Artificial Intelligence Review (2022) 55:829–894
https://doi.org/10.1007/s10462-021-09992-0

A comprehensive survey on optimizing deep learning models by metaheuristics

Bahriye Akay[1] · Dervis Karaboga[1,2] · Rustu Akay[3]

Published online: 31 March 2021

Abstract

Deep neural networks (DNNs), which are extensions of artificial neural networks, can learn higher levels of feature hierarchy established by lower level features by transforming the raw feature space to another complex feature space. Although deep networks are successful in a wide range of problems in different fields, there are some issues affecting their overall performance such as selecting appropriate values for model parameters, deciding the optimal architecture and feature representation and determining optimal weight and bias values. Recently, metaheuristic algorithms have been proposed to automate these tasks. This survey gives brief information about common basic DNN architectures including convolutional neural networks, unsupervised pre-trained models, recurrent neural networks and recursive neural networks. We formulate the optimization problems in DNN design such as architecture optimization, hyper-parameter optimization, training and feature representation level optimization. The encoding schemes used in metaheuristics to represent the network architectures are categorized. The evolutionary and selection operators, and also speed-up methods are summarized, and the main approaches to validate the results of networks designed by metaheuristics are provided. Moreover, we group the studies on the metaheuristics for deep neural networks based on the problem type considered and present the datasets mostly used in the studies for the readers. We discuss about the pros and cons of utilizing metaheuristics in deep learning field and give some future directions for connecting the metaheuristics and deep learning. To the best of our knowledge, this is the most comprehensive survey about metaheuristics used in deep learning field.

Keywords Deep neural networks · Metaheuristics · Training · Hyper-parameter optimization · Architecture optimization · Feature extraction

✉ Bahriye Akay
bahriye@erciyes.edu.tr

Dervis Karaboga
karaboga@erciyes.edu.tr

Rustu Akay
akay@erciyes.edu.tr

[1] Department of Computer Engineering, Erciyes University, 38039 Melikgazi, Kayseri, Turkey

[2] King Abdulaziz University, Jeddah, Saudi Arabia

[3] Department of Mechatronics Engineering, Erciyes University, 38039 Melikgazi, Kayseri, Turkey

1 Introduction

Artificial Intelligence (AI) algorithms can learn feature hierarchies and generalize them to new contexts, and automatically learning features at multiple levels of abstraction provides to learn complex mappings. Shallow learning algorithms extract features using artificial sampling or empirical sampling. As the data size increases, the abilities of shallow techniques turn to be insufficient on large-scale and high-dimensional data. More advanced and competent AI techniques are needed to extract features at high-level abstractions and learn complex mappings in large-scale, incompletely annotated data. One of the advanced representation learning techniques that can address these issues is deep learning (DL) (Hinton et al. 2006), which is inspired by brain activities performed in visual cortex to establish the non-linear relationships. DNNs can learn higher levels of feature hierarchy by transforming the raw feature space into another feature space. Although they had been proposed earlier, their applicability had remained limited because they require high computation budget. However, growth in computational power enabled them to be employed in many studies, recently.

A DL architecture has several specific layers corresponding to a different area of cortex. Each layer has an arbitrary number of neurons and outputs, as well as distinct initialization methods, and activation functions. The values assigned for model and learning parameters, the way of the feature representation described and the weight and bias values determined affect the overall performance of DL methods while there is a trade-off between generalization capability and computational complexity. Although Grid Search, Random Search and Bayesian Optimization (Bergstra et al. 2011) are popular to configure hyper-parameters, they are impractical when the number of parameters and the complexity of the problem is high. Besides, there is no analytic approach to automatically design the optimal architecture, designing manually or using exhausted search requires high computational cost even if high-processing facilities such as GPU and parallel computing are used. In training of DL models, using derivative-based methods is difficult to parallelize and causes to slow convergence. Recently, researchers have proposed new studies to automate the search for their design and parameters. Among these studies, neuro-evolution applies evolutionary computation to explore the huge search space and mitigate the challenges of these approaches. Nature-inspired evolutionary metaheuristics that combine a natural phenomenon and randomness can deal with dynamic changes in the problem space by transferring the problem-specific knowledge from previous generations. They can produce high-quality near-optimal solutions for large-scale problems within an acceptable time.

Tian and Fong (2016) reviewed genetic algorithm (GA) and particle swarm optimization (PSO), for traditional neural network's training and parameter optimization. Fong et al. (2017) reviewed the applications of metaheuristics in DL. Gülcü and Kuş (2019) reviewed the metaheuristic methods used to optimize hyper-parameters in Convolutional Neural Networks (CNNs). Chiroma et al. (2019) pointed out recent development issues and created a taxonomy based on nature-inspired algorithms for deep learning. These surveys report the advances in the area focusing on only some limited aspects of DL models (training or hyper-parameter optimization) or some architectures. Therefore, we have prepared a more comprehensive review considering most aspects of DL that can be treated as an optimization problem and solved by metaheuristics. As far as we know, this is the most comprehensive review about metaheuristics used in DL field. We believe that this review would be very beneficial for the researchers who prepares to study on the hybridization of metaheuristics and DL approaches.

Our motivation is to provide a review about the metaheuristic algorithms used for the optimization problems arising in deep learning field and answer the research questions listed below:

- Which optimization problems do arise in the field of deep learning?
- Which metaheuristic algorithms are used to optimize DNNs?
- Which encodings can be used to map solution space to network space?
- Which problems are solved by DNNs optimized by metaheuristics?
- Which DNN architectures are optimized by metaheuristics?
- How we can validate the results of DNNs optimized by metaheuristics?
- Which datasets are mostly used in the benchmarks for DNN optimization by metaheuristics?

To answer the research questions, first, we give a brief information about common DNN architectures widely used in engineering field and optimized by metaheuristics. Second, we group the metaheuristic algorithms based on the number of solutions evolved and the natural phenomenon underpinning the nature-inspired algorithms, including evolutionary algorithms and swarm intelligence algorithms. We provide a section for how DNNs can be designed and trained by the metaheuristics. The problem statements for hyper-parameter optimization of DNNs, training DNNs, architecture optimization, optimization at DNN feature representation level are presented to highlight decision variables and the search space. Besides, in this section, encoding schemes, which convert a network into a solution vector to be evolved by a metaheuristic, are grouped based on how mapping is achieved between genotype-phenotype spaces and network spaces. How the researchers can validate the performance of a DNN designed by the metaheuristics is also given. Next, we group the studies related to metaheuristics and DNNs based on the problem type considered. A section has also been dedicated to the description of datasets which can be used to validate the methodologies on the same experimental setup and compare with state-of-the-art algorithms.

The rest of the paper is organized as follows. In Sect. 2, the most common DL architectures are briefly introduced; in Sect. 3, the properties of metaheuristic algorithms are summarized. The forth section is dedicated to automated DNN search by the metaheuristics. The problem statements of hyper-parameter optimization, training DNNs, architecture search and optimization at feature representation level of DNNs are provided. Network encodings used to represent solutions in the metaheuristics are grouped, and how the performance of the metaheuristics can be validated is described briefly. In Sect. 6, the literature review methodology is explained and the reviewed studies are grouped according to the optimization problem arising in DNNs. Some brief information about the common-used datasets in this field is also provided. Section 7 provides a discussion and the last section is dedicated to the conclusion and future directions.

2 Deep neural network architectures

A neural network (NN) is composed of connected layers which consist of interacting neurons performing computations to achieve an intelligent behaviour or to approximate a complex function. Deep NN models can be categorized into two groups: Discriminative and Generative models. The discriminative models adopt a bottom-up approach. They model a decision boundary between classes based on the observed data and produce a class label for unseen data by calculating the maximum similarity with labeled items. The generative models adopt a top-down approach. They establish a model based on the actual distribution of each class and produce an output for unseen data based on the joint probability in an unsupervised manner. Based on architectural properties, deep neural networks can be categorized into four groups as shown in Fig. 1 (Patterson and Gibson 2017): Unsupervised Pretrained Networks (UPN), Convolutional Neural Networks (CNNs) (Fukushima 1980), Recurrent Neural Networks (RNNs) (Rumelhart et al. 1986) and Recursive Neural Networks. UPNs include Auto-Encoders (AEs) (Kramer 1991), Restricted Boltzmann Machines (RBMs) (Smolensky 1986), Deep Belief Networks (DBNs) (Hinton et al. 2006) and Generative Adversarial Networks (GANs) (Goodfellow et al. 2014). Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) is also an implementation of RNNs.

2.1 Convolutional neural networks

CNN (Fukushima 1980) is one of the most used DNN models generally used for grid-structured data, such as images. In CNNs (Fig. 2), a layer performs transformations and calculations and then propagates it to the next layer. While the dimensionality of hidden layers determines the width of the model, the depth of the model means the overall length of layers. The degree of abstraction depends on the depth of the architecture. Dense, dropout, reshape, flatten, convolution and pooling are some examples of layers in CNNs. The convolution layer weights each element in a data matrix by a probability density function. The number of filters and the filter sizes are important hyper-parameters of the convolution layer. The pooling layer is used to downsample the feature maps using operators such as max, min, average, median. Pool size and striding are to be assigned in each pooling layer. In the dense layers, outputs are calculated by transferring the weighted inputs to the activation function. The initialization mode of the weights, activation function (softmax, softplus, softsign, relu, tanh, sigmoid, hard sigmoid, linear, etc.) and the number of outputs are to be determined in each dense layer. The dropout layer assigns zeros to some randomly selected inputs to prevent overfitting during training. The reshape layer ensures that the one-dimensional feature map is conformable to the data format in other layers. The flatten layer reshapes a multi-dimensional data into a one-dimensional form.

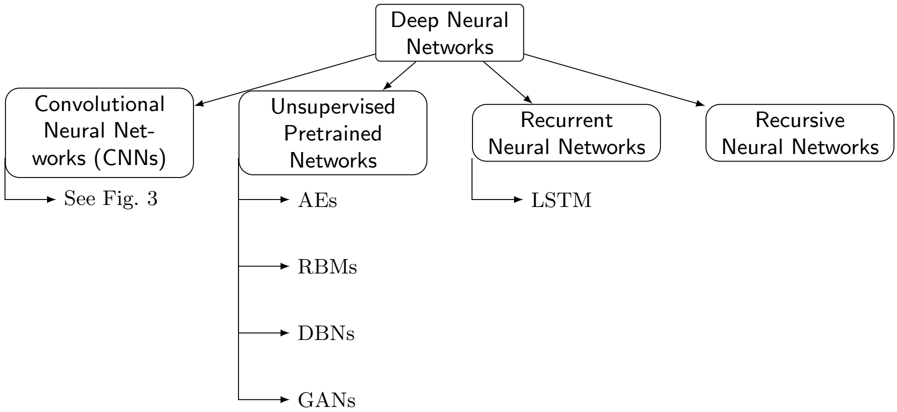

Deep Neural Networks
Convolutional Neural Networks (CNNs)
Unsupervised Pretrained Networks
Recurrent Neural Networks
Recursive Neural Networks
See Fig. 3
AEs
RBMs
DBNs
GANs
LSTM

Fig. 1 A taxonomy for DNN architectures (Patterson and Gibson 2017)

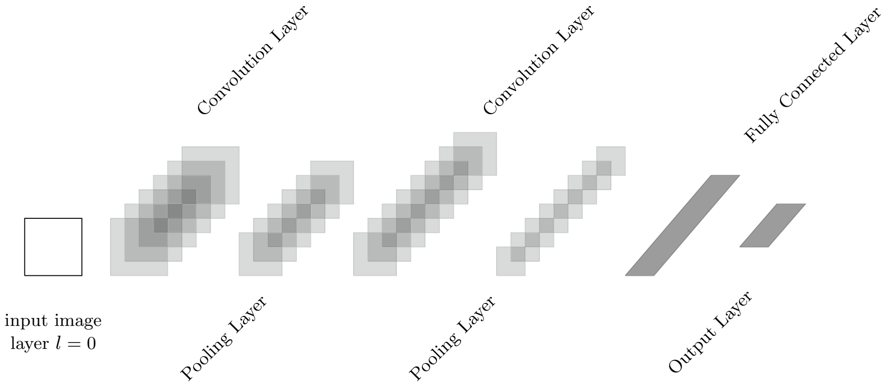


Fig. 2 Architecture of a CNN

The cost function for a sample (x, y) can be defined by Eq. 1 and error at sub-sampling layer is calculated by Eq. 2 (Shrestha and Mahmood 2019):

$$J(W, b; x, y) = \frac{1}{2}\left\|h_{w,b}(x) - y\right\|^2 \tag{1}$$

$$\delta_k^{(l)} = \text{upsample}\left(\left(W_k^{(l)}\right)^T \delta_k^{(l+1)}\right) f'(z_k^{(l)}) \tag{2}$$

where $\delta_k^{(l+1)}$is the error for $(l + 1)$th layer of a network and $f'(z_k^{(l)})$ represents the derivate of the activation function. AlexNet (Krizhevsky et al. 2012), GoogleNet (Inceptionv1) (Szegedy et al. 2015), ResNet (He et al. 2016) and VGG (Simonyan and Zisserman 2014) are some popular and widely-used CNN models. The CNN models can be categorized as in Fig. 3 (Khan et al. 2020).

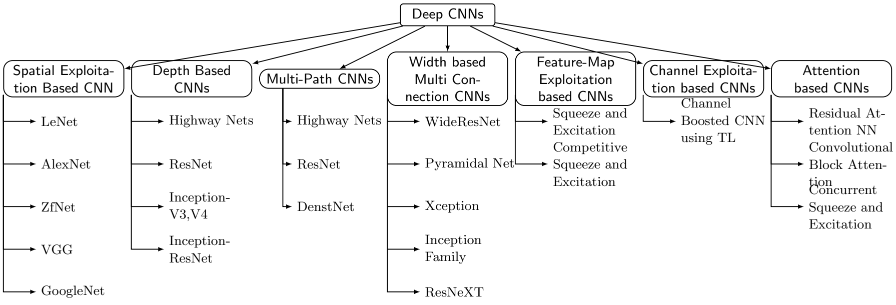


Fig. 3 A taxonomy for CNN architecture (Khan et al. 2020)

2.2 Unsupervised pretrained networks

UPNs employ greedy layer-wise unsupervised pretraining followed by a second stage for supervised fine-tuning. An unsupervised algorithm based on a nonlinear transformation trains each layer and extracts the variations in its input. This stage intensifies the weights and makes the cost function more complicated with included topological features. In the second stage, the labels are used and a training criterion is minimized (Erhan et al. 2010). UPNs have the advantage in initializing a number of layers over random initialization (Bengio 2009). Auto-Encoders (AEs) (Kramer 1991), Restricted Boltzmann Machines (RBMs) (Smolensky 1986), Deep Belief Networks (DBNs) (Hinton et al. 2006) and Generative Adversarial Networks (GANs) (Goodfellow et al. 2014) are representatives of UPNs that use training algorithms based on layer-local unsupervised criteria. These are described below briefly.

2.2.1 Auto-encoders

An auto-encoder (Kramer 1991) (Fig. 4) is a special case of feed-forward NNs that learn features by copying its input to output. AE uses greedy layer-by-layer unsupervised pretraining and fine-tuning with backpropagation (Kramer 1991). It is composed of two parts: an encoder function $h = f(x)$ that transforms data from high-dimensional space to a low dimensional space as a lossy compression, and a decoder that produces a reconstruction $r = g(h)$. The aim is to achieve $g(f(x)) = x$ nonlinear representation using an unsupervised algorithm because they regenerate the input itself rather than other output.

When the structure has linear one hidden layer, the k units in this layer learn the first k principal component of the input by minimizing mean squared error function. When the hidden layer is non-linear, AE does not act like Principal Component Analysis and can extract multi-modal variations in data by minimizing the negative log-likelihood of reconstruction given by Eq. 3, and the loss function can be defined by Eq. 4 when the inputs are binary (Bengio 2009):

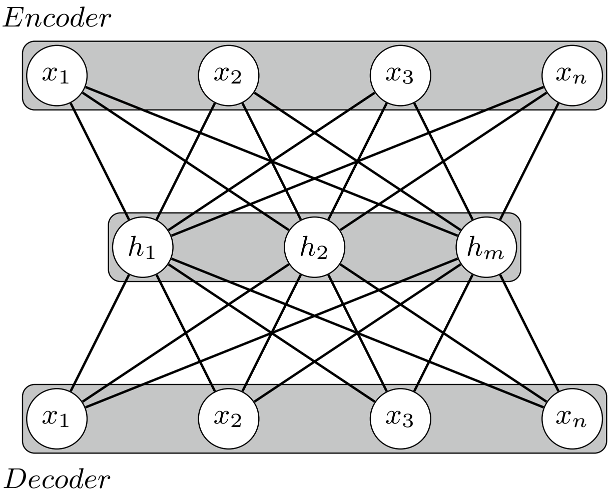


Fig. 4 A simple AE structure

$$RE = -\log P(x|f(x)) \tag{3}$$

$$-\log(P(x|f(x))) = -\sum_{i} x_i \log g_i(f(x)) + (1 - x_i)\log(1 - g_i(f(x))) \tag{4}$$

2.2.2 Restricted Boltzmann machine and deep belief networks

RBMs (Smolensky 1986) (Fig. 5) can build non-linear generative models from unlabeled data to reconstruct the input by learning the probability distribution. The architecture has a hidden layer and a visible layer in which all units are fully connected to the units in the hidden layer, and there are no connections between the units in the same layer. The energy of the RBM network is calculated by Eq. 5:

$$E(v,h) = -\sum_{i\in I} b_{v_i} v_i - \sum_{j\in H} b_{h_j} h_j - \sum_{i,j} v_i h_j \omega_{ij} \tag{5}$$

where v_i and h_j are the states of ith node. v is the set of nodes in the input layer, h is the set of nodes in the hidden layer, b_{v_i} and b_{h_j} are the biases for the input layer and hidden layer, respectively. ω_{ij} represents the weight of the edge connecting two nodes. The probability of each node is calculated by energy formula given in Eq. 6.

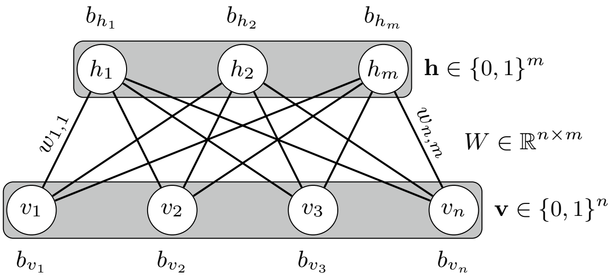
$$P(v,h) = \frac{e^{-E(v,h)}}{\sum_{v,h} e^{-E(v,h)}} \tag{6}$$

Hinton et al. (2006) showed that deeper networks could efficiently be trained using greedy layer-wise pretraining and proposed DBNs which are composed of a stacked set of RBMs with both directed edges and undirected edges. A DBN with ℓ layers models the joint distribution between observed vector x and ℓ hidden layers h^k by Eq. 7:

$$P(x, h^1, \dots, h^\ell) = \left(\prod_{k=0}^{\ell-2} P(h^k|h^{k+1})\right) P(h^{\ell-1}, h^\ell) \tag{7}$$

where $x = h^0$, $P(h^{k-1}|h^k)$ is a visible-given-hidden conditional distribution in an RBM associated with level k of the DBN, and $P(h^{\ell-1}, h^\ell)$ is the joint distribution in the top-level

Fig. 5 Architecture of an RBM



RBM. Sampling from the top level of RBM is performed using Gibbs chain. DBN is trained in greedy layer-wise fashion.

2.2.3 Generative adversarial neural networks

GANs (Goodfellow et al. 2014) are designed to train two adversarial networks, a generative model G and a discriminative model D. G, which is a differentiable function represented by a multilayer perceptron, can learn the generator's distribution p_g over data x while D estimates the probability that a sample is drawn from the training set rather than G. A prior on input noise variables $p_z(z)$ is mapped by $G(z;\theta_g)$, where θ_g is multilayer perceptron parameters. A latent code is generated by the generator network. The second network $D(x;\theta_d)$ aims to assign the correct label to both training examples and samples from G, while G is trained to generate samples that counterattack the success of D by minimizing $log(1-D(G(z)))$, which is a two-player minimax game with value function $V(G; D)$ defined by Eq. 8 (Yinka-Banjo and Ugot 2019):

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data(x)}}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_{z(z)}}\left[\log(1-D(G(z)))\right] \tag{8}$$

To avoid saturation in early cycles of learning, G can also be allowed to maximize $log(D(G(z)))$ (Goodfellow et al. 2014).

2.3 Recurrent neural networks

RNNs (Rumelhart et al. 1986) are special-purpose networks for processing sequential data, especially time series. The primary function of each layer is introducing memory rather than hierarchical processing. RNNs illustrated in Fig. 6 produce an output at each time step by Eq. 9 and have recurrent connections between hidden units (Eq. 10).

$$y_t = \omega h_t \tag{9}$$

where h_t denotes the hidden state defined by Eq. 10.

$$h_t = f(Wx_t + Uh_{t-1} + b + c) \tag{10}$$

where x_t and y_t are the input and output at time t, and f indicates the nonlinear function. Learning phase updates weights which determine the information to pass onward or to discard.

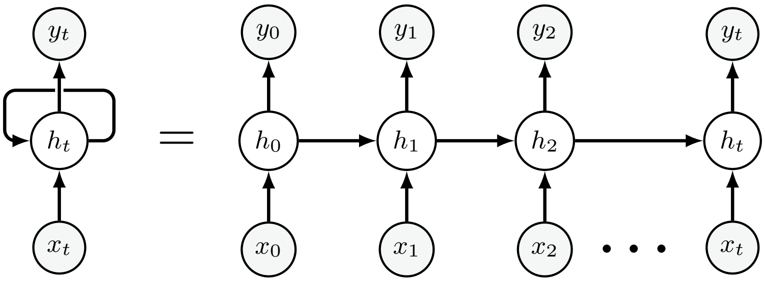


Fig. 6 Architecture of an RNN

2.3.1 Long short-term memory

LSTM (Hochreiter and Schmidhuber 1997) is an implementation of RNN, which partly addresses problems of vanishing gradients and long time delays. LSTM cells (Fig. 7.) with internal recurrence (a self-loop) are introduced to produce paths where the gradient can flow for long durations rather than element-wise nonlinearity. LSTM can obtain previous states and can be trained for applications requiring memory or state awareness. LSTM technique replaces hidden neurons by memory cells as multiple gating units control the flow of information.

In Fig. 7, x is the input, h is the hidden neuron, c is memory cell state, the symbol σ is the sigmoid function and *tanh* is the hyperbolic tangent function, operator $\oplus$ is the element-wise summation and $\otimes$ is the element-wise multiplication. The input and output gates regulate the read and write access to the cell, the forget gates learn to reset memory cells once their contents are useless. The values at the output of the gates can be given by Eqs. 11–15 (Shrestha and Mahmood 2019):

$$f_t = \sigma(W_f x_t + w_f h_{t-1} + b_f) \tag{11}$$

$$i_t = \sigma(W_i x_t + w_i h_{t-1} + b_i) \tag{12}$$

$$o_t = \sigma(W_o x_t + w_o h_{t-1} + b_o) \tag{13}$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes tanh(W_c x_t + w_c h_{t-1} + b_c) \tag{14}$$

$$h_t = o_t \otimes tanh(c_t) \tag{15}$$

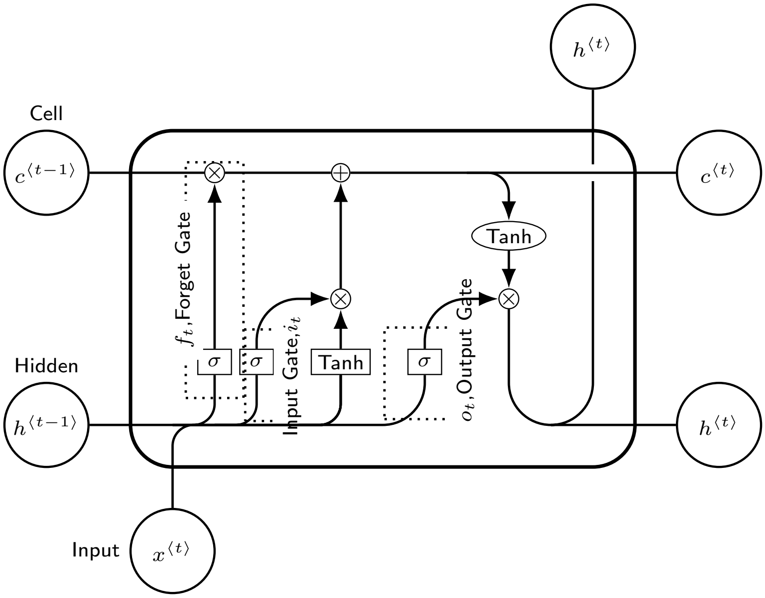


Fig. 7 Architecture of an LSTM

where *f*, *i*, *o* correspond to the forget, input and output gate vectors, *W*, *w* and *b* are weights of input, weights of recurrent output and bias, respectively.

2.4 Recursive neural networks

Recursive NNs are defined as skewed tree structures (Fig. 8), and a deep recursive NN can be constructed by stacking multiple layers of recursive layers (Irsoy and Cardie 2014).

