Survey on Evolutionary Deep Learning: Principles, Algorithms, Applications, and Open Issues

NAN LI and LIANBO MA, Northeastern University, China
GUO YU, Nanjing Tech University, China
BING XUE and MENGJIE ZHANG, Victoria University of Wellington, New Zealand
YAOCHU JIN, Bielefeld University, Germany

Over recent years, there has been a rapid development of deep learning (DL) in both industry and academia fields. However, finding the optimal hyperparameters of a DL model often needs high computational cost and human expertise. To mitigate the above issue, evolutionary computation (EC) as a powerful heuristic search approach has shown significant merits in the automated design of DL models, so-called evolutionary deep learning (EDL). This article aims to analyze EDL from the perspective of automated machine learning (AutoML). Specifically, we first illuminate EDL from DL and EC and regard EDL as an optimization problem. According to the DL pipeline, we systematically introduce EDL methods ranging from data preparation, model generation, to model deployment with a new taxonomy (i.e., what and how to evolve/optimize), and focus on the discussions of solution representation and search paradigm in handling the optimization problem by EC. Finally, key applications, open issues, and potentially promising lines of future research are suggested. This survey has reviewed recent developments of EDL and offers insightful guidelines for the development of EDL.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Computing methodologies** → **Machine learning algorithms**; • **Theory of computation** → **Evolutionary algorithms;**

Additional Key Words and Phrases: Deep learning, evolutionary computation, data preparation, model generation, model deployment

ACM Reference format:
Nan Li, Lianbo Ma, Guo Yu, Bing Xue, Mengjie Zhang, and Yaochu Jin. 2023. Survey on Evolutionary Deep Learning: Principles, Algorithms, Applications, and Open Issues. *ACM Comput. Surv.* 56, 2, Article 41 (September 2023), 34 pages.
https://doi.org/10.1145/3603704

This work is supported by the National Natural Science Foundation of China No. 62103150, the National Key R&D Program of China under Grant No. 2022YFB4500800, the Project funded by China Postdoctoral Science Foundation No. 2021M691012, the Joint Funds of the Natural Science Foundation of Liaoning Province und Grant 2021-KF-11-01, and the Fundamental Research Funds for the Central Universities No. N2117005.
Authors' addresses: N. Li and L. Ma (corresponding author), Northeastern University, No. 195, Chuangxin Road, Shenyang, Liaoning Province, China; emails: 2010500@stu.neu.edu.cn, malb@swc.neu.edu.cn; G. Yu, Nanjing Tech University, Jiangbei New District, Nanjing, Jiangsu Province, China; email: gysearch@163.com; B. Xue and M. Zhang, Victoria University of Wellington, 21 Kelburn Pde Wellington 6012, New Zeeland; emails: {bing.xue, mengjie.zhang}@ecs.vuw.ac.nz; Y. Jin, Bielefeld University, Universitaetsstreet 25·33615 Bielefeld, Germany; email: yaochu.jin@uni-bielefeld.de.

0360-0300/2023/09-ART41 $15.00
https://doi.org/10.1145/3603704

1 INTRODUCTION

Deep learning (DL) as a promising technology has been widely used in a variety of challenging tasks, such as image analysis [102] and pattern recognition [104]. However, the practitioners of DL struggle to manually design deep models and carefully find appropriate configurations by trial-and-error. An example is given in Figure 1, where domain knowledge is fed to DL in different stages such as data preparation [208], model generation [235], and model deployment [33, 35]. In fact, the manual design of novel DL models requires much knowledge and experience of human experts, which may take a large number of trials to achieve satisfied DL models. These may limit the development of DL in various applications.

In contrast, the automatic design of **deep neural networks (DNNs)** tends to be prevalent in recent decades [74, 235]. The main reason lies in the flexibility and computation efficiency of **automated machine learning (AutoML)** in data preparation (i.e., **feature selection (FS)** [208], **feature construction (FC)** [140], **feature extraction (FE)** [6], and **data augmentation (DA)** [197]), model generation (i.e., **parameter optimization (PO)** [226], **architecture optimization (AO)** [173], and **joint optimization (JO)** [74, 213, 235]), and model deployment (i.e., **model pruning (MP)** [80], and other model compression techniques [152, 153]). In this way, AutoML without manual intervention has attracted great attention and much progress has been made.

Evolutionary computation (EC) has been widely applied to automatic DL, termed as **evolutionary deep learning (EDL)** [53, 184, 227, 228], owing to its flexibility and automatically evolving mechanism [216]. In principle, EC evolves a population of individuals towards the optimum/optima via evolutionary operations and environmental selection, which is a gradient-free and robust search paradigm [91, 217]. As a result, the integration of EC and DL has become a hot research topic in both academic and industrial communities. Moreover, in Figure 2, the number of publications and citations referring to EC and DL by years from Web of Science gradually increases until around 2018, whereas it sharply rises in the recent years. Hence, more and more researchers work on the area of EDL.

In Table 1, we have listed recent surveys on automatic DL. A large number of existing surveys focus on different aspects of the DL pipeline (e.g., AO [235] or FS [208]) rather than the whole pipeline. For example, References [65, 172] concentrate on the aspect of deep model architecture search from the perspective of neuroevolution. Many others focus on specific optimization paradigms such as **reinforcement learning (RL)** [86], EC [180], and gradient [164]. However, very few of them have systematically analyzed EDL and runs the gamut of FS, FC, FE, DA, PO, AO, JO, MP, and other EC-based approaches for model deployment. To fill the gap, we aim to give a comprehensive review of EDL in detail. The main contributions of this work are as follows:

- Existing work on EDL is reviewed from the perspective of DL and EC to facilitate the understanding of readers from the communities of both ML and EC, and we also formulated EDL into an optimization problem from the perspective of EC.
- The survey describes and discusses EDL in terms of data preparation, model generation, and model deployment from a novel taxonomy, where the solution representation and the search paradigms are emphasized and systematically discussed. To the best of our knowledge, few surveys have investigated the evolutionary model deployment.
- On the basis of the comprehensive review of EDL approaches, a number of applications, open issues, and trends of EDL are discussed, which will guide the development of EDL.

The rest of this article is organized as follows: Section 2 presents an overview of EDL. In Section 3, EC-driven data preparation is presented. EC-driven model generation is discussed in Section 4. Section 5 reviews EC-driven model deployments. Section 6 describes the common

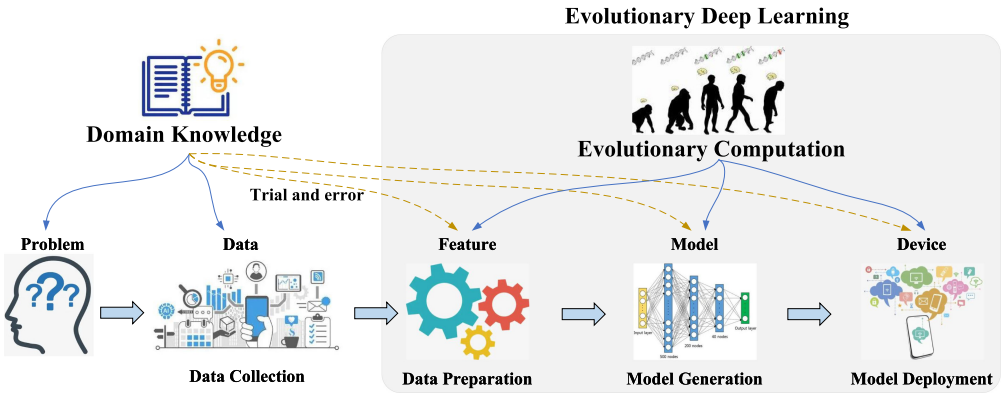

Fig. 1. An overview of EDL, driven by domain knowledge or evolutionary computation, where the life of DL gets through problem, data collection, data preparation, model generation, and model deployment.

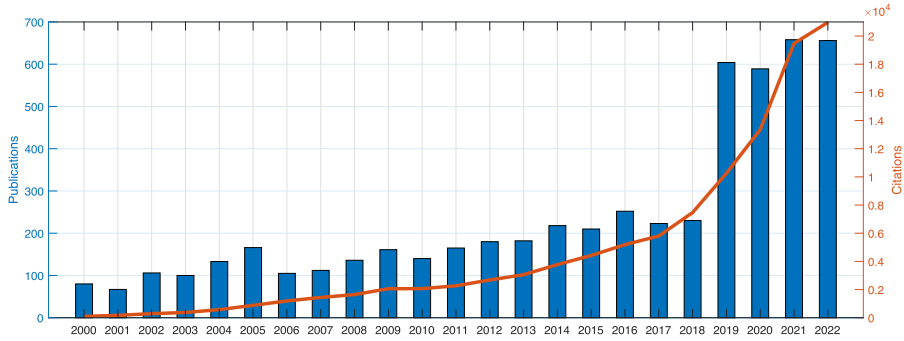

Fig. 2. Total publications and citations referring to EC and DL by years from Web of Science until April 2023.

acceleration strategies in EDL. After that, relevant applications, open issues, and the trends of EDL are discussed in Section 7. A conclusion of the article is drawn in Section 8.

2 AN OVERVIEW OF EVOLUTIONARY DEEP LEARNING

2.1 Deep Learning

DL can be described as a triplet $M = (D, T, P)$ [213], where D is the dataset used for the training of a deep model (M), and T is the targeted task. P indicates the performance of M. The aim of DL is to *boost its performance over specific task T, which is measured by P on dataset D*. In Figure 1, we can see there are three fundamental processes of DL, i.e., data preparation, model generation, and model deployment.

Data preparation: It aims to find a high-quality D to improve the performance (P) of the deep model (M) on specific tasks (T). In practice, the feature space of D may include redundant and noisy information, which harms the performance (P) of the model (M). For example, on Prostate dataset, the size of feature subset (65) selected in Reference [187] is only 1% of the total size of features (10,509).

Model generation: It targets at optimizing/generating a model (M) with desirable performance (P) for specific task (T) on the given datasets (D) [74]. For model generation, the efficiency of the

Table 1. Comparison between Existing Surveys and our Work, Where FS, FC, FE, DA, PO, AO, JO, MP, and Others Indicate Feature Selection, Feature Construction, Feature Extraction, Data Augmentation, Parameter Optimization, Architecture Optimization, Joint Optimization, Model Pruning, and other Model Compression Techniques for Model Deployment, Respectively

Survey	Type	Data preparation				Model generation			Model deployment	
		FS	FC	FE	DA	PO	AO	JO	MP	Others
[213]	AutoML	✢	✢	✢	-	✢	-	-	-	-
[74]	AutoML	✓	✓	✓	-	-	✓	✓	-	-
[214]	EDL	-	-	-	-	✓	✓	✓	✢	-
[159]	EDL	-	-	-	-	-	✓	✓	✢	-
[86]	EDL	-	-	-	-	-	✓	✓	-	-
[164]	EDL	-	-	-	-	-	✓	✓	-	-
[184]	EDL	✢	✢	✢	✢	-	✓	✓	✢	-
[118]	EDL	-	-	-	-	-	✓	✓	✢	-
[235]	EDL	-	-	-	-	✓	✓	✓	-	-
[208]	EDL	✓	✢	-	-	-	-	-	-	-
[5]	EDL	✓	✓	-	-	-	✓	✓	-	-
[7]	EDL	✢	✢	-	-	-	✓	✢	✢	-
[132]	EDL	✓	✓	-	-	-	✓	✓	-	-
[19]	EDL	✓	✓	✓	-	-	✓	-	-	-
[45]	EDL	-	-	-	-	-	✓	✢	✢	-
[60]	EDL	✢	✢	✢	-	-	✓	✢	✢	-
[172]	EDL	-	-	-	-	-	✓	✓	✢	-
[65]	EDL	-	-	-	-	-	✓	✓	✢	-
[223]	EDL	-	-	-	✓	✓	✓	✓	✢	-
Ours	EDL	✓	✓	✓	✓	✓	✓	✓	✓	✓

"✓", "✢", and "-" indicate that the content is mentioned or briefly mentioned or not mentioned in the original paper, respectively.

search and the size of the final architecture need to be considered. Thus, the design of search spaces and acceleration strategies is very important for model generation.

Model deployment: This process aims to deploy a deep model (M) to solve a deployment task T with acceptable performance (P) on input data (D) within limited computational budgets. The key issue of model deployment is how to reduce the latency, storage, and energy consumption when the number of parameters of a deep model is large, e.g., Transformer-XL Large has 257M parameters [47].

2.2 Evolutionary Computation

EC is a collection of stochastic population-based search methods inspired by evolution mechanisms such as natural selection and genetics, which does not need gradient information and is able to handle a black-box optimization problem without explicit mathematical formulations [125, 191, 219]. In principle, we can broadly divide EC methods into three categories: **evolutionary algorithms (EAs), swarm intelligence (SI)**, and other algorithms. More specifically, promising examples of EAs include **genetic algorithm (GA)** [11], **genetic programming (GP)** [21, 22], **evolutionary strategy (ES)** [82], **evolution programming (EP)** [78], and **estimation of distribution algorithm (EDA)** [48]. The representative algorithms of SI include **ant colony optimization (ACO)** [220], **particle swarm optimization (PSO)** [97], **brain storm optimization (BSO)** [87], **artificial bee colony (ABC)** [198], and so on. Other algorithms cover **memetic algorithm (MA)** [200], **artificial immune systems (AIS)** [106], and so on. EC complies with a general framework, as shown in Figure 3, which consists of three main components. In addition, we provide a detailed comparison between EC and other optimization methods (e.g., RL, Bayesian and gradient) in the Section 1 of the online supplementary material, with the aim of enabling the reader to better understand the advantages of EC.

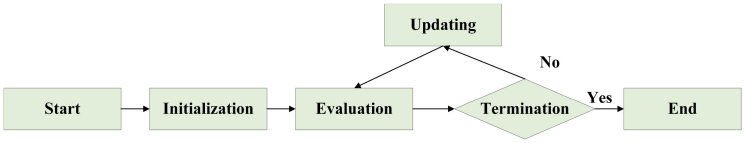

Fig. 3. A general framework of EC.

Initialization is performed to generate a population of individuals that are encoded according to the decision space (or search space and variable space) of the optimization problem, such as the feature set, model parameters, and topological structure.

Evaluation aims to calculate the fitness of individuals. In fact, the evaluation of the individuals in EDL is a computationally expensive task [131]. For example, Reference [157] used 3,000 GPU days to find a desirable architecture.

Updating aims to generates a number of offspring solutions through various reproduction operations. A new soultion is generated via velocity and position formula in PSO [187]. For GA, some reproduction operators (e.g., crossover and mutation) are used to generate new individuals [190].

