, which shows its superiority.

Aly et al. [169] proposed a technique for optimizing the deep neural network based on a recent meta-heuristic named multiple search neuroevolution (MSN). The authors divided the search space into several regions. They then searched in the regions simultaneously to keep enough

distance among them. They used the proposed technique for solving some global optimization functions. Then, train a CNN on the famous MNIST dataset. From the results, the proposed technique achieved high accuracy for the two tasks.

Peyrard and Eckle-Kohler [170] proposed a framework consisting of two summarizers for extractive multi-document summarization (MDS) based on GA artificial bee colony algorithm. The study's objective is to optimize an objective function (KL divergence and JS divergence) of input documents and a summary. The investigation yielded competitive results compared to commonly used summarization algorithms on DUC-02 and DUC-03 datasets.

6 Evaluate MH techniques combined with DNN for image classification

In this section, the results of different MH techniques applied to improve the DNN models' performance are collected from image classification applications. We focused on this application since it is one of the most popular applications used to assess the evolved DNNs. Other applications used to evaluate the evolved DNNs have been discussed in the previous sections.

Table 3 depicts the value of the accuracy obtained by different evolved DNNs using three commonly used datasets, including MNIST, IRIS, SVHN, CIFAR10, and CIFAR100.

For the MNIST, it can be noticed that the EvoDeeP framework that incorporates finite-state machine (FSM) [130] provides the highest accuracy overall the other methods. Followed by GA and grammatical evolution [114] and PSO [153] which allocate the second and third rank, respectively. However, there is no significant difference between these algorithms. For the CIFAR10, one can be observed that GA [112], sequential model-based optimization (SMBO) [166], and PSO [152] allocate the first, second, and third rank, respectively. Moreover, PSO [152], GA [112], and FA [160] achieved the best accuracies overall the other methods.

7 Problems and challenges

Notwithstanding the actual results of the reviewed researches still, some difficulties and challenges are facing DNNs and their architectures that require to be approached. To describe these challenges, the following points are given.

1. The determination of DNNs architectures: Until now, several various DNNs architectures have been utilized and employed in the literature to address hard problems. However, there is no evidence of why these structures have been chosen.
2. Absence of benchmarking and expected results: Some benchmark results are published in the literature, which is not enough. In these benchmarks, scholars have employed various deep architectures and analyzed decision trees' outcomes to provide the best training outcomes.
3. The loss of knowledge in any method that influences the determination of the entire system. This problem requires to be benchmarked to realize the best achievement.
4. Features extraction can be performed before feeding the data to DNNs via transfer learning. Then, the optimization algorithm can be selected and employed. This process aims to decrease the training and computational time.
5. Several parameters are associated with the training dataset. This can produce various errors during the training process and the error which can face in the current training dataset. Nevertheless, the DNNs model-based ANN's performance can be assisted by the capability to process the training and testing datasets.
6. Adjusting and tuning the optimization process is an essential procedure because any change in hyper-parameters influences the DNN models' effectiveness. Moreover, to trade with real-world problems by using DNNs solutions, powerful processing capability is wanted.

8 Future works

With the advancement achieved by applying developed state-of-the-art DL architectures to a variant set of applications, future directions must be addressed to overcome the problems encountered when using DL models and techniques. Dimensionality reduction is one of the most prevalent challenges needed to be handled since the number of the extracted features from DL techniques can be huge. This leads to degradation in the prediction model's performance (i.e., classifier) since most of these features are irrelevant and redundant. To handle this problem in the future, different metaheuristics can be combined with DL techniques. These MH methods aim to determine the relevant features from those extracted features and then pass the selected features to a classification algorithm or a fully connected layer in a DL architecture.

In addition, most of the presented DL techniques have been designed to deal with many samples (big data). However, few of them have been developed to handle

Table 3 Comparison between MH techniques used to evolve DNN models for image classification