Assuming that $l(\eta)$ and $r(\eta)$ are left and right children of a node η in a binary tree, a recursive NN computes the representations at each internal node η by Eq. 16:

$$x_\eta = f(W_L x_{l(\eta)} + W_R x_{r(\eta)} + b) \tag{16}$$

where W_L and W_R are square weight matrices of left and right child, respectively, and b is a bias vector. The result at the output layer can be calculated by Eq. 17: representation layer:

$$y_\eta = g(W_o x_\eta + b_o) \tag{17}$$

where W_o is the output weight matrix and b_o is the bias vector to the output layer.

When each hidden layer lies in a different space in stacked deep learners, Irsoy and Cardie (2014) proposed a stacked deep recursive NN of individual recursive by Eq. 18:

$$h_\eta^{(i)} = f(W_L^{(i)} h_{l(\eta)}^{i} + W_R^{(i)} h_{r(\eta)}^{(i)} + V^{(i)} h_\eta^{(i-1)} + b^{(i)}) \tag{18}$$

where i is the layer index of the multiple stacked layers, $W_L^{(i)}$ and $W_R^{(i)}$ are weight matrices of left and right child in layer i, and $V^{(i)}$ is the weight matrix between $(i-1)$th and ith hidden layers.

3 Metaheuristic algorithms

Most of the optimization problems in real-world are hard such that their objective functions and constraints have high nonlinearity, multi-modality, discontinuity in addition to conflicting with each other etc (Karaboga and Akay 2009). These hard problems might not be

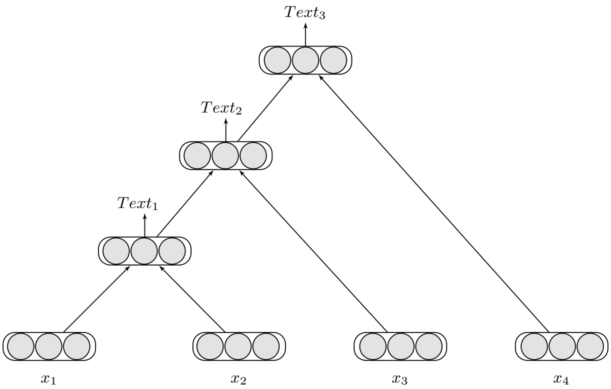


Fig. 8 Representation of a recursive NN in the form of tree structure

solved by exact methods in a reasonable time, and approximate algorithms are preferred to find good solutions. Metaheuristic algorithms are a group of approximate algorithms which mimic a natural phenomenon using operators performing intensification and diversification. They adopt an encoding scheme to represent a solution to the problem and use some search operators to change the solution in search space, systematically. These search operators try to find high-quality solutions by exploiting the current knowledge (intensification) or explore new solutions (diversification) to avoid getting stuck to local minima. A good algorithm performs intensification and diversification in a balanced manner. To guide the search for a better region in the search space, algorithms employ some greedy and stochastic selection operators.

The metaheuristic algorithms can be divided into two main groups as given in Fig. 9 depending on the number of solutions considered during the search.

3.1 Single solution-based algorithms

The first group called trajectory methods or single solution-based methods start with a single solution and construct next solution using a perturbation mechanism. Tabu Search (TS) (Glover 1986) and Simulated Annealing (SA) (Kirkpatrick et al. 1983) algorithms are popular examples of the first group.

TS proposed by Glover (1986) applies local search to construct a solution and uses a tabu list as a memory to prohibit the previous solutions. In each iteration, worse solutions have a chance to be accepted if no improvement can be obtained. SA proposed by Kirkpatrick et al. (1983) mimics metallurgical annealing in which heat of a material is increased and cooled down to optimize its crystal structure to prevent defects. In each time step, the

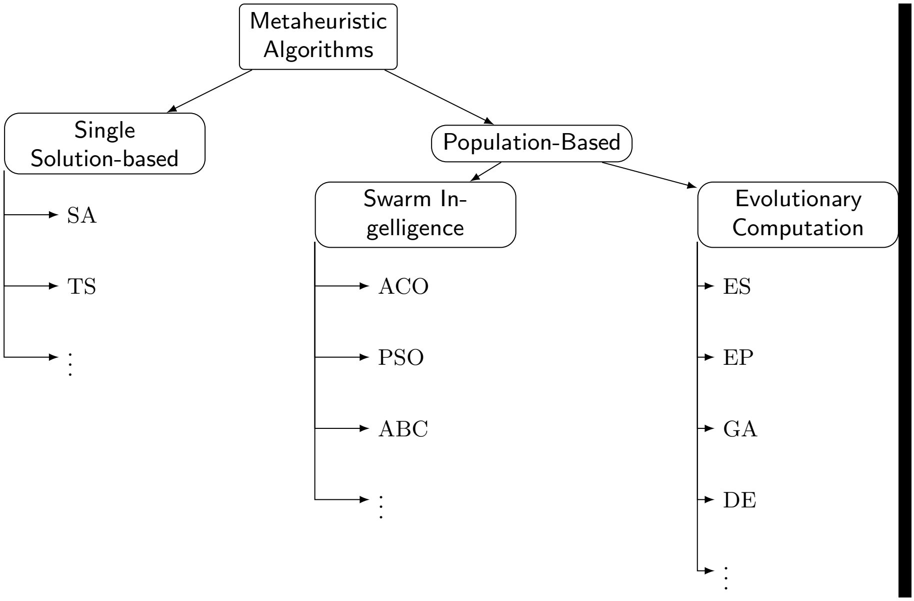


Fig. 9 A simple taxonomy of metaheuristic algorithms

algorithm generates a random solution, and the new solution is accepted or rejected according to the probability calculated based on thermodynamic principles.

3.2 Population-based algorithms

The second group uses a group of solutions called a population and tries to improve the population collectively. In the population-based algorithms, solutions might use the information due to the other solutions in the population, or the next population might be generated considering whole population. Evolutionary Computation (EC) algorithms such as Evolution Strategies (ES) (Rechenberg 1965), Evolutionary Programming (EP) (Fogel et al. 1966) and Genetic Algorithms (GA) (Holland 1975), Genetic Programming (GP) (Koza 1990) and Differential Evolution (DE) (Storn and Price 1995), and Swarm Intelligence algorithms such as Artificial Bee Colony (ABC) (Karaboga 2005), Particle Swarm Optimization (PSO) (Kennedy and Eberhart 1995), Ant Colony Optimization (ACO) (Dorigo et al. 1996) algorithms are the most popular members of population-based algorithms.

3.2.1 Evolutionary computation algorithms

EC algorithms mimic natural selection and genetic operations during the search, and they can be grouped as ES by Rechenberg (1965), EP by Fogel et al. (1966), GA by Holland (1975) and GP by Koza (1990). For each solution in the population, EC algorithms generate offsprings by genetic recombination, and mutation operators and next population is formed by the solutions corresponding to better regions in the search space. DE proposed by Storn and Price (1995) is an evolutionary algorithm proposed to optimize the function of real-valued decision variables.

3.2.2 Swarm intelligence algorithms

Swarm intelligence algorithms mimic the collective behaviour of creatures to establish a global response through local interactions of agents in a swarm. They adopt self-organization and division of labour, where self-organization is characterized by positive feedback to intensify better patterns, negative feedback to abandon the patterns exploited sufficiently, multiple interactions between agents and fluctuations to bring diversity to swarm. PSO developed by Kennedy and Eberhart (1995) simulates the flocking of birds and schooling of fishes. The algorithm changes the velocity and location of each particle using the particle's self-experience and the global information of all particles. ACO proposed by Dorigo et al. (1996) is an analogy of ants' path finding between the nest and source using the pheromone trail. In each iteration, ants change their positions based on probabilities proportional with pheromone amount between the nodes. An edge visited by more ants will have higher pheromone, and a higher amount of pheromone will attract more ants while the pheromone is evaporating by time. ABC proposed by Karaboga (2005) is another well-known swarm intelligence algorithm which simulates the foraging behaviour of honey bees. There are three types of bees allocated for foraging task: employed bees, onlooker bees and scout bees, which all try to maximize the nectar amount unloaded to the hive. There are some other metaheuristics as well as the modifications of these algorithms.

4 Automated DNN search by metaheuristics

While solving a specific problem by using a DNN, there are some design decisions that influence the performance of the DNN. Selecting the hyper-parameters of a DNN, training a DNN to find the optimal weights and bias values, deciding the optimal architecture parameters of a DNN, or reducing the dimensionality in feature representation level of a DNN can be considered as optimization problems. Metaheuristic algorithms can be used for solving these types of optimization problems. In each evolution cycle, present solution/s is/are assigned a fitness value, and new solution/s is/are generated by reproduction operators. The selection operators of metaheuristics guide the search to promising regions of the search space based on the information of solutions, and this makes the metaheuristics efficient tools to solve complex problems.

4.1 Problem statements

The optimization problems encountered in DNN design and training are formulated in the subsections.

4.1.1 Hyper-parameter optimization

Most DL models have model-specific hyper-parameters that have to be tuned before the model starts the learning phase. Some of these parameters are related to macro structure of DNN (outer level) while some are cell or block-level parameters (inner level). Training the model with these parameters is very influential in the behaviour and performance of the network. Assuming that λ_h is a hyper-parameter vector, the hyper-parameter optimization can be defined by Eq. 19 (Li et al. 2017):

$$\lambda_h^* = \underset{\lambda_h \in \wedge}{arg\,min}\, Err(x_{test}, F_{network}(x_{train}, A_{\lambda_h}, \theta))) \tag{19}$$

where Err is the error on the test set, $\wedge_h$ is hyper-parameter space, A_{λ_h} is learning algorithm, θ represents the model parameters, x_{train} and x_{test} are train and test datasets, respectively. By deciding the optimal hyper-parameter values, the error of the model is minimized.

Running different configurations manually and evaluating them to decide the best parameter setting is quite labor-intensive and time-consuming especially when the search space is high-dimensional. It should also be noted that a tuned hyper-parameter setting by an experienced user is application-dependent. Therefore, automatic hyper-parameter optimization is needed to reduce computational cost and user involvement. The steepest gradient descent algorithm is not suitable to optimize hyper-parameter configurations. Although Grid Search, Random Search (RS) and Bayesian Optimization are popular to configure hyper-parameters, they are impractical as the DNN model gets more complicated and deeper or when the number of parameters and the complexity of the problem is high (Nakisa et al. 2018). Some studies evaluate the hyper-parameters on a low dimensional model and apply the values to the real model (Hinz et al. 2018). Metaheuristic algorithms have been used to obtain a faster convergence and to find a near-optimal configuration in a reasonable time. They systematically search the hyper-parameter space by evolving the population at each iteration.

4.1.2 Training DNNs

Although deep learning models can be successfully applied to solve many problems, training them is complicated for several reasons. Some studies employ derivative methods such as stochastic gradient descent, conjugate gradient, Hessian-free optimization to determine the optimal values of weights and biases. These conventional training methods suffer from premature convergence and vanishing (or exploding) gradients problem which means that the gradients calculated layer-wise in a cascading manner are decreasing or increasing exponentially and tend to explode. Because the weights are updated based on the gradient values, the training might go to stagnation or overshoot, especially when nonlinear activation functions ranging in a small scale are employed, and hence, the solutions may not converge to the optimum. The gradient-based algorithms can get stuck to saddle points arising due to flat regions in the search space. Besides, steep edges in the search space surface leads overshoot during weight updates. Parallelizing the derivative-based training algorithms and distributing to computing facilities to reduce the computational cost is also hard. Therefore, the metaheuristic algorithms are introduced in the training of deep learning methods. Training a DNN can be expressed as an optimization problem defined by Eq. 20:

$$\{w, b\}^* = \underset{\omega, b \in \mathbb{R}}{arg\,min}\, Err(x_{test}, F_{network}(x_{train}, \omega, b)) \tag{20}$$

4.1.3 Architecture optimization (architecture search)

Deep learning models require to define an application-dependent architecture that affects the model accuracy. These parameters include the model-specific parameters such as convolutional layers' parameters, fully-connected layers' parameters, and parameters of the training process itself. The problem of finding an optimal CNN architecture (Eq. 21) can be considered as an expensive combinatorial optimization problem.

$$\{\lambda_N, \lambda_{L_i}, w, b\}^* = \underset{\omega, b \in \mathbb{R}, \lambda_N \in \wedge_N, \lambda_{L_i} \in \wedge_{L_i}}{arg\,min} Err(x_{test}, F_{network}(x_{train}, \omega, b, \lambda_N, \lambda_{L_i})) \tag{21}$$

where $i = 1, 2, \dots, LN$, LN is the number of layers, λ_N is the global parameter set of the network, λ_{L_i} parameter set for ith layer $\wedge_N$ and $\wedge_{L_i}$ are parameter spaces for global parameters and layer-wise parameters, respectively.

If a compact model is established, the accuracy may be adversely affected whilst if a complex model is adopted, training time and computational cost will be higher. Additionally, when the network is expanded with large number of computations units, this time the network might get overfitting to the outliers in the data. There is a trade-off between generalization capability and computational complexity. In one-shot models (Brock et al. 2017; Bender et al. 2018), a large network that represents a wide variety of architectures is trained, and weight sharing is applied between sub-models instead of training separate models from scratch. Architecture optimization is hard because it involves optimizing too many real, integer, binary and categoric variables. Depending on the values of outer level parameters, the number of inner-level parameters to be optimized changes and therefore, the search space has variable-size dimension. There is no analytic approach to automate the design of DL model architecture. Since there are many possible configurations, an exhaustive search is too expensive. Therefore, researchers have employed metaheuristics to automatically search for optimal architecture to maximize performance metrics defined in Sect. 4.6.

4.1.4 Optimization at feature representation level

Metaheuristic algorithms have been used in the feature representation level of deep models or dimensionality reduction. They find a subset of features by maximizing a fitness function measuring how good the subset F_s represents the whole set F (Eq. 22).

$$F_s^* = \underset{F_s \in F}{arg\,max}\, I(F_s) \tag{22}$$

where F is the feature set, F_s is a subset of F and I is a fitness function such as mutual information. The features extracted are transferred to the input layer of deep models.

4.2 Representations used in metaheuristics for encoding DNN parameters

Encoding is converting the network parameters or structure into a solution representation of the metaheuristic, so that the reproduction operators can operate on the solutions and generate candidate solutions. Adopting an efficient representation and encoding strategy is a critical design issue because search operators are decided in accordance with the encoding selected, and they affect the performance of the metaheuristic and the DNN produced. The suitable representation is related with how the DNN optimization is handled. If the network topology is fixed, and hyper-parameters or weights are optimized, a fixed-length representation might be employed. However, when the topology is evolved during search, a variable-length representation is required to be able to encode larger or smaller networks constructed by the search operators. The encoding of networks can be divided into three main groups: direct, indirect and implicit encoding. For encoding solutions based on these encoding types, bit strings, real-valued vectors, index vectors for categorical data, adjacency matrices, context-free grammar, directed cyclic graphs or trees can be used.

In the direct encoding, an evaluation of genotype-phenotype mapping for the network is employed. There are a variety of direct encoding types, including, but not limited to, connection-based, cell-based, layer-based, module-based, pathway-based, block-based, operation-based (Fekiač et al. 2011; Kassahun et al. 2007). An example of direct encoding for a chained CNN architecture and its hyper-parameters is shown in Fig. 10.

- Connection-based: in a predetermined network structure, each switch connection can be represented by one bit, or weights can be represented by bit strings or real values. If the connectivity matrix of a network is represented by a matrix, when the network includes n neurons, the matrix is of size n^2, which can cause the scalability problem. After reproduction, the solutions may not be feasible and need repairing mechanisms.
- Cell-based: the individual contains all the information about the nodes and their connectivity. A list can contain nodes and their connections, or a tree can represent the nodes and connection weights. The complete architecture is constructed by stacking the cells based on a predefined rule (Koza and Rice 1991; Schiffmann 2000; Suganuma et al. 2017; Pham et al. 2018; Zoph et al. 2018; Real et al. 2019).

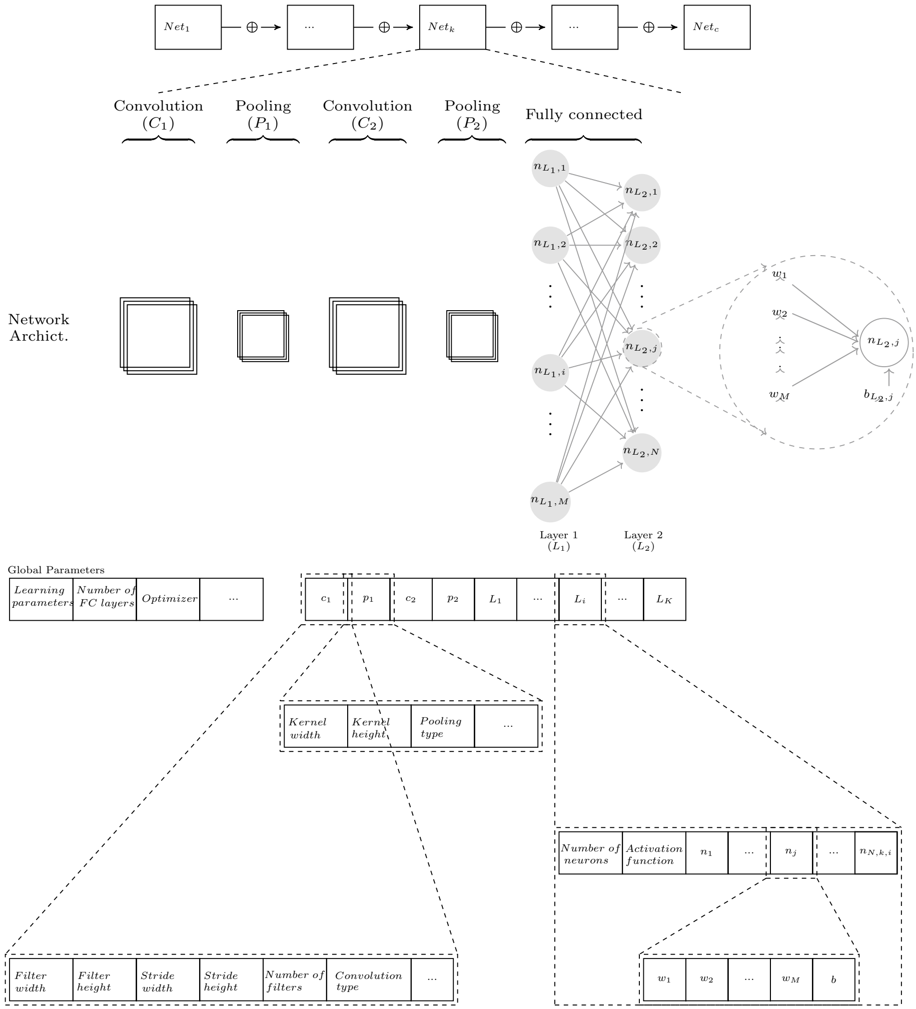


Fig. 10 A direct representation for a chained CNN architecture and hyper-parameters

- Layer-based: the individual encodes the layers' parameters together with the global network parameters (Jain et al. 2018; Baldominos et al. 2019; Mattioli et al. 2019; Sun et al. 2019b; Pandey and Kaur 2018; Martín et al. 2018).
- Module-based: it can be seen as a special case of cell-based representation. Because some modules are repeating in the networks such as GoogleNet and ResNet, the modules and the blueprints are encoded in the chromosomes. The blueprint chromosomes correspond to a graph of nodes pointing a specific module while the module chromosomes correspond to a network. Blueprint and module chromosomes are assembled together to construct a large network (Miikkulainen et al. 2017).
- Pathway-based: a set of paths represented by context-free grammar is used to describe the paths from the inputs to the outputs. This encoding can be preferred especially for recurrent networks (Jacob and Rehder 1993; Cai et al. 2018a). Especially for chain-structured structures such as Inception models (Szegedy et al. 2016), ResNets (He

et al. 2016) and DenseNets (Huang et al. 2017), path-based representation and evolution modifies the path topologies in the network and maintains functionality by means of weight reusing. Bidirectional tree-structured representation can be used for efficient exploration (Cai et al. 2018a) (Fig. 11).

- Block-based: in block-based networks, two-dimensional arrays holds different internal configurations depending on the structure settings and weights (Zhong et al. 2018; Junior and Yen 2019; Wang et al. 2019b). Sun et al. (2018a) proposed block based definition for CNN architecture which is composed of ResNet, DenseNet blocks and pooling layer unit.
- Operation-based: phases are encoded by a directed acyclic graph describing the operations within a phase such as convolution, pooling, batch-normalization or a sequence of operations. Skip connections are introduced for bypassing some blocks. The same phases are not repeated in network construction (Xie and Yuille 2017; Lu et al. 2019).
- Multi-level encoding: one or more encoding types can be used jointly to represent structures, nodes, pathways and connections. Stanley and Miikkulainen (2002) represented augmenting topologies by connection genes which refer to two node genes being connected and neuron genes which specifies the in-node, the out-node, the weight of the connection, whether or not the connection gene is expressed (an enable bit), and an innovation number, which allows finding corresponding genes during crossover, as shown in Fig. 12.

In the indirect encoding, phenotype can be converted to a genotype in a smaller search space based on a production rule. L-System, cellular encoding, developmental and generative encoding, connective compositional pattern producing networks are some examples of indirect encoding.

- L-System is a mathematical model simulating the natural growth (Lindenmayer 1968). Information exchange between neighboring cells arises by continually determining rewriting steps to make a neural network based on grammar production rules.
- Cellular Encoding is similar to L-system as it has grammar based production rule. It starts with an initial cell and constructs a graph grammar network until it contains only terminals (Gruau 1994).

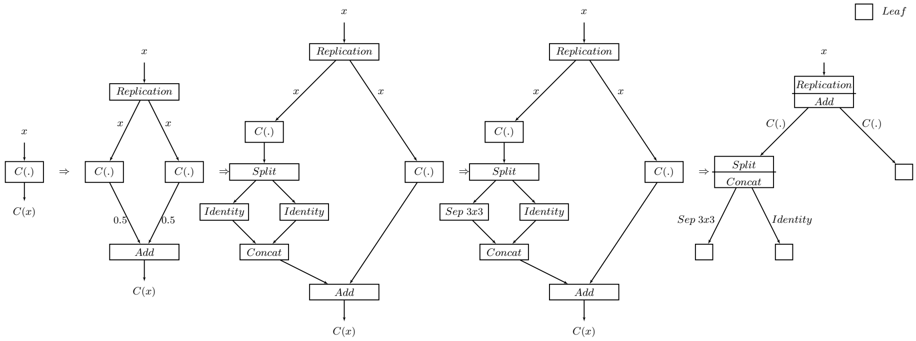
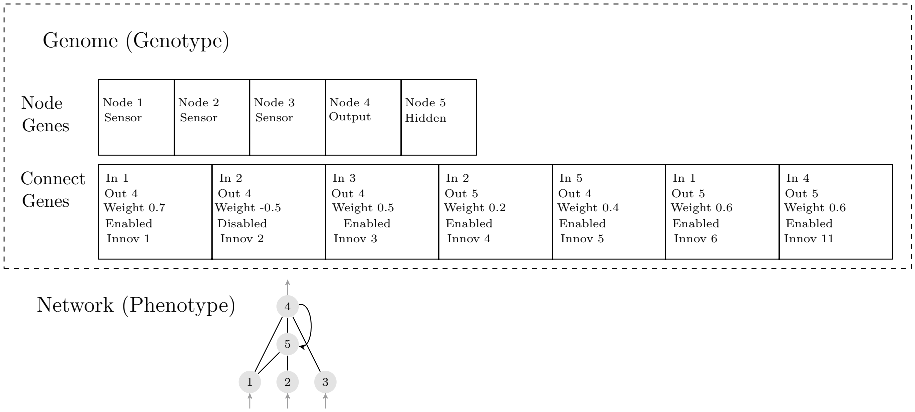


Fig. 11 Transforming a single layer to a tree-structured motif via path-level transformation operations (Cai et al. 2018a)



Node 1 Sensor	Node 2 Sensor	Node 3 Sensor	Node 4 Output	Node 5 Hidden

In 1 Out 4 Weight 0.7 Enabled Innov 1	In 2 Out 4 Weight -0.5 Disabled Innov 2	In 3 Out 4 Weight 0.5 Enabled Innov 3	In 2 Out 5 Weight 0.2 Enabled Innov 4	In 5 Out 4 Weight 0.4 Enabled Innov 5	In 1 Out 5 Weight 0.6 Enabled Innov 6	In 4 Out 5 Weight 0.6 Enabled Innov 11

Fig. 12 Augmenting topologies encoding (Stanley and Miikkulainen 2002)

- Developmental and generative encoding mimics DNA mapping to a mature phenotype through a growth process and reactivating genes during the development process. The connectivity patterns are generated by applying the right activation functions and repetition with variation motifs (Gauci and Stanley 2007).
- Connective compositional pattern producing networks exploit geometry and represent connectivity patterns as functions of Cartesian space (Stanley et al. 2009).

Another group of encoding is implicit encoding in which a network is represented based on biological gene regulatory (Floreano et al. 2008) or based on the parameters of Gaussian distribution where the weights are assumed to be sampled (Sun et al. 2019a).

While designing a search space, various architectures can be represented using a single one-shot model as a super-graph of DNN components. The template network is trained to predict the validation accuracies of the architectures. The pretrained one-shot model is used to validate the architectures, and the best architectures are re-trained from scratch (Bender et al. 2018). An example of one-shot model is shown in Fig. 13.