2.3 Evolutionary Deep Learning

2.3.1 EDL from Two Perspectives. In contrast to traditional DL that heavily relies on expert or domain knowledge to build deep model, EDL is to automatically design the deep model through an evolutionary process [159, 180, 214, 227].

From the perspective of DL: Traditional DL needs a lot of expert knowledge in inventing and analyzing a learning tool to a specific dataset or task. In contrast, EDL can be seen as a human-friendly learning tool that can automatically find appropriate deep models on given datasets or tasks [213]. In other words, *EDL concentrates on how easy a learning tool can be used.*

From the perspective of EC: The configurations of a model are represented as an individual and the performance as the objective to be optimized. EC plays an important role in the optimization driven by evolutionary mechanisms [218]. Namely, *EDL can be seen as an evolutionary optimization process to find the optimal configurations of the deep model with high performance.*

From the above analysis, EDL not only aims to increase the adaptability of a deep model towards learning tasks via the automatic construction approach (from the perspective of DL), but also tries to achieve the optimal model under the designed objectives or constraints (from the perspective of EC).

2.3.2 Definition and Framework of EDL. According to the above discussion in Section 2.3.1 and following Reference [213], we can define EDL as follows:

$$\begin{aligned} \underset{config.}{\text{Max}} \quad & \text{Learning tools performance } (P) \text{ for target tasks } (T) \text{ on datasets } (D) \\ s.t. \quad & \begin{cases} \text{No assistance from humans} \\ \text{Limited computational budgets} \end{cases}, \end{aligned} \tag{1}$$

where $config.$ indicates the configurations that form the decision space of an optimization problem. The problem is to maximize the objective (i.e., learning tools' performance P) of target tasks T on datasets D under the constraints of no assistance from humans and limited computational resources. Accordingly, three aspects are taken into account in the design of EDL.

Desirable generalization performance: EDL should have desirable generalization performance across given datasets and tasks.

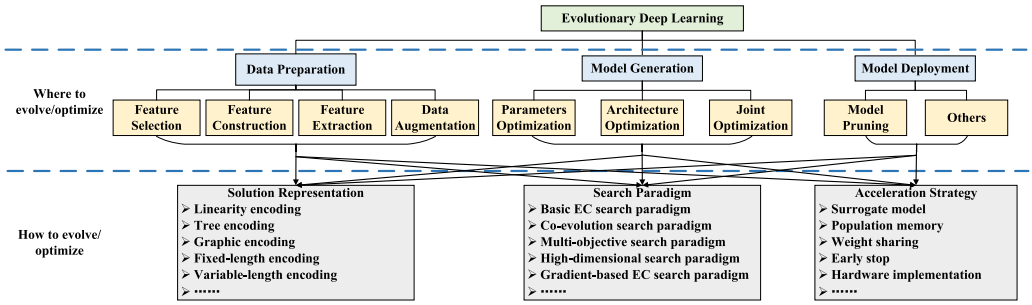

Fig. 4. A taxonomy of EDL approaches.

High search efficiency: EDL is able to find optimal or desirable configuration within a limited computational budgets (e.g., hardware, latency, energy consumption) under different designed objectives (e.g., high accuracy, small model size).

Without human assistance: EDL is able to automatically configure without human intervention.

Following the EC framework described in Figure 3, we present a general optimization framework of EDL as follows:

Step 1 Initialization: A population of individuals are initialized according to the designed encoding scheme.

Step 2 Evaluation: Each individual is evaluated according to the objectives (e.g., high accuracy, small model size) or constraints (e.g., energy consumption).

Step 3 Updating: A required number of new solutions are generated from previous generation via various updating operations.

Step 4 Termination condition: Go to Step 2 if the predefined termination condition is unsatisfied; Otherwise, go to Step 5.

Step 5 Output: Output the solution with the best performance.

2.3.3 Taxonomy of EDL Approaches. In this section, a novel taxonomy of EDL approaches is proposed according to "what to evolve/optimize" and "how to evolve/optimize," as shown in Figure 4.

"What to evolve/optimize": We may be concerned about "what EDL can do" or "what kinds of problems EDL can tackle." In data preparation, there are four key issues to be resolved, including the FS, FC, FE, DA. In model generation, PO, AO, and JO become the critical issues [235], while model deployment is involved with the issues of MP and other compression technologies.

"How to evolve/optimize": The answer to the question is designing appropriate solution representation and search paradigm for EC. The representation schemes are designed for the encoding of individuals, search paradigms for the achievement of optimal configurations, acceleration strategies for the reduction of time or resources consumption.

According to the above taxonomy, we will elaborately introduce EDL in data preparation, model generation, and model deployment in Sections 3, 4, and 5, respectively, and acceleration strategies of EDL are provided in Section 6.

3 DATA PREPARATION

Generally, to enhance the prediction performance, the deep learning model usually integrates complex feature extraction structure to get task-specific features for the downstream tasks, which in

fact increases the black-box feature (i.e., uninterpretable) as well as influences the convergence performance. Therefore, the additional feature engineering step can reduce the complexity of the deep model structure and thereby enhance the performance of the model (e.g., alleviating under-fitting and performance collapse) [51].

Data preparation is an important step in DL pipeline, aiming to improve the quality of the data for deep models by filtering out the irrelevant features of the data or creating the new features/data based on original data [107, 208]. Various EC-based techniques have been proposed to reduce data dimensionality, speed up learning process, or improve model performance [208]. The common techniques can be categorized into FS [144], FC [18], FE [149], and DA [39].

3.1 Feature Selection

3.1.1 Problem Formulation. FS is to automatically select a representative subset F by removing irrelevant or redundant features, aiming to minimize model loss $\mathcal{L}(.)$. However, the search space grows exponentially with the increase of features. If the original feature set has M features, then there are 2^M solutions in the search space. In addition, the interactions between features may seriously impact the FS performance [208]. The FS problem can be defined as:

$$\begin{gathered} min \mathcal{L}(F) \\ s.t. \quad F = S(F_o, x_m), \quad |F| < M \\ F_o = \left(f_1^o, f_2^o, \ldots, f_M^o\right), \quad x_m \in \{0,1\} \quad \forall m \in \{1,2,\ldots,M\}, \end{gathered} \tag{2}$$

where $S(.)$ is a selection function. F_o indicates the original feature set. $x_m = 1$ indicates that the mth corresponding feature is selected. Otherwise, the corresponding feature is discarded. In the following, we will review existing work on solution representations and search paradigms in EC for FS.

3.1.2 Solution Representations. Generally, there are three different categories of solution representations.

Linear encoding: This encoding uses vectors or strings to store feature information. For example, in Reference [52], a fixed-length binary vector was used to express whether a feature is selected or not, where "1" indicates a corresponding feature is selected, and "0" is the opposite. In Reference [76], a binary index was used to indicate the corresponding feature.

Tree-based encoding: In canonical GP, all leaf nodes/terminal nodes represent the selected features and non-terminal nodes represent functions (e.g., arithmetic or logic operators) [101]. For automatic classification on high-dimensional data, Krawiec et al. [101] proposed a tree-based encoding to select a subset of highly discriminative features, where each feature consisted of sibling leaf nodes and their paternal function node. On the basis of the tree-based encoding, Muni et al. [138] proposed a multi-tree GP method for online FS.

Graph-based encoding: In Reference [145], the feature space of the high-dimensional data is represented by a graph, and each node of the graph represents a feature. A feature subset is composed of visited nodes of the graph, i.e., the path of node composition or subgraph. Yu et al. [220] converted FS to the optimal path problem in a directed graph, where the value of the node was "1" or "0" to indicate whether the feature was selected or not.

3.1.3 Search Paradigms. In FS, representative types of search paradigms are introduced as follows:

Basic EC search paradigm: In FS, typical evolutionary search methods have been widely used, such as GA [41, 52], GP [101], PSO [144, 192], ACO [127], and ABC [198]. Besides, some other studies [95] combined ACO with DE to seek optimal feature subsets, where the solutions searched by the ACO were fed into the DE to further explore the optimal solution. In Reference [41], a

family of FS methods based on different variants of GA was developed to improve the accuracy of content-based image retrieval systems.

Co-evolution search paradigm: In co-evolution search paradigm for FS, at least two populations are simultaneously evolved and interacted toward the optimal subset of features [155, 200]. For example, a divide-and-conquer strategy was developed in Reference [200] to manage two subpopulations. One subpopulation was to conduct an evolution process of classifier design, while the other one was to search for an optimal subset of features.

Multi-objective search paradigm: This type of search paradigms are driven by two or more conflicting objectives [32, 73, 207], such as the maximization of the accuracy of a classifier and minimization of the size of a feature subset. On the basis of the above two conflicting objectives, Xue et al. [207] designed a multi-objective PSO algorithm for FS and obtained a set of Pareto nondominated candidate solutions for FS after the multi-objective search.

3.1.4 Summary. Table S2 of the online supplementary material documents the classical FS works based on the solution representations and search paradigms. GA and GP are widely applied to FS. GA early serves for low-dimensional (i.e., ≤1,000) datasets [77, 208]. Recently, many GA-based approaches have been proposed to solve high-dimensional FS [32]. Nevertheless, GP is commonly applied to large-scale/high-dimensional FS, since it is flexible in feature representation [208]. Especially, GP outperforms GA on some small but high-dimensional datasets, e.g., Brain Tumor-2 [97] with 10,367 features but only 50 samples. In addition, PSO has been proved with faster convergence rate to an optimal feature subset than GAs and GP [230]. The graph representation of ACO outperforms GA and GP on flexibility, but the challenge of ACO is how to design appropriate graph encoding for large-scale scenarios [184, 208].

3.2 Feature Construction

3.2.1 Problem Formulation. FC aims to create new high-level features F_a (size = N) from the original features F_o (size = M) [189] via appropriate function operators $T_o(.)$ (e.g., conjunction and average) [140, 209], so the high-level features are more easily discriminative than the original ones. After FC, the additional feature set F_a is generated based on the original feature set F_o by function operators. The original feature set and the additional features are combined to form more expressive feature set F (size = $M + N$) for the model training [136]. FC can be represented as:

$$\begin{aligned} & & & min \mathcal{L}(F) & & \\ s.t. \quad & F = F_o + F_a = \left(f_1^o, f_2^o, \ldots, f_M^o, f_1^a, f_2^a, \ldots, f_N^a\right) & , \\ F_o = \left(f_1^o, f_2^o, \ldots, f_M^o\right), \quad & F_a = T_o(F_o) = \left(f_1^a, f_2^a, \ldots, f_N^a\right) & & \end{aligned} \tag{3}$$

where f_i^o denotes ith feature in F_o and f_i^a represents ith feature in F_a. FC is a complicated combinatorial optimization problem, where search space increases exponentially along with the total number of original features and the function operators. In the following subsections, we will describe the EC-based FC methods in terms of both solution representations and search paradigms.

3.2.2 Solution Representations. Existing EC-based approaches for FC can be categorized into three groups.

Linear encoding: Reference [209] used n-bit (n is the total number of original features) binary vector to represent each particle, where "0" indicated the corresponding feature not applied to build the new high-level feature while "1" was in the opposite. On the basis of the encoding, a local search was performed to select candidate operators from a predefined function set to construct a new high-level feature.

Tree-based encoding: Tree-based encoding is natural for FC, where leaf nodes represent the feature information and internal nodes represent operators. Many studies [18, 189] have demonstrated the effectiveness of tree encoding in FC. For example, Bhanu et al. [18] designed a GP-based coevolutionary FC procedure to improve the discriminative ability of classifiers. In Reference [189], an individual in EC was represented by a multi-tree encoding with multiple high-level features.

Graph-based encoding: In this encoding, the nodes and edges represent features and operators (e.g., "+", "-", "*", "/"), respectively. Teller et al. [185] applied an arbitrary directed graph to represent all features and operators, where each possible high-level feature can be represented as a subgraph of this directed graph. For linear GP, features and operations form a many-to-many directed acyclic graph, in which each feature is loaded into predefined registers and register's value can be used in multiple operators [57]. However, graph encoding becomes inefficient on high-dimensional feature sets, since the complexity of graph traversal exacerbates the difficulty of FC.

3.2.3 Search Paradigms. There are four categories of search paradigms for FC in existing works.

Basic EC search paradigm: Existing studies include but are not limited to GA [190] and GP [141, 183, 189]. For example, Reference [141] designed GP-based FC to reduce the feature (input) dimensions of a classifier. Especially, GP has been also widely used to construct new features, where each individual following the form of a GP tree usually represents a constructed high-level feature [66].

Co-evolution search paradigm: It can be decomposed to FC subproblem and classifier design subproblem, and each subproblem is solved with a standalone subpopulation by an EC-based method [18, 160]. For example, Reference [160] decomposed construction problem into two subproblems (i.e., FC, and object detection), where the FC was solved by evolving a population of pixel (i.e., feature) and the object detection was optimized using **object detection algorithm (ODA)** [160].

Multi-features search paradigm: Unlike early methods [167, 188–190] only constructing one high-level feature in a single search process, this sort of paradigm is able to create multiple high-level features. For example, Ahmed et al. [3] employed Fisher criterion together with p-value measure as the discriminant information between classes, based on which multiple features were constructed through multiple GP trees.

Multi-objective search paradigm: In this search paradigm, the number of features and classification accuracy are commonly taken into account as the objective functions for multi-objective evolutionary optimization [25, 71]. Especially, Hammami et al. [71] constructed a set of high-level features by optimizing a **multi-objective optimization problem (MOP)** with three objectives (i.e., the number of features, the mutual information, and classification accuracy) with Pareto dominance relationship.

3.2.4 Summary. Existing works on FC are listed in Table S3 of the online supplementary material based on the solution representations and search paradigms. GP-based approaches are popular in FC due to the flexible representation of features and operations. In addition, the hybrid of evolutionary algorithms also attracts much attention for FC. However, there is still plenty of room for the improvement of efficiency in constructing features in high-dimensional or large-scale scenarios, where a large number of computational resources are needed [183, 188]. Notably, FC often requires more computational overhead than FS, since FC commonly performs after the FS and the quality of the selected features may influence the performance of FC.