The MH algorithm	MNIST	IRIS	CIFAR10	SVHN	CIFAR100
EvoDeeP finite-state machine (FSM) [130]	99.93	NA	NA	NA	NA
Two different types of co-evolutionary processes [131]	98.7	98.3	NA	NA	NA
GA [109]	96.66	NA	NA	NA	NA
GA [110]	99.66	NA	92.9	98.03	70.97
GA based on ResNet blocks and DenseNet blocks [113]	NA	NA	95.3	NA	77.6
GA [112]	NA	NA	96.78	NA	79.47
GA and grammatical evolution [114]	99.63	NA	NA	NA	NA
GA [115]	94.53	NA	NA	NA	NA
GA [119]	92	NA	77	NA	NA
Cartesian genetic programming (CGP) [122]	NA	NA	94.02	NA	NA
DE [128]	98.54	NA		NA	NA
EA [134]	99.36	NA	75.8	NA	NA
EA and Random search [135]	NA	NA	92.57	NA	NA
GA [136]	NA	NA	94.7	NA	75.63
GA and grammatical evolution [137]	99.73	NA	NA	NA	NA
GA [138]	99.64	NA	89.23	NA	66.77
PSO [149]	97	NA	NA	NA	NA
PSO [150]	99.12	NA	71.69	NA	NA
PSO [152]	NA	NA	96.42	81.44	98.16
PSO [153]	99.68	NA	NA	NA	NA
ACO [158]	99.54	NA	87.3	NA	NA
FA [160]	NA	NA	96	NA	77.75
Evolutionary optimization [161]	NA	NA	94.1	NA	74.3
GA-PSO [162]	NA	NA	95.63	NA	NA
Hypercube-based NeuroEvolution [164]	92.1	NA	NA	NA	NA
Genetic algorithm and L-BFGS [165]	96.41	NA	NA	NA	NA
Sequential model-based optimization (SMBO) [166]	NA	NA	96.59	NA	NA
ScheduledDropPath [167]	NA	NA	96.41	NA	NA
Statistically driven coral reef optimization algorithm [168]	NA	NA	93.48	NA	NA
Multiple search neuroevolution (MSN) [169]	90	NA	NA	NA	NA

training a model with few samples. This problem occurred when applied DL models to some applications that generate a small number of observations, including engineering, chemistry, bioinformatics, and medicine.

Although experts designed recent proposed state-of-the-art architectures. Few of these architectures have been explored by incorporating MH or NAS techniques such in YOLOv4 [171], FP-NAS [172], and auto-deep lab [173]. A huge range of architectures based on transformers of GANs suffer from the long training time, computation cost, a high number of training parameters, sequence length, and many more issues, which are still open for further optimization using MH and NAS techniques.

Finally, the evolutionary algorithms still need enhancement before applying them to the DL techniques. Since most of them have a high ability to explore or exploit the search space, it is a challenging task to find the EC that can balance between these two abilities. In addition, most of the MH techniques ranked top in CEC's competitions have not been applied to improve or optimize the DL design or parameters.

9 Conclusion

A deep neural network (DNN) is an artificial intelligence approach that has been utilized successfully in various applications. DNN is an artificial neural network (ANN) with multiple layers to determine the optimal output of a given input. The DNN structuring produces computational models that consist of many processing layers to learn data representations. Also, DNN is recognized as a sensible

technique to derive knowledge from the massive volume of data. Hence, DNN is one of the most promising research fields in machine learning and artificial intelligence domains. Based on the literature, some efforts have been made to realize how and why DNN achieves such essential results. Moreover, DNN has been employed in medical image analysis, classification rule, automatic speech recognition, electromyography recognition, graphics object detection, image recognition, drug discovery and toxicology, voice recognition, natural language processing, bioinformatics, financial fraud detection, and big data analytics.

The architecture of DNNs may contain several layers with variant types and depth. Each of these layers can extract useful information from the input data. Meanwhile, a wide range of DNNs, DNNs blocks, and training techniques made building DNNs models more difficult. Furthermore, various hyper-parameters to select to define how each layer will be trained added an extra layer of complexity to the model building or the architecture selection process. For example, a convolutional neural network can contain different hyper-parameters, including several convolutional layers, the number of kernels, kernel size, activation function, pooling operation, and pooling size. A dense layer can include the number of neurons, activation function, and layer type. Also, to train a DNN model, the number of epochs should be set alongside the batch size, initialization function, optimizer, and learning rate. From the previous literature, it can be concluded that determining the optimal configuration from those parameters is complex and can be considered an NP-hard problem.

In this paper, a comprehensive survey of existing metaheuristic optimization methods used to evolve DNNs in various research tasks, areas, and applications (such as in training DNNs fine-tuning DNNs, networks architecture search (NAS), and DNNs hyper-parameters search) is presented. We focused on recent SI and EC algorithms applied to DNNs, present their advantage and disadvantage, and find out some future research directions for connecting the gaps between optimization methods and DNNs.

It has been noticed from the comparison between the MH techniques that have been used to improve the performance of DNN models that the Evolutionary algorithms are wider used than swarm and hybrid techniques. However, the quality of the swarm techniques, such as the particle swarm algorithm (PSO), can provide results better than evolutionary algorithms. Moreover, the MH techniques are less applied to unsupervised learning-based DNNs. In most scenarios, MH techniques have been applied to image classification tasks. Thus, there is still room to apply these techniques in different areas and assess their performance on different and more challenging real-world datasets. Mainly DNNs require high computational resources and time. In this case, a few MH techniques have been explored to balance the computational resources, computation time, and performance.