The representation can also be grouped into flat and hierarchical representations (Liu et al. 2017b).

- Flat Representation: Each architecture $A = assemble(G, o)$ is represented as a single-source, single-sink directed acyclic graph in which each node corresponds to a feature

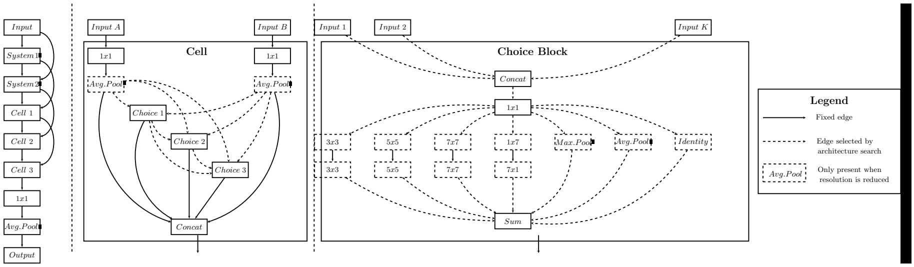


Fig. 13 Diagram of a one-shot architecture (Bender et al. 2018)

map, and each edge is associated with available primitive operators, $o = \{o_1, o_2, \dots\}$, such as convolution, pooling etc. G is adjacency matrix where $G_{ij} = k$ if o_k operation is performed between nodes i and j. To calculate the feature map of a node, the feature maps of the predecessor nodes of the current node are concatenated in depth-wise manner or element-wise manner.

- Hierarchical Representation: In this representation there are several motifs at different levels of hierarchy. Assuming that the hierarchy has L levels and ℓth level includes $M_\ell$motifs. Lower level motifs correspond to operations used to construct the high level motifs. The lowest level, $\ell = 1$, contains a set of primitive operations while the highest level, $\ell = L$, corresponds to the entire full architecture as a composition of previous layers (Liu et al. 2017b, 2018). The set of operations in level $\ell -$ lcan be defined recursively as $o_m^{(\ell)} = \text{assemble}(G_m^{(\ell)}, o^{(\ell-1)}), \forall \ell = 2, \dots, L$where $o^{(\ell-1)} = \left\{o_1^{(\ell-1)}, o_2^{(\ell-1)}, \dots, o_{M(\ell-1)}^{(\ell-1)}\right\}$ (Liu et al. 2017b). An example of this operation is shown in Fig. 14.

A problem related to the network encoding is that multiple different genotypes can be mapped to the same genotype. This is called competing conventions problem and can mislead the search process (Stanley and Miikkulainen 2002; Hoekstra 2011).

4.3 Evolutionary operators used for evolving DNN architecture and parameters

The initial solutions in metaheuristics can be generated randomly, by applying a large number of mutations (Liu et al. 2017b) to a trivial genotype, using a constructive heuristic (Lorenzo and Nalepa 2018) or a rule (Zoph and Le 2016), by morphing apriori networks for neural network architecture search. Elsken et al. (2018) and Real et al. (2017) inherit knowledge from a parent network by using network morphism and obtained speed-up compared to random initialization. To find optimal parameters in one-shot or separate models, initial solutions are generated in real space or discrete space based on the type of the parameter. To evolve the solutions and guide the search towards the optimum, some operators compatible with the representation can be used during search, including arithmetic and binary crossover, mutation operators.

Crossover combines the information of two or more individuals to create two new networks or parameter set to exploit the information in the architecture space (Miikkulainen et al. 2017). It achieves information sharing among the solutions in the population. Single point crossover, n-point crossover, uniform crossover, shuffle crossover, arithmetic crossover, flat crossover, unimodal normal distribution crossover, fuzzy connective based crossover, simplex crossover, average bound crossover, geometrical crossover, partially mapped

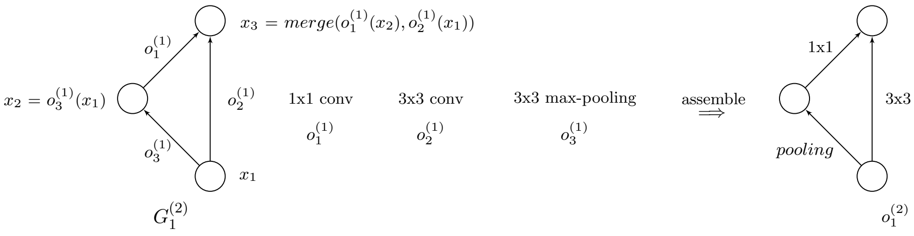


Fig. 14 An example of how level-1 primitive operations are merged into a level-2 motif (Liu et al. 2017b)

crossover, order-based crossover, and hybrid crossover operators (Lim et al. 2017) can be used to produce new solutions exploiting the information of current solutions. When an integer representation is adopted, position-based, edge-based, order-based, subset-based, graph partition and swap-based crossover operators can be used while swap, semantic, cut and merge, matrix addition, distance-based and probability based operators are suitable when tree structures are used to represent solutions (Pavai and Geetha 2016). Although crossover operator is very helpful for numeric and binary problems, it was shown that it does not provide significant improvement in combinatorial problems compared to mutation operator (Osaba et al. 2014; Salih and Moshaiov 2016). For variable-size chromosomes, Sun et al. (2019a) proposed unit alignment and unit restore to align the chromosomes to the top based on their orders and then restore them after applying reproduction.

Mutation operator can be applied for different purposes during evolution similar to human designer actions, including layer deepening, layer widening, kernel widening, branch layer, inserting SE-skip connection, inserting dense connection (Chen et al. 2015a; Wistuba 2018; Zhu et al. 2019), altering learning rate, reseting weights, inserting convolution, removing convolution, altering stride, altering number of channels, changing filter size, inserting one to one identity connection, adding skip, removing skip (Real et al. 2017). Cai et al. (2018a) presented an RNN which iteratively generates transformation operations based on current network topology and reinforcement.

4.4 Selection operators and diversity preserving

The selection operator in evolutionary algorithms is an analogy of natural selection leading survival of the fittest individuals. It means that during evolution, genetic information of the best solutions are assigned higher chance to be transferred to the next generations by means of reproduction. A very strong selection that consider only the fittest solutions can cause genetic drift and stagnation in the population due to reduced diversity. Although a weak selection can maintain diversity in the population, it may slow down convergence. There are different types of selection operators. Some of them are used to sample the individuals to be reproduced while some are used to determine solutions to be survived in the next generation population (Real et al. 2017, 2019; Liu et al. 2017b), $p[t+1] = s(v(p[t]))$, where $p[t]$ is the population at generation t, $v(.)$ is the variation operator and $s(.)$ is the selection operator. Truncation selection, stochastic universal sampling, tournament selection, roulette wheel selection, rank-based selection, greedy selection, elitism and Boltzmann selection can be used in evolutionary search.

Especially for population-based metaheuristics, diversity is an important factor in the initialization and during evolution of the population (Moriarty and Miikkulainen 1997). Diversity preserving mechanisms can be integrated into the algorithms by applying weak selection strategies for avoiding the individuals to be very similar or applying mutation with high rate to increase diversity in the population. Fitness sharing (Goldberg et al. 1987) to prevent genetic drift, niching (Mahfoud 1995) to reduce the effect of genetic drift, and crowding mechanisms (De Jong 1975) are also helpful to maintain the diversity.

When an offspring is generated in the initialization or after reproduction operators, it may not satisfy all of the problem-specific constraints. These solutions are classified as infeasible solutions. The constraint handling techniques may apply a distinction between feasible and infeasible solutions, penalize the constraint violations or use a repairing mechanism to transform infeasible solutions into feasible solutions (Koziel and Michalewicz 1999).

4.5 Speed-up methods

Because finding the optimal architecture, parameters and weights requires thousands of GPU days, some speed-up methods have been introduced. To reduce the training time, a downscaled model and data can be used (Li et al. 2017; Zoph et al. 2018), a learning curve can be extrapolated based on the results of previous epochs (Baker et al. 2017), warm-start can be used by weight inheritance/network morphisms (Real et al. 2017; Elsken et al. 2018; Cai et al. 2018a), one-shot models can be used (Brock et al. 2017; Pham et al. 2018; Bender et al. 2018) or surrogate models can be used to approximate the objective functions (Gaier and Ha 2019). Parallel implementation of the approaches can also be used to gain a speed-up (Desell 2017; Real et al. 2017; Martinez et al. 2018).

4.6 Validating the performance of the designed DNN architectures

In order to validate the performance of a network designed by metaheuristics, experiments might rely on the performance metrics, including loss function such as cross entropy, validation error, predictive accuracy, precision, recall, and F_1; network size metrics, including the number of parameters, neurons, depth, layers; memory consumption metrics, including the number of processing units, floating point operations per second (FLOP), memory usage and network inference time. One or more of these metrics can be considered during optimization in a single objective or multi-objective manner based on penalty, decomposition or Pareto-based approaches (Coello 2003). Some of these metrics can be treated as constraints and constraint handling techniques are used during optimization to guide the search towards feasible region (Koziel and Michalewicz 1999). When the objective function calculation is expensive, surrogate models can be alternative to reduce computational cost based on approximation (Jin 2011).

When a study focuses on architecture optimization, although there is no state-of-the-art algorithm validated on architecture search (Li and Talwalkar 2020), the efficiency of the metaheuristics can be verified against random search, grid search, tree-based Bayesian optimization (Bergstra et al. 2011; Hutter et al. 2011), Monte Carlo tree search (Negrinho and Gordon 2017), reinforcement learning (Zoph and Le 2016; Baker et al. 2016) or discrete gradient estimators (Xie et al. 2018). Jin et al. (2019) proposed an approach based on network morphism and Bayesian optimization for architecture search.

5 Literature review

5.1 Methodology and paper inclusion/exclusion criteria

In this section, we provide a review of the studies on utilizing metaheuristic algorithms to solve the optimization problems encountered in DL field. We searched databases Elsevier, Springer, IEEExplore, Web of Science, Scopus and Google Scholar in October 2019 by filtering with some combinations of the keywords deep learning, deep neural network, optimization, evolutionary, metaheuristic, hyper-parameter, parameter tuning, training, architecture, topology, structure. We reached 179 publications at the end of the search. The results included articles, book chapters and conference papers reporting original studies. The date range of papers was 2012 when the first paper was appeared and 2019 when the search was conducted. 77 of 179 publications were excluded because they used other optimization methods rather than employing a metaheuristic or they were written in some other languages rather than English. Within 102 publications included, there were 4 survey papers. The number of studies on training DL models using metaheuristics is 21, the number of publications on hyper-parameter optimization of DL models using metaheuristics is 30, the number of studies on architecture (topology) optimization of DL models using metaheuristics is 39, and the number of publications on using metaheuristics at feature representation level is 8.

In the subsequent sections, we summarize how metaheuristics are applied to solve optimization problems encountered in the deep learning field. The DNN architecture used, the encoding scheme, the fitness function and search operators tailored for the metaheuristics are presented as much as possible provided that they are described in the original paper. The datasets on which the study was validated and the results are also presented in the rest of the paper.

5.2 Studies using metaheuristics for hyper-parameter optimization of deep neural networks

5.2.1 Articles

Zhang et al. (2017) introduced MOEA/D integrated to DBN ensemble for remaining useful life estimation, which is a core task in condition-based maintenance (CBM) in industrial applications. The proposed method was incorporated with DE operator and Gaussian mutation to evolve multiple DBNs optimizing accuracy and diversity. Each DBN with a fixed number of hidden layers was trained using contrastive divergence followed by BP. The number of hidden neurons per hidden layer, weight cost, and learning rates used by contrastive divergence and BP were encoded as decision variables. The proposed method was evaluated on NASA's C-MAPSS aero-engine data set (Saxena and Goebel 2008), which contained four sub-datasets. From the results, it was concluded that the proposed approach demonstrated outstanding performance in comparison with some existing approaches.

The study by Delowar Hossain and Capi (2017) and another study by Hossain et al. (2018) presented an evolutionary learning method that combined GA and DBNN for robot object recognition and grasping. The DBNN consisted of a visible layer, three hidden layers, and an output layer. The parameters of DBNN including the number of hidden units, the number of epochs, learning rates and momentum of the hidden layers are optimized using GA algorithm. Visible units were set to the activation probabilities, while hidden units were set to the binary values. They generated a dataset of images of six robot graspable objects in different orientations, positions, and lighting conditions in the experimental environment. They considered the error rates and network training time as the objective to be minimized and reported that their method was efficient at assigning robotic tasks.

Lopez-Rincon et al. (2018) proposed an evolutionary optimized CNN classifier for tumor-specific microRNA signatures. In the approach, the number of layers was fixed to 3, and each layer was described by output channels, convolution window, max-pooling window. Together with the output channels in the fully connected rectifier layer, 10 hyper-parameters were tuned by EA, whose fitness was the average accuracy of CNN on a 10-fold validation process. The approach was validated on a real-world dataset containing miRNA sequencing isoform values taken from The Cancer Genome Atlas (TCGA) (2006) featuring 8129 patients, for 29 different classes of tumors, using 1046 different biomarkers. The presented approach was compared against 21 state-of-the-art classifiers. The hyper-parameters optimized by EA helped CNN to improve average validation accuracy to 96.6%, and their approach was shown to be the best classifier on the considered problem.

Nakisa et al. (2018) presented a framework to automate the search for LSTM hyper-parameters including the number of hidden neurons and batch size using the DE algorithm. The fitness function is the accuracy of emotion classification. The performance evaluation and comparison with state-of-the-art algorithms (PSO, SA, RS, and TPE) were performed using a new dataset collected from a lightweight wireless EEG headset (Emotiv) and smart wristband (Empatica E4) which can measure EEG and BVP signals. This performance was evaluated based on four-quadrant dimensional emotions: High Arousal Positive emotions (HA-P), Low Arousal-Positive emotions (LA-P), High Arousal-Negative emotions (HA-N), and Low Arousal- Negative emotions (LA-N). The experimental results showed that the average accuracy of the system based on the optimized LSTM network using DE and PSO algorithms attained progress in each time interval and could improve emotion classification significantly. Considering the average time, the processing time for the SA algorithm was lower than the others, DE consumed the most processing time overall iterations, however, its performance was higher than all other algorithms. It was noted that after a number of iterations (100 iterations), the performance of the system using ECs (PSO, SA, and DE) did not change significantly due to the occurrence of a premature convergence problem.

Loussaief and Abdelkrim (2018) proposed a GA-based CNN model to learn optimal hyper-parameters. For a CNN with D_p convolutional layers (CNN depth), the genetic algorithm optimized $2 * D_p$ variables which are D_p pairs of values (Filter Number per Layer and Filter Size per Layer). The experiments on the Caltech-256 (Griffin et al. 2007) dataset proved the ability of the GA to enhance the CNN model with a classification accuracy of 90%. They inserted a batch normalization layer after each convolutional layer and improved the quality of the network training. GA simulation produced a pretrained CNN performing an accuracy of 98.94%.

Soon et al. (2018) trained CNN by tuning its parameters for the vehicle logo recognition system. For this purpose, the hyper-parameters of CNN were selected by the PSO algorithm on The XMU dataset (Huang et al. 2015). Optimized hyper-parameters were the learning rate (α), spatial size of convolutional kernels, and the number of convolutional kernels in each convolutional layer i. They reported that the proposed CNN framework optimized by PSO achieved better accuracy compared to other handcrafted feature extraction methods.

Hinz et al. (2018) aimed to reduce the time needed for hyper-parameter optimization of deep CNNs. The parameters considered in the optimization process were the learning rate, the number of convolutional and fully connected layers, the number of filters per convolutional layer and their size, the number of units per fully connected layer, the batch size, and the L1 and L2 regularization parameters. They identified suitable value ranges in lower-dimensional data representations and then increased the dimensionality of the input later during optimization. They performed experiments with random search, TPE, sequential model-based algorithm, and GA. The approaches were validated on Cohn-Kanade (CK+) data set (Lucey et al. 2010) to classify the displayed emotion, the STL-10 data set (Coates et al. 2011), the Flowers 102 dataset (Nilsback and Zisserman 2008). All three optimization algorithms (random search, TPE, GA) produced similar values for the different hyper-parameters for all resolutions. There were minor differences in the number of filters and units per convolutional or fully connected layer. The hyper-parameters found by the GA and TPE produced better results compared to the results achieved by RS.

de Rosa and Papa (2019) presented a GP approach for hyper-parameter tuning in DBNs for binary image reconstruction. Hyper-parameters were encoded as terminal nodes, and internal nodes were the combinations of mathematical operators. The individual (solution) vector was composed of four design variables: learning rate (η), number of hidden units (n), weight decay (λ), and momentum (α). In the experiments, the difference between the original and reconstructed images was used as the fitness function. Each DBN is trained with three distinct learning algorithms: contrastive divergence, persistent contrastive divergence (PCD), and fast persistent contrastive divergence (FPCD). GP was compared to nine approaches including RS, RS-Hyperopt, Hyper-TPE, HS, IHS, Global-Best HS (GHS), Novel GHS (NGHS), Self-Adaptive GHS (SGHS), Parameter-Setting-Free HS (PSF-HS). Performances of the methods were evaluated on the MNIST, CalTech 101 Silhouettes (Marlin et al. 2010), Semeion Handwritten Digit datasets (Semeion 2008). Depending on the results, GP achieved the best results, and it was statistically similar to IHS in some cases. On the Semeion Handwritten Digit dataset, only GP achieved the best results.

Yoo (2019) proposed a search algorithm called a univariate dynamic encoding algorithm that combines local search and global search to improve the training speed of a network with several parameters to configure. In the local search, a bisectional search and a uni-directional search was performed while in global search, the multi-start method was employed. The batch size, the number of epochs, and the learning rate were the hyper-parameters optimized. The cost function was the average of the difference between the decoded value and the original image for the AE, and the inverse of the accuracy for the CNN. The proposed method was tested for AE and CNN on the MNIST dataset and its performance was compared with those of SA, GA, and PSO, and it was shown that the proposed method achieved fast convergence rate and less computational cost.

Wang et al. (2019b) proposed a cPSO to optimize the hyper-parameters of CNNs. cPSO-CNN employed a confidence function defined by a normal compound distribution to improve exploration capability and updates scalar acceleration coefficients according to the variant ranges of CNN hyper-parameters. A linear prediction model predicted the ranking of the PSO particles to reduce the cost of fitness function evaluation. In the first part of the experiments, the first convolutional layers of AlexNet and the other CNNs including VGGNet-16, VGGNet-19, GoogleNet, ResNet-52, ResNet-101, DenseNet-121 were optimized on CIFAR-10 dataset using CER as the performance metric. From the experiments, the approach was effective with all CNN architectures. In the second part of the experiments, eight layers of AlexNet were optimized using cPSO and the results were compared to those obtained by state-of-the-art methods, including GA, PSO, SA, Nelder-Mead. It was demonstrated that cPSO-CNN performed competitively according to both performance metrics and overall computation cost.

Sabar et al. (2019) proposed a hyper-heuristic parameter optimization framework (HHPO) to set DBN parameters. HHPO iteratively selects suitable heuristics from a heuristic set using Multi-Armed Bandit (MAB), applies the heuristic to tune the DBN for a better fit with the current search space. In this study, the DBN parameters optimized by the framework were learning rate, weight decay, penalty parameter, and the number of hidden units, and MSE was utilized as the cost function. The proposed approach was compared with PSO, HS, IHS, and FFA on the MNIST, CalTech 101 Silhouettes, Semeion datasets. The proposed HHPO achieved the best test MSE across almost all datasets except for on Semeion data with a one layer DBN trained using CD. The p-value results obtained by the Wilcoxon test demonstrated that HHPO was statistically better than other algorithms.

Lin et al. (2019) proposed a DBN optimized with GA for predicting wind speed and weather-related data collected from Taiwan's central weather bureau. The seasonal autoregressive integrated moving average (SARIMA) method and the least squares support vector regression for time series with genetic algorithms (LSSVRTSGA) were used to forecast wind speed in a time series, and the least-squares support vector regression with genetic algorithms (LSSVRGA). In DBN, the momentum and learning rate was considered in both unsupervised learning and supervised learning stages. Two parameters of the LSSVR and LSSVRTS models were tuned by GA using negative RMSE values as the fitness function. DBNGA models outperformed the other models according to forecasting accuracy.

Guo et al. (2019) combined GA and TS in hyper-parameter tuning to avoid wasting redundant computational budget arising from repeated searches. They reported that grid search is not suitable as the number of hyper-parameters increases. To avoid repeated searches in GAs, the proposed method integrated a Tabu list to GA (TabuGA) and was validated on the MNIST and Flower5 (Mamaev 2018) datasets. It was indicated that the Flower5 dataset was more sensitive to the values of hyper-parameters than the MNIST dataset was. The experiment on the MNIST dataset was performed on the classic LeNet-5 convolutional neural network while the experiment on the Flower5 dataset was carried out using a four-layer CNN structure. In the LeNet architecture, learning rate, batch size, number of F1 units, dropout rate, l2 weight decay parameters were optimized. In the CNN architecture, learning rate, batch size, number of F1 units, number of F2 units, number of F3 units, number of F4 units, l2 weight decay, dropout rate1, dropout rate2, dropout rate3, dropout rate4 parameters were evolved during optimization. Both results of experiments on MNIST and Flower-5 showed that TabuGA was preferable by reducing the required number of evaluations to achieve the highest classification accuracy. The proposed TabuGA was superior to the existing popular methods, including RS, SA, CMA-ES, TPE, GA, TS.

Yuliyono and Girsang (2019) proposed ABC to automate the search for hyper-parameters of LSTM for bitcoin price prediction. The sliding window size, number of LSTM units, dropout rate, regularizer, regularizer rate, optimizer and learning rate were encoded in a solution and RMSE was used as the fitness function. ABC-LSTM revealed an RMSE of 189.61 while LSTM without tuning resulted in an RMSE of 236.17, and the ABC-LSTM model outperformed the latter.

5.2.2 Conference papers

Zhang et al. (2015) applied an ensemble of deep learning network (DBN) and MOEA/D in diagnosis problems with multivariate sensory data. A multi-objective ensemble learning was conducted to find optimal ensemble weights of deep NNs. Diversity and accuracy as the conflicting objectives were employed with MOEA/D to adjust the ensemble weights. Turbofan engine degradation dataset (Saxena and Goebel 2008) was used to validate the efficacy of the proposed model. The results indicated that the average accuracy of multi-objective DBN ensemble learning schemes outperformed mean and majority voting ensemble schemes.

Rosa et al. (2016) used the Firefly Algorithm (FFA), PSO, HS, IHS algorithms to properly fine-tune RBM and DBN hyper-parameters in the learning step, including learning rate η, weight decay λ, penalty parameter α, and the number of hidden units n. They considered three DBN structures: one layer, two layers, and three layers. The experiments were performed on MNIST (Lecun et al. 1998), CalTech 101 Silhouettes (Li et al. 2003), Semeion (Semeion 2008) datasets. The experiment results showed that FFA produced the best results with fewer number of layers and less computational effort compared to the other optimization techniques considered in the study.

Nalepa and Lorenzo (2017) introduced PSO and parallel PSO techniques for automatically tuning DNN hyper-parameters, including the receptive field size, number of receptive fields, stride size, receptive field size. The experiments were performed for several DNN architectures on multi-class benchmark datasets, MNIST (Lecun et al. 1998) and CIFAR-10 (Krizhevsky 2009). They also analyzed the convergence abilities of their sequential and parallel PSO for optimizing hyper-parameters in DNNs using the Markov-chain theory. The convergence time, the number of processed generations, and the number of visited points in the search space were examined according to the values of DNN hyper-parameters. It was shown that adding new particles to the swarm increased the computational cost significantly while improving the generalization performance of the final DNN slightly. Therefore, smaller swarms were noted to be preferable in practice.

Hossain and Capi (2017) integrated DBN with NSGA-II for real-time object recognition and robot grasping tasks. The number of hidden units and the number of epochs in each hidden layer were considered as decision variables to optimize accuracy and network training time. Normalization and shuffling operations were performed in preprocessing step. As a result of the optimization step, the number of hidden units was determined as 515, 707 and 1024 in three hidden layers, respectively. The number of epochs iterated in three hidden layers was 129, 153 and 184, respectively. Experimental results showed that the approach proposed method was efficient in object recognition and robot grasping tasks.

Qolomany et al. (2017) utilized a PSO algorithm to optimize parameter settings to reduce computational resources in the tuning process. The deep learning models were trained to predict the number of occupants and their locations on a dataset from a Wi-Fi campus network. PSO was used to optimize the number of hidden layers and the number of neurons in each layer for deep learning models. They reported that the proposed method was an efficient approach for tuning the parameters and decreased the training time compared to the grid search method.

Fujino et al. (2017) proposed a GA-based deep learning technique for the recognition of human sketches. They focused on the evolution of hyper-parameters in AlexNet, including the number of filters, filter sizes and, max-pooling sizes. In order to reduce the training time of GA-based CNN evolution, a memory was integrated into GA to retrieve the fitness of a solution when it was encountered in the subsequent generations instead of recalculating the fitness. The proposed approach was applied to sketch recognition dataset that contains 250 object categories (Eitz et al. 2012). The proposed method showed higher performance even on smaller size images and provided high generalization ability.