3.3 Feature Extraction

3.3.1 Problem Formulation. FE is to reduce the feature dimensions by altering the original features/data via some transformation functions [74]. Traditional extractors include **principal**

component analysis (PCA) [1] and **linear discriminant analysis (LDA)** [85]. However, they cannot keep somewhat important information after the transformation [1] and it is tedious to tune their hyperparameters (e.g., number of retained features) to find the best extraction. Thus, automatically finding high-quality map functions by EC-based approaches to achieve informative feature set tends to be popular. FE task is explained as:

$$\begin{aligned} & min \mathcal{L}\left(B\right) \\ s.t. \qquad & B = T_o\left(F_o\right) = \left(b_1, b_2, \ldots, b_N\right), \qquad M > N \\ & F_o = \left(f_1^o, f_2^o, \ldots, f_M^o\right), \end{aligned} \tag{4}$$

where B is the new features extracted from the original features F_o by the transformation function T_o. The new features B generated by FE are usually used directly for model training while in FC, the new features are combined with and the original features to form a more expressive feature set for parameter learning [136]. Besides, traditional feature extraction (e.g., local binary pattern [69] and histogram of oriented gradient [43]) may generate high-level features from low-level data (e.g., image pixels), while FC aims to construct new high-level features from the original low-level features. In the following, we will introduce the EC-based FE methods in terms of both solution representations and search paradigms.

3.3.2 Solution Representations. There are two typical ways for solution representation in EC-driven FE.

Linear encoding: In this encoding, map functions [6, 158] or function parameters [232] are encoded as a linear format. For example, Wissam et al. [6] predefined three sets of track functions (i.e., trace functions, diametric functions, and circus functions) for FE, and the optimal combination between the functions was obtained by an EC-based method. In Reference [232], the hyperparameters of map functions were encoded by some linear vectors that were constructed by a number of optimal projection basis vectors obtained via EC.

Tree-based encoding: In tree-based encoding, leaf nodes represent original features or constants, while the non-leaf nodes are some operators for FE including common arithmetic, logical operators (i.e., "+", "/", "∪") or other transformation operators (e.g., uLBP, and SobelY). In EC-driven FE, an individual represents a feature extractor or map function [149, 231]. Especially, an EC-based framework was developed in Reference [231] to search for features and sequences of operations by use of tree-based encoding.

3.3.3 Search Paradigms. In this section, some common search paradigms for FE are introduced.

Basic EC search paradigm: EC has been successfully utilized in various FE tasks [20, 222]. For example, Zhao et al. [234] introduced bagging concept to an evolutionary algorithm for FE. The work in Reference [233] developed an **evolutionary discriminant feature extraction (EDFE)** algorithm by combining GA with subspace analysis, which can reduce the complexity of the search space and improve the classification performance.

Co-evolution search paradigm: In FE, finding the optimal extractor is an optimization problem, which can be decomposed into a series of subproblems [70, 98]. For example, Hajati et al. [70] proposed a co-evolutionary method for FE. Specifically, a subpopulation was evolved to optimize the classifier-related subproblem (i.e., classifier construction), and the other subpopulation made use of genetic information from the first population for the optimization of a feature-related subproblem (i.e., FE).

Multi-objective search paradigm: In multi-objective FE, the model accuracy, computational time, complexity, and robustness are often taken into account as the objectives [24, 231]. Cano et al. [24] proposed a Pareto-based multi-objective GP algorithm for FE and data visualization,

where the objectives were to minimize the complexity of data transformation (i.e., tree size) and maximize the recognition performance (i.e., accuracy).

3.3.4 Summary. Table S4 of the online supplementary material presents existing works on FE according to the solution representations and search paradigms. In existing studies, many efficient searching and balancing strategies—driven by EC approaches to achieve satisfactory solutions at significantly reduced computation overheads—have been developed in recent years [24, 128, 168, 231]. However, the performance of extractors may be limited with existing encoding methods and predefined operation sets. Therefore, it is essential to develop efficient algorithms, operation control strategies, and representation for high-dimensional FE.

3.4 Data Augmentation

3.4.1 Problem Formulation. DA aims to enhance the generalizability and robustness of a deep model via increasing the volume, quality, and diversity of training data (i.e., additional training data D_a). Assume that an original training data D_o consists of instances I_i with corresponding labels L_i. The task of DA can be expressed as follows: it is to generate additional instance $I_i^{\sim}$ using the original instance I_i without altering the corresponding label L_i by a transformation function T_A (.). Meanwhile, the deep models can obtain smaller loss $\mathcal{L}(D_o, D_a)$ by using D_a and D_o. Therefore, the DA task can be depicted using the following formula:

$$\begin{aligned} &min \mathcal{L}\left(D_o, D_a\right) \\ s.t.\ D_a = \left(I_i^{\sim}, L_i\right) &= T_A\left(I_i, L_i\right), \quad I_i^{\sim} \in R^n, \quad I_i \in R^n \end{aligned}, \tag{5}$$

where n is the dimensionality of training data. Note that the output of the transformation function in DA is different from FC and FE. The generated instances have the same dimensionality as the original instance (i.e., $I_i^{\sim} \in R^n$ and $I_i \in R^n$) [137]. In the following, we will introduce the EC-based DA methods in terms of both solution representations and search paradigms.

3.4.2 Solution Representations. The direct encoding and indirect encoding are the two popular ways in the EC-based DA approaches.

Direct encoding: This method directly encodes the variables that need to be optimized [39, 130]. For example, Reference [39] directly encoded the basic elements (e.g., eyebrows, eyes, and nose) of the face, and a GA was used to select the best combination of the basic elements, which can improve the performance of face detector.

Indirect encoding: It is based on pre-designed rules or mappings to transform complex variables into a simple space. For example, Reference [64] proposed a tree-based rule to encode the transformation function, where each node indicated an image processing filter.

3.4.3 Search Paradigms. EC-based methods for DA can be divided into two categories according to whether the method combines with other generators, including pure EC search paradigms and generator-based EC search paradigms.

Pure EC: This paradigm utilizes reproduction operators (e.g., crossover and mutation) to generate new data, where the fitness evaluation is performed using an additional model. In other words, each individual in the population represents new training data. For example, Silvan et al. [130] employed a well-trained discriminator to evaluate the fitness value of the individuals (i.e., additional training data).

Generator-based EC: In this paradigm, EC is mainly used to optimize the generator (i.e., transformation function). It has emerged as a mainstream approach that EC optimizes **generative adversarial networks (GANs)** to obtain high-quality data [61, 129]. For example, Mehta et al. [129] proposed an improved EA to find the best architecture of GANs for DA. In addition, very little

existing work on finding optimal combinations of transformation functions has also been proposed [64, 151]. For example, Reference [151] designed an EC-based DA method to select the optimal combination among a collection of transformation functions (e.g., addElementwise, dropout and cutout).

3.4.4 Summary. Table S5 of the online supplementary material lists the main works about EC-based DA. The pure EC search paradigm cannot handle complex data formats due to limited encoding strategies, and they often use a two-stage process that increases the algorithmic complexity. Although the generator-based search paradigm can handle complex data formats, they highly depend on the distribution and volume of the original data [130, 151]. Thus, it is desirable to design DA methods for small and complex data.

4 MODEL GENERATION

Model generation is to search for optimal models with desirable learning capability on given tasks [74, 213]. In this section, we introduce corresponding PO, AO, and JO from solution representation to search paradigms. Readers interested in other model generation approaches (e.g., RL-based and gradient-based approaches) can refer to References [86, 159].

4.1 Parameter Optimization

4.1.1 Problem Formulation. PO targets at searching for the best parameter set (i.e., weights W^*) for a predefined architecture A. The loss function $\mathcal{L}(.)$ (e.g., the cross-entropy loss function) measures the performance of the model with optimized parameters (i.e., W in Equation (6)) on given datasets. The general PO can be formulated as:

$$W^* = \arg\min_{W} \mathcal{L}\left(W, A\right), \tag{6}$$

where W is usually large-scale (millions of model parameters) and highly non-convex.

4.1.2 Solution Representations. There are two typical EC-based representation schemes for PO, including direct encoding and indirect encoding [74].

Direct encoding: The model parameters are directly represented via a vector or matrix, in which each element represents a specific parameter [83, 93]. For example, a chromosome with 64 real numbers is used to directly represent the network corresponding weights, where the first 63 real numbers are used to encode three convolution masks of size 1×21. The last real number was the random seed of a generator for the initialization of a fully connected network [83]. This encoding approach may require a huge computational overhead to represent and optimize the large-scale weights.

Indirect encoding: This encoding approach represents only a subset of the model parameters via a deterministic transformation [99, 109]. In Reference [99], the weight information was encoded as a set of Fourier coefficients in the frequency domain to reduce dimensionality of representation by ignoring high-frequency coefficients. Although this method is able to speed up the search process, the loss of parameter information may occur due to the incomplete information representation, which may degrade the model performance.

4.1.3 Search Paradigms. EC-based methods for PO can be divided into two categories according to whether or not method combines with the gradient approach, i.e., pure EC and gradient-based EC.

Pure EC paradigms optimize model parameters only via evolutionary search, including the basic EC search paradigm and co-evolution search paradigm.

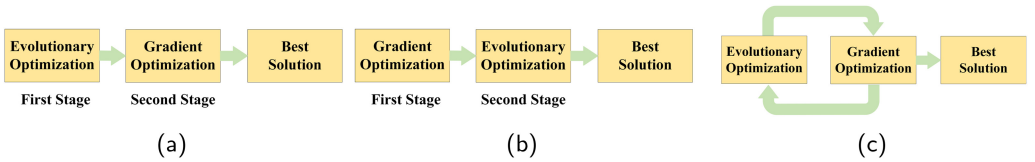

Fig. 5. Three hybrid ways of gradient-based ECs.

- **Basic EC search paradigm:** In addition to GA [93, 135], some heuristic algorithms such as PSO [4], ABC [92], and ACO [171] are also commonly utilized for PO. For example, Karaboga et al. [92] adopted ABC to find a set of weights for a **feed-forward neural network (FNN)** on targeted tasks.
- **Co-evolution search paradigm:** Co-evolution search is conducted on the subproblems of the original optimization problem (e.g., synapse-based and neuron-based problems [27, 28]). For example, Chandra et al. [28] regarded a single hidden layer as a subcomponent in the initialization phase, which will be merged with the individuals with the best fitness from different sub-populations to constitute new neural networks during the co-evolution optimization process.

Gradient-based EC combines basic EC with the gradient-based method to enhance the exploitation ability in optimizing model parameters. According to the execution order, there are three hybrid ways.

- The first hybridization approach is shown in Figure (5(a)), where the EC is used to identify the optimal parameters for model, then the parameters are further optimized using gradient-based method to find the final optimal solution [29, 229]. For example, a **genetic adaptive momentum estimation algorithm (GADAM)** was proposed in Reference [226] by incorporating Adam and GA into a unified learning scheme, where Adam was an adaptive moment estimation method with first-order gradient.
- The second hybridization approach is given in Figure (5(b)), where the gradient-based method is used to produce a set of parameters for the initialization of the population used in EC [203]. For example, Reference [94] first trained an RL agent through a gradient-based method, then the parameters of the RL were used as the initial population to feed the EC. As a result, the parameters will be further optimized by the EC.
- The third approach is presented in Figure (5(c)), which iteratively applies EC and gradient-based method during the optimization to find the optimal parameters. Following this framework, when the method is used and which method is chosen vary in different studies [40, 210].

4.1.4 Summary. Table S6 of the online supplementary material summarizes some EC-based PO methods according to solution representations and search paradigms. The direct encoding is straightforward and able to keep more information than the indirect encoding. Compared to gradient-based methods easily trapped into local optima, EC shows more powerful ability in global search. Here, several scenarios are introduced as follows, where EC is applied to PO:

Small-scale scenario: Reference [135] shows that pure EC approaches outperform gradient-based methods in search effectiveness on some small-scale problems, where the models are with small numbers of parameters or simple architectures (e.g., FNN).

Large-scale scenario: The performance of pure EC approaches might not be promising in large-scale learning models, while a better way is to utilize the hybridization of EC and gradient-

based methods. Such hybrid methods can alleviate the issue of getting trapped in local optima and increase the effectiveness of subsequent exploitation [210].

In addition to above scenarios, EC-based method can be used to train the DNN when the exact gradient information of the loss function is difficult to be acquired [150]. For example, the rewards of policy network are sparse or deceptive in **deep reinforcement learning (DRL)** so the gradient information is unattainable. Reference [38] introduced the **novelty search (NS)** and the **quality diversity (QD)** to the **evolution strategies (ES)** for the policy network.

4.2 Architecture Optimization

4.2.1 Problem Formulation. AO, also termed as **neural architecture search (NAS)**, is to search promising network architectures with good performance such as model accuracy on given tasks. AO can be formulated as follows:

$$\begin{cases} A^* = \arg\min_{A \in \mathcal{A}} \mathcal{L}_{eval}\left(W^*, A\right) \\ s.t. \quad W^* = \arg\min_{W} \mathcal{L}_{train}\left(W, A\right), \end{cases} \tag{7}$$

where weights W^* are obtained by training architecture A on the training data while minimizing the loss function $\mathcal{L}_{train}$, and A^* indicates the architecture with the minimum loss function $\mathcal{L}_{eval}$ on fitness evaluation set (e.g., training set or validation set), searched from the search space $\mathcal{A}$. Thereby, this optimization is a bi-level optimization problem [118, 235], where AO is subject to the PO [123]. Since the current AO works are mainly focused on CNN, we will discuss the solution representations, the search paradigms, and acceleration strategies of CNN. Due to the page limit, the design of the search space of AO are not introduced here, but interested readers can check References [118, 159], which have details about search space design.

4.2.2 Solution Representations. According to varying lengths of encodings, we can classify the encoding strategies into fixed-length and variable-length encodings.

Fixed-length encoding: The length of each individual is fixed during the evolution. For example, a fixed-length vector is designed to represent the model architecture of CNN [206], where a subset of elements in the vector represents an architectural unit (e.g., convolutional, pooling or fully connected layer) of a CNN. Such encoding may be easily adapted to evolutionary operations (e.g., crossover and mutation) of EC [206], but it has to specify an appropriate maximal length, which is usually unknown in advance and needs to predefine based on domain expertise.

Variable-length encoding: Different from the fixed-length approach, the variable-length encoding strategy does not require a prior knowledge about the optimal depth of model architecture and actually could be a way to reduce the complexity of the search space. The flexible design of this encoding may encode more detailed information about the architecture into a solution vector, and the optimal length of the solution is automatically found during the search process [118]. In Reference [30], the entire **variational autoencoder (VAE)** was divided into four blocks, including h-block, μ-block, σ-block, and t-block, while the variable-length chromosomes consisted of different quantities and types of layers. Notably, variable-length encoding is not straightforward to apply standard genetic operators (e.g., crossover).