Future aspects and challenges of this research include optimizing DNN structure using other recent optimization algorithms and modifying them by introducing more robust methods such as hybridizing them or adding new operators to these algorithms. These propositions can improve the optimization algorithms' ability to solve significant problems in DNNs and enhance their performance. Furthermore, DNNs can be employed in several other applications, such as military, financial fraud detection, mobile advertising, recommendation systems, drug discovery and toxicology, visual art processing, computer vision, and natural language processing.

Acknowledgements This work is supported by the Hubei Provincinal Science and Technology Major Project of China under Grant No. 2020AEA011 and the Key Research & Developement Plan of Hubei Province of China under Grant No. 2020BAB100.

Declaration

Conflict of interest The authors declare that they have no conflict of interest.

References

1. McCulloch Warren S, Walter P (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 5(4):115–133
2. Frank R (1958) The perceptron: a probabilistic model for information storage and organization in the brain. Psychol Rev 65(6):386
3. Rumelhart David E, Hinton Geoffrey E, Williams Ronald J (1985) Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science
4. Khotimah C, Purnami SW, Prastyo DD, Chosuvivatwong V, Spriplung H (1905) Additive survival least square support vector machines: a simulation study and its application to cervical cancer prediction. In: Proceedings of the 13th IMT-GT International Conference on Mathematics, Statistics and their Applications (ICMSA), AIP Conference Proceedings
5. Yann LC, Yoshua B, Geoffrey H (2015) Deep learning. Nature 521(7553):436–444
6. Geoffrey H (2018) Deep learning-a technology with the potential to transform health care. Jama 320(11):1101–1102
7. Krizhevsky A, Sutskever I, Hinton GE (2012). 2012 alexnet. Adv Neural Inf Process Syst, pp 1–9
8. Zhong Z, Jin L, Xie Z (2015) High performance offline handwritten chinese character recognition using googlenet and directional feature maps. In: 2015 13th International Conference on Document Analysis and Recognition (ICDAR), pp 846–850. IEEE
9. Targ S, Almeida D, Lyman K (2016) Resnet in resnet: Generalizing residual architectures. *arXiv preprint*arXiv: 1603.08029,