Bochinski et al. (2017) introduced an EA-based hyper-parameter optimization by a committee of multiple CNNs. An individual was represented by the hyper-parameters h, which is a set of the configuration tuple of the ith layer and the layer sizes. A one-point crossover and the variation and the elimination or creation of a gene mutation operators were applied in $(\mu + \lambda)$ EA. They compared the performances of the evolutionary-based hyper-parameter optimization for a committee of independent CNNs. The experiments were validated on the MNIST dataset and significant improvements were reported over the state-of-the-art.

Sinha et al. (2018) optimized CNN parameters by PSO on CIFAR-10 (Krizhevsky 2009) and road-side vegetation dataset (RSVD). Each particle was represented by a binary vector of 4 input image sizes, 8 filter sizes, 8 number of filters, and 2 architectures. Top-1 score error measurement was used to evaluate the performance of each approach. CNN optimized by the proposed approach revealed a higher accuracy than the grid search and standard Alexnet classification for the CIFAR-10 dataset. For RSVD, the best network obtained a higher accuracy than the grid search approach and class-semantic color-texture textons approach. It was shown that increasing the swarm size and the number of iterations increase accuracy.

Sun et al. (2018b) investigated stacked AE models, including AEs, SAEs, DAEs, and CAEs optimized using PSO and compared them to the grid search method. Stacked AEs were preferred due to their small number of hyper-parameters that the grid search method can exploit all combinations within the given experimental time slot. In the proposed method, the hyper-parameters including the number of neurons n, weight balance factor λ, predefined sparsity ρ of SAE, sparsity balance factor β of SAE, corruption level z of DAE, contractive term balance factor γ of CAE were considered during optimization. The experiments were validated on the MINST dataset variations (Larochelle et al. 2007) including the MNIST with Background Images (MBI), Random Background (MRB), Rotated Digits (MRD), with RD plus Background Images (MRDBI), the Rectangle, the Rectangle Images (RI), and the Convex Sets (CS) benchmarks. They reported that AE models optimized by PSO could achieve the comparative classification accuracy but only took 10% to 1% computational complexity to that of grid search.

Agbehadji et al. (2018) introduced an approach to adjust the learning rate of LSTM by using two different aspects of Kestrel bird behavior (KSA). Six biological datasets (Carcinom, SMK_CAN_187, Tox_171, CLL_SUB_111, Glioma, Lung) (Li et al. 2018) were used in the experiments. The results showed that out of the six datasets, KSA has the best learning parameter (highlighted in bold) in three datasets.

Sabar et al. (2017) proposed an evolutionary hyper-heuristic (EHH) framework for automatic parameter optimization of DBN, which was tedious and time-consuming when handled manually. EHH performed a high-level strategy and a pool of large sets of evolutionary operators such as crossover and mutation and employed a non-parametric statistical test to identify a subset of effective operators. The number of hidden units, learning rate, penalty parameter, and weight decay were the hyper-parameters optimized using the mean squared error as the objective function. Three datasets, MNIST, CalTech 101 Silhouettes, Semeion Handwritten Digit datasets were used to evaluate the performance of the proposed approach. Three different DBN models using three different learning algorithms were tested, and the results of the proposed approach were compared to those of existing metaheuristics based DBN optimization methods. The results demonstrated that the proposed approach was competitive compared to state-of-the-art methods.

Sehgal et al. (2019) employed GA for tuning hyper-parameters used in Deep Deterministic Policy Gradient (DDPG) combined with Hindsight Experience Replay (HER) to speed up the learning agent. GA maximized task performance and minimized the number of training epochs considering the following parameters: discounting factor γ, polyak-averaging coefficient τ, learning rate for critic network α_{critic}, learning rate for actor-network α_{actor}, percent of times a random action is taken ϵ. The experiments were validated on fetch-reach, slide, push, pick and place and door opening robotic manipulation tasks. They reported that the proposed method could find the parameter values yielding faster learning and better or similar performance at the chosen tasks.

5.2.3 Others

Jaderberg et al. (2017) introduced a Population-Based Training (PBT) algorithm to jointly optimize a population of models and their hyper-parameters to maximize performance with a fixed computational budget. PBT was validated on reinforcement learning, supervised learning for machine translation, and training of Generative Adversarial Networks. They evaluated the GAN model by a variant of the Inception score on a pretrained CIFAR classifier, which used a much smaller network and made evaluation much faster. PBT optimized the discriminator's learning rate and the generator's learning rate separately. In all cases, PBT produced stable training and a better final performance by the automatic discovery of hyper-parameter schedules.

Steinholtz (2018) evaluated the Nelder-Mead Algorithm, PSO, Bayesian Optimization with Gaussian Processes (BO-GP) and Tree-structured Parzen Estimator (TPE) for two hyper-parameter optimization problem instances. In the first experiment, LeNet5 structure was adopted for the CIFAR-10 dataset. In the second experiment, an RNN model of LSTM-units was applied to a language modeling problem on the PTB dataset to predict the next word in a sentence by a given history of previous words. In LeNet structure, learning Rate, number of filters C1, number of filters C2, receptive field size C1, receptive field size C2, receptive field size P1, receptive field size P2, stride size P1, stride size P2, number of neurons FC1, number of neurons FC2, dropout probability FC2 parameters were optimized. In LSTM, learning rate, learning rate decay, pre-decay epochs, maximum gradient norm, initialization scale, number of layers, layer Size, number of steps, dropout probability parameters were optimized. Experimental results showed that the TPE algorithm achieved the highest performance according to mean solution quality for both problems. The NM, PSO and BO-GP algorithms outperformed the RS for the first experiment.

These studies are summarized in Table 1.

5.3 Studies using metaheuristics for training deep neural networks

5.3.1 Articles

Rere et al. (2015) proposed SA to train CNN on MNIST (Lecun et al. 1998) dataset that consists of digital images of handwriting. The best solution produced by SA was stored as the weights and bias values of CNN, and then the output of CNN was used to compute the loss function. When various neighbourhood sizes including 10, 20 and 50 were investigated, the percentage accuracy of CNN trained by SA was better when the neighbourhood size was 50. They concluded that although the computation time increased, the accuracy of the model was improved by the proposed method compared to the traditional CNN.

Rere et al. (2016) proposed SA, DE, and Harmony Search (HS) to improve the accuracy of CNN by computing the loss function of vector solution or the standard error on the training set. The experiments were performed on MNIST and CIFAR-10 datasets. The proposed methods and the original CNN were compared to LeNet-5 architecture. CNNSA produced the best accuracy for all epochs. The computation time of CNNSA was in the range of 1.01 times compared to the original CNN. They concluded that although the proposed methods showed an increase in the computation time, the accuracy achieved by them was also improved up to 7.14 percentage.

Syulistyo et al. (2016) utilized PSO in CNNs to improve the recognition accuracy of neural networks on the handwritten dataset from MNIST. The system gained 95.08% in 4 epochs, which was better than the conventional CNN and DBN without increasing the execution time. Besides, they concluded that CNN trained by PSO has better accuracy compared to CNN trained by SA and CNNPSO had the advantages of fast searching ability with the minimum iteration of CNN. In terms of execution time, DBN using

Table 1 The list of studies related to tuning hyper-parameters of deep neural networks using metaheuristics

No	Year	Type	Study	DL model	Metaheuristics	Dataset
1	2015	C	Rosa et al. (2015)	CNN	HS, IHS, GHS, SGHS	MNIST (Lecun et al. 1998), CIFAR-10 (Krizhevsky 2009)
2	2015	C	Zhang et al. (2015)	DBN	MOEA/D	NASA's C-MAPSS data set (Saxena and Goebel 2008)
3	2016	C	Rosa et al. (2016)	DBN	FFA, PSO, HS, IHS	MNIST (Lecun et al. 1998) CalTech 101 Silhouettes (Li et al. 2003) Semeion (Semeion 2008)
4	2017	C	Nalepa and Lorenzo (2017)	DNN, LeNet-4	PSO	MNIST (Lecun et al. 1998), CIFAR-10 (Krizhevsky 2009)
5	2017	C	Hossain and Capi (2017)	DBNN	NSGA-II	Object recognition, Robot grasping task
6	2017	A	Zhang et al. (2017)	DBN	MOEA/D	NASA's C-MAPSS data set (Saxena and Goebel 2008)
7	2017	A	Delowar Hossain and Capi (2017)	DBNN	GA	1800 images of six objects
8	2017	A	Hossain et al. (2018)	DBNN	GA	1800 images of six objects
9	2017	R	Jaderberg et al. (2017)	GAN	PBT	CIFAR-10 (Krizhevsky 2009) ImageNet (Russakovsky et al. 2015)
10	2017	C	Qolomany et al. (2017)	DNN	PSO	UoH Wifi (Qolomany et al. 2017)
11	2017	C	Fujino et al. (2017)	AlexNet	GA	Human Sketches (Eitz et al. 2012)
12	2017	C	Bochinski et al. (2017)	CNN	EA	MNIST (Lecun et al. 1998)
13	2018	A	Lopez-Rincon et al. (2018)	CNN	EA	miRNA sequences (The Cancer Genome Atlas (TCGA) 2006)
14	2018	A	Nakisa et al. (2018)	LSTM	DE	Emotiv, Empatica E4 (Nakisa et al. 2018)
15	2018	C	Sinha et al. (2018)	CNN	PSO	CIFAR-10 (Krizhevsky 2009), RSVD (Sinha et al. 2018)
16	2018	C	Sun et al. (2018b)	AE Models	PSO	MINST variations (Larochelle et al. 2007)
17	2018	C	Agbehadji et al. (2018)	LSTM	KSA	Biological Dataset (Li et al. 2018)
18	2018	A	Loussaief and Abdelkrim (2018)	CNN	GA	Caltech-256 (Griffin et al. 2007)
19	2018	T	Steinholtz (2018)	LeNet-5, LSTM	TPE, NM, BO-GP, PSO	CIFAR-10 (Krizhevsky 2009), PTB (Mitchell et al. 1999)
20	2018	A	Soon et al. (2018)	CNN	PSO	XMU VLR (Huang et al. 2015)
21	2018	A	Hinz et al. (2018)	CNN	RS, TPE, SMAC, GA	CK+ (Lucey et al. 2010), STL-10 (Coates et al. 2011) Flowers 102 (Nilsback and Zisserman 2008)

Table 1 (continued)

No	Year	Type	Study	DL model	Metaheuristics	Dataset
22	2019	A	de Rosa and Papa (2019)	DBN	GP	MNIST (Lecun et al. 1998) CalTech 101 Silhouettes (Marlin et al. 2010) Semeion (Semeion 2008)
23	2019	A	Yoo (2019)	CNN, AE	uDEAS, PSO, SA, GA	MNIST (Lecun et al. 1998)
24	2019	C	Sabar et al. (2017)	DBN	EHH, FFA, ES, PSO, IES	MNIST (Lecun et al. 1998) CalTech 101 Silhouettes (Marlin et al. 2010) Semeion (Semeion 2008)
25	2019	A	Wang et al. (2019b)	AlexNet, VGGNet-16, VGGNet-19, GoogleNet, ResNet-52, RestNet-101, DenseNet-121	cPSO-CNN	CIFAR-10 (Krizhevsky 2009)
26	2019	C	Sehgal et al. (2019)	DDPG	GA	Robotic manipulation
27	2019	A	Sabar et al. (2019)	DBN	HHPO	MNIST (Lecun et al. 1998) CalTech 101 Silhouettes (Marlin et al. 2010) Semeion (Semeion 2008)
28	2019	A	Lin et al. (2019)	DBN	GA	Wind speed and weather data in Taiwan (TCWB 2019)
29	2019	A	Guo et al. (2019)	LeNet-5, CNN	RS, SA, CMA-ES TPE, GA, TS, TabuGA	MNIST (Lecun et al. 1998), Flower5 (Mamaev 2018)
30	2019	A	Yuliyono and Girsang (2019)	LSTM	ABC	Bitcoin Price Prediction

A: Article, C: Conference Paper, O: Others

contrastive divergence was very fast while CNNPSO consumed more time due to extra calculations in the PSO algorithm.

Badem et al. (2017) combined ABC and L-BFGS method for tuning the parameters of a DNN. The DNN architecture used in the study included one or more AE layers cascaded to a softmax layer. The proposed method combines the exploitation ability of L-BFGS's and the exploration capability of ABC to avoid getting trapped minima. The performance of the proposed method was validated on the 15 benchmark data sets from UCI (Dua and Graff 2017) and compared with the state-of-the-art classifiers, including MLP, SVM, KNN, DT and NB. It was reported that the proposed hybrid strategy achieved better classification performance on a wide range of the data sets and boosted the performance of the DNN classifier on the data sets consisting of binary and multiple classes.

Deepa and Baranilingesan (2018) designed a DLNN model predictive controller to investigate the performance of a non-linear continuous stirred tank reactor (CSTR). The train data was collected from a state-space model of a CSTR. The weights of DLNN were tuned by a hybrid model of PSO and Gravitational Search Algorithm (GSA) to avoid the random initialization of the weights. The hybrid model aimed to overcome the inefficiency of each individual algorithm in exploration and exploitation. The effectiveness of the proposed approach was compared with those of some state-of-the-art techniques, and it was reported to achieve better minimal integral square error.

Rashid et al. (2018) implemented an LSTM to overcome the weaknesses of NNs related to learning speed, error convergence and accuracy due to long-term dependencies. They employed HS, Gray Wolf Optimizer (GWO), Sine Cosine Algorithm (SCA), and Ant Lion Optimization Algorithm (ALOA) to train LSTM for classification of real and medical time series data sets (Breast Cancer Wisconsin Data Set and Epileptic Seizure Recognition Data Set). Classification accuracy measure was used as a cost function in the training phase and the 5-fold cross-validation is performed to verify the results. They concluded that as the number of individuals in the population increased, the performances of the algorithms were improved. The performances of the models were not affected by the increase in the number of neurons in the hidden layer for breast cancer dataset while those deteriorated for epileptic seizure recognition dataset. The results indicated that training time, communication overhead and accuracy were improved.

Banharnsakun (2018) proposed a distributed ABC algorithm to discover initial weights of a CNN classifier based on solutions. The method was compared to other existing algorithms including the ordinary CNN, the CNN-SA, CNN-GA and the CNN-PSO. The method was validated on MNIST dataset, and a sevenfold cross-validation method was performed during training. Experimental results showed that the proposed method enhanced the CNN performance in both recognition accuracy and computing time compared to other approaches in the study.

Lakshmanaprabu et al. (2019) presented a study on the classification of lung CT images using an Optimal Deep Neural Network (ODNN) and Linear Discriminate Analysis (LDA). Features were extracted from the images in the database comprised of 50 low-dosage and recorded lung cancer CT images (VIA/I-ELCAP 2019) and reduced by LDA method. These features were given to ODNN, which worked with the help of DBN and RBM. The best solutions obtained from Modified Gravitational Search Algorithm (MGSA) (Rashedi et al. 2009) were set as the weights of ODNN. The proposed model was compared with existing classifiers like MLP, RBF, SVM, ANN, KNN, DNN. The comparative results showed that ODNN produced the accuracy of 94.56%, a sensitivity of 96.2%, a specificity of 94.2%, and outperformed the other methods.

Wang et al. (2019a) combined the back-propagation algorithm and EA for optimizing generative adversarial networks (GANs). They aimed to remove training problems such as instability and mode collapse. A population of generators was evolved to play the adversarial game with the discriminator. Each individual was updated based on different adversarial training objectives. The fitness of each individual was measured by the quality and diversity of generated samples. The performance of E-GAN was evaluated on CIFAR-10, LSUN bedroom (Yu et al. 2015), and CelebA (Liu et al. 2015) datasets. They modified both the generative and discriminator networks based on the DCGAN architecture. The batch norm layers were removed from the generator, and more feature channels were connected to each convolutional layer. Maximum Mean Discrepancy, the Inception score and the Frechet Inception distance were used as quantitative metrics. From the experiments, it was concluded that the proposed generative network improved the training stability and achieves good performance in image generation.

5.3.2 Conference papers

David and Greental (2014) applied GA to train deep NNs. They represented the multiple sets of encoding and decoding weights as a chromosome in GA. The root mean squared error (RMSE) was calculated in the training phase. Results indicated that the proposed approach produced an efficient deep AE and a sparser neural network. The worst half of GA-population was replaced with the best chromosomes in the population. A comparison was conducted between the results of GA, and the least reconstruction error (best tuned) of backpropagation algorithm repeated 10 times. According to the results, the GA-assisted method yielded a smaller reconstruction error, as well as a sparser network. The representation quality of the networks was compared by a Support Vector Machine with a radial basis function (RBF) kernel. The traditional AE achieved a 1.85% classification error by SVM, while the GA-assisted method produced a 1.44% classification error.

Ayumi et al. (2016) presented a Microcanonical Annealing (MCA) to optimize CNN according to the accuracy and the loss function. From the experimental results on MNIST dataset, they reported an increase in computation time (1.02×–1.38×) while the proposed method enhanced the performance (up to 4.60%) compared to traditional CNN. On the CIFAR10 (Krizhevsky 2009) dataset, the proposed method achieved 99.14% accuracy. They compared the proposed approach with some approaches including CNN using Harmony Search (HS), Fractional Pooling, Large ALL-CNN, Spatially Sparse CNN, LSUV and CNN. The proposed approach was shown to have better accuracy against these methods.

Tirumala et al. (2016) presented an analysis of optimizing the learning process inspired by the existing unconventional approaches. The study dealt with the problem of slow training procedure by introducing EC co-evolution strategies to reduce training time. Competitive co-evolution (CCEA) and Cooperative co-evolution (COCA) both combined the information of other individuals to improve efficiency. In the proposed method, there were two populations: one is for a competitive strategy to identify the fittest individual, and the other one is for a cooperative strategy to construct an optimal solution. They interchanged their best solutions by a migration strategy. An automatic self-destruction process was executed at regular intervals to remove non-active individuals. The experiments were performed on a 5-layered DNN trained layer-wise by gradient descent and back-propagation for the entire network. An evolved DNN-CCEA and a DNN-COCA were employed for classification purpose on MNIST and IRIS data sets. Their approaches improved the classification accuracy on both MNIST and IRIS datasets, and they empirically shown that the training process of DNN with EC approach was faster than the regular approach considerably.

Liu et al. (2017a) proposed an evolutionary gradient descent algorithm (EGD) which combined ES and the gradient descent methods to be used in the training phase of deep belief networks which was composed of the stacked RBMs and a softmax classifier. The experiments were validated on MNIST and Wine datasets, and the results were compared to those of DBNs optimized by the traditional gradient descent method, the combination of the stacked RBMs and SVM, and the combination of the stacked RBMs and the ES-based softmax classifier. On MNIST and Wine datasets, the EGD-based softmax classifier obtained the lowest error while SVM produced the lowest time cost. It was observed that the difference of training and test errors for the EGD-based softmax classifier was far below that for SVM. It was shown that the EGD-based softmax classifier could avoid overfitting problem better. The experimental results demonstrated that the proposed method had higher classification accuracy in less time and anti over-fitting ability compared to other classification methods. The proposed method was reported to have accelerated the convergence rate and improved the exploitation ability of the gradient descent method.

Alvernaz and Togelius (2017) introduced a scalable low-dimensional and hierarchal representation and analyzed it in the VizDoom environment built on the classic FPS Doom. In the approach, an autoencoder generated a compressed representation of a three-dimensional video game. The AE was trained using the backpropagation algorithm to reproduce input images of the environment. The compressed representation was cascaded to a smaller behaviour-generating NN optimized by CMA-ES method. The paper highlighted that training with compressed data had advantages overtraining with the raw visual data. They concluded that using EAs on the compressed representation could improve agent behaviour.

Chhabra et al. (2017) proposed a hybrid PSO algorithm to tackle with the training complexity problem of CNNs by reducing the number of training epochs. The proposed CNNPSO was compared with other deep learning methods based on the accuracy and the error on MNIST dataset. In the approach, CNN was first trained with backpropagation to establish some predefined weights for the neural network, and then these weights were encoded as one-hot vector corresponding to particles updated by PSO updates, iteratively. The proposed algorithm avoided the local minima problem of the regular backpropagation and improved the accuracy 3–4% in less number of epochs.

Khalifa et al. (2017) employed a PSO algorithm to train the output layer weights in ConvNet in order to optimize the accuracy on MNIST data sets. The first six layers were optimized by SGD algorithm while the last layer was optimized by PSO. The method produced better accuracy compared to the ConvNet which used SGD in all layers.

Muñoz-Ordóñez et al. (2018) designed a framework called Deep Neuroevolution Framework (DNF) based on a range of different metaheuristics including PSO, Global-best Harmony Search (GBHS) and DE algorithms. The framework was applied to regression and classification of handwritten dataset MNIST. In the experiments, Adam and RMSProb optimizers were integrated with the same DNF architecture and compared to each other. They compared the best results of the metaheuristics and their memetic version using the cross entropy-based cost function. When GBHS used RMSProp as its local optimizer, it obtained a better performance than when the Adam optimizer was used. In the second experiments, the effect of the number of training periods in the local search within GBHS was evaluated. It was stated that GBHS obtained better convergence rate, and it was superior over RMSProp as the number of training epochs increases.

Kösten et al. (2018) used an improved PSO in the training of LeNet-5 architecture for classification purpose on MNIST dataset. Each particle was a vector of all parameters in the network to be trained. It was observed that the classification performance varied based on the number of particles. When the number of particles was 20, the accuracy was about 90% while that was 96.29 when the number of particles was 64%.

Cai et al. (2018b) introduced Nondominated Sorting Genetic Algorithm II (NSGA-II) in the training of stacked AE to avoid the vanishing gradient (or exploding gradient) and local optima problems of gradient-based methods. In the approach, encoder's weights and decoder's weights of the auto-encoder were expressed as a closed-form solution, and the weights were optimized by NSGA-II. They introduced a Ridge Regression (RR) classifier to finish the final classification. The proposed approach was compared with existing architectures on MNIST, CIFAR-10, USPS, Pendigits, Optdigits datasets. PCA was applied to reduce the feature dimension to 20 to all the datasets, and then the max-min scale was performed to normalize each dataset. In training, NSGA-II algorithm optimized two objective functions. The first one is the L1-norm of the hidden-layer outputs, which guarantees sparsity and alleviates overfitting. The second objective function was the mean square error between target outputs and the actual decoder's outputs. Once NSGA-II was applied to optimize objective functions, then the solution which maximizes the second objective was selected as the final solution. The approach was compared to two ELM-based methods: Extreme Learning Machine and Multi-layer Extreme Learning Machine, two Stacked AE, neural networks trained by Stochastic Gradient Descent: SAE and SAE with L1 constraint, Multi-layer Perception trained by SGD and two traditional algorithms: Ridge Regression and Logistic Regression classifiers. The proposed method was shown to be more stable compared to ML-ELM and SAE-L1 when the depth was increased. Based on sparsity investigation, SEvoAE restricted the redundant neurons straightforwardly, which alleviated the overfitting problem. According to test classification accuracy on 7 datasets, SEvoAE produced an improvement of 1.21,2.05, and 0.98% compared to ML-ELM, SAE and SAE-L1, respectively. They concluded that the resulting architecture could obtain better results with a simple structure, and optimizing the multiple objectives simultaneously avoided adjusting hyper-parameter.

Zavalnyi et al. (2018) extended their previous work on favourable gradients optimized by ES. The training loss was considered as the estimation of the objective function value (fitness). Two different optimizers were used: SGD and RMSprop. The approach was evaluated on the MNIST dataset. A comparison was performed among SGD with backpropagation, SGD with favourable gradients and RMSprop with favourable gradients. SGD with backpropagation achieved good results much faster. RMSprop with favourable gradients performed better than SGD does. In the case of using SGD optimizer, the number of evaluations needed for ES was higher than that needed for backpropagation. Because using only favourable gradients was quite slow, the combination of ES and backpropagation was also investigated for MNIST and CIFAR10 datasets. In the hybrid model, favourable gradients were used on every 25th batch, and backpropagation was used for the rest of the optimization steps. The combination of two methods improved the training results on both datasets. From the results, it was observed that when ES was applied to optimizing favourable gradients used by local optimizers, more stable convergence results were obtained.