Since the neural network architectures are composed of basic units and connections between them, both of them are to be encoded, as suggested in Reference [118].

(1) Encoding hyperparameters of basic units. In CNNs, there are many hyperparameters to be specified for each unit (e.g., layer, block, or cell), such as feature map size, type of convolution layer, and filter size [180]. In Reference [177], DenseBlock only had to set two hyperparameters (e.g., block type and specific parameter of internal unit) to configure the block that can be seen as

a microcosm of a complete CNN model. The parameterization of a cell is more flexible than that of a block, since it can be configured via a combination of different primitive layers [178].

(2) Encoding connections between units. In general, there are two kinds of model architectures according to the connection patterns of basic units: linear topological architectures and non-linear topological architectures [212]. The linear pattern of architecture consists of sequential basic units, and the non-linear pattern allows for skip or loop connections in the architecture [118].

- **Linear topological architecture:** The linear topology widely appears in the construction of layer-wise and block-wise search spaces. Due to the simplicity of linear topology, basic units can be stacked one-by-one by a linear piecing method. In this way, the skeleton of an architecture can be built up effectively [30, 177] regardless of the complexity of the internal of basic units.
- **Non-linear topological architecture:** Compared to the linear architecture, the non-linear topological architecture receives much more attention due to its flexibility to construct well-performing architectures [193, 195, 206], such as macro structures composed of basic units and micro structures within basic units. There are two typical encoding approaches for non-linear topological architectures: one is to use adjacent matrix to represent the connections in non-linear architectures, where "1" of the matrix denotes the existence of the connection between two units, and "0" goes the opposite. In Reference [121], skip connections are represented by a matrix where constraints can be set in place to guarantee valid encoding and avoid recurrent edges while performing skip connections. Note that adjacent matrix has a limitation that the number of basic units needs to be fixed in advance [96]. Another one is to utilize an ordered pair to represent a directed acyclic graph and then encode the connections between unites. The ordered pair can be formulated as $G = (V, E)$, where V is a set of vertices and E is a directed edge in the acyclic graph, and it has been applied in Reference [84] to encode the connections.

4.2.3 Search Paradigms. In this section, the commonly used EC-based search paradigms for AO are introduced.

Basic EC search paradigm: Many basic EC algorithms have been widely applied in existing AO methods, such as GA [96] and PSO [179]. A general framework of EC is presented in Figure 3.

Incremental search paradigm: A model architecture can be built in an incremental way where model elements (e.g., layers and connections) are gradually added to the model during the evolutionary process [112, 194]. This way allows to find parts of architecture at different optimization stages, which reduces the computational burden on acquiring a complete model at once [112]. For example, Wang et al. [194] used an incremental approach to stack blocks for building architectures, which improved the capacity of the final architecture via a progressive process.

Co-evolution search paradigm: AO is decomposed into the optimizations of a blueprint and its components [110, 146]. Specifically, the blueprint plays a role in specifying the topological connection patterns of its components, and an optimal architecture is acquired by cooperatively optimizing the blueprint and its components. For example, O'Neill et al. [146] proposed a co-evolution search paradigm for AO, where the candidate blueprints and components were sampled from two populations and then combined to form new architectures.

Multi-objective search paradigm: This paradigm targets at searching for a set of Pareto optimal architectures based on multiple criteria and finding the final solutions according to some practical considerations, such as computational environment [124, 139]. This paradigm becomes popular in practical applications, since many objectives are required to be considered, such as the accuracy, inference time, model size, and energy consumption. In Reference [139], NSGA-II and RL

were used to explore model architectures with respect to the model accuracy and model complexity (e.g., the number of model parameters and multiply-adds operators).

4.2.4 Summary. Table S7 of the online supplementary material shows the AO approaches, which are classified according to the search paradigms and solution representations. Obviously, the basic EC search paradigms and fixed-length encodings dominate AO. Most AO methods are based on basic EC search paradigms on an global search space [118]. There are also some other automatic search techniques such as RL-based [86], Bayesian-based [201], and gradient-based [114] methods, for architecture search.

RL-based methods can be regarded as an incremental search, where a policy function is learned by using a reward-prediction error to drive the generation of incremental architecture. Due to huge state space and action space, the environment modeling and exploration-learning procedures become extremely difficult, leading to inefficient sampling for search space [221], which brings tremendous pressures for the exploration of agents. In particular, existing RL-based methods for AO generally enable agents to learn from scratch when dealing with new search space, which is difficult to leverage pre-acquired prior knowledge to assist explorations [242]. In addition, there are a large number of hyper-parameters (e.g., discount factor) in RL-based methods. Tuning these hyperparameters also requires a huge overhead [118, 180]. Besides, they transform a multi-objective optimization problem into a single-objective problem via *a priori* or expert knowledge, so they are unable to find a Pareto optimal set to the target tasks.

Bayesian-based methods are a common tool for hyperparametric optimization problems with low dimensions. In comparison to EC-based methods, they are much more efficient on the condition that a proper distance function has to be designed to evaluate the similarities between two subnets. However, the computational cost of Gaussian process grows exponentially and its accuracy decreases when the dimensionality of the problem increases.

Gradient-based methods, taking AO problem as a continuous differentiable problem instead of a discrete one, are able to efficiently search architectures with proper weight parameters. Unfortunately, their GPU costs are usually very high due to a large number of parameters to be updated in gradient-based algorithms [114].

In contrast, EC-based methods benefit from less hyperparameters to be optimized and no distance functions to be designed. The related experiments demonstrate that RL-based methods require more computational resources than EA-based methods under the same settings. In addition, EC-based methods can be applied to AO with multiple objectives and constraints. Although there are many acceleration strategies in EC-based methods, they still suffer from high computational overheads.

4.3 Joint Optimization

4.3.1 Problem Formulation. The independent optimization of architecture or parameters is difficult to achieve the optimal model on give tasks. Hence, joint optimization methods have been developed to search for the optimal configuration of architecture (A^*), and parameters (W^*, associated weights). The optimization problem can be defined in Equation (8):

$$(W^*, A^*) = \arg\min_{W,A} L(W, A), \tag{8}$$

where L is the loss function. In the following, we will introduce the joint optimization regarding the solution representations and search paradigms and then discuss the pros and cons of EC-based methods in comparison to others.

4.3.2 Solution Representations. There are three typical classes of encoding schemes used for joint optimization.

Linear encoding: This is a simple but effective encoding strategy, which has been widely used in many studies to build architecture with high performance [8, 62]. In Reference [62], a variable-length binary vector was used to represent weights and structure of neural networks, where the weights utilize direct encoding.

Tree-based encoding: In this encoding, the topology and weights of an architecture can be represented by a tree structure with a number of nodes and edges [67, 224]. In Reference [225], the mechanism of **Reverse Encoding Tree (RET)** was developed to ensure the robustness of a deep model, where the topological information of an architecture was represented by a combination of nodes and the weight information was recorded on the edges.

Graph-based encoding: In this encoding, the nodes of a graph represent neurons or other network units, and the edges are used to record the weight information [26, 72]. For example, a graph incidence matrix was developed in Reference [147] to encode a neural network. The size of the matrix was set to $(N_i + N_h + N_o) \times (N_i + N_h + N_o)$, where N_i, N_h, and N_o indicate the numbers of input, hidden, and output nodes, respectively. In the graph incidence matrix, real numbers represented the weight and biases, and "0" meant that there was no connection between two nodes.

4.3.3 Search Paradigms. There are a number of effective search paradigms for joint optimization, and the EC-based search paradigms are in the spotlight.

Basic EC search paradigm: Some basic EC search methods have been employed to handle joint optimization problems [16, 17, 54, 147, 173]. In Reference [147], an architecture and its corresponding weights were simultaneously optimized by an EC-based method using linear and graph encodings. In **neuro-evolution of augmenting topologies (NEAT)** [173], the architecture of a small network is evolved by an incremental mechanism, while the weights are optimized by an EC-based method. NEAT is able to ensure the lowest dimensional search space over all generations. Some representative studies on NEAT are presented in References [16, 17].

Multi-objective search paradigm: Multi-objective optimization on model design has been developed in many studies (e.g., artificial neural network [161] and recurrent neural network [170]). For example, Smith et al. [170] built a bi-objective optimization (i.e., the minimizations of the **mean squared error (MSE)** on a training dataset and the number of connections in the network) to search for optimal weights and connections of network architectures. The chromosome of an individual was composed of two parts, where the one with Boolean type represented the structure of a network, and the other with real values represented the weights.

4.3.4 Summary. Existing works about JO are summarized in Table S8 of the online supplementary material. Linearity encoding is used to be prevalent in the joint optimization of small-scale neural networks [147, 215]. However, with the increase of the scale of neural networks, encoding of high-dimensional vector or matrix of weights is not realistic. Therefore, recent studies are more on efficient encoding (e.g., indirect encoding). For example, a complex mapping with acceptable accuracy loss is designed in References [44, 100] to construct weight vectors with arbitrary size. EC-based approaches with capability of searching the optimal solution have been developed to configure a DL model for the specific task. However, they often encounter a prohibitive computational cost, which is even higher than that of AO. Hence, designing efficient EC-based approaches for architecture and parameter search deserves much investigation.

5 MODEL DEPLOYMENT

The large-scale DNNs are not straightforward to be deployed into devices (e.g., smartphones) with limited computation and storage resources (e.g., battery capacity and memory size). To solve this

issue, various model compression approaches have been proposed to reduce the model size and inference time, such as pruning, model distillation, and quantization [35, 169]. However, they need much expert knowledge and a lot of efforts on the manual compression of neural network models. In contrast, EC-based approaches are automation approaches and have been recently introduced to achieve automated model compression. We have observed that most of them concentrate on the area of MP.

5.1 Model Pruning

5.1.1 Problem Formulation. DNN is commonly an over-parameterized model, which has redundant and non-informative components (e.g., weights, channels, and filters). To address this issue, researchers have designed various pruning approaches (e.g., channel pruning [75]) to obtain a configuration (i.e., A^* and W^*) of lightweight deep model with high accuracy. MP can be formulated as:

$$\begin{cases} (A^*, W^*) = \arg\min \phi = \mathcal{L}_u\left(A_u, W_u\right) - \mathcal{L}_p(A_p, W_p) \\ \qquad\qquad s.t. \quad (A_p, W_p) = \mathcal{P}(A_u, W_u), \end{cases} \tag{9}$$

where $\mathcal{L}_u$ and $\mathcal{L}_p$ denote loss functions of unpruned and pruned network, respectively. ϕ indicates the gap between the loss function of pruned and unpruned network. $\mathcal{P}$ is the pruning function that is used to generate different configurations of the pruned network (i.e., A_p and W_p) based on the configurations of unpruned network (i.e., A_u and W_u), where $A_\#$ and $W_\#$ indicate architecture and weight configuration. The common pruning functions are mainly weight pruning, channel pruning, shape pruning, filter pruning, and layer pruning [111]. This study aims to introduce EC-based methods for MP, and readers interested in traditional pruning methods may refer to the surveys cited in Reference [35] to get more details. In Table S9 of the online supplementary material, we have summarized these two categories of search paradigms as well as their corresponding ways of encoding.

5.1.2 Solution Representations. For MP, binary encoding is one of the most popular approaches among these solution representations, where each element corresponds to the network component (e.g., channel). In Reference [199], the network pruning task was formulated as a binary programming problem, where a binary variable was directly associated with each convolution filter to determine whether or not the filter took effect. Although binary representation is straightforward, the length of the representation becomes large when the model complexity (i.e., the number of units) improves, and the overhead of exploration will also increase.

To address the above issue, some efficient solution representations (i.e., indirect encoding) have been developed. For example, Liu et al. [116] used N digits to record the number of compressed layers. The first digit represented the number of compressed layers, and following digits recorded the selected compression operator index of each layer. This way can significantly improve the search efficiency. In Reference [119], encoding vectors are used to represent the number of channels in each layer for original networks. Then, a meta-network is constructed to generate the weights according to network encoding vectors. By stochastically inputting different structure encodings, the meta-network gradually learns to generate weights for various pruned structures.

5.1.3 Search Paradigms. The search paradigms in MP studies can be categorized into two main groups.

Basic EC search paradigm: A number of studies introduce single-objective EC search paradigm for MP [182, 204]. For example, Wu et al. [204] first analyzed the pruning sensitivity on weights via **differential evolution (DE)**, and then the model was compressed by iteratively performing the weight pruning process according to the weight sensitivity. In addition, this method

adopted a recovery strategy to increase the pruned model performance during the fine-tuning phase.

Multi-objective search paradigm: Recently, this sort of search paradigm has been adopted to MP, which is able to provide users with a set of Pareto lightweight models [236, 239]. For example, Zhou et al. [237] considered two objectives (i.e., minimizing convolutional filters and maximizing the model performance) for biomedical image segmentation. During the MP, a classical multi-objective optimization algorithm (NSGA-II [46]) was used find the optimal set of non-dominated solutions, where the optimization was based on a binary string encoding (each bit represents a filter).

5.2 Other EC-based Model Deployment Methods

There are several other EC-based model compression methods for model deployment. In the following, some typical methods are introduced, including knowledge distillation, low-rank factorization, and EC for hybrid techniques.

5.2.1 Knowledge Distillation. **Knowledge distillation (KD)** [68] aims to get a small light network but with good generalization capability. The basic idea is to transfer the knowledge learned from a big cumbersome network (or teacher network) with good generalization ability to a small but light network (or student network).

However, knowledge distillation may be seriously influenced when there is a big gap in the learning capability between the teacher and student networks. In other words, if the difference is large, then the student network may not be able to learn knowledge from the teacher network. Recently, several EC-based approaches have been proposed to mitigate the above issue of knowledge distillation. For example, Reference [174] constructed KD as a multi-objective optimization problem, aiming to enable student networks to achieve tradeoffs between two conflicting objectives minimizing inference latency (i.e., latency) and minimizing accuracy loss (i.e., error). Four hyperparameters of student models were evolved in this approach. Jiao et al. [88] proposed a GA-based approach to find the optimal layer mapping for task-agnostic **bidirectional encoder representation from transformers (BERT)** distillation. This method encoded the layer mapping as a binary string, where each element represented whether the student network required to learn from the teacher network. Their experimental results showed that the proposed approach was able to narrow the performance between the teacher and student networks on given tasks, where student network achieved state-of-the-art accuracy on GLUE tasks.