10. Iandola F, Moskewicz M, Karayev S, Girshick R, Darrell T, Keutzer K (2014) Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint*arXiv: 1404.1869
11. Mingxing T (1905) Le Quoc V (2019) Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv, Efficientnet, p 11946
12. Faruk E (2020) A novel clustering method built on random weight artificial neural networks and differential evolution. Soft Comput, pp 1–12
13. Laith A, Gandomi Amir H, Abd EM, Al HH, Mahmoud O, Mohammad A, Khasawneh Ahmad M (2021) Advances in metaheuristic optimization algorithms in big data text clustering. Electronics 10(2):101
14. Laith A, Gandomi Amir H, Abd EM, Hussien Abdelazim G, Khasawneh Ahmad M, Mohammad A, Houssein Essam H (2020) Nature-inspired optimization algorithms for text document clustering–a comprehensive analysis. Algorithms 13(12):345
15. Nigam A, Friederich P, Krenn M, Aspuru-Guzik A (2019) Augmenting genetic algorithms with deep neural networks for exploring the chemical space. arXiv preprint. arXiv: 1909.11655
16. Shukla P, Kumar H, Nandi GC (2020) Robotic grasp manipulation using evolutionary computing and deep reinforcement learning. arXiv preprint. arXiv: 2001.05443
17. Yuan Y, Fuchun S, Huaping L, Hongjiu Y (2014) Low-frequency robust control for singularly perturbed system. IET Control Theory Appl 9(2):203–210
18. Thomas W, Helmut B (2017) A mathematical theory of deep convolutional neural networks for feature extraction. IEEE Trans Inf Theory 64(3):1845–1866
19. van den Oord A, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior A, Koray K (2016). Wavenet A generative model for raw audio. arXiv preprint. arXiv: 1609.03499,
20. Kumar K, Kumar R, de Boissiere T, Gestin L, Zhen TW, Sotelo J, de Brébisson A, Bengio Y, Courville AC (2019) Melgan: Generative adversarial networks for conditional waveform synthesis. In: Advances in Neural Information Processing Systems, pp 14910–14921
21. Andrew H, Mark S, Grace C, Liang-Chieh C, Bo C, Mingxing T, Weijun W, Yukun Z, Ruoming P, Vijay V et al (2019) Searching for mobilenetv3. In: Proceedings of the IEEE International Conference on Computer Vision, pp 1314–1324
22. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint. arXiv: 1409.1556
23. Kaiming H, Xiangyu Z, Shaoqing R, Jian S (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
24. Xie S, Girshick Ross, Dollár Piotr, Tu Zhuowen , He Kaiming (2017) Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp 1492–1500
25. Foret P, Kleiner A, Mobahi H, Neyshabur B (2020) Sharpness-aware minimization for efficiently improving generalization. arXiv preprint. arXiv: 2010.01412
26. Szegedy C, Liu W , Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
27. Soria PX, Riba E, Sappa A (2020) Dense extreme inception network: Towards a robust cnn model for edge detection. In: The IEEE Winter Conference on Applications of Computer Vision, pp 1923–1932
28. Huang G, Liu Z, Van Der Maaten L, Weinberger Kilian Q (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708
29. Iandola Forrest N, Han Song, Moskewicz Matthew W, Ashraf Khalid, Dally William J , Keutzer Kurt (2016) Squeezenet: Alexnet-level accuracy with 50x fewer parameters and> 0.5 mb model size. arXiv preprint. arXiv: 1602.07360
30. Howard Andrew G, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint. arXiv: 1704.04861
31. Ma N, Zhang X, Zheng H-T, Sun J (2018) Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European conference on computer vision (ECCV), pp 116–131
32. Zhang X, Zhou X, Lin M, Sun J (2018) Shufflenet: an extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 6848–6856
33. Bochkovskiy A, Wang C-Y, Mark Liao H-Y (2020) Yolov4: Optimal speed and accuracy of object detection. arXiv preprint. arXiv: 2004.10934
34. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788
35. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg Alexander C (2016) Ssd: Single shot multibox detector. In: European conference on computer vision, pp 21–37. Springer
36. Lin T-Y, Goyal P, Girshick R, He K, Dollár P (2017) Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp 2980–2988
37. Merity S (2019) Single headed attention rnn: Stop thinking with your head. arXiv preprint.arXiv: 1911.11423
38. Kalchbrenner N, Elsen E, Simonyan K, Noury S, Casagrande N, Lockhart E, Stimberg F, van den Oord A, Dieleman S, Kavukcuoglu K (2018) Efficient neural audio synthesis. arXiv preprint. arXiv: 1802.08435
39. Melis G, Kočiskỳ T, Blunsom P (2019) Mogrifier lstm. arXiv preprint.arXiv: 1909.01792
40. Ning J, Jiaxian W, Xiang M, Ke Y, Yuchang M (2020) Multitask learning model based on multi-scale cnn and lstm for sentiment classification. IEEE Access 8:77060–77072
41. Ganapathy K (2020) A study of genetic algorithms for hyperparameter optimization of neural networks in machine translation. arXiv preprint. arXiv: 2009.08928
42. Jingzhao Z, Tianxing H, Suvrit S (1905) Jadbabaie A (2019) A theoretical justification for adaptivity. arXiv preprint arXiv, Why gradient clipping accelerates training, p 11881
43. Young T, Hazarika D, Poria S, Cambria E (2018) Recent trends in deep learning based natural language processing. IEEE Comput Intell Magazine 13(3):55–75
44. Kumar SA, Sandeep C, Kumar SD (2020) Sentimental short sentences classification by using cnn deep learning model with fine tuned word2vec. Proc Comput Sci 167:1139–1147
45. Ashish V, Noam S, Niki P, Jakob U, Llion J, Aidan NG, Łukasz K, Illia P (2017) Attention is all you need. In: Advances in neural information processing systems, pp 5998–6008
46. Shoeybi M, Patwary M, Puri R , LeGresley P, Casper J, Catanzaro B (2019) Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. arXiv preprint. arXiv: 1909.08053
47. Devlin J, Chang M-W, Lee K, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint. arXiv: 1810.04805