5.3.3 Others

Lamos-Sweeney (2012) investigated GA in the training phase of deep networks, especially with a large number of layers. Layer-wise training was adopted to reduce the computational cost and increase the flexibility of the algorithm. The reconstruction error of the algorithm

Table 2 The list of studies related to training deep neural networks using metaheuristics

No	Year	Type	Study	DL model	Metaheuristics	Dataset
1	2012	O	Lamos-Sweeney (2012)	DNN	GA	Handwriting, Cat, Face
2	2014	C	David and Greental (2014)	AE	GA	MNIST (Lecun et al. 1998)
3	2015	A	Rere et al. (2015)	CNN	SA	MNIST (Lecun et al. 1998)
4	2016	C	Ayumi et al. (2016)	CNN	MCA	MNIST (Lecun et al. 1998), CIFAR-10 (Krizhevsky 2009)
5	2016	A	Rere et al. (2016)	LeNet-5	SA, DE, HS	MNIST (Lecun et al. 1998), CIFAR-10 (Krizhevsky 2009)
6	2016	A	Syulistyo et al. (2016)	CNN	PSO, SA	MNIST (Lecun et al. 1998)
7	2016	C	Tirumala et al. (2016)	CNN	CCEA, COCA	MNIST (Lecun et al. 1998), Iris
8	2017	A	Badem et al. (2017)	DNN	L-BFGS + ABC	15 datasets from UCI (Dua and Graff 2017)
9	2017	C	Liu et al. (2017a)	DBN	EGD	MNIST (Lecun et al. 1998), Wine (Dua and Graff 2017)
10	2017	C	Alvernaz and Togelius (2017)	AE + BGN	CMA-ES	Doom Game
11	2017	C	Chhabra et al. (2017)	CNN	PSO	MNIST (Lecun et al. 1998)
12	2017	C	Khalifa et al. (2017)	ConvNet	PSO	MNIST (Lecun et al. 1998)
13	2018	A	Deepa and Baranilingesan (2018)	DLNN	PSO + GSA	the state space model of CSTR
14	2018	A	Rashid et al. (2018)	RNN + LSTM	HS, GWO, SCA, ALOA	Breast Cancer (Dua and Graff 2017) Epileptic Seizure Recognition (Andrzejak et al. 2001)
15	2018	C	Muñoz-Ordóñez et al. (2018)	DNF	GBHS, PSO, DE	MNIST (Lecun et al. 1998)
16	2018	C	Kösten et al. (2018)	LeNet-5	IPSO	MNIST (Lecun et al. 1998)
17	2018	C	Cai et al. (2018b)	AE	NSGA-II	MNIST (Lecun et al. 1998) CIFAR-10 (Krizhevsky 2009), USPS (Hull 1994) Pendigits and Optdigits (Dua and Graff 2017)
18	2018	C	Zavalnyi et al. (2018)	DNN	BP + ES	MNIST (Lecun et al. 1998), CIFAR-10 (Krizhevsky 2009)
19	2018	A	Banharnsakun (2018)	CNN	ABC	MNIST (Lecun et al. 1998)
20	2019	A	Lakshmanaprabu et al. (2019)	ODNN DBN + RBM	MGSA	Lung CT scans (VIA/I-ELCAP 2019)
21	2019	A	Wang et al. (2019a)	GAN	EA	CIFAR-10 (Krizhevsky 2009) LSUN bedroom (Yu et al. 2015) CelebA (Liu et al. 2015)

A: Article, C: Conference Paper, O: Others

was used to determine the fitness of each individual. In the study, data compression and object classification problems were considered on datasets such as handwriting, face and cat datasets. It was reported that if the resulting size of the deep network was insufficient to contain the data in the testing set, training error increased drastically.

These studies are summarized in Table 2.

5.4 Studies using metaheuristics for architecture optimization of deep neural networks

5.4.1 Articles

Miikkulainen et al. (2017) proposed a new approach called, CoDeepNeat, by extending NEAT Stanley and Miikkulainen (2002) for co-evolutionary optimization of DNN components, topologies, and hyper-parameters. The fitness function was established based on training error. There were two populations of modules and blueprints that were evolved independently. Each module denoted a small DNN and blueprint chromosome was a graph of pointers (nodes) each for a specific module type. Although the module and blueprint chromosomes were evolved separately, they were combined together and a larger network was constructed during evaluation. The approach was used to evolve a CNN topology on CIFAR-10 dataset. The approach was integrated with two types of mutations switching the connections between nodes and layers and applied to an LSTM topology optimization on Penn Treebank benchmark (Mitchell et al. 1999) task of language modeling. It was also utilized in a real-world application of automated image captioning on MSCOCO dataset (Chen et al. 2015b) and a new dataset.

(Liu et al. 2017b) introduced hierarchical representation which encodes several motifs at different levels of hierarchy, where lower-level motifs were used as building blocks (operations) during the construction of higher-level motifs as stated in Sect. 4.2. Convolutions with larger receptive fields and channels were defined based on primitive operations including 1×1 convolution of C channels, 3×3 depth-wise convolution, 3×3 separable convolution of C channels, 3×3 max-pooling, 3×3 average-pooling and identity in a chain structure. Wider convolutions with more channels can be obtained by concatenating the outputs of multiple convolutions. All convolutional operations were followed by batch normalization and ReLU activation. A diversification-based scheme was used to generate the initial population by applying a large number of random mutations. A single mutation of a hierarchical genotype consisted of a set of actions, including sampling a target non-primitive level ≥ 2, sampling a target motif m in the target level, sampling a random successor node i in the target motif, sampling a random predecessor node j in the target motif, replacing the current operation between j and i with a randomly sampled operation. Tournament selection was applied to select parents and determine surviving offspring. The architecture was trained for a fixed number of iterations and the average accuracy was used as fitness function. The experiments were performed on MINST (Krizhevsky 2009) and ImageNet Russakovsky et al. (2015) datasets. It was shown that efficient results could be obtained using simple search algorithms based on a well-designed architecture representation.

Sun et al. (2018a) presented GA to evolve CNN architectures based on ResNet blocks and DenseNet blocks without any intervention in pre and post processing steps. The fully-connected layers were not considered in the approach. A variable-length encoding scheme was employed not to restrict the depth of the CNN. For ResNet block and DenseNet block, the number of blocks, the input spatial size and the output spatial size, an additional parameter *k* for DenseNet block. Pooling type was encoded for pooling layer. The one-point crossover operator and cascade adjustments were performed. Adding block, removing block and modifying the information operations were used in mutation operator and again some adjustments were performed. The approach was validated on CIFAR10 and CIFAR100 datasets Krizhevsky (2009). Experimental results revealed that the proposed approach was better than state-of-the-art CNNs in terms of the classification accuracy and computation time.

Liu et al. (2018) proposed a layer-wise structure learning method with high representation and better generalization ability. Therefore, the visible data was defined by the structure of many hidden units. An MOEA was adopted to optimize representation ability and connecting the sparsity of the network simultaneously. The experiments were implemented on single-layer level, hierarchical level, and application level, respectively. In the experiments, on the single-layer level, the improved MOEA optimized single-layer structure. In the experiments on the hierarchical level, the single-layer structures were stacked, and the deep structured network was evaluated on the the MNIST and CIFAR-10 datasets with different training samples. The experiments demonstrated the effectiveness of the improved MOEA and structure learning model.

Liu et al. (2019) presented a GA-based network evolution approach to optimize CNN network structures automatically. The evolutionary process was expedited through an experience-based greedy exploration strategy and transfer learning. A possible solution included the type of the optimizers, the choice of the activation function, the number of layers, and the number of neurons and evolution was applied to the complementary-gene set. The Peak Signal-to-Noise Ratio (PSNR) was used as the fitness function. The proposed model was evaluated on denoising of 10,775 computed tomography perfusion (CTP) images. The results of the proposed approach outperformed those of state-of-the-art methods at various noise levels.

Junior and Yen (2019) proposed a PSO algorithm with variable-length particles with virtually no size limitation. The encoding strategy represented an architecture with two blocks: one block included convolutional and pooling layers, while the other included only fully connected layers. The operators handled two solutions with a different number of layers and parameters. The fitness evaluation was performed by the loss function. The loss function of each particle was compared to the cross-entropy loss. The method was validated on MNIST, MINST variations, Rectangle, Convex (Larochelle et al. 2007) and Fashion (Xiao et al. 2017) datasets. The results demonstrated that the proposed method could find CNN models which were capable of achieving better results compared to more complex and complicated architectures.

Baldominos et al. (2019) improved the performance of the models by using committees using GA and grammatical evolution (GE) to automatically tuning the structure of CNNs that maximize accuracy. The GA encoding included a 69-bit binary string using gray encoding. Input configuration, convolutional layers, dense layers, and learning process parameters were encoded in a gene solution. Convolutional layer parameters were composed of the number of convolutional layers, for each convolutional layer, the number of kernels, kernel size, pooling size (or no pooling), and activation function. Dense layer parameters included the number of dense layers, for each dense layer, type of the layer (feedforward or recurrent), number of neurons, activation function, weights regularization (L1 or L2), and dropout rate. Learning process parameters included the optimizer and learning rate. The classification error of the network was assigned as the individual fitness, and the adjusted fitness provided by the niching was computed. Experiments were validated on the MNIST database, and then the optimal structure was transferred to EMNIST

(Cohen et al. 2017). The study reported that the error produced by a committee was less than the average error rate of its constituents individually.

Mattioli et al. (2019) used GAs in deep neural networks architecture selection. One descriptor for convolutional layers and second descriptor for fully connected layers were encoded in a solution. Convolutional layer descriptors consisted of the activation function, number of filters, kernel size, stride, and pooling size, while fully connected layer descriptors included the number of neurons and the dropout probability. The fitness function was represented by the test accuracy of the solution. MNIST and CIFAR-10 datasets were investigated to validate the model. Using a fitness predictor in the evaluation yielded a significant reduction in execution time, and the proposed method was able to find competing topologies compared to state-of-the-art methods.

Sun et al. (2019b) proposed a PSO model that used an encoding strategy for particles with nonidentical lengths to optimize stacking AEs. Each particle contained different numbers of convolutional layers and pooling layers. Each convolution filter had filter width, filter height, stride width, stride height, convolution type, number of feature maps, the coefficient ℓ_2 parameters whilst each pooling layer had kernel width, kernel height, stride width, stride height parameters. The objective function for training was defined by the reconstruction error. The experiments were conducted on MNIST, CIFAR-10, STL-10 (Coates et al. 2011), CalTech 101 (Fei-Fei et al. 2007). The experimental results showed that the proposed method outperformed the algorithms considered in the study in terms of accuracies.

Sun et al. (2019a) implemented a new variable-length encoding scheme, a new genetic operator and a slacked selection operator to evolve CNN architectures for image classification problems. The first part of the chromosome contained the convolutional layers and the pooling layers, and the second part is the full connection layers. Instead of representing hundreds of thousands connection weights of fully connected layer, the standard derivation and mean value of the connection weights were encoded and the connection weights were sampled from the corresponding Gaussian distribution based on these values. In each training step, the fitness of an individual was calculated by three components: classification error based on the current weights and the batch data, the mean and standard derivations errors of these batch data, and the number of weights. The approach utilized a slack version of the binary tournament selection according to thresholds for the mean values of individuals and the parameter numbers. The simulated crossover operator applied unit alignment and unit restore to produce offspring from parents with different chromosome lengths. For a selected mutation point, addition, deletion or modification was applied with a probability of 1/3. They also employed an environmental selection to address elitism and the diversity. The proposed approach was validated on Fashion (Xiao et al. 2017), MNIST Krizhevsky (2009) and MNIST variations Larochelle et al. (2007) datasets and compared to some state-of-the-art methods. It was reported that their approach outperformed them on almost all these datasets in addition to yielding a much smaller number of parameters.

Wei et al. (2019) proposed hybridization of PSO and GA algorithms to optimize the DBN's network structure for intrusion detection classification. PSO algorithm was combined with genetic operators with self-adjusting crossover probability and mutation probability. Accuracy, FPRate, FNRate, and DRate were used as evaluation indicators for optimization algorithm performance. The experimental results showed that the optimized intrusion detection model had higher detection speed and detection accuracy.

Shi et al. (2019) presented a PSO algorithm to optimize the number of hidden layer nodes of DNN for digital modulation recognition. The aim was to improve the performance of recognition under the condition of low signal-to-noise ratio (SNR). Experimental results demonstrated that the recognition rate was improved by 9.4% and 8.8% compared to conventional DNN and support vector machine (SVM) when SNR was 0dB and 1dB, respectively. Compared with GA, the proposed method was more effective in optimizing DNN.

Wang et al. (2019b) proposed a new encoding strategy coding architecture and shortcut connections separately. The proposed encoding strategy was incorporated with a combination of PSO and GA algorithms to optimize CNN architecture. The number of blocks and for each block, the number of convolutional layers, and the growth rate were encoded in the first dimension and then the shortcut connections were encoded in a binary vector at the second-level encoding. The accuracy of the trained model was treated as the fitness function. The proposed algorithm was evaluated on three widely used benchmark datasets from MNIST variations (Larochelle et al. 2007) and CIFAR-10, and compared with 12 peer Non-EC based competitors and one EC based competitor. The experimental results demonstrated that the proposed method outperformed the other methods in terms of classification accuracy.

Ye (2017) used PSO and the steepest gradient descent algorithm combination to determine the optimal network structure parameters, including learning rate, dropout rate, momentum, weight decay, and the number of neurons of each hidden layer. In the proposed approach, parameters corresponding to the structure were coded as a set of real-number m-dimensional vectors. The training accuracy or the last loss value were used as the score (fitness value) for the individual of the PSO algorithm. The approach was validated for classification purpose on MNIST (Lecun et al. 1998) and regression purpose on the Kaggle Competition biological dataset (KGD) (Kaggle 2017). Results showed that PSO combined with the steepest gradient descent algorithm produced better network structure and generalization ability.

Chiba et al. (2019) utilized Improved GA and SA to build a DNN automatically for anomaly Network Intrusion Detection System (IDS). Each solution represented parameters including normalization, activation function, number of nodes in hidden layer 1, number of nodes in hidden layer 2, learning rate, and momentum term. AUC metric was considered as fitness function to guide the search problem. Experiments were carried out on three IDS datasets, CICIDS2017 (CICIDS2017 2017), NSL-KDD (NSL-KDD 2019) and CIDDS-001 (CIDDS-001 2019) dataset and the results demonstrated that the proposed model could detect intrusions with high detection accuracy and low false alarm rate, and it was superior over state-of-the-art methods.

5.4.2 Conference papers

Lander and Shang (2015) proposed an EA to optimize network weights and structure of AEs in which each hidden node and its corresponding connections and weights were encoded in a solution of EA. The proposed method employed a mini-batch variant that accelerated search on large datasets. Starting from a pool of encoded solutions, the proposed method first applies backpropagation in training each AE to minimize the reconstruction error. Then, EA evolved the population in the feature and structure space of AE. Experiments were validated on the MNIST dataset, and it was reported that mini-batch and Evo-batch had compelling influence as data size grows. The proposed method produced an easily tunable AE and revealed smaller reconstruction errors against multi-start approaches.

Tanaka et al. (2016) explored network structure of LSTM recurrent neural network for speech recognition. They applied CMA-ES in which solutions encode layer-wise parameters. WER and computational time were minimized simultaneously by a Pareto based approach. Experiments were performed on the Corpus of Spontaneous Japanese (CSJ) (Furui et al. 2000) and the individual headset microphone (IHM) subset of the AMI meeting speech transcription corpus (Carletta et al. 2006). It was shown that CMA-ES improved WER on the datasets (CSJ and AMI) and multi-objective optimization reduced WER and computational time jointly.

Suganuma et al. (2017) proposed a Cartesian GP (CGP) encoding based evolution approach to assemble a ResNet architecture. The encoding scheme represented as directed acyclic graphs with a two-dimensional grid defined on computational nodes and was able to map variable-length network structures and skip connections. Highly functional modules, such as convolutional blocks and tensor concatenation, were adopted as the node functions in CGP. There were six types of node functions called ConvBlock, ResBlock, max pooling, average pooling, concatenation, and summation. The summation and concatenation operations were provided to represent shortcut connections or branching layers. A point mutation operator randomly changed the type and connections of each node. If the fitnesses of the offsprings did not improve, a parent was modified by the neutral mutation, which changed only the genes of the inactive nodes without the modification of the phenotype. After the CGP process, the best CNN architecture was trained using SGD. The performance of the validation dataset was used as the fitness of the individual. It was tested on CIFAR-10 and CIFAR-100 (Krizhevsky 2009) datasets and it was seen that the proposed approach could automatically find the competitive CNN architecture compared with the state-of-the-art models

Vidnerova and Neruda (2017) designed an algorithm for the optimization of a network architecture based on ES. In the approach, ES was used to search for the optimal architecture of DNN, while the weights were learned by gradient-based technique. The model implemented as sequential was built layer-by-layer. $I = ([size_1, drop_1, act_1, \sigma_1^{size}, \sigma_1^{drop}]_1, \ldots, [size_H, drop_H, act_H, \sigma_H^{size}, \sigma_H^{drop}]_H)$ where H was the number of hidden layers, $size_i$ was the number of neurons in corresponding layer that was dense (fully connected) layer, $drop_i$ was the dropout rate (zero value represents no dropout), $act_i \in \{relu, tanh, sigmoid, hardsigmoid, linear\}$ stand for activation function, and σ_i^{size} and σ_i^{drop} were strategy coefficients corresponding to size and dropout. In this paper, only fully connected feedforward layer was optimized. The approach was tested on MNIST and a real air pollution monitoring dataset (Vito et al. 2012). On the air pollution data set, the algorithm outperformed Support Vector Regression and manually tuned architectures. For the MNIST dataset, the network with ReLU and hard sigmoid unit outperformed the baseline solution. The main drawback of the proposed method was its computational cost.

Martín et al. (2017) proposed a GA to evolve the parameters and the architecture of a DNN to maximize the malware classification accuracy and to minimize the complexity of the model jointly. Each genome was composed of global network parameters and a variable-size section, which contained a set of layers. The global parameters included the optimizer type, the maximum number of epochs or iterations, the number of samples, while the layer-related parameters were the number of outputs, the initialization function, and the activation function. The fitness of each individual was calculated by accuracy. The proposed model was tested on a specific problem of determining the malware family of different malicious samples. The experimental results showed that the proposed approach successfully determined the parameters and enhanced the results achieved by Support Vector Machines trained with different kernel functions.

Xie and Yuille (2017) proposed an encoding scheme using fixed-length binary string initialized by sampling from a Bernoulli distribution. The approach utilized standard roulette wheel selection, bit flipping mutation and crossover operators. Recognition accuracy was used as objective function during training. It was validated on small scale CIFAR10 dataset and the learned patterns were transfered to large scale ImageNET dataset. Each solution represented a family of networks with S stages of ordered K_S nodes corresponding to convolutional operations that were performed after element-wise summing summation. They obtained interesting results although it was also noted that the fraction of network structures was limited.

Desell (2017) presented an evolutionary algorithm based algorithm to evolve structure of CNNs on MNIST dataset. The approach was built on the structure of CNN can be evolved by determining the filter sizes and how filters are connected. It was implemented in a multi-threaded manner. CNN genomes are evaluated in workers. When the number of genomes in the workers reaches to the predetermined population size, mutation (adding adges, enabling edges, disabling edges, splitting edges, and change node sizes) or crossover operations are performed. The experiments on MNIST and TinyImage (Torralba et al. 2008) datasets have shown that the approach improved the train and test accuracy compared to human designed benchmark neural networks.

Real et al. (2017) presented EA based technique to minimize manual crafting effort in end-to-end evolution and training as a one-shot model. Their approach considered no fixed depth, arbitrary skip connections, and numerical parameters that had few restrictions on the values they can take. To be able to reach a large scale architecture, a massively-parallel, asynchronous, lock-free infrastructure was used. Each architecture was encoded as a graph in which the vertices represent rank-3 tensors or activations. Activation functions that were applied at the vertices could be either batch-normalization with rectified linear units (ReLUs) or plain linear units. Edges in the graph corresponded to identity connections or convolutions and parameters. The learning-rate value was also encoded in the chromosome. To generate offspring, mutation operator was applied by randomly selecting one of the actions, including altering learning rate, identity, resetting weights, inserting convolution at a random location, removing convolution, altering stride, altering number of channels, changing filter size, inserting one to one identity connection, adding skip connection between randomly selected layers, removing skip connections randomly. The approach was validated on CIFAR-10 and -100 datasets and it was shown that evolutionary algorithm based evolution could explore a large space and construct accurate networks.

Kramer (2018) introduced convolutional highways which are based on multiple stacked convolutional layers for feature preprocessing. Each convolutional highway layer has two gates for the flow of information, followed by dense layers and a final softmax layer. The structure and hyper-parameters (kernel size of all highways, the kernel size of the max pooling layers, and the activation function types of all highways and of the dense layers) of convolutional highways were optimized using binary encoding (1+1)-EA which used adaptive flipping mutation based on Rechenberg's rule and a niching mechanism. EA evolved the learning rate of the network and applied Adam as gradient descent optimizer. The categorical cross-entropy was considered as fitness function. The approach was verified on MNIST (Krizhevsky 2009) data set.

Bibaeva (2018) implemented RS, GA, SA, MA to optimize CNN architecture on MNIST and CIFAR-10 datasets. Binary encoding of the entire architecture includes activation function, pooling type, filter size, step size, and the number of outputs. All layers were fixed to the same length according to layer count and input image size. It was assumed that evolving one excellent architecture and tuning its training parameters afterward was less time consuming than evolving many architectures together with their training parameters. Thus, each CNN was trained with Stochastic Gradient Descent using standard values for batch size, momentum, weight decay etc. It was shown the proposed approach achieved competitive results with state-of-the-art performance without user involvement and MA and SA algorithms outperformed the RS and GA for hyper-parameter optimization.

Falco et al. (2018) optimized architecture of DNN through the DE algorithm for Obstructive Sleep Anpnea classification. To decrease execution times, a distributed version of the DE algorithm was introduced and the size of the training set was reduced by a box-based reduction. The number of hidden layers, the number of neurons in each layer, the activation function and the number of learning steps were considered in the optimization. Experiments were carried out on a medical data set about Obstructive Sleep Anpnea (McNicholas and Levy 2000). It was shown that sub-optimal DNN hyper-parameters values were obtained in a much lower time without a detriment in the classification accuracy over the test set.

Pandey and Kaur (2018) used a GA to determine the best hyper-parameter settings for LSTM models. The population was initialized with random combinations of the number of layers, neurons per layer, dense layer activation function and network optimizer etc. The proposed model was applied in inhibiting the extensive spread of fake news and clickbait by utilizing the lexical and semantic features of the corresponding text. It was reported that LSTM designed based on GA provided an accuracy of 94.36% in LSTM and 95.61% in BiLSTM.

Martinez et al. (2018) integrated Dakota optimization library, TensorFlow, and Galaxy HPC workflow management tool to perform massively parallel function evaluations in a GA to avoid extensive hyper-parameter searches for multivariate regression for spectrum reconstruction problem. A deep regression was performed on input features for reconstructing a vibrational spectrum on specific HUMS configurations in rotorcraft. The number of neurons in the three hidden layers and the dropout rates in the three hidden layers, a total of six design variables were considered in the optimization. An initial random population was generated and genetic operators were employed to create the next generation. In order to evaluate the solutions, MSE between the predicted spectrum and the true spectrum was calculated as the loss function. An experiment was provided information about the influence of each hyper-parameter over the objective function. It was reported that the number of units in the first hidden layer correlated with the cost function significantly. GA improved the validation accuracy of several models by 5–7% in less than 72 h. Compared with random generated and hand-tuned models, their approach was significantly faster and better for tuning hyper-parameter configurations. It was noted that fine-tuning of mutation, crossover, and the initial population size was required to avoid local minima during training.

Ding et al. (2018) employed GA to learn the CNN structure for HSI classification. In this study, the first encoding area contained 3 nodes, the second encoding area contains 4 nodes, and the final stage contains 5 nodes. After the encoding process, GA was used to determine the best CNNs structure. The method was validated on the Indian Pines scene and Pavia University scene images. Experimental results demonstrated the effectiveness of the proposed CNNs framework in HSI classification accuracy.

Martín et al. (2018) proposed an EA to evolve the architecture of a DNN to optimize the classification accuracy. Individuals were generated by Finite-State Machine defining possible paths. Each genome was a set of global parameters which defines the general behaviour of the network, and a sequence of layers with an arbitrary number of them. Layer parameters were different for each layer in the architecture. Some of the latter were common to any kind of layer while others were specific to a type of layer. The proposed model was tested on the MNIST handwritten digits images. It was shown that EA was able to construct the DNN architecture with a 98.93% accuracy in the best run.

Tian et al. (2018a) proposed an approach based on GA to select or regenerate the best pre-trained CNN models for different visual datasets. Each individual represented a possible combination of the CNN model defined with a finite-length string. The fitness function was defined by the average F1-score from the validation data as the feedback to evaluate. In each generation, a set of pre-trained CNN models was selected, and an automatically created network with several dense layers was employed to leverage the deep feature representations. Then, a linear SVM classifier was trained on the deep features. They validated the approach on the CIFAR-10, Network Cameras (Pouyanfar et al. 2018), Youtube video (Tian et al. 2018b) datasets. The experimental results demonstrated that the proposed method outperforms several existing approaches in visual data classification.

Wang et al. (2018) utilized PSO to search for the optimal architecture of CNNs. The encoding strategy was inspired by Network IP addresses to be able to encode all the types of CNN layers. Each subnet can be used to define a specific type of CNN layer while the encoded information can vary according to types of layers. There were three parameters related to the convolution layers: filter size, number of feature maps, and stride size. As the particle length of PSO was fixed after initialization, another layer type called the disabled layer was introduced in order to cope with the variable-length CNN architectures. The mean value of the accuracies for each individual represented the individual fitness. The proposed algorithm was compared with 12 state-of-art methods on MINST variations including MNIST Basic, the MNIST with Rotated Digits plus Background Images (MRDBI) and the Convex Sets (CS) (Larochelle et al. 2007). The experimental results showed that the proposed algorithm achieved a competing accuracy and outperformed the other methods on the MDRBI benchmark dataset, being the second-best on the MNIST benchmark dataset and ranking above the middle line on the CS benchmark dataset.

Gibb et al. (2018) used GA to evolve the structure parameters of a CNN used in the detection of the concrete images (Cha et al. 2017) containing cracks. CNN in the study had three convolution layers, two pooling/subsampling layers, and two activation layers. GA was employed to tune the hyper-parameters, including convolution layer sizes, pooling layer sizes, activation function types, network depth, optimization function type, and learning rate of the optimization function. Stochastic gradient descent (SGD) was used for updating weights in training. The accuracy metric was assigned as the fitness function. Experimental results showed that it was possible to generalize the process of optimizing a CNN for image classification through the use of a GA.