5.2.2 Low-rank Factorization. DNNs often involve a huge number of weights, which may impact the inference speed and seriously increase the storage overhead of the DNN. The weights can be viewed as a matrix W with $m \times n$ dimensions. The low-rank approach is commonly applied to the weight matrix (W) after the DNN is fully trained. For example, singular value decomposition [58] is a typical low-rank factorization method, where W is decomposed as follows:

$$W{=}USV^T, \tag{10}$$

where $U \in R^{m\times m}$, $V^T \in R^{n\times n}$ are orthogonal matrices and $S \in R^{m\times n}$ is a diagonal matrix.

Notably, most of the existing low-rank factorization methods rely on domain expertise and experience for the selection of hyperparameters (e.g., the rank and sparsity of weight matrix) to get appropriate compression results without serious performance degradation [79, 181, 202]. Accordingly, EC-based methods have been introduced to solve the above challenge [81, 199]. For example, Huang et al. [81] presented a multi-objective evolution approach to automatically optimize rank and sparsity for weight matrix without human intervention, where two objectives were taken into account including the minimization of the model classification error rate and maximization of the

model compression rate. They therefore generated a set of approximately compressed models with different compression rates to mitigate the expensive training process.

5.2.3 Joint Optimization. Many compression techniques (e.g., quantization) can be easily applied on top of other techniques (e.g., pruning and low-rank factorization). For example, pruning first and then quantification can obtain a lightweight model with faster inference. Similarly, EC can optimize more than one model compression method at the same time. In the following, we will briefly review such works.

Phan et al. [152] designed efficient 1-Bit CNNs, which combined quantization with a compact model. Specifically, they first created a number of strong baseline binary networks, which had abundant random group combinations at each convolutional layer. Then, they adopted evolutionary search to seek an optimal group convolution combination with accuracy above threshold. Finally, the obtained binary models were trained from scratch to achieve the final lightweight network. Different from Reference [152], Polino et al. [153] jointly utilized weight quantization and distillation to compress large networks (i.e., teacher network) into small networks (i.e., student network), where the latency and model error were regarded as the objectives during the optimization.

Recently, Zhou et al. [238] developed an evolutionary algorithm-based method for shallowing DNNs at block levels (ESNB). In ESNB, a prior knowledge was extracted from the original model to guide the population initialization. Then, an evolutionary multi-objective optimization method was performed to minimize the number of blocks and the accuracy drop (i.e., loss). After that, knowledge distillation was employed to compensate for the performance degradation via matching output of the pruned model with the softened and hardened output of the original model.

5.2.4 Summary. Similar to architecture optimization in terms of search techniques, RL-based methods, Bayesian-based methods, and gradient-based methods have also been applied to model deployment. Specifically, RL-based methods explore the search space using a predefined controller model. However, controller construction is a complex task, which results in slow convergence of model compression [9]. Bayesian-based methods use Gaussian Process to construct the relationship between accuracy and the compressed components. Unfortunately, Bayesian-based methods are highly dependent on well-designed distance functions between different solutions and the precision of the Gaussian Process [196]. The gradient-based methods aim to explore the search space of discretization by gradient information. However, such methods often require a large amount of memory space to store an over-parameterized model, and using gradient information to determine the results of model compression may not be the best solution in many cases [211].

In contrast, EC-based methods use flexible and variable populations to drive model compression. However, the additional model (e.g., controller and Gaussian Process) and gradient information are not required during the search process [75, 204]. However, there is still much room for improvement in addressing the huge computational overhead of evolutionary model deployment due to the large search space and expensive evaluation costs. Acceleration strategies may be able to alleviate the issue, and related works have been discussed in Section 6.

6 ACCELERATION STRATEGIES

EDL is a high computational overhead task, mainly due to the large search space and highly time-consuming evaluation [213]. To overcome this challenge, various acceleration strategies [14, 179] have been developed to accelerate the optimization. In this section, we summarize the speed-up strategies from the aspects of algorithm design to the hardware implementation, as shown in Figure 6.

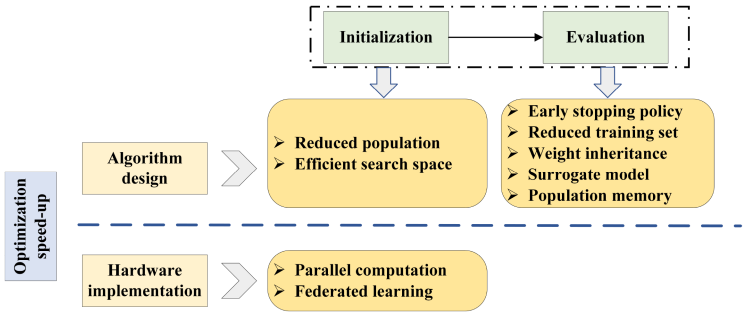

Fig. 6. Overview of acceleration strategies.

From the algorithm design point of view, we summarized a number of acceleration strategies from population initialization to evaluations.

- **Initialization:**

(1) *Reduced population*: The simplest way of acceleration during the initialization stage is to set the population with a small size. In other words, less evaluations are required with a smaller size of population, since the evaluation of a candidate architecture is time-consuming [213]. As a result, some studies [13, 14] use small populations with a fixed size to speed up their evolutionary process, such as CARS (size = 32) [212], ESNB (size = 30) [238], and ECDNN (size = 60) [237]. In contrast, some other studies use dynamic sizes of populations during the optimization. In Reference [56], the population size is dynamically changed to reach a balance between algorithmic efficiency and population diversity.

(2) *Efficient search space*: Another way is to design efficient search space to speed up the search process. For example, an architecture constructed on the basis of cell-wise search space [118] is composed of many similar structures of cells and only representative cells need to be optimized, which contributes to significant computational speed-up.

- **Evaluations:**

(1) **Early stopping policy:** A relatively small number of training epochs are used to reduce the training cost (i.e., early stopping policy), since the training time is reduced [2, 179, 186].

(2) **Reduced training set:** Some methods are designed to reduce the size of the training set to improve training efficiency at the expense of a little accuracy [115, 165]. Besides, low-resolution data (e.g., ImageNet 32) [36] is also commonly used as the training set to accelerate the search process for the optimal architecture.

(3) **Weight inheritance**[1]**:**

- *Supernet-based inheritance*: It uses an over-parameterized and pre-trained supernet to encode all candidate architectures (i.e., subnets). In other words, the subnets share weights of the identical structures from the supernet, and they are directly evaluated on the validation dataset to obtain their model accuracy [42, 50, 154, 165].
- *Parent-based inheritance*: It inherits weights from previously trained networks (i.e., parental networks) instead of a supernet, since offspring individuals retain some identical parts of their parental architectures [49, 103, 157, 157, 240]. As a result, offspring

[1]In References [105, 162, 163], Rumelhart/Hinton/LeCun used the term "weight sharing" to mean that different network connections/links share the same set of weights, and pointed out that "weight sharing" is the core of shared weight NNs/CNNs. More recent [123, 205] use of this term refers to "weight/parameter replications" or "weight inheritance."

architectures can inherit the weights of the identical parts and no longer need to be trained from scratch.
  - In MP, the weights of candidate networks can be inherited from the pre-trained model (i.e., pruned model). Different from the supernet-based inheritance methods, this pre-trained model is not over-parameterized. For example, He et al. [75] used classical baselines (e.g., ResNet and DenseNet) as the model to be pruned. During the search process, all candidate solutions inherit the weights of the baseline models and fine-tune them to obtain accuracy.

(4) **Surrogate model**: Since the evaluation of an architecture is time-consuming [118, 213], cheap surrogate models have been introduced in AO as performance predictors to reduce the computational time [123].
  - *Online performance predictors*: They are trained online on the datasets sampled from past several epochs [118], including a sequence of data pairs with different training epochs and their corresponding performance of these epochs [123]. After that, they will be used for the performance prediction on new architectures. To reduce the true evaluations of architectures, some performance predictors directly predict whether a candidate architecture can be survived into next iteration through a trained ranking or classification method, such as classification-wise AO [126]. Reference [142] designed SVM-based surrogate model for feature selection, where training data for surrogate model was built by difference method.
  - *Offline performance predictors*: They are regression models mapping the architectures to specific performance. The predictors can be trained in an offline manner, so they are able to predict the performance of architectures during the entire search process. Consequently, they can significantly reduce the computational burden [112, 175, 176]. For FS, Liu et al. [117] designed surrogate models for each low-dimensional subspace divided from the high-dimensional space to accelerate the search process in a parallel way.

(5) **Population memory:** Population memory is used to store elite individuals from different generations during the optimization [10, 63, 133, 180]. When a new individual is generated, it does not need to be evaluated again if it is the same as an individual in the memory. In other words, the performance of individuals sharing the same architectures is the same and can be acquired via the population memory instead of training from scratch. This mechanism relies on the fact that similar or same individuals may repeatedly appear in different generations.

According to the above introduction, we can conclude that many of them improve the search efficiency at the expense of sub-optimality. For example, a small population cannot well cover a multi-objective optimal front. Parameter sharing may lead to the biased search due to much similarities among the individuals. Highly accurate surrogates need a large number of training data, which are commonly time-consuming. Population memory heavily relies on the random emergence of similar or same individuals.

Hardware implementation: Importantly, a powerful hardware platform can significantly speed up the search process under the reasonable utilization of computing resources (e.g., cloud computing [34] and volunteer computers [15]). Parallel computation is a powerful tool to decompose large search problems into small sub-problems, which can be simultaneously optimized by several cheaper hardware [89, 90]. For example, Lorenzo et al. [122] proposed a parallel PSO algorithm to search for optimal architecture of CNN. The security of the computing device also becomes an important consideration. For this reason, an emerging decentralized privacy-preserving framework is applied to AO, which unites multiple local clients to collaboratively learn a shared

Table 2. Different Acceleration Strategies

Algorithm design	Initialization	Reduced population		Model generation: [14, 56, 115, 212] Model deployment: [237, 238]
	Evaluation	Early stopping policy		Model generation: [2, 12, 14, 59, 134, 148, 179]
		Reduced training set		Model generation: [115, 165, 194]
		Weight inheritance	Supernet-based sharing	Model generation: [42, 50, 154, 165]
			Parent-based sharing	Model generation: [2, 31, 49, 59, 103, 166, 240]
			Other	Model deployment: [75, 238]
		Surrogate model	Online performance predictors	Data preparation: [142, 143] Model generation: [112, 120, 123, 126]
			Offline performance predictors	Data preparation: [23, 55, 117] Model generation: [113, 156, 176]
		Population memory		Data preparation: [10, 133] Model generation: [37, 108, 131, 180]
Hardware implementation				Model generation: [34, 89, 90, 241]

global model trained on the parameters or gradients of the local models, instead of the raw data. For example, Zhu et al. [241] first proposed a real-time federated NAS that can not only optimize the model architecture but also reduce the computational payload. Specifically, the decentralized system is able to accelerate the algorithm efficiency of federated NAS. Besides, data encryption is employed on the transmitted data (parameters or gradients of the local models) between the clients and the server to ensure the privacy even though all of the training is performed in local. Accordingly, federated NAS is highly efficient and secure, which may become a new hot research topic.

Table 2 lists the common acceleration strategies to improve the algorithm efficiency. It is noted that multiple strategies can be utilized together to improve the computational speedup. For example, Lu et al. [123] employed supernet and learning curve performance predictor in NAS, while Liu et al. [115] leveraged a small population size and a small dataset to reduce the time overhead of evaluations. Besides, we can observe that the acceleration strategies are mainly focused on model generation, and few acceleration methods are designed for data preparation and model deployment. Designing more acceleration strategies for specific optimization scenarios is an interesting direction.

7 APPLICATIONS, OPEN ISSUES, AND TRENDS

7.1 Applications

EDL algorithms have been widely used in various real-world applications. In practice, great developments have been achieved in **computer vision (CV), natural language processing**

(NLP) and other practical applications (e.g., crisis prediction and disease prediction). The corresponding descriptions are presented in Section 3.1 of the online supplementary material due to the page limit.

7.2 Open Issues

EDL is a hot research topic in both fields of DL and EC. There are a large number of publications on various top conferences and journals, such as ICCV, CVPR, GECCO, TPAMI, TEVC, TCYB, and TNNLS (see the reference list). Yet some challenges remain to be resolved (e.g., **acceleration strategies, effectiveness, large-scale datasets, safety, privacy, ethicality, and end-to-end EDL**). The presentations of these open issues are provided in Section 3.2 of online supplementary material because of the page limit.

7.3 Challenges and Future Trends

Although remarkable progress has been made in EDL, there are still many promising lines of research, such as **fair comparisons, interpretability, and exploring more scenarios.** The above mentioned contents are introduced in Section 3.3 of online supplementary material.

8 CONCLUSIONS

With the development of ML and EC, many EDL approaches have been proposed to automatically optimize the parameters or architectures of deep models following the EC optimization framework. EDL approaches show competitive performance in robust and search capability, in comparison with the manually designed approaches. Therefore, EDL has become a hot research topic.

In this survey, we first introduced EDL from the perspective of DL and EC to facilitate the understanding of readers from the communities of ML and EC. Then, we formulated EDL as a complex optimization problem and provided a comprehensive survey of EC techniques in solving EDL optimization problems in terms of data preparation, model generation to model deployment to form a new taxonomy (i.e., what, where, and how to evolve/optimize in EDL). Specifiically, we discussed the solution representations and search paradigms of EDL at different stages of its pipeline in detail. Then, the pros and cons of EC-based approaches in comparison to non-EC based ones are discussed. Subsequently, various applications are summarized to show the potential ability of EDL in handling real-world problems.

Although EDL approaches have achieved great progress in AutoML, there are still a number of challenging issues to be resolved. For example, effective acceleration strategies are essential to reduce the expensive optimization process. Another issue is to handle large-scale datasets and how to perform fair comparisons between different EDL approaches or non-EC-based methods. More investigations are required to theoretically analyze or interpret the search ability of EDL. In addition, a lot of efforts are required on improving the performance of EDL on both benchmarks (e.g., large-scale and small-scale data) and real-world applications. Last, the development of end-to-end EDL is challenging but deserves much efforts.

REFERENCES

[1] Hervé Abdi and Lynne J. Williams. 2010. Principal component analysis. *Comput. Stat.* 2, 4 (2010), 433–459.

[2] Amr Ahmed, Saad Mohamed Darwish, and Mohamed M. El-Sherbiny. 2019. A novel automatic CNN architecture design approach based on genetic algorithm. In *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics*. 473–482.

[3] Soha Ahmed, Mengjie Zhang, Lifeng Peng, and Bing Xue. 2014. Multiple feature construction for effective biomarker identification and classification using genetic programming. In *Proceedings of the Genetic Evolutionary Computation Conference*. 249–256.