48. Liu Y, Ott M , Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V (2019) Roberta: a robustly optimized bert pretraining approach. arXiv preprint. arXiv: 1907.11692
49. Sanh V, Debut L, Chaumond J, Wolf T (2019) Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint. arXiv: 1910.01108
50. Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, Stoyanov V, Zettlemoyer L (2019) Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint. arXiv: 1910.13461
51. Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Zhou Y., Li W., Liu Peter J (2019) Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint. arXiv: 1910.10683
52. Nikita K, Łukasz K (2001) Levskaya Anselm (2020). The efficient transformer. arXiv preprint arXiv, Reformer, p 04451
53. Brown Tom B, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A et al (2020) Language models are few-shot learners. arXiv preprint. arXiv: 2005.14165
54. Shin H-C, Zhang Y, Bakhturina E, Puri R, Patwary M, Shoeybi M, Mani R (2020) Biomegatron: Larger biomedical domain language model. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 4700–4706
55. Jimmy L, Rodrigo N (2010) Yates Andrew (2020). Bert and beyond. arXiv preprint arXiv, Pretrained transformers for text ranking, p 06467
56. Liu X, Duh K, Liu L, Gao J (2020) Very deep transformers for neural machine translation. arXiv preprint. arXiv: 2008.07772
57. Siddhant G, Thuy V, Alessandro M (2020) Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection. Proc AAAI Conf Artif Intell 34:7780–7788
58. Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507
59. Goodfellow I , Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, pp 2672–2680
60. Karras T, Laine S, Aittala M, Hellsten J, Lehtinen J, Aila T (2020) Analyzing and improving the image quality of stylegan. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 8110–8119
61. Vahdat A, Kautz J (2020) Nvae. A deep hierarchical variational autoencoder. arXiv preprint. arXiv:2007.03898
62. Guo J, Lu S, Cai H, Zhang W, Yu Y, Wang J (2018) Long text generation via adversarial training with leaked information. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 32
63. Karras T, Laine S, Aila T (2019) A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4401–4410
64. Schonfeld E, Schiele B, Khoreva A (2020) A u-net based discriminator for generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 8207–8216
65. Ronneberger OF, Philipp BT (2015) U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, pp 234–241. Springer
66. Abualigah L, Diabat A (2021) Advances in sine cosine algorithm: a comprehensive survey. Artificial Intelligence Review, pp 1–42
67. Abualigah L, Diabat A (2020) A comprehensive survey of the grasshopper optimization algorithm: results, variants, and applications. Neural Computing and Applications, pp 1–24
68. Laith A, Ali D, Zong WG (2020) A comprehensive survey of the harmony search algorithm in clustering applications. Appl Sci 10(11):3827
69. Abualigah L (2020) Group search optimizer: a nature-inspired meta-heuristic optimization algorithm with its results, variants, and applications. Neural Computing and Applications, pp 1–24
70. Abualigah L (2020) Multi-verse optimizer algorithm: a comprehensive survey of its results, variants, and applications. Neural Computing Applications, pp 1–21
71. Abualigah L, Diabat A, Mirjalili S, Abd Elaziz M, Gandomi AH (2021) The arithmetic optimization algorithm. Comput Methods Appl Mech Eng
72. Xin Y, Yong L, Guangming L (1999) Evolutionary programming made faster. IEEE Trans Evolut Comput 3(2):82–102
73. John H (1975) Adaptation in artificial and natural systems. The University of Michigan Press, Ann Arbor
74. Goldberg David E, Holland John H (1988) Genetic algorithms and machine learning. Mach Learn 3(2):95–99
75. Michalewicz Z (1996) Evolution strategies and other methods . In: Genetic Algorithms+ Data Structures= Evolution Programs, pp 159–177. Springer
76. Rainer S, Kenneth P (1997) Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359
77. Koza John R (1994) Genetic programming as a means for programming computers by natural selection. Stat Comput 4(2):87–112
78. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995. MHS'95., pp 39–43. IEEE,
79. Seyedali M, Gandomi Amir H, Zahra MS, Shahrzad S, Hossam F, Mohammad MS (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191
80. Faramarzi A, Heidarinejad M, Mirjalili S, Gandomi AH (2020) Marine predators algorithm: A nature-inspired metaheuristic. Exp Syst Appl, pp 113377,
81. Seyedali M, Andrew L (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
82. Hadi E, Ali S, Ardeshir B, Mohd H (2012) Water cycle algorithm-a novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput Struct 110:151–166
83. Kenichi T, Keiichiro Y (2011) Primary study of spiral dynamics inspired optimization. IEEJ Trans Electrical Electron Eng 6(S1):S98–S100
84. Bayraktar Z, Komurcu M, Werner DH (2010) Wind driven optimization (wdo): a novel nature-inspired optimization algorithm and its application to electromagnetics. In: Antennas and Propagation Society International Symposium (APSURSI), 2010 IEEE, pp 1–4. IEEE
85. Ali K (2014) Advances in metaheuristic algorithms for optimal design of structures. Springer, Berlin
86. İlker BŞ, Shu-Chering F (2003) An electromagnetism-like mechanism for global optimization. J Global Optim 25(3):263–282
87. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. Acta Mech 213(3–4):267–289
88. Scott K, Daniel Gelatt C, Mario PV (1983) Optimization by simulated annealing. Science 220(4598):671–680
89. Esmat R, Hossein N-P, Saeid S (2009) Gsa: a gravitational search algorithm. Inf Sci 179(13):2232–2248