Evans et al. (2018) proposed a GP approach to design CNNs automatically. A strongly-typed GP was used to enforce a tiered structure. The stages were roughly broken into tiers and then stacked to simulate the original image classification pipeline. The fitness function used was the proportion of correctly classified instances. The proposed approach was tested on four benchmark image datasets, Cars (Agarwal et al. 2004), Jaffe (Cheng et al. 2010), Faces (Sung and Poggio 1998), Pedestrian (Munder and Gavrila 2006). the experimental results showed that the method produced competitive performance compared to the state-of-the-art techniques and could automatically generate highly interpretable evolved programs.

Loni et al. (2018a) proposed a multi-objective framework to discover near-optimal DNN architectures in terms of the accuracy and the network size. The main architectural hyper-parameters of DNNs were encoded using direct encoding, and the recombination relied on one-point crossover operation. The proposed approach was integrated with NSGA-II to optimize both objectives jointly. The method was validated on

MNIST and CIFAR-10 datasets. The evaluation results demonstrated the effectiveness of the approach in terms of pure kernel time and communication time.

Loni et al. (2018b) presented a multi-objective CGP for encoding the genome as a directed graph that is mapped in a two-dimensional matrix. This representation defined two node types, active and inactive. The functionality of each node block was a composition of max-pooling, average pooling, concatenation, activation, summation, ConvBlock, and ResBlock. The convBlock included an ordinary convolution layer, batch normalization and activation, respectively. Moreover, ResBlock was a bit more complicated where it consisted of ConvBlock, convolution layer, batch normalization, and summation to perform element-wise addition on ConvBlock input and the normalization output. Finally, the output node was always a fully connected layer with the softmax activation function. The Score factor was calculated as the fitness function. The framework was validated on CIFAR-10 then compared to cutting-edge architectures. The results indicated that the proposed method improved the network architecture and efficiency.

Kumar and Batra (2018) optimized RNN with GA for various combinations of number network parameters, including the number of hidden layers, number of neurons per layer, activation function, and optimizer. The model was trained with a dataset that is sliced according to the best window size. RMSE on the validation set was utilized as the fitness in GA. The results were evaluated based on training time, error, and accuracy. With sample data streams, simple RNN produced higher accuracy than LSTM and GRU for single hidden layer architecture. It was reported that as the RNN went deeper, LSTM and GRU outperformed simple RNN, and accuracy was improved for all the architectures. LSTM showed better performance compared to Simple RNN and GRU with an equal number of layers, and the optimized RNN demonstrated competing performance with LSTM. The optimized version of RNN improved training time significantly, with a slight reduction in the accuracy.

Assuncao et al. (2018) investigated compressed data forms generated by AEs using EAs. The candidate solutions were represented with variable-length ordered lists of integers, where each integer represented the number of neurons of a specific layer. For each individual, the number of layers was selected, and only then the number of neurons in each layer was sampled. To train the AEs, the error between the reconstructed signal and the mean signal of the class was utilized as the fitness function. Experiments were performed on the MNIST dataset, and the results demonstrated that the proposed approach could reduce dimensionality with the advantage of superior performance on the original uncompressed images.

Ai et al. (2019) presented an LSTM for forecasting peak demands in a home/neighbourhood energy management system to obtain a more reliable prediction automatically. They selected network structures through empirical or enumerative approaches. In the study, an evolutionary ensemble method was introduced to train LSTM networks in order to solve the problems, including diverse learning efficiency and falling into a local minimum. Mean absolute percentage error (MAPE) was used to evaluate the peak demand prediction of an anonymous western Norwegian domestic household. They concluded that the proposed model achieved better performance than single LSTM network, and the evolutionary parameters had variant impacts on the model performance.

These studies are listed in Table 3.

Table 3 The list of studies related to discovering the optimal architecture of deep neural networks using metaheuristics

No	Year	Type	Study	DL model	Metaheuristics	Dataset
1	2015	C	Lander and Shang (2015)	AE	EA	MNIST (Lecun et al. 1998)
2	2016	C	Tanaka et al. (2016)	LSTM	CMA-ES	CSJ (Furui et al. 2000) and AMI corpus (Carletta et al. 2006)
3	2017	C	Suganuma et al. (2017)	CNN	CGP	CIFAR-10, CIFAR-100 (Krizhevsky 2009)
4	2017	C	Vidnerova and Neruda (2017)	DNN	ES	MNIST (Lecun et al. 1998), Air pollution dataset (Vito et al. 2012)
5	2017	A	Ye (2017)	CNN	PSO	MNIST (Lecun et al. 1998), KCD (Kaggle 2017)
6	2017	C	Martín et al. (2017)	DNN	GA	Malware Dataset
7	2017	C	Xie and Yuille (2017)	CNN	EA	CIFAR-10 (Krizhevsky 2009), ImageNet (Russakovsky et al. 2015)
8	2017	A	Miikkulainen et al. (2017)	CNN, LSTM	EA	CIFAR10 (Krizhevsky 2009), Penn Treebank (Mitchell et al. 1999) MCOCO (Chen et al. 2015b)
9	2017	C	Desell (2017)	CNN	EA	MNIST (Lecun et al. 1998), TinyImage (Torralba et al. 2008)
10	2017	C	Real et al. (2017)	CNN	EA	CIFAR-10,-100 (Krizhevsky 2009)
11	2017	A	(Liu et al. 2017b)	CNN	EA	MINST (Krizhevsky 2009), ImageNet (Russakovsky et al. 2015)
12	2018	C	(Kramer 2018)	CNN	EA	MINST (Krizhevsky 2009)
13	2018	C	Sun et al. (2018a)	CNN	GA	MINST (Krizhevsky 2009)
14	2018	C	Bibaeva (2018)	CNN	RS, GA, SA, MA	MNIST (Lecun et al. 1998), 8491845
15	2018	C	Falco et al. (2018)	DNN	DE	Obstructive Sleep Anpnea Dataset (McNicholas and Levy 2000)
16	2018	C	Pandey and Kaur (2018)	LSTM	GA	Clickbait (Pandey and Kaur 2018)
17	2018	C	Martinez et al. (2018)	DNN	GA	Rotorcraft Health State (Wade et al. 2015)
18	2018	C	Ding et al. (2018)	CNN	GA	Indian Pines scene, Pavia University scene
19	2018	A	Martín et al. (2018)	DNN	EA	MNIST (Lecun et al. 1998)
20	2018	A	Liu et al. (2018)	AE, RBM, SR-RBM, DAE	MOEA/D	MNIST (Lecun et al. 1998), CIFAR-10 (Krizhevsky 2009)
21	2018	C	Tian et al. (2018a)	CNN	GA	CIFAR-10 (Krizhevsky 2009), NetCam (Pouyanfar et al. 2018) Youtube (Tian et al. 2018b)
22	2018	C	Wang et al. (2018)	CNN	IPPSO	MINST variations, MRDBI, CS (Larochelle et al. 2007)
23	2018	C	Gibb et al. (2018)	CNN	GA	Concrete Crack (Cha et al. 2017)

Table 3 (continued)

No	Year	Type	Study	DL model	Metaheuristics	Dataset
24	2018	C	Evans et al. (2018)	CNN	GP	Cars (Agarwal et al. 2004), Jaffe (Cheng et al. 2010) Faces (Sung and Poggio 1998), Pedestrian (Munder and Gavrila 2006)
25	2018	C	Loni et al. (2018a)	DNN	NSGA-II	MNIST (Lecun et al. 1998), CIFAR-10 (Krizhevsky 2009)
26	2018	C	Loni et al. (2018b)	CNN	CGP	CIFAR-10 (Krizhevsky 2009)
27	2018	C	Kumar and Batra (2018)	RNN	GA	Streaming data
28	2018	C	Assuncao et al. (2018)	AE	EA	MNIST (Lecun et al. 1998)
29	2019	C	Ai et al. (2019)	LSTM	EA	Western Norwegian domestic household (Ai et al. 2019)
30	2019	A	Liu et al. (2019)	CNN	GA	CTP dataset (Liu et al. 2019)
31	2019	A	Junior and Yen (2019)	CNN	PSO	MNIST (Lecun et al. 1998), Fashion (Xiao et al. 2017) MINST variations, Rectangle, Convex (Larochelle et al. 2007)
32	2019	A	Baldominos et al. (2019)	CNN	GA + GE	MNIST (Lecun et al. 1998), EMNIST (Cohen et al. 2017)
33	2019	A	Mattioli et al. (2019)	CNN	GA	MNIST (Lecun et al. 1998), CIFAR-10 (Krizhevsky 2009)
34	2019	A	Sun et al. (2019b)	AE	PSO	MNIST (Lecun et al. 1998), CIFAR-10 (Krizhevsky 2009) STL-10 (Coates et al. 2011), CalTech 101 (Fei-Fei et al. 2007)
35	2019	A	Sun et al. (2019a)	CNN	EA	MNIST (Lecun et al. 1998), Fashion (Xiao et al. 2017) MINST variations, Rectangle, Convex (Larochelle et al. 2007)
36	2019	A	Wei et al. (2019)	DBN	PSO + GA	NSL-KDD (NSL-KDD 2019)
37	2019	A	Shi et al. (2019)	DNN	PSO	Modulated signals
38	2019	A	Wang et al. (2019b)	CNN	GA + PSO	MINST variations (Larochelle et al. 2007), CIFAR-10 (Krizhevsky 2009)
39	2019	A	Chiba et al. (2019)	DNN	IGA + SA	CICIDS2017 (CICIDS2017 2017), NSL-KDD (NSL-KDD 2019) CIDDS- 001 (CIDDS-001 2019)

A: Article, C: Conference Paper, O: Others

5.5 Studies using metaheuristics at feature representation level of deep neural networks

5.5.1 Articles

Miranda et al. (2014) proposed to use a combination of EA and PSO in AE to reduce the dimensionality of the search space without loss of accuracy. The EA obtained distinct solutions through iterations instead of random sampling to obtain a denser representation. The solutions were transferred to AE to encode/decode. While the particles were updated in the reduced space, the fitness was calculated according to the original space. The experimental results illustrated that searching in a compressed space could discover more promising regions compared to searching the original space.

Gong et al. (2015) proposed a multi-objective sparse feature learning (MO-SFL) model to avoid choosing the constant, which controlled the sparsity of representations. MO-SFL handled two objectives, the sparsity of hidden units and reconstruction error. The sparse feature learning problem was expressed as a MOP and a multi-objective induced learning procedure (Sa-MODE/D) was designed according to the learning process of SR-RBM. The solution found by Sa-MODE/D was transferred to the fast updating step in which the reconstruction error was minimized using the stochastic gradient descent algorithm. The experiments were performed on MNIST and Berkeley (Ranzato et al. 2006) datasets and demonstrated that useful features could be learned by the multi-objective sparse feature learning model.

Ghamisi et al. (2016) used Fractional Order Darwinian PSO (FODPSO) to improve the overall accuracy of CNN. The selected most informative bands by PSO were given to the designed CNN to produce the final classification map. The overall accuracy was expressed as the fitness value. In this study, Indian Pines captured by AVIRIS over a rural area in NW Indiana, and Pavia dataset captured by ROSIS-03 were used. Results indicated that the proposed model could classify hyper-spectral data via a CNN trained with a limited number of training samples.

Hosseini et al. (2017) utilized principal component analysis (PCA), independent component analysis (ICA), and Differential Search Algorithm (DSA) to extract features in BCI seizure prediction task and localization from scalp EEG and ECoG big data. The EEG data was decorrelated by PCA while I-ICA separated the remaining higher-order dependencies and DSA found the global minimum. The approach was extended on existing deep-learning structures. Stacked AE and CNN were applied to BCI seizure prediction and localization from scalp EEG and ECoG big data. The results on the ECoG datasets illustrated the advantage of the proposed approach in patient-specific BCI prediction.

The approach proposed by Trivedi et al. (2018) used a sparse AE for feature extraction and a CNN tuned by the GA algorithm for the softmax classifier. After color transformations and each image was resized to 28×28, a sparse AE was employed to learn the approximation of input data and then to obtain the best features. The study aimed to extract low-dimensional features by reducing the number of nodes and imposing the sparsity parameter. In the sparse AE used in the study, the input layer was of dimension 1024, and the output layer was of dimension 81. The sparse AE was trained using L-BFGS gradient optimization using the cost function of the sum of square errors, weight decay, and KL distance. The CNN architecture was executed once AE output was obtained. After mean pooling from the pooling layer of CNN architecture, the inputs of a fully-connected layer were obtained by GA and the weights were tuned by L-BFGS. They concluded that incorporating the GA with CNN architecture improved accuracy, while increasing the number of iterations caused an increase in the length of the chromosome, which made the GA encoding technique less efficient.

Tian and Liu (2019) integrated a CNN and immune algorithm principles to improve the extraction of rolling bearing fault features. CNN was used to extract the characteristics of the time domain and frequency domain signals of rolling bearings. A cloning strategy and a mutation operation were applied to enhance the efficiency of the learning stage. The experiments were investigated on the rolling bearing data by Case Western Reserve University Data Center (Smith and Randall 2015). The raw data was the vibration acceleration signal, which was composed of hashed data points. The experimental results showed that the proposed model was useful in classification and recognition.

5.5.2 Conference papers

Blanco et al. (2018) developed a CNN to classify network connections using only features available at a network node, distinguishing between normal data and a set of attack classes. In order to select the optimal layout of the input features, GA was used by reducing the number of different features if required. Kappa metric was used as the fitness function because it considered the imbalance of the instances in different classes among the training dataset. The tests were performed on two different public datasets with different ratio of attacks: UNSW (Nour and Slay 2015) (10 classes) and NSL-KDD (NSL-KDD 2019) (4 classes). Experiments on (NSL-KDD 2019) produced better results because it could proportionate between the different classes, obtaining a cross-validated multi-class classifier with K of 0.95.

Silva et al. (2018) proposed a binary PSO that searches the possible combinations to find out a fusion rule at the feature level. A comparison was performed among sum, multiplication, and minimum fusion rules and the proposed approach. The last layers of VGG model trained for face recognition were replaced to best represent the number of classes on the training database. The experiments were performed using iris and periocular region (NICE.II competition database) (Proenca et al. 2010). The experiments in the NICE.II competition databases showed that the transfer learning representation for iris modality achieved a new state-of-the-art when the fusion at feature level by PSO was performed on periocular and iris modalities, decidability of 3.45 and 5.55% of EER.

These studies are summarized in Table 4.

5.6 The most used datasets in the reviewed studies

Complex models such as DL models require large-scale datasets in training to avoid overfitting and generate consistent results over unseen data. Researchers created datasets by collecting data from some domains, including handwriting, face, eeg, car, iris, flower etc., and some of the datasets are labeled by domain experts for classification purposes. In this section, some brief information is given about the most used public datasets used in the studies we reviewed, although there are more datasets in the literature.

MNIST handwritten digit recognition database (Lecun et al. 1998): It is an extended version of the NIST database. Each sample is a 28 × 28 grayscale image and contains a target class label between 0 and 9 corresponding to each digit. It has 60,000 training images and 10,000 testing images.

Table 4 The list of studies related to employing Metaheuristics at feature representation level of deep neural networks

No	Year	Type	Study	DL models	Metaheuristic	Dataset
1	2014	A	Miranda et al. (2014)	AE	EA + PSO	Wind-hydro coordination
2	2015	A	Gong et al. (2015)	AE, SR-RBM	Sa-MODE/D	MNIST (Lecun et al. 1998), Berkeley (Ranzato et al. 2006)
3	2016	A	Ghamisi et al. (2016)	CNN	PSO	Indian Pines scene, Pavia University scene
4	2017	A	Hosseini et al. (2017)	CNN, AE	DSA	EEG (Hosseini et al. 2017), ECoG datasets (Brinkmann et al. 2009)
5	2018	A	Trivedi et al. (2018)	CNN	L-BFGS + GA	Handwriting dataset (Bhattacharya and Chaudhuri 2005)
6	2018	C	Blanco et al. (2018)	CNN	GA	UNSW (Nour and Slay 2015), NSL-KDD (NSL-KDD 2019)
7	2018	C	Silva et al. (2018)	VGG	PSO	Ubiris v2 (Proenca et al. 2010)
8	2019	A	Tian and Liu (2019)	CNN	CSA	Rolling Bearing Data (Smith and Randall 2015)

A: Article, C: Conference Paper, O: Others

MNIST variations (Larochelle et al. 2007): There are the MNIST Basic (MB), the MNIST with Background Images (MBI), Random Background (MRB), Rotated Digits (MRD), with RD plus Background Images (MRDBI), the Rectangle, the Rectangle Images (RI), and the Convex Sets (CS) benchmarks. The rectangle or not for the Rectangle and RI benchmarks, and the convex or not for the Convex benchmark) sets are for the shape recognition purpose. These datasets are more challenging than the original MNIST because they also contain irrelevant information to the problem.

Fashion-MNIST (Xiao et al. 2017): It consists of 60,000 training and 10,000 test images for the cloth classification purpose. Each sample is 28 × 28 gray scale image assigned to one of 10 classes.

EMNIST Database (Cohen et al. 2017): It is an extended version of MNIST and consists of both digits and letters. It contains over 81,4255 images for both digit and letter classification. There are 62 classes in the dataset.

Semeion Handwritten Digit (Semeion 2008): This dataset contains 1593 grayscale images from handwritten 0–9 digits. Each sample is 16 × 16 in a grayscale of 256 values. USPS (Hull 1994): The dataset includes 7291 train and 2007 test images assigned to one of 0–9 digit classes.

CIFAR10 (Krizhevsky 2009): Each sample is 32 × 32 color images and contains a target of 10 classes, where each class has 6000 images. This dataset consists of 60,000 color images. There are five batches for training, composed of 50,000 images, and one batch of test images consists of 10,000 images. CIFAR-10 and CIFAR-100 are labeled subsets of the 80 million tiny images. CIFAR 100 (Krizhevsky 2009) is similar to CIFAR-10 except it has 100 classes where each one has 600 images. There are 50,000 training and 10,000 test images.

CalTech 101 dataset (Li et al. 2003): This dataset contains 9146 images from Google images assigned to 101 classes corresponding to objects and shapes. CalTech 101 Silhouettes (Marlin et al. 2010) is generated based on CalTech 101 dataset, comprising silhouettes of images from 101 classes with a resolution of 28 × 28. The training and testing sets are composed of 1185 and 2307 samples, respectively. CalTech-256 (Griffin et al. 2007) is an extended version of CalTech 101 by including more categories and more number of images in each category. Artifacts in the predecessor dataset are also avoided in CalTech-256 datasets.

STL-10 data set (Coates et al. 2011): This dataset consists of 96 × 96 images acquired from ImageNet. There are 10 classes, and each class has 500 images for training and 800 images for testing purposes.

The CICIDS 2017 dataset (CICIDS2017 2017): The dataset satisfies indispensable characteristics of a valid intrusion detection system dataset, namely Anonymity, Attack Diversity, Complete Capture, Complete Interaction, Complete Network Configuration, Available Protocols, Complete Traffic, Feature Set, Metadata, Heterogeneity, and Labeling. It contains the abstract behaviour of 25 users based on the HTTP, HTTPS, FTP, SSH, and email protocols. Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS attacks are performed. 83 statistical features such duration, number of packets, number of bytes etc. There are a total of 2,827,876 flows in the dataset. NSL-KDD Dataset (NSL-KDD 2019): Each sample in the dataset is related to a network connection and consists of 41 features. There are 4,898,431 records and the records are labeled as normal or one of the specific attack names which fall into four major categories: DOS, U2R, R2L, PROBE. CIDDS-001 Dataset (CIDDS-001 2019): (Coburg Network Intrusion Detection Dataset) is a labeled dataset of flow in a Cloud environment based on OpenStack platform. It consists of three logs files (attack logs, client configurations, and client logs) and traffic data from two servers where each server traffic comprises of 4 four week captured traffic data. It contains 14 attributes, the first 10 attributes are the default NetFlow attributes and the last four attributes are additional attributes. A total of 32 million of normal and attack flows are captured in the dataset within four weeks. There are 92 types of attacks in the dataset.

The XMU dataset (Huang et al. 2015): It consists of 13 classes of vehicle logo, namely Buick, Chery, Citroen, Honda, Hyundai, Lexus, Mazda, Peugeot, Toyota, and Volkswagen, Proton, Nissan, and Perodua. Each image is 70 × 70 pixels in a greyscale format. There are 13,000 training images and 1950 test images These vehicle logo images encompass various outdoor imaging conditions such as distortions, illumination variance, and translation.

CK+ Dataset (Lucey et al. 2010): Cohn-Kanade (CK+) data set consists of 593 images depicting facial expressions of 210 adults, and the task is to classify the emotions including neutral, sadness, surprise, happiness, fear, anger, contempt and disgust. The resolution of the images is 640 × 490.

Faces Dataset by AT&T (Sung and Poggio 1998): This dataset contains a set of face images taken between April 1992 and April 1994 at the lab. There are 10 samples for each of 40 classes, 400 in total. Each image is 92 × 112 pixels, with 256 gray levels per pixel. The images were taken at different times, varying the lighting, facial expressions (open/closed eyes, smiling/not smiling), and facial details (glasses/no glasses).

Ubiris.v2 (Proenca et al. 2010) has 11,102 images from 261 classes. The images were acquired under different distances, angles, lightning in order to simulate real noise conditions. The images have 800 × 700 resolution. The database was used in the NICE.II competition for the iris recognition purpose. The dataset is balanced regarding the number of images per subject.

Flower Dataset (Nilsback and Zisserman 2008): This dataset has 8189 flower images of 102 different flower types. There are 2040 training images and 6149 testing images. Each image is 500 × 500 pixels.

An overview of the most used dataset in the reviewed studies is presented in Table 5.

6 Discussion

Deep learning models are similar to neural networks, but have much more parameters and many design issues at both representation level and connected dense layers. While weights and bias values in network models are numeric, there are many discrete and categorical valued parameters, which makes derivation-based methods unsuitable for optimizing the problem. Deep learning models have some hyper-parameters to be appropriately tuned. Since running each different configuration manually is quite labor-intensive, and time-consuming, automatic hyper-parameter optimization is needed to reduce computational cost. Designing an efficient architecture is another significant issue to be handled. Furthermore, at feature representation level, there are some tasks to be optimized, such as feature selection, feature dimension reduction, thresholding, determining sparsity-inducing term, deciding optimal input layout. Therefore, researchers introduced metaheuristics to solve these optimization problems arising in the deep learning field. This paper reviews the use of metaheuristic algorithms in hyper-parameter optimization, training, architecture design, and at feature representation level of deep learning models. By reviewing the studies in the previous section, we can deduce the results given below:

Table 5 An overview of the most used datasets used in the studies reviewed

Dataset	Input size	Classes	Train size	Test size	Domain
MNIST (Lecun et al. 1998)	28 × 28	10	60,000	10,000	Hand-written digits
MNIST-RD (Larochelle et al. 2007)	28 × 28	10	12,000	50,000	Handwritten digits
MNIST-RB (Larochelle et al. 2007)	28 × 28	10	12,000	50,000	Handwritten digits
MNIST-BI (Larochelle et al. 2007)	28 × 28	10	12,000	50,000	Handwritten digits
MNIST-RD + BI (Larochelle et al. 2007)	28 × 28	10	12,000	50,000	Handwritten digits
EMNIST Database (Cohen et al. 2017)	28 × 28	62	697,932	116,323	Handwritten digits and letters
Semeion (Semeion 2008)	16 × 16	10	1593		Handwritten digits
USPS (Hull 1994)	16 × 16	10	7291	2007	Handwritten digits
CIFAR-10 (Krizhevsky 2009)	32 × 32	10	50,000	10,000	Object recognition
CIFAR-100 (Krizhevsky 2009)	32 × 32	100	50,000	10,000	Object recognition
Caltech-101 (Li et al. 2003)	300 × 200	101	9146		Object detection
CalTech-101 Silhouettes (Marlin et al. 2010)	28 × 28	101	1185	2307	Object recognition
CalTech-256 (Griffin et al. 2007)	351 pixels in average	256	30,607		Object recognition
STL-10 data set (Coates et al. 2011)	96 × 96	10	5000	8000	Object recognition
Rectangles (Larochelle et al. 2007)	28 × 28	2	12000	50,000	Shape recognition
Rectangle-I (Larochelle et al. 2007)	28 × 28	2	12,000	50,000	Shape recognition
Convex (Larochelle et al. 2007)	28 × 28	2	8000	50,000	Shape recognition
The CICIDS 2017 dataset (CICIDS2017 2017)	83	15	2,827,876		IDS
NSL-KDD Dataset (NSL-KDD 2019)	41	5	4,898,431		IDS
CIDDS-001 Dataset (CIDDS-001 2019)	14	5	32 million		IDS
MNIST-Fashion (Xiao et al. 2017)	28 × 28	10	60,000	10,000	Cloth classification
XMU (Huang et al. 2015)	70 × 70	10	13,000	1950	Vehicle logo recognition
CK+ Dataset (Lucey et al. 2010)	640 × 490	8	593		Emotion classification
Faces Dataset by AT&T (Sung and Poggio 1998)	92 × 112	40	400		Faces
Ubiris.v2 (Proenca et al. 2010)	800 × 700	261	11,102		Iris recognition
Flower Dataset (Nilsback and Zisserman 2008)	500 × 500	102	2040	6149	Flower classification

- According to our search (October 2019), the number of papers on training DL models using metaheuristics is 21, the number of papers on hyper-parameter optimization of DL models using metaheuristics is 30, the number of papers on architecture (topology) optimization of DL models using metaheuristics is 39, and the number of papers on using metaheuristics at feature representation level is 8. It is seen from the number of publications that particularly hyper-parameter optimization and architectural design by metaheuristics are hot topics in the field. Based on these studies, it can be stated that metaheuristics can automatically generate highly interpretable evolved networks compared to cutting-edge architectures.
- As seen in Fig. 15, starting from 2012, there is an acceleration in the number of studies related to the optimization of the deep learning models. Most of the DNNs optimized by metaheuristics target image classification problems.
- The number of studies grouped depending on DL architectures used in the reviewed publications is shown in Fig. 16. Traditional CNN is the most investigated architecture, followed by DNN, AE, DBN, LSTM. Simplicity of CNN architecture, existing open-source libraries, and computational burden affect the architecture choice of the researchers. Unsupervised pretraining models and generative models are also widely used in the publications. More complex networks are employed in a limited number of studies due to their requirement for high computing facilities.
- Many kinds of metaheuristics have been applied to solve various optimization problems arising in the field of deep learning. The number of studies with respect to the metaheuristic employed in the studies is given in Fig. 17. It should be noted that modified and hybrid algorithms are treated as a separate method. GA is the most used metaheuristic in deep learning optimization since the solution representation in GA is more suitable to encode network structures and it is also best-known metaheuristic among researchers.