[4] Buthainah Al-kazemi and Chilukuri Krishna Mohan. 2002. Training feedforward neural networks using multi-phase particle swarm optimization. In *Proceedings of the International Conference on Neural Information Processing*. 2615–2619.
[5] Harith Al-Sahaf, Ying Bi, Qi Chen, Andrew Lensen, Yi Mei, Yanan Sun, Binh Tran, Bing Xue, and Mengjie Zhang. 2019. A survey on evolutionary machine learning. *J. R. Soc. N. Z.* 49, 2 (2019), 205–228.
[6] Wissam A. Albukhanajer, Johann A. Briffa, and Yaochu Jin. 2015. Evolutionary multiobjective image feature extraction in the presence of noise. *IEEE Trans. Cybern.* 45, 9 (2015), 1757–1768.
[7] Stamatios-Aggelos N. Alexandropoulos and Christos K. Aridas. 2019. Multi-objective evolutionary optimization algorithms for machine learning: A recent survey. *Approxim. Optim.* 145, 4 (2019), 35–55.
[8] Ibrahim Aljarah, Hossam Faris, and Seyed Mohammad Mirjalili. 2018. Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Comput.* 22, 1 (2018), 1–15.
[9] Manoj Alwani, Yang Wang, and Vashisht Madhavan. 2022. DECORE: Deep compression with reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 12349–12359.
[10] Malek Alzaqebah, Khaoula Briki, Nashat Alrefai, Sami Brini, Sana Jawarneh, Mutasem K. Alsmadi, Rami Mustafa A. Mohammad, Ibrahim ALmarashdeh, Fahad A. Alghamdi, Nahier Aldhafferi et al. 2021. Memory based Cuckoo search algorithm for feature selection of gene expression dataset. *Inform. Med. Unlocked* 24 (2021), 100572.
[11] Hayden Andersen, Sean Stevenson, Tuan Ha, Xiaoying Gao, and Bing Xue. 2021. Evolving neural networks for text classification using genetic algorithm-based approaches. In *Proceedings of the IEEE Congress on Evolutionary Computation*. 1241–1248.
[12] Filipe Assuno, Joao Correia, and Rúben Conceição. 2019. Automatic design of artificial neural networks for gamma-ray detection. *IEEE Access* 7 (2019), 110531–110540.
[13] Filipe Assuno, Nuno Lourenço, P. Machado, and Bernardete Ribeiro. 2018. Evolving the topology of large scale deep neural networks. In *Proceedings of the European Conference on Genetic Programming*. 19–34.
[14] Filipe Assuno, Nuno Lourenço, Penousal Machado, and Bernardete Ribeiro. 2019. Fast denser: Efficient deep neuroevolution. In *Proceedings of the European Conference on Genetic Programming*. 197–212.
[15] Medha Atre, Birendra Jha, and Ashwini Rao. 2021. Distributed deep learning using volunteer computing-like paradigm. In *Proceedings of the International Parallel and Distributed Processing Symposium*. 933–942.
[16] Shohag Barman and Yung-Keun Kwon. 2020. A neuro-evolution approach to infer a Boolean network from time-series gene expressions. *Bioinformatics* 36, 2 (2020), i762–i769.
[17] Amir Behjat and Sharat Chidambaran. 2019. Adaptive Genomic Evolution of Neural network Topologies (AGENT) for state-to-action mapping in autonomous agents. In *Proceedings of the International Conference on Robotics and Automation*. 9638–9644.
[18] Bir Bhanu and Krzysztof Krawiec. 2002. Coevolutionary construction of features for transformation of representation in machine learning. In *Proceedings of the Genetic Evolutionary Computation Conference*. 249–254.
[19] Ying Bi, Bing Xue, Pablo Mesejo, Stefano Cagnoni, and Mengjie Zhang. 2023. A survey on evolutionary computation for computer vision and image analysis: Past, present, and future trends. *IEEE Trans. Evol. Comput.* 27, 1 (2023), 5–25.
[20] Ying Bi, Bing Xue, and Mengjie Zhang. 2018. An automatic feature extraction approach to image classification using genetic programming. In *Proceedings of the International Conference on Applications of Evolutionary Computation*. 421–438.
[21] Ying Bi, Bing Xue, and Mengjie Zhang. 2021. *Genetic Programming for Image Classification: An Automated Approach to Feature Learning*, Vol. 24. Springer Nature.
[22] Ying Bi, Bing Xue, and Mengjie Zhang. 2022. Genetic programming-based evolutionary deep learning for data-efficient image classification. *IEEE Trans. Evol. Comput.* (2022). Early Access. DOI: https://doi.org/10.1109/TEVC.2022.3214503
[23] Ying Bi, Bing Xue, and Mengjie Zhang. 2023. Instance selection-based surrogate-assisted genetic programming for feature learning in image classification. *IEEE Trans. Cybern.* 53, 2 (2023), 1118–1132.
[24] Alberto Cano, Sebastián Ventura, and Krzysztof J. Cios. 2017. Multi-objective genetic programming for feature extraction and data visualization. *Soft Comput.* 21, 8 (2017), 2069–2089.
[25] Mauro Castelli, Luca Manzoni, and Leonardo Vanneschi. 2011. Multi objective genetic programming for feature construction in classification problems. In *Proceedings of the International Conference on Learning and Intelligent Optimization*. 503–506.
[26] Zheng-Yi Chai, ChuanHua Yang, and Ya-Lun Li. 2022. Communication efficiency optimization in federated learning based on multi-objective evolutionary algorithm. *Evol. Intell.* 16, 11 (2022), 1033–1044.
[27] Rohitash Chandra. 2015. Competition and collaboration in cooperative coevolution of Elman recurrent neural networks for time-series prediction. *IEEE Trans. Neural Netw. Learn. Syst.* 26, 12 (2015), 3123–3136.
[28] Rohitash Chandra and Mengjie Zhang. 2012. Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction. *Neurocomputing* 86 (2012), 116–123.

[29] Qi Chen, Bing Xue, and Mengjie Zhang. 2015. Generalisation and domain adaptation in GP with gradient descent for symbolic regression. In *Proceedings of the IEEE Congress on Evolutionary Computation*. 1137–1144.
[30] Xiangru Chen, Yanan Sun, Mengjie Zhang, and Dezhong Peng. 2020. Evolving deep convolutional variational autoencoders for image classification. *IEEE Trans. Evol. Comput.* 25, 5 (2020), 815–829.
[31] Yukang Chen, Gaofeng Meng, Qian Zhang, Shiming Xiang, and Chang Huang. 2019. RENAS: Reinforced Evolutionary Neural Architecture Search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4787–4796.
[32] Fan Cheng, Feixiang Chu, Yi Xu, and Lei Zhang. 2021. A steering-matrix-based multiobjective evolutionary algorithm for high-dimensional feature selection. *IEEE Trans. Cybern.* 52, 9 (2021), 9695–9708.
[33] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2017. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282* (2017).
[34] Zouhair Chiba, Noreddine Abghour, Khalid Moussaid, Amina El Omri, and Mohamed Rida. 2019. Intelligent approach to build a deep neural network based IDS for cloud environment using combination of machine learning algorithms. *Comput. Secur.* 86 (2019), 291–317.
[35] Tejalal Choudhary, Vipul Mishra, Anurag Goswami, and Jagannathan Sarangapani. 2020. A comprehensive survey on model compression and acceleration. *Artif. Intell. Rev.* 53, 7 (2020), 5113–5155.
[36] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. 2017. A downsampled variant of ImageNet as an alternative to the CIFAR datasets. *arXiv preprint arXiv:1707.08819* (2017).
[37] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Hailong Ma. 2020. Multi-objective reinforced evolution in mobile neural architecture search. In *Proceedings of the European Conference on Computer Vision*. 99–113.
[38] Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2018. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 5032–5043.
[39] Joao Correia, Tiago Martins, and Penousal Machado. 2019. Evolutionary data augmentation in deep face detection. In *Proceedings of the Genetic Evolutionary Computation Conference*. 163–164.
[40] Xiaodong Cui, Wei Zhang, Zoltán Tüske, and Michael Picheny. 2018. Evolutionary stochastic gradient descent for optimization of deep neural networks. *Proc. Adv. Neural Inf. Process. Syst.* 31 (2018), 6051–6061.
[41] Sérgio Francisco Da Silva and João do E. S. Batista Neto. 2011. Improving the ranking quality of medical image retrieval using a genetic feature selection method. *Decis. Support Syst.* 51, 4 (2011), 810–820.
[42] Binay Dahal and Justin Zhijun Zhan. 2020. Effective mutation and recombination for evolving convolutional networks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 1–6.
[43] N. Dalal and B. Triggs. 2005. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 886–893.
[44] David B. D'Ambrosio and Kenneth O. Stanley. 2007. A novel generative encoding for exploiting neural network sensor and output geometry. In *Proceedings of the Genetic Evolutionary Computation Conference*. 974–981.
[45] Ashraf Darwish, Aboul Ella Hassanien, and Swagatam Das. 2020. A survey of swarm and evolutionary computing approaches for deep learning. *Artif. Intell. Rev.* 53, 3 (2020), 1767–1812.
[46] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* 6, 2 (2002), 182–197.
[47] Thomas Dowdell and Hongyu Zhang. 2020. Language modelling for source code with transformer-XL. *arXiv preprint arXiv:2007.15813* (2020).
[48] Ke-Lin Du and M. N. S. Swamy. 2016. *Estimation of Distribution Algorithms*. Springer International Publishing, Cham, 105–119.
[49] Thomas Elsken, Jan-Hendrik Metzen, and Frank Hutter. 2017. Simple and efficient architecture search for convolutional neural networks. *arXiv preprint arXiv:1711.04528* (2017).
[50] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Efficient multi-objective neural architecture search via Lamarckian evolution. In *Proceedings of the International Conference on Learning Representations*. Retrieved from https://arxiv.org/abs/1804.09081.
[51] Raquel Espinosa, Fernando Jiménez, and José Palma. 2023. Surrogate-assisted and filter-based multiobjective evolutionary feature selection for deep learning. *IEEE Trans. Neural Netw. Learn. Syst.* (2023). Early Access. DOI: https://doi.org/10.1109/TNNLS.2023.3234629
[52] Pablo A. Estévez and Rodrigo E. Caballero. 1998. A niching genetic algorithm for selecting features for neural network classifiers. In *Proceedings of the International Conference on Artificial Neural Networks*. 311–316.
[53] Benjamin Evans, Harith Al-Sahaf, Bing Xue, and Mengjie Zhang. 2018. Evolutionary deep learning: A genetic programming approach to image classification. In *Proceedings of the IEEE Congress on Evolutionary Computation*. 1538–1545.

[54] Tresna Maulana Fahrudin, Iwan Syarif, and Ali Ridho Barakbah. 2016. Ant colony algorithm for feature selection on microarray datasets. In *Proceedings of the International Electronics Symposium*. 351–356.
[55] Qinglan Fan, Ying Bi, Bing Xue, and Mengjie Zhang. 2022. A global and local surrogate-assisted genetic programming approach to image classification. *IEEE Trans. Evol. Comput.* (2022). Early Access. DOI: https://doi.org/10.1109/TEVC.2022.3214607
[56] Zhun Fan, Jiahong Wei, Guijie Zhu, Jiajie Mo, and Wenji Li. 2020. Evolutionary neural architecture search for retinal vessel segmentation. *arXiv preprint arXiv:2001.06678* (2020).
[57] Christopher Fogelberg and Mengjie Zhang. 2005. Linear genetic programming for multi-class object classification. In *Proceedings of the Australian Joint Conference on Artificial Intelligence*. 369–379.
[58] Luigi Fortuna and Mattia Frasca. 2021. Singular value decomposition. *Optim. Robot. Control* 14, 2 (2021), 51–58.
[59] Luc Frachon, Wei Pang, and George M. Coghill. 2019. ImmuNeCS: Neural committee search by an artificial immune system. *arXiv preprint arXiv:1911.07729* (2019).
[60] Alex A. Freitas. 2003. A survey of evolutionary algorithms for data mining and knowledge discovery. In *Advances in Evolutionary Computing*. Springer, 819–845.
[61] Ying Fu, Min Gong, Guang Yang, Hong Wei, and Jiliu Zhou. 2021. Evolutionary GAN–based data augmentation for cardiac magnetic resonance image. *Comput., Mater. Contin.* 1, 68 (2021), 1359–1374.
[62] Saya Fujino, Taichi Hatanaka, Naoki Mori, and Keinosuke Matsumoto. 2019. Evolutionary deep learning based on deep convolutional neural network for anime storyboard recognition. *Neurocomputing* 338 (2019), 393–398.
[63] Saya Fujino, Naoki Mori, and Keinosuke Matsumoto. 2017. Deep convolutional networks for human sketches by means of the evolutionary deep learning. In *Proceedings of the International Conference on Soft Computing and Intelligent Systems*. 1–5.
[64] Kosaku Fujita, Masayuki Kobayashi, and Tomoharu Nagao. 2018. Data augmentation using evolutionary image processing. In *Proceedings of the International Conference on Digital Image Computing: Techniques and Applications*. 1–6.
[65] Edgar Galván and Peter Mooney. 2021. Neuroevolution in deep neural networks: Current trends and future challenges. *IEEE Trans.* 2, 6 (2021), 476–493.
[66] David García, Antonio González Muñoz, and Raúl Pérez. 2011. A two-step approach of feature construction for a genetic learning algorithm. In *Proceedings of the International Conference on Fuzzy Systems*. 1255–1262.
[67] Wolfgang Golubski and Thomas Feuring. 1999. Evolving neural network structures by means of genetic programming. In *Proceedings of the European Conference on Genetic Programming*. 211–220.
[68] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *Int. J. Comput. Vis.* 129, 6 (2021), 1789–1819.
[69] Zhenhua Guo, Lei Zhang, and David Zhang. 2010. A completed modeling of local binary pattern operator for texture classification. *IEEE Trans. Image Process.* 19, 6 (2010), 1657–1663.
[70] Farshid Hajati, Caro Lucas, and Yongsheng Gao. 2010. Face localization using an effective co-evolutionary genetic algorithm. In *Proceedings of the International Conference on Digital Image Computing: Techniques and Applications*. 522–527.
[71] Marwa Hammami, Slim Bechikh, and Chih-Cheng Hung. 2018. A multi-objective hybrid filter-wrapper evolutionary approach for feature construction on high-dimensional data. In *Proceedings of the IEEE Congress on Evolutionary Computation*. 1–8.
[72] Sang-Jun Han and Sung-Bae Cho. 2006. Evolutionary neural networks for anomaly detection based on the behavior of a program. *IEEE Trans. Syst. Man Cybern.* 36, 3 (2006), 559–570.
[73] Emrah Hancer, Bing Xue, Mengjie Zhang, and Dervis Karaboga. 2015. A multi-objective artificial bee colony approach to feature selection using fuzzy mutual information. In *Proceedings of the IEEE Congress on Evolutionary Computation*. 2420–2427.
[74] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A survey of the state-of-the-art. *Knowl.-based Syst.* 212 (2021), 106622.
[75] Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 1398–1406.
[76] Jin-Hyuk Hong and Sung-Bae Cho. 2006. Efficient huge-scale feature selection with speciated genetic algorithm. *Pattern Recognit. Lett.* 27, 2 (2006), 143–150.
[77] Mohamed Hosni, Ginés García-Mateos, and Juan Carrillo-de Gea. 2020. A mapping study of ensemble classification methods in lung cancer decision support systems. *Med. Biol. Eng. Comput.* 58, 10 (2020), 2177–2193.
[78] Yingtung Hsiao. 2004. Multiobjective evolution programming method for feeder reconfiguration. *IEEE Trans. Power. Syst.* 19, 1 (2004), 594–599.
[79] Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. 2021. Language model compression with weighted low-rank factorization. In *Proceedings of the International Conference on Learning Representations*. Retrieved from https://arxiv.org/abs/2207.00112.