90. Abualigah L, Yousri D, Abd Elaziz M, Ewees AA, Al-qanes, MA, Gandomi AH (2021) Aquila Optimizer: A novel meta-heuristic optimization Algorithm. Comput Indus Eng. https://doi.org/10.1016/j.cie.2021.107250
91. Tahani M, Babayan N (2018) Flow regime algorithm (fra): a physics-based meta-heuristics algorithm. Knowledge and Information Systems, pp 1–38
92. Ali Husseinzadeh Kashan (2015) A new metaheuristic for optimization: optics inspired optimization (oio). Comput Operat Res 55:99–125
93. Lam Albert YS, Li Victor OK (2010) Chemical-reaction-inspired metaheuristic for optimization. IEEE Trans Evolut Comput 14(3):381–399
94. Laith A, Mohammad S, Mohammad A, Seyedali M, Abd EM (2020) Ant lion optimizer: A comprehensive survey of its variants and applications. Arch. Comput, Methods Eng
95. Abualigah L, Abd EM, Hussien AG, Alsalibi B, Jafar J, Seyed M, Gandomi AH (2020) Lightning search algorithm: a comprehensive survey. Appl Intell, pp 1–24
96. Ravipudi VR, Vimal JS, Vakharia DP (2011) Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43(3):303–315
97. Reza M, Khodakaram S (2018) Volleyball premier league algorithm. Appl Soft Comput 64:161–185
98. Dai C, Zhu Y, Chen W (2006) Seeker optimization algorithm. In: International Conference on Computational and Information Science, pp 167–176. Springer
99. Naser Moosavian and Babak Kasaee Roodsari (2014) Soccer league competition algorithm: a novel meta-heuristic algorithm for optimal design of water distribution networks. Swarm Evolut Comput 17:14–24
100. Kashan AH (2009) League championship algorithm: a new algorithm for numerical function optimization. In: 2009 International Conference of Soft Computing and Pattern Recognition, pp 43–48. IEEE
101. Fogel DB (1995) Phenotypes, genotypes, and operators in evolutionary computation. In Proceedings 1995 IEEE Int. Conf. Evolutionary Computation (ICEC'95), pp 193–198
102. Kriegman S, Cheney N, Corucci F, Bongard JC (2017) A minimal developmental model can increase evolvability in soft robots. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp 131–138
103. Parker A, Nitschke G (2017) Autonomous intersection driving with neuro-evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp 133–134
104. Radu-Emil P, Radu-Codrut D (2019) Nature-inspired optimization algorithms for fuzzy controlled servo systems. Butterworth-Heinemann, Oxford
105. Constantin P, Radu-Emil P, Daniel I, Lucian-Ovidiu F, Radu-Codrut D, Florin D (2013) Optimal robot path planning using gravitational search algorithm. Int J Artif Intell 10(S13):1–20
106. Kaya Y, Faruk Ertugru O (2017) Determining the optimal number of body-worn sensors for human activity recognition. Soft Comput 21(17):5053–5060
107. Justesen N, Risi S (2017) Continual online evolutionary planning for in-game build order adaptation in starcraft. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp 187–194
108. Fan Z , Wei J, Zhu G, Mo J, Li W (2020) Evolutionary neural architecture search for retinal vessel segmentation. arXiv, pages arXiv–2001
109. Abeer A-H, Shawki A (2017) Design space exploration of convolutional neural networks based on evolutionary algorithms. J Comput Vis Imag Syst, vol 3, no 1
110. Xie L, Yuille A (2017) Genetic cnn. In: Proceedings of the IEEE international conference on computer vision, pp 1379–1388
111. Felbinger GC (2017) A genetic approach to design convolutional neural networks for the purpose of a ball detection on the nao robotic system. Project Work
112. Yanan S, Bing X, Mengjie Z, Gary GY, Jiancheng L (2020) Automatically designing cnn architectures using the genetic algorithm for image classification. IEEE Trans Cybern
113. Sun Y, Xue B, Zhang M, Yen GG (2018) Automatically evolving cnn architectures based on blocks. arXiv preprint. arXiv: 1810.11875
114. Alejandro B, Yago S, Pedro I (2018) Evolutionary convolutional neural networks: an application to handwriting recognition. Neurocomputing 283:38–52
115. Sun Y, Xue B, Zhang M, Yen GG (2019) Evolving deep convolutional neural networks for image classification. IEEE Trans Evolut Comput
116. Stanley Kenneth O, D'Ambrosio David B, Jason G (2009) A hypercube-based encoding for evolving large-scale neural networks. Artificial Life 15(2):185–212
117. Yang C, An Z, Li C, Diao B, Xu Y (2019) Multi-objective pruning for cnns using genetic algorithm. In: International Conference on Artificial Neural Networks, pp 299–305. Springer
118. Jones D, Schroeder A, Nitschke G (2019) Evolutionary deep learning to identify galaxies in the zone of avoidance. arXiv preprint. arXiv: 1903.07461
119. da Silveira BJ, Iochins GB, Dorn M (2020) Neuroevolution of neural network architectures using codeepneat and keras. arXiv preprint. arXiv: 2002.04634
120. Miikkulainen R, Liang J , Meyerson E, Rawal A, Fink D, Francon O, Raju B, Shahrzad H, Navruzyan Arshak,Duffy Nigel , et al (2019). Evolving deep neural networks. In: Artificial Intelligence in the Age of Neural Networks and Brain Computing, pp 293–312. Elsevier
121. Agapitos A, O'Neill M, Nicolau M, Fagan D, Kattan A, Brabazon A, Curran K (2015) Deep evolution of image representations for handwritten digit recognition. In: 2015 IEEE Congress on Evolutionary Computation (CEC), pp 2452–2459. IEEE
122. Masanori S, Shinichi S, Tomoharu N (2017) A genetic programming approach to designing convolutional neural network architectures. In: Proceedings of the genetic and evolutionary computation conference, pp 497–504
123. Pak-Kan W, Kwong-Sak L, Man-Leung W (2019) Probabilistic grammar-based neuroevolution for physiological signal classification of ventricular tachycardia. Expert Syst Appl 135:237–248
124. Chong Z, Pin L, Kai Qin A, Kay Chen T (2016) Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. IEEE Trans Neural Networks Learn Syst 28(10):2306–2318
125. Anupam T, Dipti S, Krishnendu S, Abhiroop G (2016) A survey of multiobjective evolutionary algorithms based on decomposition. IEEE Trans Evolut Comput 21(3):440–462
126. Swagatam D, Sankha SM, Ponnuthurai NS (2016) Recent advances in differential evolution-an updated survey. Swarm Evolut Comput 27:1–30
127. Tae JC, Chang WA (2017) An improved differential evolution algorithm and its application to large-scale artificial neural networks. In: Journal of Physics: Conference Series, vol 806, p 012010. IOP Publishing
128. Bin W, Yanan S, Bing X, Mengjie Z (2018) A hybrid differential evolution approach to designing deep convolutional neural networks for image classification. In: Australasian Joint Conference on Artificial Intelligence, pp 237–250. Springer