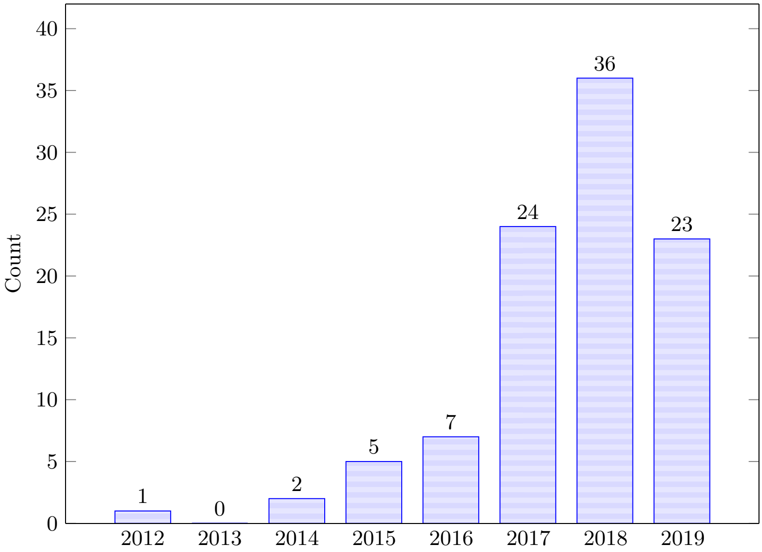


Fig. 15 The number of studies by year

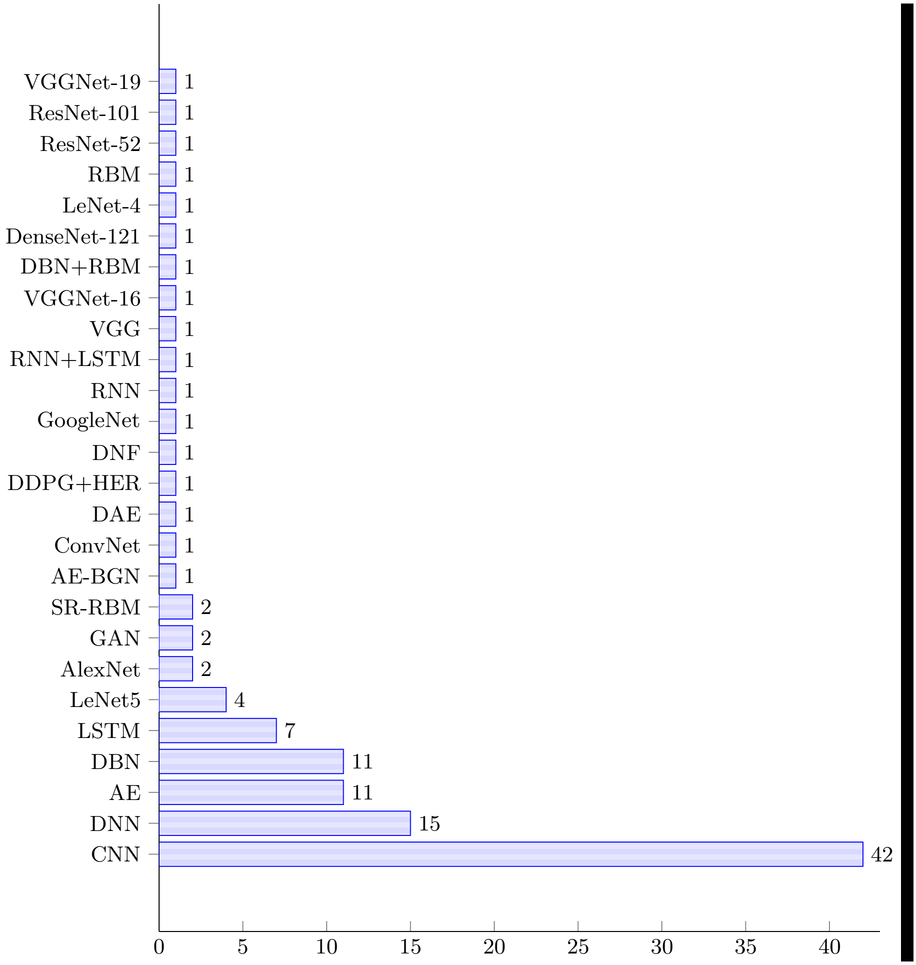


Fig. 16 The number of studies with respect to the DL model

- Efficient metaheuristic algorithms with global search ability do not suffer from the problems of gradient-based approaches, including premature convergence and vanishing (or exploding) gradients problem. Besides, they can be applied to mixed programming problems with real, binary, categorical and discrete decision variables, such as network architecture search.
- Adopting multi-objective heuristics is appropriate for simultaneously handling multiple conflicting objectives such as obtaining a sparse network, improving network accuracy, minimizing computation complexity. Non-dominated sorting and decomposition-based multi-objective techniques generate well-distributed solutions for this kind of problems.
- To reduce the computational cost, population-based metaheuristics are more convenient to be distributed to parallel computing facilities compared to derivative-based training algorithms.
- Although indirect encoding schemes are flexible and can yield efficient compact networks, they are hard to be incorporated into the metaheuristics compared to direct

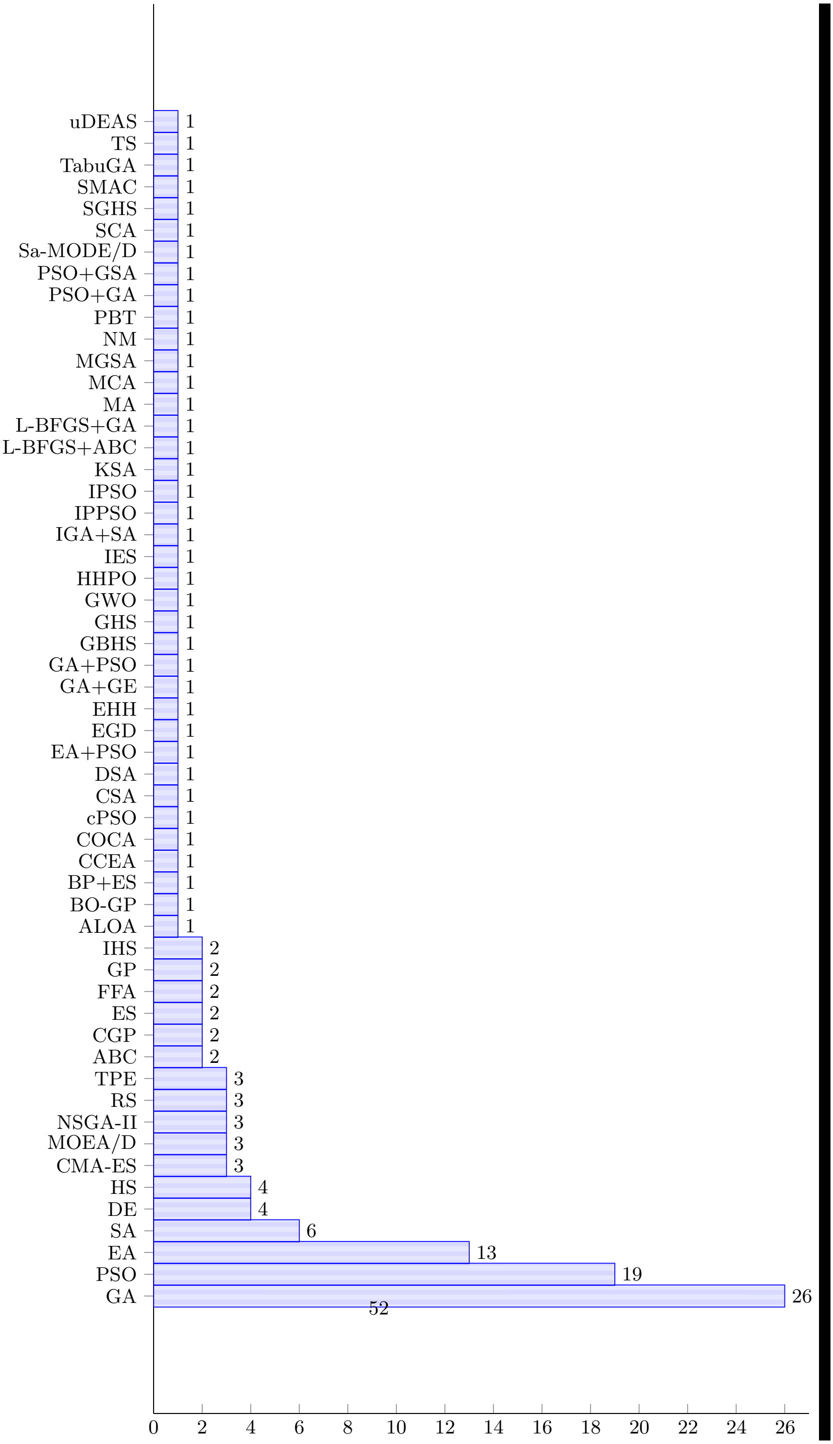


Fig. 17 The number of studies with respect to the metaheuristic

encoding schemes. Automatic programming approaches, such as GP or ABCP, has a good potential in generating network architectures because its solution representation fits indirect encoding.
- Hybrid approaches which optimize the network architecture using metaheuristics while training the network using gradient-based methods may be helpful for some problems. Similarly, hybridization of two metaheuristics with different search characteristics can improve the overall exploitation and exploration ability compared to the ability of each individual metaheuristics. Integrating local search heuristics with metaheuristics enhances the intensification ability of search.
- As there is no unique algorithm to solve any kind of problems, ensemble or hybrid methods have been also proposed to extend the problem types that can be solved. The error of the models using committees is better than the average error rate of its constituents individually.
- There is a variety of public datasets in the literature. MNIST and its variations, CIFAR, and CalTech are the most used datasets in the experiments to validate the approaches. Novel approaches can be validated on these datasets. However, the comparisons with state-of-the-art algorithms would be more fair if the papers gave more detail about the conditions they performed the studies, such as initialization values, data sampling, data separation, augmentation, model parameters, preprocessing steps, etc.

7 Conclusion and future directions

In this paper, we aimed to provide brief information about the basis of common DNN networks and give recent advances in the field of deep learning by using metaheuristics. We defined main optimization problems in DL field and presented representation schemes to encode a DNN structure for metaheuristics. And then, ee reported the studies on the metaheuristics used for optimization problems in deep learning field and discussed the advantages and drawbacks of the approaches. Based on our review and discussion, it is concluded that using metaheuristics in deep learning is open for further developments. The future directions can be listed as below:

- Designing efficient and compact architectures
- New projections and kernel trick methods at feature level to deal with nonlinearity and heterogeneity
- Efficient automatic labeling approaches on huge datasets based on self-supervised models and unsupervised generative models
- Resource-efficient hardware implementations, especially the hardware implementation of hierarchical and coupled architectures
- Implementing more elaborate and multi-level encoding schemes that do not have competing conventions problem, and appropriate efficient intensification and global exploration operators
- Studying on self-adaptive metaheuristics for tuning the control parameters automatically based on population trajectory
- Aiming efficient parallelization of metaheuristics distributed to high computing facilities (GPUs, TPUs) in order to reduce computational cost of the fitness functions due to large-scale architecture and big data

- Efficient constructive heuristics and morphological operators for morphologic perturbation of the networks during search by metaheuristics
- Introducing dynamic optimization methods that can cope with changing conditions and network structure to avoid overfitting
- Enhancing metaheuristics that increase diversity and boost the exploration to deal with real, binary, discrete, categoric decision variables

We hope this paper will be very useful for readers performing researches in the deep learning and metaheuristics fields for further advances.

References

Agarwal S, Awan A, Roth D (2004) Learning to detect objects in images via a sparse, part-based representation. IEEE Trans Pattern Anal Mach Intell 26(11):1475–1490

Agbehadji IE, Millham R, Fong SJ, Yang H (2018) Kestrel-based search algorithm (KSA) for parameter tuning unto long short term memory (LSTM) network for feature selection in classification of high-dimensional bioinformatics datasets. In: Federated conference on computer science and information systems (FedCSIS), pp 15–20

Ai S, Chakravorty A, Rong C (2019) Evolutionary ensemble LSTM based household peak demand prediction. In: International conference on artificial intelligence in information and communication (ICAIIC), pp 1–6

Alvernaz S, Togelius J (2017) Autoencoder-augmented neuroevolution for visual doom playing. In: IEEE conference on computational intelligence and games (CIG), pp 1–8

Andrzejak RG, Lehnertz K, Mormann F, Rieke C, David P, Elger CE (2001) Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: dependence on recording region and brain state. Phys Rev E 64

Assuncao F, Sereno D, Lourenco N, Machado P, Ribeiro B (2018) Automatic evolution of autoencoders for compressed representations. In: IEEE congress on evolutionary computation (CEC), pp 1–8

Ayumi V, Rere LMR, Fanany MI, Arymurthy AM (2016) Optimization of convolutional neural network using microcanonical annealing algorithm. In: International conference on advanced computer science and information systems (ICACSIS). IEEE, pp 506–511

Badem H, Basturk A, Caliskan A, Yuksel ME (2017) A new efficient training strategy for deep neural networks by hybridization of artificial bee colony and limited-memory BFGS optimization algorithms. Neurocomputing 266:506–526

Baker B, Gupta O, Naik N, Raskar R (2016) Designing neural network architectures using reinforcement learning. arXiv preprint arXiv:1611.02167

Baker B, Gupta O, Raskar R, Naik N (2017) Accelerating neural architecture search using performance prediction. arXiv preprint arXiv:1705.10823

Baldominos A, Saez Y, Isasi P (2019) Hybridizing evolutionary computation and deep neural networks: an approach to handwriting recognition using committees and transfer learning. Complexity 2019:1–16

Banharnsakun A (2018) Towards improving the convolutional neural networks for deep learning using the distributed artificial bee colony method. Int J Mach Learn Cybern 10:1301–1311

Bender G, Kindermans P-J, Zoph B, Vasudevan V, Le Q (2018) Understanding and simplifying one-shot architecture search. In: International conference on machine learning, pp 550–559

Bengio Y (2009) Learning deep architectures for AI. Found Trends Mach Learn 2(1):1–127

Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyper-parameter optimization. In: Proceedings of the 24th international conference on neural information processing systems, NIPS'11, USA. Curran Associates Inc., pp 2546–2554

Bhattacharya U, Chaudhuri BB (2005) Databases for research on recognition of handwritten characters of Indian scripts. In: Eighth international conference on document analysis and recognition (ICDAR'05), vol 2, pp 789–793

Bibaeva V (2018) Using metaheuristics for hyper-parameter optimization of convolutional neural networks. In: IEEE 28th international workshop on machine learning for signal processing (MLSP), pp 1–6

Blanco R, Malagón P, Cilla JJ, Moya JM (2018) Multiclass network attack classifier using CNN tuned with genetic algorithms. In: 28th International symposium on power and timing modeling, optimization and simulation (PATMOS), pp 177–182

Bochinski E, Senst T, Sikora T (2017) Hyper-parameter optimization for convolutional neural network committees based on evolutionary algorithms. In: IEEE international conference on image processing (ICIP), pp 3924–3928

Brinkmann BH, Bower MR, Stengel KA, Worrell GA, Stead M (2009) Large-scale electrophysiology: acquisition, compression, encryption, and storage of big data. J. Neurosci Methods 180(1):185–192

Brock A, Lim T, Ritchie JM, Weston N (2017) Smash: one-shot model architecture search through hypernetworks. arXiv preprint arXiv:1708.05344

Cai H, Yang J, Zhang W, Han S, Yu Y (2018) Path-level network transformation for efficient architecture search. arXiv preprint arXiv:1806.02639

Cai Y, Cai Z, Zeng M, Liu X, Wu J, Wang G (2018) A novel deep learning approach: Stacked evolutionary auto-encoder. In: International joint conference on neural networks (IJCNN), pp 1–8

Carletta J, Ashby S, Bourban S, Flynn M, Guillemot M, Hain T, Kadlec J, Karaiskos V, Kraaij W, Kronenthal M, Lathoud G, Lincoln M, Lisowska A, andWilfried Post IM, Reidsma D, Wellner P (2006) The AMI meeting corpus: a pre-announcement. In: Proceedings of the second international conference on machine learning for multimodal interaction, pp 28–39

Cha Y-J, Choi W, Bykztrk O (2017) Deep learning-based crack damage detection using convolutional neural networks. Comput Aided Civ Infrastruct Eng 32(5):361–378

Chen T, Goodfellow I, Shlens J (2015a) Net2net: accelerating learning via knowledge transfer. arXiv preprint arXiv:1511.05641

Chen X, Fang H, Lin T-Y, Vedantam R, Gupta S, Dollár P, Zitnick CL (2015b) Microsoft coco captions: data collection and evaluation server. arXiv preprint arXiv:1504.00325

Cheng F, Yu J, Xiong H (2010) Facial expression recognition in Jaffe dataset based on Gaussian process classification. IEEE Trans Neural Netw 21(10):1685–1690

Chhabra Y, Varshney S, Ankita (2017). Hybrid particle swarm training for convolution neural network (CNN). In: Tenth international conference on contemporary computing (IC3), pp 1–3

Chiba Z, Abghour N, Moussaid K, Rida M (2019) Intelligent approach to build a deep neural network based IDS for cloud environment using combination of machine learning algorithms. Comput Secur 86:291–317

Chiroma H, Gital AY, Rana N, Abdulhamid SM, Muhammad AN, Umar AY, Abubakar AI (2019) Nature inspired meta-heuristic algorithms for deep learning: recent progress and novel perspective. In: Advances in intelligent systems and computing. Springer, pp 59–70

CICIDS2017 (2017) Cicids2017 data set

CIDDS-001 (2019) Cidds-001 dataset, hochschule coburg

Coates A, Ng AY, Lee H (2011) An analysis of single-layer networks in unsupervised feature learning. International conference on artificial intelligence and statistics. Pt. Lauderdale, Florida, USA, pp 215–223

Coello CAC (2003) Evolutionary multi-objective optimization: a critical review. In: Evolutionary optimization. Springer, pp 117–146

Cohen G, Afshar S, Tapson J, van Schaik A (2017) EMNIST: an extension of MNIST to handwritten letters

David OE, Greental I (2014) Genetic algorithms for evolving deep neural networks. In: Proceedings of the 2014 conference companion on genetic and evolutionary computation companion-GECCO Comp14. ACM Press, pp 1451–1452

De Jong K (1975) An analysis of the behavior of a class of genetic adaptive systems. Ph.D. thesis, University of Michigan

de Rosa GH, Papa JP (2019) Soft-tempering deep belief networks parameters through genetic programming. J Artif Intell Syst 1(1):43–59

Deepa S, Baranilingesan I (2018) Optimized deep learning neural network predictive controller for continuous stirred tank reactor. Comput Electr Eng 71:782–797

Delowar Hossain, Capi G (2017) Genetic algorithm based deep learning parameters tuning for robot object recognition and grasping. Zenodo

Desell T (2017) Large scale evolution of convolutional neural networks using volunteer computing. In: Proceedings of the genetic and evolutionary computation conference companion, GECCO '17. Association for Computing Machinery, New York, pp 127–128

Ding C, Li W, Zhang L, Tian C, Wei W, Zhang Y (2018) Hyperspectral image classification based on convolutional neural networks with adaptive network structure. In: International conference on orange technologies (ICOT), pp 1–5

Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. Trans Syst Man Cyber B 26(1):29–41

Dua D, Graff C (2017) UCI machine learning repository

Eitz M, Hays J, Alexa M (2012) How do humans sketch objects? ACM Trans Graph (Proc SIGGRAPH) 31(4):44:1–44:10

Elsken T, Metzen JH, Hutter F (2018) Efficient multi-objective neural architecture search via lamarckian evolution. arXiv preprint arXiv:1804.09081

Erhan D, Courville A, Bengio Y, Vincent P (2010) Why does unsupervised pre-training help deep learning? In: Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp 201–208

Evans B, Al-Sahaf H, Xue B, Zhang M (2018) Evolutionary deep learning: a genetic programming approach to image classification. In: IEEE congress on evolutionary computation (CEC), pp 1–6

Falco ID, Pietro GD, Sannino G, Scafuri U, Tarantino E, Cioppa AD, Trunfio GA (2018) Deep neural network hyper-parameter setting for classification of obstructive sleep apnea episodes. In: IEEE symposium on computers and communications (ISCC), pp 01187–01192

Fei-Fei L, Fergus R, Perona P (2007) Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. Comput Vis Image Underst 106(1):59–70

Fekiač J, Zelinka I, Burguillo JC (2011) A review of methods for encoding neural network topologies in evolutionary computation. In: Proceedings of 25th European conference on modeling and simulation ECMS, pp 410–416

Floreano D, Dürr P, Mattiussi C (2008) Neuroevolution: from architectures to learning. Evol Intell 1(1):47–62

Fogel LJ, Owens AJ, Walsh MJ (1966) Artificial intelligence through simulated evolution. Wiley, New York

Fong S, Deb S, Yang XS (2017) How meta-heuristic algorithms contribute to deep learning in the hype of big data analytics. Advances in intelligent systems and computing. Springer, Singapore, pp 3–25

Fujino S, Mori N, Matsumoto K (2017) Deep convolutional networks for human sketches by means of the evolutionary deep learning. In: Joint 17th world congress of international fuzzy systems association and 9th international conference on soft computing and intelligent systems (IFSA-SCIS), pp 1–5

Fukushima K (1980) Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybern 36(4):193–202

Furui S, Maekawa K, Isahara H (2000) A japanese national project on spontaneous speech corpus and processing technology. In: Proceedings of ASR’00, pp 244–248

Gaier A, Ha D (2019) Weight agnostic neural networks. In: Advances in neural information processing systems, pp 5364–5378

Gauci J, Stanley K (2007) Generating large-scale neural networks through discovering geometric regularities. In: Proceedings of the 9th annual conference on genetic and evolutionary computation, pp 997–1004

Ghamisi P, Chen Y, Zhu XX (2016) A self-improving convolution neural network for the classification of hyperspectral data. IEEE Geosci Remote Sens Lett 13(10):1537–1541

Gibb S, La HM, Louis S (2018) A genetic algorithm for convolutional network structure optimization for concrete crack detection. In: IEEE congress on evolutionary computation (CEC), pp 1–8

Glover F (1986) Future paths for integer programming and links to artificial intelligence. Comput Oper Res 13(5):533–549

Goldberg DE, Richardson J et al (1987) Genetic algorithms with sharing for multimodal function optimization. In: Genetic algorithms and their applications: proceedings of the second international conference on genetic algorithms. Lawrence Erlbaum, Hillsdale, pp 41–49

Gong M, Liu J, Li H, Cai Q, Su L (2015) A multiobjective sparse feature learning model for deep neural networks. IEEE Trans Neural Netw Learn Syst 26(12):3263–3277

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

Griffin G, Holub A, Perona P (2007) Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology

Gruau F (1994) Neural network synthesis using cellular encoding and the genetic algorithm

Gülcü A, Kuş Z (2019) Konvolüsyonel sinir ağlarında hiper-parametre optimizasyonu yöntemlerinin incelenmesi. Gazi Üniversitesi Fen Bilimleri Dergisi Part C: Tasarım ve Teknoloji 7(2):503–522

Guo B, Hu J, Wu W, Peng Q, Wu F (2019) The tabu_genetic algorithm: a novel method for hyper-parameter optimization of learning algorithms. Electronics 8(5):579

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 770–778

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778

Hinton GE, Osindero S, Teh Y-W (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554

Hinz T, Navarro-Guerrero N, Magg S, Wermter S (2018) Speeding up the hyperparameter optimization of deep convolutional neural networks. Int J Comput Intell Appl 17(02):1850008