[80] Bin Hu, Tianming Zhao, Yucheng Xie, Yan Wang, and Xiaonan Guo. 2021. MIXP: Efficient deep neural networks pruning for further FLOPs compression via neuron bond. In *Proceedings of the International Joint Conference on Neural Networks*. 1–8.
[81] Junhao Huang, Weize Sun, and Lei Huang. 2020. Deep neural networks compression learning based on multiobjective evolutionary algorithms. *Neurocomputing* 378 (2020), 260–269.
[82] Renke Huang, Wei Gao, Rui Fan, and Qiuhua Huang. 2022. A guided evolutionary strategy based Static Var Compensator control approach for inter-area oscillation damping. *IEEE Trans. Industr. Inform.* 19, 3 (2022), 2596–2607.
[83] Earnest Paul Ijjina and Krishna Mohan Chalavadi. 2016. Human action recognition using genetic algorithms and convolutional neural networks. *Pattern Recognit.* 59 (2016), 199–212.
[84] William Irwin-Harris, Yanan Sun, Bing Xue, and Mengjie Zhang. 2019. A graph-based encoding for evolutionary convolutional neural network architecture design. In *Proceedings of the IEEE Congress on Evolutionary Computation*. 546–553.
[85] Alan Julian Izenman. 2013. Linear discriminant analysis. In *Modern Multivariate Statistical Techniques*. Springer, 237–280.
[86] Yesmina Jaâfra, Jean Luc Laurent, Aline Deruyver, and Mohamed Saber Naceur. 2019. Reinforcement learning for neural architecture search: A review. *Image Vis. Comput.* 89 (2019), 57–66.
[87] Yanxiang Jiang, Xuan Chen, Fu-Chun Zheng, Dusit Niyato, and Xiaohu You. 2021. Brain storm optimization-based edge caching in fog radio access networks. *IEEE Trans. Vehic. Technol.* 70, 2 (2021), 1807–1820.
[88] Xiaoqi Jiao, Huating Chang, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Improving task-agnostic BERT distillation with layer mapping search. *Neurocomputing* 461 (2020), 194–203.
[89] Haifeng Jin, Qingquan Song, and Xia Hu. 2018. Auto-Keras: Efficient neural architecture search with network morphism. *arXiv preprint arXiv:1806.10282* (2018).
[90] Haifeng Jin, Qingquan Song, and Xia Hu. 2019. Auto-Keras: An efficient neural architecture search system. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1946–1956.
[91] Yaochu Jin. 2006. *Multi-Objective Machine Learning*. Springer Science.
[92] Dervis Karaboga, Bahriye Akay, and Celal Öztürk. 2007. Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. In *Proceedings of the International Conference on Modeling Decisions for Artificial Intelligence*. 318–329.
[93] Asha Gowda Karegowda and A. S. Manjunath. 2011. Application of genetic algorithm optimized neural network connection weights for medical diagnosis of PIMA Indians diabetes. *Int. J. Soft Comput.* 2, 2 (2011), 15–23.
[94] Shauharda Khadka and Kagan Tumer. 2018. Evolution-guided policy gradient in reinforcement learning. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 1196–1208.
[95] Rami N. Khushaba, Ahmed Al-Ani, Akram AlSukker, and Adel Al-Jumaily. 2008. A combined ant colony and differential evolution feature selection algorithm. In *Proceedings of the International Conference on Ant Colony Optimization and Swarm Intelligence*. 1–12.
[96] Hiroaki Kitano. 1990. Designing neural networks using genetic algorithms with graph generation system. *Complex Syst.* 4, 4 (1990), 225–238.
[97] Madhusudhan Kongovi, Juan Carlos Guzman, and Venu Dasigi. 2002. Text categorization: An experiment using phrases. In *Proceedings of the European Conference on Information Retrieval*. Springer, 213–228.
[98] Manabu Kotani and Daisuke Kato. 2004. Feature extraction using coevolutionary genetic programming. In *Proceedings of the IEEE Congress on Evolutionary Computation*. 614–619.
[99] Jan Koutník, Faustino J. Gomez, and Jürgen Schmidhuber. 2010. Evolving neural networks in compressed weight space. In *Proceedings of the Genetic Evolutionary Computation Conference*. 619–626.
[100] Jan Koutník, Jürgen Schmidhuber, and Faustino J. Gomez. 2014. Evolving deep unsupervised convolutional networks for vision-based reinforcement learning. In *Proceedings of the Genetic Evolutionary Computation Conference*. 541–548.
[101] Krzysztof Krawiec. 2002. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genet. Program. Evolvable Mach.* 3, 4 (2002), 329–343.
[102] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 1097–1105.
[103] Arkadiusz Kwasigroch, Michał Grochowski, and Mateusz Mikolajczyk. 2019. Deep neural network architecture search using network morphism. In *Proceedings of the International Conference on Methods and Models in Automation and Robotics*. 30–35.
[104] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.
[105] Yann LeCun, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, and Wayne Hubbard. 1989. Backpropagation applied to handwritten zip code recognition. *Neural Comput.* 1, 4 (1989), 541–551.

[106] Jun-qing Li, Zheng-min Liu, Chengdong Li, and Zhi-xin Zheng. 2021. Improved artificial immune system algorithm for type-2 fuzzy flexible job shop scheduling problem. *IEEE Trans. Fuzzy Syst.* 29, 11 (2021), 3234–3248.
[107] Nan Li, Lianbo Ma, Tiejun Xing, Guo Yu, Chen Wang, Yingyou Wen, Shi Cheng, and Shangce Gao. 2023. Automatic design of machine learning via evolutionary computation: A survey. *Appl. Soft Comput.* 143 (2023), 110412.
[108] Qing Li, Wei Zhang, Lin Zhao, Xia Wu, and Tianming Liu. 2022. Evolutional neural architecture search for optimization of spatiotemporal brain network decomposition. *IEEE. Trans. Biomed. Eng.* 69, 2 (2022), 624–634.
[109] Youru Li, Zhenfeng Zhu, Deqiang Kong, Hua Han, and Yao Zhao. 2019. EA-LSTM: Evolutionary attention-based LSTM for time series prediction. *Knowl.-based Syst.* 181 (2019), 104785.
[110] Jason Zhi Liang, Elliot Meyerson, Babak Hodjat, Daniel Fink, Karl Mutch, and Risto Miikkulainen. 2019. Evolutionary neural AutoML for deep learning. In *Proceedings of the Genetic Evolutionary Computation Conference*. 401–409.
[111] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. 2021. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing* 461 (2021), 370–403.
[112] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. 2018. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision*. 19–34.
[113] Chia-Hsiang Liu, Yu-Shin Han, Yuan-Yao Sung, Yi Lee, Hung-Yueh Chiang, and Kai-Chiang Wu. 2021. FOX-NAS: Fast, on-device and explainable neural architecture search. In *Proceedings of the IEEE International Conference on Computer Vision*. 789–797.
[114] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. DARTS: Differentiable architecture search. In *Proceedings of the International Conference on Learning Representations*. https://arxiv.org/abs/1806.09055.
[115] Peng Liu, Mohammad D. El Basha, Yangjunyi Li, Yao Xiao, Pina C. Sanelli, and Ruogu Fang. 2019. Deep evolutionary networks with expedited genetic algorithms for medical image denoising. *Med. Image Anal.* 54 (2019), 306–315.
[116] Sicong Liu and Bin Guo. 2021. AdaSpring: Context-adaptive and runtime-evolutionary deep model compression for mobile applications. *Proc. ACM Interact., Mobile, Wearable Ubiquitous Tech.*, Vol. 5. ACM, 1–22.
[117] Shulei Liu, Handing Wang, Wei Peng, and Wen Yao. 2022. A surrogate-assisted evolutionary feature selection algorithm with parallel random grouping for high-dimensional classification. *IEEE Trans. Evol. Comput.* 26, 5 (2022), 1087–1101. DOI: https://doi.org/10.1109/TEVC.2022.3149601
[118] Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G. Yen, and Kay Chen Tan. 2021. A survey on evolutionary neural architecture search. *IEEE Trans. Neural Netw. Learn. Syst.* 34, 2 (2021), 550–570.
[119] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, K. Cheng, and Jian Sun. 2019. MetaPruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE International Conference on Computer Vision*. 3295–3304.
[120] Eugenio Lomurno, Stefano Samele, Matteo Matteucci, and Danilo Ardagna. 2021. Pareto-optimal progressive neural architecture search. In *Proceedings of the Genetic Evolutionary Computation Conference*. 1726–1734.
[121] Pablo Ribalta Lorenzo and Jakub Nalepa. 2018. Memetic evolution of deep neural networks. In *Proceedings of the Genetic Evolutionary Computation Conference*. 505–512.
[122] Pablo Ribalta Lorenzo, Jakub Nalepa, Luciano Sánchez Ramos, and José Ranilla. 2017. Hyper-parameter selection in deep neural networks using parallel particle swarm optimization. In *Proceedings of the Genetic Evolutionary Computation Conference*. 1864–1871.
[123] Zhichao Lu, Gautam Sreekumar, Erik Goodman, Wolfgang Banzhaf, Kalyanmoy Deb, and Vishnu Naresh Boddeti. 2021. Neural architecture transfer. *IEEE IEEE Trans. Pattern Anal. Mach. Intell.* 43, 9 (2021), 2971–2989.
[124] Zhichao Lu, Ian Whalen, Vishnu Naresh Boddeti, Yashesh D. Dhebar, and Kalyanmoy Deb. 2019. NSGA-Net: Neural architecture search using multi-objective genetic algorithm. In *Proceedings of the Genetic Evolutionary Computation Conference*. 419–427.
[125] Lianbo Ma, Min Huang, Shengxiang Yang, Rui Wang, and Xingwei Wang. 2022. An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization. *IEEE Trans. Cybern.* 52, 7 (2022), 6684–6696.
[126] Lianbo Ma, Nan Li, Guo Yu, Xiaoyu Geng, Min Huang, and Xingwei Wang. 2021. How to simplify search: Classification-wise pareto evolution for one-shot neural architecture search. *arXiv preprint arXiv:2109.07582* (2021).
[127] Wenping Ma, Xiaobo Zhou, Hao Zhu, Longwei Li, and Licheng Jiao. 2021. A two-stage hybrid ant colony optimization for high-dimensional feature selection. *Pattern Recognit.* 116 (2021), 107933.
[128] Stefano Mauceri, James Sweeney, Miguel Nicolau, and James McDermott. 2021. Feature extraction by grammatical evolution for one-class time series classification. *Genet. Program. Evolvable Mach.* 22, 3 (2021), 267–295.
[129] Kaitav Nayankumar Mehta, Ziad Kobti, Kathryn A. Pfaff, and Susan Fox. 2019. Data augmentation using CA evolved GANs. *IEEE Symp. Comput. Commun.* (2019), 1087–1092.
[130] Silvan Mertes, Alice Baird, Dominik Schiller, Björn W. Schuller, and Elisabeth André. 2020. An evolutionary-based generative approach for audio data augmentation. In *Proceedings of the IEEE International Conference on Multimedia and Signal Processing*. 1–6.