129. Peng L, Shan L, Rui L, Lin W (2018) Effective long short-term memory with differential evolution algorithm for electricity price prediction. Energy 162:1301–1314
130. Alejandro M, Raúl L-C, Félix F-H, Valery N, David C (2018) Evodeep: a new evolutionary approach for automatic deep neural networks parametrisation. J Parallel Distribut Comput 117:180–191
131. Tirumala S S, Ali S, Ramesh C P (2016) Evolving deep neural networks: A new prospect. In: 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), pp 69–74. IEEE
132. Alexander O, AbdElRahman E, Travis D (2019) Investigating recurrent neural network memory structures using neuro-evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp 446–455
133. ElSaid A,Benson S, Patwardhan S, Stadem D, Desell T (2019) Evolving recurrent neural networks for time series data prediction of coal plant parameters. In: International Conference on the Applications of Evolutionary Computation (Part of EvoStar), pp 488–503. Springer
134. Badan F, Sekanina L (2019) Optimizing convolutional neural networks for embedded systems by means of neuroevolution. In: International Conference on Theory and Practice of Natural Computing, pp 109–121. Springer
135. Irwin-Harris W, Sun Y, Xue B, Zhang M (2019) A graph-based encoding for evolutionary convolutional neural network architecture design. In: 2019 IEEE Congress on Evolutionary Computation (CEC), pp 546–553. IEEE
136. Bakhshi A, Noman N, Chen Z, Zamani M, Chalup S (2019)Fast automatic optimisation of cnn architectures for image classification using genetic algorithm. In: 2019 IEEE Congress on Evolutionary Computation (CEC), pp 1283–1290. IEEE
137. Alejandro B, Yago S, Isasi P (2019) Hybridizing evolutionary computation and deep neural networks: an approach to handwriting recognition using committees and transfer learning. Complexity
138. Benteng M, Xiang L, Yong X, Yanning Z (2020) Autonomous deep learning: a genetic dcnn designer for image classification. Neurocomputing 379:152–161
139. Corne DW , Reynolds AP, Bonabeau E (2012) Swarm intelligence
140. Yang X-S (2010) Nature-inspired metaheuristic algorithms. Luniver press
141. Hossam F, Ibrahim A, Al-Betar MA, Mirjalili S (2018) Grey wolf optimizer: a review of recent variants and applications. Neural Comput Appl 30(2):413–435
142. Bonyadi M, Reza MZ (2017) Particle swarm optimization for single objective continuous space problems: a review
143. Dorigo M, Stützle T (2019) Ant colony optimization: overview and recent advances. In: Handbook of metaheuristics, pp 311–351. Springer
144. Farhad Soleimanian Gharehchopogh and Hojjat Gholizadeh (2019) A comprehensive survey: whale optimization algorithm and its applications. Swarm Evolut Comput 48:1–24
145. Mohit J, Vijander S, Asha R (2019) A novel nature-inspired algorithm for optimization: squirrel search algorithm. Swarm Evolut Comput 44:148–175
146. Khalifa MH, Ammar M, Ouarda W, Alimi AM (2017) Particle swarm optimization for deep learning of convolution neural network. In: 2017 Sudan Conference on Computer Science and Information Technology (SCCSIT), pp 1–5. IEEE
147. Fei Y (2017) Particle swarm optimization-based automatic parameter selection for deep neural networks and its applications in large-scale and high-dimensional data. PloS ONE 12(12):e0188746
148. Qolomany B, Maabreh M, Al-Fuqaha A, Gupta A, Benhaddou D (2017) Parameters optimization of deep learning models using particle swarm optimization. In: 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), pp 1285–1290. IEEE
149. Wang B, Sun Y, Xue B, Zhang M (2018) Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification. In: 2018 IEEE Congress on Evolutionary Computation (CEC), pp 1–8. IEEE
150. De Rosa GH, Papa João P, Xin-S Y (2018) Handling dropout probability estimation in convolution neural networks using meta-heuristics. Soft Comput 22(18):6147–6156
151. Enrique A, Marco T (2002) Parallelism and evolutionary algorithms. IEEE Trans Evolut Comput 6(5):443–462
152. Bin W, Bing X , Mengjie Z (2019) Particle swarm optimisation for evolving deep neural networks for image classification by evolving and stacking transferable blocks. arXiv preprint. arXiv: 1907.12659
153. Junior FEF, Yen GG (2019) Particle swarm optimization of deep neural networks architectures for image classification. Swarm Evolut Comput 49:62–74
154. de Pinho P, Cesar A, Nedjah N, de Macedo ML (2020) Detection and classification of pulmonary nodules using deep learning and swarm intelligence. Multimedia Tools Appl 79(21):15437–15465
155. Desell T, Clachar S, Higgins J, Wild B (2015) Evolving deep recurrent neural networks using ant colony optimization. In: European Conference on Evolutionary Computation in Combinatorial Optimization, pp 86–98. Springer
156. ElSaid A, Wild B, Jamiy FE, Higgins J, Desell T (2017) Optimizing lstm rnns using aco to predict turbine engine vibration. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp 21–22
157. ElSaid A, Jamiy FE, Higgins J, Wild B, Desell T (2018) Using ant colony optimization to optimize long short-term memory recurrent neural networks. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp 13–20
158. Byla E, Pang W (2019). Deepswarm: Optimising convolutional neural networks using swarm intelligence. In: UK Workshop on Computational Intelligence, pp 119–130. Springer
159. ElSaid AA , Ororbia Alexander G, Desell Travis J (2019) The ant swarm neuro-evolution procedure for optimizing recurrent networks. arXiv preprint. arXiv:1909.11849
160. Sharaf AI , Radwan E-SF (2020) 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, pp 99–118. Springer
161. Bingham G, Macke W, Miikkulainen R (2020) Evolutionary optimization of deep learning activation functions. arXiv preprint. arXiv: 2002.07224
162. Wang B, Sun Y, Xue B, Zhang M (2019). A hybrid ga-pso method for evolving architecture and short connections of deep convolutional neural networks. In: Pacific Rim International Conference on Artificial Intelligence, pp 650–663. Springer
163. Yanan S, Yen Gary G, Zhang Y (2018) Evolving unsupervised deep neural networks for learning meaningful representations. IEEE Trans Evolut Comput 23(1):89–103
164. Verbancsics P , Harguess J (2015) Image classification using generative neuro evolution for deep learning. In: 2015 IEEE winter conference on applications of computer vision, pp 488–493. IEEE
165. Adarsh T, Siddhant S, Apoorva M, Anupam S, Ritu T (2018) Hybrid evolutionary approach for Devanagari handwritten numeral recognition using convolutional neural network. Procedia Comput Sci 125:525–532