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780

Hoekstra V (2011) An overview of neuroevolution techniques. Technical report

Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor

Hossain D, Capi G (2017) Multiobjective evolution for deep learning and its robotic applications. In: 8th international conference on information, intelligence, systems applications (IISA), pp 1–6

Hossain D, Capi G, Jindai M (2018) Optimizing deep learning parameters using genetic algorithm for object recognition and robot grasping. J Electron Sci Technol 16(1):11–15

Hosseini M, Pompili D, Elisevich K, Soltanian-Zadeh H (2017) Optimized deep learning for EEG big data and seizure prediction BCI via internet of things. IEEE Trans Big Data 3(4):392–404

Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708

Huang Y, Wu R, Sun Y et al (2015) Vehicle logo recognition system based on convolutional neural networks with a pretraining strategy. IEEE Trans Intell Transp Syst 16(4):1951–19604

Hull JJ (1994) A database for handwritten text recognition research. IEEE Trans Pattern Anal Mach Intell 16(5):550–554

Hutter F, Hoos HH, Leyton-Brown K (2011) Sequential model-based optimization for general algorithm configuration. In: International conference on learning and intelligent optimization. Springer, pp 507–523

Irsoy O, Cardie C (2014) Deep recursive neural networks for compositionality in language. In: Advances in neural information processing systems, pp 2096–2104

Jacob C, Rehder J (1993) Evolution of neural net architectures by a hierarchical grammar-based genetic system. In: Artificial neural nets and genetic algorithms. Springer, pp 72–79

Jaderberg M, Dalibard V, Osindero S, Czarnecki WM, Donahue J, Razavi A, Vinyals O, Green T, Dunning I, Simonyan K, Fernando C, Kavukcuoglu K (2017) Population based training of neural networks

Jain A, Phanishayee A, Mars J, Tang L, Pekhimenko G (2018) Gist: efficient data encoding for deep neural network training. In: ACM/IEEE 45th annual international symposium on computer architecture (ISCA), pp 776–789

Jin H, Song Q, Hu X (2019) Auto-keras: an efficient neural architecture search system. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1946–1956

Jin Y (2011) Surrogate-assisted evolutionary computation: recent advances and future challenges. Swarm Evol Comput 1(2):61–70

Junior FEF, Yen GG (2019) Particle swarm optimization of deep neural networks architectures for image classification. Swarm Evol Comput 49:62–74

Kaggle (2017) Kaggle competition dataset

Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Computer Engineering Department, Engineering Faculty, Erciyes University

Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. Appl Math Comput 214(1):108–132

Kassahun Y, Edgington M, Metzen JH, Sommer G, Kirchner F (2007) A common genetic encoding for both direct and indirect encodings of networks. In: Proceedings of the 9th annual conference on genetic and evolutionary computation, pp 1029–1036

Kennedy J, Eberhart RC (1995) Particle swarm optimization. IEEE international conference on neural networks 4:1942–1948

Khalifa MH, Ammar M, Ouarda W, Alimi AM (2017) Particle swarm optimization for deep learning of convolution neural network. In: Sudan conference on computer science and information technology (SCCSIT), pp 1–5

Khan A, Sohail A, Zahoora U, Qureshi AS (2020) A survey of the recent architectures of deep convolutional neural networks. Artif Intell Rev 53(8):5455–5516

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680

Koza JR (1990) Genetic programming: a paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STAN–CS–90–1314, Computer Science Department, Stanford University

Koza JR, Rice JP (1991) Genetic generation of both the weights and architecture for a neural network. In: IJCNN-91-seattle international joint conference on neural networks, vol 2. IEEE, pp 397–404

Koziel S, Michalewicz Z (1999) Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. Evol Comput 7(1):19–44

Kramer MA (1991) Nonlinear principal component analysis using autoassociative neural networks. AIChE J 37(2):233–243

Kramer O (2018) Evolution of convolutional highway networks. Springer, Berlin, pp 395–404

Krizhevsky A (2009) Learning multiple layers of features from tiny images. Technical Report 5, University of Toronto

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105

Kösten MM, Barut M, Acir N (2018) Deep neural network training with iPSO algorithm. In: 26th Signal processing and communications applications conference (SIU), pp 1–4

Kumar P, Batra S (2018) Meta-heuristic based optimized deep neural network for streaming data prediction. In: International conference on advances in computing, communication control and networking (ICACCCN), pp 1079–1085

Lakshmanaprabu S, Sachi NM, Shankar K, Arunkumar N, Gustavo R (2019) Optimal deep learning model for classification of lung cancer on CT images. Future Gener Comput Syst 92:374–382

Lamos-Sweeney JD (2012) Deep learning using genetic algorithms

Lander S, Shang Y (2015) EvoAE: a new evolutionary method for training autoencoders for deep learning networks. In: IEEE 39th annual computer software and applications conference, vol 2, pp 790–795

Larochelle H, Erhan D, Courville A, Bergstra J, Bengio Y (2007) An empirical evaluation of deep architectures on problems with many factors of variation. In: Proceedings of the 24th international conference on machine learning, ICML ’07. Association for Computing Machinery, New York, pp 473-480

Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324

Li FF, Andreetto M, Ranzato MA, Perona P (2003) Caltech-101 silhouttes dataset. Technical Report 7694, California Institute of Technology

Li J, Cheng K, Wang S, Morstatter F, Trevino RP, Tang J, Liu H (2018) Feature selection: a data perspective. ACM Comput Surv (CSUR) 50(6):94

Li L, Jamieson K, DeSalvo G, Rostamizadeh A, Talwalkar A (2017) Hyperband: a novel bandit-based approach to hyperparameter optimization. J Mach Learn Res 18(1):6765–6816

Li L, Talwalkar A (2020) Random search and reproducibility for neural architecture search. In: Uncertainty in artificial intelligence. PMLR, pp 367–377

Li Y, Lu G, Zhou L, Jiao L (2017) Quantum inspired high dimensional hyperparameter optimization of machine learning model. In: International smart cities conference (ISC2), pp 1–6

Lim SM, Sultan ABM, Sulaiman MN, Mustapha A, Leong K (2017) Crossover and mutation operators of genetic algorithms. Int J Mach Learn Comput 7(1):9–12

Lin K, Pai P, Ting Y (2019) Deep belief networks with genetic algorithms in forecasting wind speed. IEEE Access 7:99244–99253

Lindenmayer A (1968) Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs. J Theor Biol 18(3):300–315

Liu G, Xiao L, Xiong C (2017) Image classification with deep belief networks and improved gradient descent. In: IEEE international conference on computational science and engineering (CSE) and IEEE international conference on embedded and ubiquitous computing (EUC), vol 1, pp 375–380

Liu H, Simonyan K, Vinyals O, Fernando C, Kavukcuoglu K (2017) Hierarchical representations for efficient architecture search. arXiv preprint arXiv:1711.00436

Liu J, Gong M, Miao Q, Wang X, Li H (2018) Structure learning for deep neural networks based on multiobjective optimization. IEEE Trans Neural Netw Learn Syst 29(6):2450–2463

Liu P, Basha MDE, Li Y, Xiao Y, Sanelli PC, Fang R (2019) Deep evolutionary networks with expedited genetic algorithms for medical image denoising. Med Image Anal 54:306–315

Liu Z, Luo P, Wang X, Tang X (2015) Deep learning face attributes in the wild. In: Proceedings of the 2015 IEEE international conference on computer vision (ICCV), ICCV ’15, pp 3730–3738

Loni M, Daneshtalab M, Sjödin M (2018a) ADONN: adaptive design of optimized deep neural networks for embedded systems. In: 21st Euromicro conference on digital system design (DSD), pp 397–404

Loni M, Majd A, Loni A, Daneshtalab M, Sjödin M, Troubitsyna E (2018b) Designing compact convolutional neural network for embedded stereo vision systems. In: IEEE 12th international symposium on embedded multicore/many-core systems-on-chip (MCSoC), pp 244–251

Lopez-Rincon A, Tonda A, Elati M, Schwander O, Piwowarski B, Gallinari P (2018) Evolutionary optimization of convolutional neural networks for cancer miRNA biomarkers classification. Appl Soft Comput 65:91–100

Lorenzo PR, Nalepa J (2018) Memetic evolution of deep neural networks. In: Proceedings of the genetic and evolutionary computation conference, pp 505–512

Loussaief S, Abdelkrim A (2018) Convolutional neural network hyper-parameters optimization based on genetic algorithms. Int J Adv Comput Sci Appl 9(10):252–266

Lu Z, Whalen I, Boddeti V, Dhebar Y, Deb K, Goodman E, Banzhaf W (2019) Nsga-net: neural architecture search using multi-objective genetic algorithm. In: Proceedings of the genetic and evolutionary computation conference, pp 419–427

Lucey P, Cohn JF, Kanade T, Saragih J, Ambadar Z, Matthews I (2010) The extended cohn-kanade dataset (ck+): a complete dataset for action unit and emotion-specified expression. IEEE conference on computer vision and pattern recognition. California, USA, San Francisco, pp 94–101

Mahfoud SW (1995) Niching methods for genetic algorithms. Ph.D. thesis. University of Illinois at Urbana Champaign, Urbana

Mamaev A (2018) Flowers recognition dataset

Marlin B, Swersky K, Chen B, Freitas N (2010) Inductive principles for restricted Boltzmann machine learning. J Mach Learn Res Proc Track 9:509–516

Martín A, Lara-Cabrera R, Fuentes-Hurtado F, Naranjo V, Camacho D (2018) EvoDeep: a new evolutionary approach for automatic deep neural networks parametrisation. J Parallel Distrib Comput 117:180–191

Martinez D, Brewer W, Behm G, Strelzoff A, Wilson A, Wade D (2018) Deep learning evolutionary optimization for regression of rotorcraft vibrational spectra. In: IEEE/ACM machine learning in HPC environments (MLHPC), pp 57–66

Martín A, Fuentes-Hurtado F, Naranjo V, Camacho D (2017) Evolving deep neural networks architectures for android malware classification. In: IEEE congress on evolutionary computation (CEC), pp 1659–1666

Mattioli F, Caetano D, Cardoso A, Naves E, Lamounier E (2019) An experiment on the use of genetic algorithms for topology selection in deep learning. J Electr Comput Eng 2019:1–12

McNicholas W, Levy P (2000) Sleep-related breathing disorders: definitions and measurements. Eur Respir J 15(6):988

Miikkulainen R, Liang JZ, Meyerson E, Rawal A, Fink D, Francon O, Raju B, Shahrzad H, Navruzyan A, Duffy N et al (2017) Evolving deep neural networks. corr abs/1703.00548 (2017). arXiv preprintarXiv:1703.00548

Miranda V, da Hora Martins J, Palma V (2014) Optimizing large scale problems with metaheuristics in a reduced space mapped by autoencoders-application to the wind-hydro coordination. IEEE Trans Power Syst 29(6):3078–3085

Mitchell M, Santorini B, Marcinkiewicz MA, Taylor A (1999) Treebank-3 ldc99t42. Phila Linguist Data Consort 3:2

Moriarty DE, Miikkulainen R (1997) Forming neural networks through efficient and adaptive coevolution. Evol Comput 5(4):373–399

Munder S, Gavrila DM (2006) An experimental study on pedestrian classification. IEEE Trans Pattern Anal Mach Intell 28(11):1863–1868

Muñoz-Ordóñez J, Cobos C, Mendoza M, Herrera-Viedma E, Herrera F, Tabik S (2018) Framework for the training of deep neural networks in TensorFlow using metaheuristics. In: Intelligent data engineering and automated learning—IDEAL 2018, pp 801–811. Springer

Nakisa B, Rastgoo MN, Rakotonirainy A, Maire F, Chandran V (2018) Long short term memory hyperparameter optimization for a neural network based emotion recognition framework. IEEE Access 6:49325–49338

Nalepa J, Lorenzo PR (2017) Convergence analysis of PSO for hyper-parameter selection in deep neural networks. In: Advances on P2P, parallel, grid, cloud and internet computing, pp 284–295. Springer

Negrinho R, Gordon G (2017) Deeparchitect: Automatically designing and training deep architectures. arXiv preprint arXiv:1704.08792

Nilsback M-E, Zisserman A (2008) Automated flower classification over a large number of classes. Conference on computer vision, graphics image processing. Madurai, India, pp 722–729

Nour M, Slay J (2015) Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: IEEE military communications and information systems conference (MilCIS), pp 1–6

NSL-KDD (2019) Dataset of nsl-kdd

Osaba E, Carballedo R, Diaz F, Onieva E, De La Iglesia I, Perallos A (2014) Crossover versus mutation: a comparative analysis of the evolutionary strategy of genetic algorithms applied to combinatorial optimization problems. Sci World J. https://doi.org/10.1155/2014/154676

Pandey S, Kaur G (2018) Curious to click it?-Identifying clickbait using deep learning and evolutionary algorithm. In: International conference on advances in computing, communications and informatics (ICACCI), pp 1481–1487

Patterson J, Gibson A (2017) Deep learning: a practitioner's approach. O'Reilly Media, Inc

Pavai G, Geetha T (2016) A survey on crossover operators. ACM Comput Surv (CSUR) 49(4):1–43

Pham H, Guan MY, Zoph B, Le QV, Dean J (2018) Efficient neural architecture search via parameter sharing. arXiv preprint arXiv:1802.03268

Pouyanfar S, Tao Y, Mohan A, Tian H, Kaseb AS, Gauen K, Dailey R, Aghajanzadeh S, Lu Y-H, Chen S-C, Shyu M-L (2018) Dynamic sampling in convolutional neural networks for imbalanced data classification. In: IEEE international conference on multimedia information processing and retrieval, pp 112–117

Proenca H, Filipe S, Santos R, Oliveira J, Alexandre LA (2010) The UBIRIS. v2: a database of visible wavelength iris images captured on-the-move and at-a-distance. IEEE Trans Pattern Anal Mach Intell 32(8):1529

Qolomany B, Maabreh M, Al-Fuqaha A, Gupta A, Benhaddou D (2017) Parameters optimization of deep learning models using particle swarm optimization. In: 13th International wireless communications and mobile computing conference (IWCMC), pp 1285–1290

Ranzato M, Poultney C, Chopra S, LeCun Y (2006) Efficient learning of sparse representations with an energy-based model. In: Proceedings of advances in neural information processing systems, Vancouver, BC, Canada, pp 1137–1144

Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248 (Special section on high order fuzzy sets)

Rashid TA, Fattah P, Awla DK (2018) Using accuracy measure for improving the training of LSTM with metaheuristic algorithms. Procedia Comput Sci 140:324–333

Real E, Aggarwal A, Huang Y, Le QV (2019) Regularized evolution for image classifier architecture search. Proc AAAI Conf Artif Intell 33:4780–4789

Real E, Moore S, Selle A, Saxena S, Suematsu YL, Tan J, Le QV, Kurakin A (2017) Large-scale evolution of image classifiers. In: Proceedings of machine learning research, PMLR, vol 70. International Convention Centre, Sydney, pp 2902–2911

Rechenberg I (1965) Cybernetic solution path of an experimental problem. Library translation 1122, Royal Aircraft Establishment, Farnborough, Hants, UK

Rere LMR, Fanany MI, Arymurthy AM (2016) Metaheuristic algorithms for convolution neural network. Comput Intell Neurosci 2016:1–13

Rere LR, Fanany MI, Arymurthy AM (2015) Simulated annealing algorithm for deep learning. Procedia Comput Sci 72:137–144

Rosa G, Papa J, Costa K, Passos L, Pereira C, Yang X-S (2016) Learning parameters in deep belief networks through firefly algorithm. In: Artificial neural networks in pattern recognition. Springer, pp 138–149

Rosa G, Papa J, Marana A, Scheirer W, Cox D (2015) Fine-tuning convolutional neural networks using harmony search. In: Pardo A, Kittler J (eds) Progress in pattern recognition, image analysis, computer vision, and applications. Springer, Cham, pp 683–690

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323(6088):533–536

Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2015) ImageNet large scale visual recognition challenge. Int J Comput Vis (IJCV) 115(3):211–252

Sabar NR, Turky A, Song A, Sattar A (2017) Optimising deep belief networks by hyper-heuristic approach. In: IEEE congress on evolutionary computation (CEC), pp 2738–2745

Sabar NR, Turky A, Song A, Sattar A (2019) An evolutionary hyper-heuristic to optimise deep belief networks for image reconstruction. Appl Soft Comput 97

Salih A, Moshaiov A (2016) Multi-objective neuro-evolution: should the main reproduction mechanism be crossover or mutation? In: IEEE international conference on systems, man, and cybernetics (SMC). IEEE, pp 004585–004590

Saxena A, Goebel K (2008) Turbofan engine degradation simulation data set

Schiffmann W (2000) Encoding feedforward networks for topology optimization by simulated evolution. In: Fourth international conference on knowledge-based intelligent engineering systems and allied technologies, KES'2000. Proceedings (Cat. No. 00TH8516), vol 1. IEEE, pp 361–364

Sehgal A, La H, Louis S, Nguyen H (2019) Deep reinforcement learning using genetic algorithm for parameter optimization. In: Third IEEE international conference on robotic computing (IRC). IEEE, pp 596–601

Semeion (2008) Semeion handwritten digit data set

Shi W, Liu D, Cheng X, Li Y, Zhao Y (2019) Particle swarm optimization-based deep neural network for digital modulation recognition. IEEE Access 7:104591–104600

Shrestha A, Mahmood A (2019) Review of deep learning algorithms and architectures. IEEE Access 7:53040–53065

Silva PH, Luz E, Zanlorensi LA, Menotti D, Moreira G (2018) Multimodal feature level fusion based on particle swarm optimization with deep transfer learning. In: IEEE congress on evolutionary computation (CEC), pp 1–8

Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556

Sinha T, Haidar A, Verma B (2018) Particle swarm optimization based approach for finding optimal values of convolutional neural network parameters. In: IEEE congress on evolutionary computation (CEC), pp 1–6

Smith WA, Randall RB (2015) Rolling element bearing diagnostics using the case western reserve university data: a benchmark study. Mech Syst Signal Process 64(12):100–131

Smolensky P (1986) Chapter 6: Information processing in dynamical systems: foundations of harmony theory. In: Rumelhart DE, McClelland JL, Group PR (eds) Parallel distributed processing: explorations in the microstructure of cognition: foundations, vol 1. MIT Press, Cambridge, pp 194–281

Soon FC, Khaw HY, Chuah JH, Kanesan J (2018) Hyper-parameters optimisation of deep CNN architecture for vehicle logo recognition. IET Intell Transp Syst 12(8):939–946

Stanley KO, D’Ambrosio DB, Gauci J (2009) A hypercube-based encoding for evolving large-scale neural networks. Artif Life 15(2):185–212

Stanley KO, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. Evol Comput 10(2):99–127

Steinholtz OS (2018) A comparative study of black-box optimization algorithms for tuning of hyper-parameters in deep neural networks

Storn R, Price K (1995) Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, International Computer Science Institute, Berkley

Suganuma M, Shirakawa S, Nagao T (2017) A genetic programming approach to designing convolutional neural network architectures. In: Proceedings of the genetic and evolutionary computation conference, GECCO ’17. Association for Computing Machinery, New York, pp 497–504

Sun Y, Xue B, Zhang M, Yen GG (2018) Automatically evolving cnn architectures based on blocks. arXiv preprint arXiv:1810.11875

Sun Y, Xue B, Zhang M, Yen GG (2018) An experimental study on hyper-parameter optimization for stacked auto-encoders. In: IEEE congress on evolutionary computation (CEC), pp 1–8

Sun Y, Xue B, Zhang M, Yen GG (2019a) Evolving deep convolutional neural networks for image classification. IEEE Trans Evolut Comput 24:394–407

Sun Y, Xue B, Zhang M, Yen GG (2019b) A particle swarm optimization-based flexible convolutional autoencoder for image classification. IEEE Trans Neural Netw Learn Syst 30(8):2295–2309

Sung K-K, Poggio T (1998) Example-based learning for view-based human face detection. IEEE Trans Pattern Anal Mach Intell 20(1):39–51

Syulistyo AR, Purnomo DMJ, Rachmadi MF, Wibowo A (2016) Particle swarm optimization (PSO) for training optimization on convolutional neural networK (CNN). Jurnal Ilmu Komputer dan Informasi 9(1):52

Szegedy C, Ioffe S, Vanhoucke V, Alemi A (2016) Inception-v4, inception-resnet and the impact of residual connections on learning. arXiv preprint arXiv:1602.07261

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

Tanaka T, Moriya T, Shinozaki T, Watanabe S, Hori T, Duh K (2016) Automated structure discovery and parameter tuning of neural network language model based on evolution strategy. In: IEEE spoken language technology workshop (SLT), pp 665–671

TCWB (2019) Wind speed and weather-related data at the Penghu station in Taiwan

The Cancer Genome Atlas (TCGA) (2006) The cancer genome atlas (TCGA)

Tian H, Pouyanfar S, Chen J, Chen S, Iyengar SS (2018) Automatic convolutional neural network selection for image classification using genetic algorithms. In: IEEE international conference on information reuse and integration (IRI), pp 444–451

Tian H, Tao Y, Pouyanfar S, Chen S-C, Shyu M-L (2018) Multimodal deep representation learning for video classification. World Wide Web 1:1–17

Tian Y, Liu X (2019) A deep adaptive learning method for rolling bearing fault diagnosis using immunity. Tsinghua Sci Technol 24(6):750–762

Tian Z, Fong S (2016) Survey of meta-heuristic algorithms for deep learning training. In: Optimization algorithms—methods and applications. InTech, pp 195–220

Tirumala SS, Ali S, Ramesh CP (2016) Evolving deep neural networks: a new prospect. In: 12th International conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD), pp 69–74

Torralba A, Fergus R, Freeman WT (2008) 80 million tiny images: a large data set for nonparametric object and scene recognition. IEEE Trans Pattern Anal Mach Intell 30(11):1958–1970

Trivedi A, Srivastava S, Mishra A, Shukla A, Tiwari R (2018) Hybrid evolutionary approach for Devanagari handwritten numeral recognition using convolutional neural network. Procedia Comput Sci 125:525–532

VIA/I-ELCAP (2019) Elcap public lung image database

Vidnerova P, Neruda R (2017) Evolution strategies for deep neural network models design. In: CEUR workshop proceedings, Proceedings of the 17th conference on information technologies—applications and theory, ITAT 2017, pp 159–166

Vito SD, Fattoruso G, Pardo M, Tortorella F, Francia GD (2012) Semi-supervised learning techniques in artificial olfaction: a novel approach to classification problems and drift counteraction. IEEE Sens J 12(11):3215–3224

Wade D, Vongpaseuth T, Lugos R, Ayscue J, Wilson A, Antolick L, Brower N, Krick S, Szelistowski M, Albarado K (2015) Machine learning algorithms for hums improvement on rotorcraft components. In: AHS Forum 71, at Virginia Beach, VA

Wang B, Sun Y, Xue B, Zhang M (2018) Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification. In: IEEE congress on evolutionary computation (CEC), pp 1–8

Wang C, Xu C, Yao X, Tao D (2019) Evolutionary generative adversarial networks. IEEE Trans Evolut Comput 23:921–934

Wang Y, Zhang H, Zhang G (2019) cPSO-CNN: an efficient PSO-based algorithm for fine-tuning hyper-parameters of convolutional neural networks. Swarm Evol Comput 49:114–123

Wei P, Li Y, Zhang Z, Hu T, Li Z, Liu D (2019) An optimization method for intrusion detection classification model based on deep belief network. IEEE Access 7:87593–87605

Wistuba M (2018) Deep learning architecture search by neuro-cell-based evolution with function-preserving mutations. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, pp 243–258

Xiao H, Rasul K, Vollgraf R (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms

Xie L, Yuille A (2017) Genetic CNN. In: Proceedings of the IEEE international conference on computer vision (ICCV)

Xie S, Zheng H, Liu C, Lin L (2018) SNAS: stochastic neural architecture search. arXiv e-prints

Ye F (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)

Yinka-Banjo C, Ugot O-A (2019) A review of generative adversarial networks and its application in cyber-security. Artif Intell Rev 53:1721–1736

Yoo Y (2019) Hyperparameter optimization of deep neural network using univariate dynamic encoding algorithm for searches. Knowl-Based Syst 178:74–83

Yu F, Seff A, Zhang Y, Song S, Funkhouser T, Xiao J (2015) LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365

Yuliyono AD, Girsang AS (2019) Artificial bee colony-optimized LSTM for bitcoin price prediction. Adv Sci Technol Eng Syst J 4(5):375–383

Zavalnyi O, Zhao G, Savchenko Y, Xiao W (2018) Experimental evaluation of metaheuristic optimization of gradients as an alternative to backpropagation. In: IEEE 4th International conference on computer and communications (ICCC), pp 2095–2099

Zhang C, Lim P, Qin AK, Tan KC (2017) Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. IEEE Trans Neural Netw Learn Syst 28(10):2306–2318

Zhang C, Sun JH, Tan KC (2015) Deep belief networks ensemble with multi-objective optimization for failure diagnosis. In: IEEE international conference on systems, man, and cybernetics, pp 32–37

Zhong Z, Yan J, Wu W, Shao J, Liu C-L (2018) Practical block-wise neural network architecture generation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2423–2432

Zhu H, An Z, Yang C, Xu K, Zhao E, Xu Y (2019) EENA: efficient evolution of neural architecture. In: Proceedings of the IEEE international conference on computer vision workshops

Zoph B, Le QV (2016) Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578

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