166. Chenxi L, Barret Z, Maxim N, Jonathon S, Wei H, Li-Jia L, Li F-F, Alan Y, Jonathan H, Kevin M (2018) Progressive neural architecture search. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 19–34
167. Zoph B, Vasudevan V, Shlens J, Le QV (2018) Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 8697–8710
168. Alejandro M, Manuel VV, Antonio GP, David C, César H-M (2020) Optimising convolutional neural networks using a hybrid statistically-driven coral reef optimisation algorithm. Appl Soft Comput 90:106144
169. Aly A, Weikersdorfer D, Delaunay C (2019) Optimizing deep neural networks with multiple search neuroevolution. arXiv preprint. arXiv: 1901.05988
170. Peyrard M, Eckle-Kohler J (2016) A general optimization framework for multi-document summarization using genetic algorithms and swarm intelligence. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pp 247–257
171. Chen S-L, Lin S-C, Huang Y, Jen C-W, Lin Z-L, Su S-F (2020) A vision-based dual-axis positioning system with yolov4 and improved genetic algorithms. In: 2020 Fourth IEEE International Conference on Robotic Computing (IRC), pp 127–134. IEEE
172. Zhicheng Y, Xiaoliang D, Peizhao Z, Tian Yuandong W, Bichen, (2011) Feiszli Matt (2020). Fast probabilistic neural architecture search. arXiv preprint arXiv, Fp-nas, p 10949
173. Liu C, Chen L-C, Schroff F, Adam H, Hua W, Yuille W, Li F-F (2019) Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 82–92