[131] Erfan Miahi, Seyed Abolghasem Mirroshandel, and Alexis Nasr. 2022. Genetic neural architecture search for automatic assessment of human sperm images. *Expert Syst. Appl.* 188 (2022), 115937.
[132] Seyedali Mirjalili, Hossam Faris, and Ibrahim Aljarah. 2019. *Evolutionary Machine Learning Techniques*. Springer.
[133] Kamlesh Mistry, Li Zhang, Siew Chin Neoh, Chee Peng Lim, and Ben Fielding. 2016. A micro-GA embedded PSO feature selection approach to intelligent facial emotion recognition. *IEEE Trans. Cybern.* 47, 6 (2016), 1496–1509.
[134] Hyunho Mo, Leonardo Lucio Custode, and Giovanni Iacca. 2021. Evolutionary neural architecture search for remaining useful life prediction. *Appl. Soft Comput.* 108 (2021), 107474.
[135] David J. Montana and Lawrence Davis. 1989. Training feedforward neural networks using genetic algorithms. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 762–767.
[136] Hiroshi Motoda and Huan Liu. 2002. Feature selection extraction and construction. *Commun. IICM* 5, 2 (2002), 67–72.
[137] Alhassan G. Mumuni and Fuseini Mumuni. 2022. Data augmentation: A comprehensive survey of modern approaches. *Array* 16 (2022), 100258.
[138] Durga Prasad Muni, Nikhil R. Pal, and Jyotirmay Das. 2006. Genetic programming for simultaneous feature selection and classifier design. *IEEE Trans. Syst. Man. Cybern.* 36, 1 (2006), 106–117.
[139] Mehdi Neshat, Meysam Majidi Nezhad, Ehsan Abbasnejad, Lina Bertling Tjernberg, Davide Astiaso Garcia, Bradley Alexander, and Markus Wagner. 2020. An evolutionary deep learning method for short-term wind speed prediction: A case study of the Lillgrund offshore wind farm. *arXiv preprint arXiv:abs/2002.09106* (2020).
[140] Kourosh Neshatian, Mengjie Zhang, and Peter Andreae. 2012. A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming. *IEEE Trans. Evol. Comput.* 16, 5 (2012), 645–661.
[141] Kourosh Neshatian, Mengjie Zhang, and Mark Johnston. 2007. Feature construction and dimension reduction using genetic programming. In *Proceedings of the Australian Conference on Artificial Intelligence*. 242–253.
[142] Bach Hoai Nguyen, Bing Xue, and Mengjie Zhang. 2022. A constrained competitive swarm optimiser with an SVM-based surrogate model for feature selection. *IEEE Trans. Evol. Comput.* (2022). Early Access. DOI: https://doi.org/10.1109/TEVC.2022.3197427
[143] Hoai Bach Nguyen, Bing Xue, and Peter Andreae. 2017. Surrogate-model based particle swarm optimisation with local search for feature selection in classification. In *Proceedings of the European Conference on Genetic Programming*. Springer, 487–505.
[144] Hoai Bach Nguyen, Bing Xue, Ivy Liu, and Mengjie Zhang. 2014. PSO and statistical clustering for feature selection: A new representation. In *Proceedings of the Asia-Pacific Conference on Simulated Evolutionary Learning*. 569–581.
[145] Noel M. O'Boyle and David S. Palmer. 2008. Simultaneous feature selection and parameter optimisation using an artificial ant colony: Case study of melting point prediction. *Chem. Cent. J.* 2, 1 (2008), 1–15.
[146] Damien O'Neill, Bing Xue, and Mengjie Zhang. 2018. Co-evolution of novel tree-like ANNs and activation functions: An observational study. In *Proceedings of the Australian Conference on Artificial Intelligence*. 616–629.
[147] Tatt Hee Oong and Nor Ashidi Mat Isa. 2011. Adaptive evolutionary artificial neural networks for pattern classification. *IEEE Trans. Neural Netw. Syst.* 22, 11 (2011), 1823–1836.
[148] Patxi Ortego, Alberto Diez-Olivan, Javier Del Ser, and Fernando Veiga. 2020. Evolutionary LSTM-FCN networks for pattern classification in industrial processes. *Swarm Evol. Comput.* 54 (2020), 100650.
[149] Bo Peng, Shuting Wan, Ying Bi, Bing Xue, and Mengjie Zhang. 2021. Automatic feature extraction and construction using genetic programming for rotating machinery fault diagnosis. *IEEE Trans. Cybern.* 51, 10 (2021), 4909–4923.
[150] Yiming Peng, Gang Chen, Harman Singh, and Mengjie Zhang. 2018. NEAT for large-scale reinforcement learning through evolutionary feature learning and policy gradient search. In *Proceedings of the Genetic Evolutionary Computation Conference*. 490–497.
[151] Sofia Pereira, João Correia, and Penousal Machado. 2022. Evolving data augmentation strategies. In *Proceedings of the International Conference on Applications of Evolutionary Computing*. 337–351.
[152] Hai T. Phan, Zechun Liu, Dang The Huynh, Marios Savvides, Kwang-Ting Cheng, and Zhiqiang Shen. 2020. Binarizing MobileNet via evolution-based searching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 13417–13426.
[153] Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. In *Proceedings of the International Conference on Learning Representations*. https://arxiv.org/abs/1802.05668.
[154] Elad Rapaport, Oren Shriki, and Rami Puzis. 2019. EEGNAS: Neural architecture search for electroencephalography data analysis and decoding. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 3–20.
[155] A. N. M. Bazlur Rashid, Mohiuddin Ahmed, Leslie F. Sikos, and Paul Haskell-Dowland. 2020. Cooperative co-evolution for feature selection in big data with random feature grouping. *J. Big Data* 7, 1 (2020), 1–42.
[156] Aditya Rawal and Risto Miikkulainen. 2018. From nodes to networks: Evolving recurrent neural networks. *arXiv preprint arXiv:1803.04439* (2018).
[157] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. 2017. Large-scale evolution of image classifiers. In *Proceedings of the International Conference on Machine Learning*. 2902–2911.

[158] Mohammad Saleh Refahi, A. Mir, and Jalal A. Nasiri. 2020. A novel fusion based on the evolutionary features for protein fold recognition using support vector machines. *Sci. Rep.* 10, 1 (2020), 1–13.
[159] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, and Zhihui Li. 2021. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Comput. Surv.* 54, 4 (2021), 1–34.
[160] Mark E. Roberts and Ela Claridge. 2005. A multistage approach to cooperatively coevolving feature construction and object detection. In *Proceedings of the International Conference on Applications of Evolutionary Computing*. 396–406.
[161] Shahin Rostami and Ferrante Neri. 2016. Covariance matrix adaptation Pareto archived evolution strategy with hypervolume-sorted adaptive grid algorithm. *Integr. Comput. Aided. Eng.* 23, 4 (2016), 313–329.
[162] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1985. *Learning Internal Representations by Error Propagation.* Technical Report. California University San Diego La Jolla Institute for Cognitive Science.
[163] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323, 6088 (1986), 533–536.
[164] Santanu Santra, Jun-Wei Hsieh, and Chi-Fang Lin. 2021. Gradient descent effects on differential neural architecture search: A survey. *IEEE Access* 9 (2021), 89602–89618.
[165] Dolly Sapra and Andy D. Pimentel. 2020. Constrained evolutionary piecemeal training to design convolutional neural networks. In *Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. 709–721.
[166] Christoph Schorn, Thomas Elsken, Sebastian Vogel, and Armin Runge. 2020. Automated design of error-resilient and hardware-efficient deep neural networks. *Neural. Comput. Appl.* 32, 24 (2020), 18327–18345.
[167] Leila Shila Shafti and E. Islas Pérez. 2008. Data reduction by genetic algorithms and non-algebraic feature construction: A case study. In *Proceedings of the International Conference on Hybrid Intelligent Systems*. 573–578.
[168] Pratistha Shakya, Eamonn Kennedy, Christopher Rose, and Jacob K. Rotein. 2021. High-dimensional time series feature extraction for low-cost machine olfaction. *IEEE Sens. J.* 21, 3 (2021), 2495–2504.
[169] Mingzhu Shen, Kai Han, Chunjing Xu, and Yunhe Wang. 2019. Searching for accurate binary neural architectures. In *Proceedings of the IEEE International Conference on Computer Vision*. 2041–2044.
[170] Christopher Smith and Yaochu Jin. 2014. Evolutionary multi-objective generation of recurrent neural network ensembles for time series prediction. *Neurocomputing* 143 (2014), 302–311.
[171] Krzysztof Socha and Christian Blum. 2007. An ant colony optimization algorithm for continuous optimization: Application to feed-forward neural network training. *Neural. Comput. Appl.* 16, 3 (2007), 235–247.
[172] Kenneth O. Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. 2019. Designing neural networks through neuroevolution. *Nat. Mach. Intell.* 1, 1 (2019), 24–35.
[173] Kenneth O. Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evol. Comput.* 10, 2 (2002), 99–127.
[174] Rob Stewart, Andrew Nowlan, Pascal Bacchus, Quentin Ducasse, and Ekaterina Komendantskaya. 2021. Optimising hardware accelerated neural networks with quantisation and a knowledge distillation evolutionary algorithm. *Electronics* 10, 4 (2021).
[175] Yanan Sun, Xian Sun, Yuhan Fang, Gary G. Yen, and Yuqiao Liu. 2021. A novel training protocol for performance predictors of evolutionary neural architecture search algorithms. *IEEE Trans. Evol. Comput.* 25, 3 (2021), 524–536.
[176] Yanan Sun, Handing Wang, Bing Xue, Yaochu Jin, Gary G. Yen, and Mengjie Zhang. 2019. Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor. *IEEE Trans. Evol. Comput.* 24, 2 (2019), 350–364.
[177] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G. Yen. 2019. Completely automated CNN architecture design based on blocks. *IEEE Trans. Neural Netw. Learn. Syst.* 31, 4 (2019), 1242–1254.
[178] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G. Yen. 2019. Evolving deep convolutional neural networks for image classification. *IEEE Trans. Evol. Comput.* 24, 2 (2019), 394–407.
[179] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G. Yen. 2019. A particle swarm optimization-based flexible convolutional autoencoder for image classification. *IEEE Trans. Neural Netw. Learn. Syst.* 30, 8 (2019), 2295–2309.
[180] Yanan Sun, Bing Xue, Mengjie Zhang, Gary G. Yen, and Jiancheng Lv. 2020. Automatically designing CNN architectures using the genetic algorithm for image classification. *IEEE Trans. Cybern.* 50, 9 (2020), 3840–3854.
[181] Sridhar Swaminathan, Deepak Garg, Rajkumar Kannan, and Frédéric Andrès. 2020. Sparse low rank factorization for deep neural network compression. *Neurocomputing* 398 (2020), 185–196.
[182] Yajiao Tang, Junkai Ji, Yulin Zhu, Shangce Gao, Zheng Tang, and Yuki Todo. 2019. A differential evolution-oriented pruning neural network model for bankruptcy prediction. In *Complexity*, Vol. 2019. 8682124:1–8682124:21.
[183] Hassan Tariq, Elf Eldridge, and Ian Welch. 2018. An efficient approach for feature construction of high-dimensional microarray data by random projections. *PLoS One* 13, 4 (2018), e0196385.
[184] Akbar Telikani, Amirhessam Tahmassebi, Wolfgang Banzhaf, and Amir H. Gandomi. 2021. Evolutionary machine learning: A survey. *ACM Comput. Surv.* 54, 8 (2021), 1–35.

[185] Astro Teller and Manuela Veloso. 1996. PADO: A new learning architecture for object recognition. *Symbol. Visual Learn.* 4, 18 (1996), 81–116.
[186] Haiman Tian, ShuChing Chen, MeiLing Shyu, and Stuart Harvey Rubin. 2019. Automated neural network construction with similarity sensitive evolutionary algorithms. In *Proceedings of the IEEE International Conference on Information Reuse and Integration for Data Science.* 283–290.
[187] Binh Tran, Bing Xue, and Mengjie Zhang. 2018. A new representation in PSO for discretization-based feature selection. *IEEE Trans. Cybern.* 48, 6 (2018), 1733–1746.
[188] Binh Tran, Bing Xue, and Mengjie Zhang. 2019. Genetic programming for multiple-feature construction on high-dimensional classification. *Pattern Recognit.* 93 (2019), 404–417.
[189] Binh Tran, Mengjie Zhang, and Bing Xue. 2016. Multiple feature construction in classification on high-dimensional data using GP. In *IEEE Symposium Series on Computational Intelligence.* 1–8.
[190] Haleh Vafaie and Kenneth De Jong. 1998. Feature space transformation using genetic algorithms. *IEEE Intell. Syst. Applic.* 13, 2 (1998), 57–65.
[191] Gustavo A. Vargas-Hákim, Efrén Mezura-Montes, and Héctor-Gabriel Acosta-Mesa. 2022. A review on convolutional neural networks encodings for neuroevolution. *IEEE Trans. Evol. Comput.* 26, 1 (2022), 12–27.
[192] Susana M. Vieira, Luís F. Mendonça, Goncalo J. Farinha, and João M. C. Sousa. 2013. Modified binary PSO for feature selection using SVM applied to mortality prediction of septic patients. *Appl. Soft Comput.* 13, 8 (2013), 3494–3504.
[193] Bin Wang, Bing Xue, and Mengjie Zhang. 2020. Particle swarm optimization for evolving deep convolutional neural networks for image classification: Single- and multi-objective approaches. In *Deep Neural Evolution.* Springer, 155–184.
[194] Bin Wang, Bing Xue, and Mengjie Zhang. 2020. Particle swarm optimization for evolving deep neural networks for image classification by evolving and stacking transferable blocks. In *Proceedings of the IEEE Congress on Evolutionary Computation.* 1–8.
[195] Bin Wang, Bing Xue, and Mengjie Zhang. 2021. Surrogate-assisted particle swarm optimization for evolving variable-length transferable blocks for image classification. *IEEE Trans. Neural Netw. Learn. Syst.* 33, 8 (2021), 3727–3740.
[196] Jiaxing Wang, Haoli Bai, Jiaxiang Wu, and Jian Cheng. 2020. Bayesian automatic model compression. *IEEE J. Select. Topics Signal Process.* 14, 4 (2020), 727–736.
[197] Shuyan Wang, Chunyan Wen, and Jiaze Sun. 2016. Test data augmentation method based on adaptive particle swarm optimization algorithm. *J. Netw. Comput. Applic.* 36, 9 (2016), 2492.
[198] Xiao-han Wang, Yong Zhang, and Xiao-yan Sun. 2020. Multi-objective feature selection based on artificial bee colony: An acceleration approach with variable sample size. *Appl. Soft Comput.* 88 (2020), 106041.
[199] Yunhe Wang, Chang Xu, Jiayan Qiu, Chao Xu, and Dacheng Tao. 2018. Towards evolutionary compression. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2476–2485.
[200] Yun Wen and Hua Xu. 2011. A cooperative coevolution-based Pittsburgh learning classifier system embedded with memetic feature selection. In *Proceedings of the IEEE Congress on Evolutionary Computation.* 2415–2422.
[201] Colin White, Willie Neiswanger, and Yash Savani. 2021. BANANAS: Bayesian optimization with neural architectures for neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence.* 10293–10301.
[202] Genta Indra Winata, Andrea Madotto, Jamin Shin, Elham J. Barezi, and Pascale Fung. 2019. On the effectiveness of low-rank matrix factorization for LSTM model compression. *arXiv* abs/1908.09982 (2019).
[203] Min Wu, Wanjuan Su, Luefeng Chen, and Zhentao Liu. 2021. Weight-adapted convolution neural network for facial expression recognition in human-robot interaction. *IEEE Trans. Syst. Man Cybern.* 51, 3 (2021), 1473–1484.
[204] Tao Wu, Xiaoyang Li, Deyun Zhou, Na Li, and Jiao Shi. 2021. Differential evolution based layer-wise weight pruning for compressing deep neural networks. *Sensors* 21, 3 (2021), 880.
[205] Lingxi Xie, Xin Chen, Kaifeng Bi, Longhui Wei, Yuhui Xu, Zhengsu Chen, Lanfei Wang, Anxiang Xiao, Jianlong Chang, Xiaopeng Zhang, and Qi Tian. 2022. Weight-sharing neural architecture search: A battle to shrink the optimization gap. *ACM Comput. Surv.* 54, 9 (2022), 1–37.
[206] Lingxi Xie and Alan Yuille. 2017. Genetic CNN. In *Proceedings of the IEEE International Conference on Computer Vision.* 1379–1388.
[207] Bing Xue, Mengjie Zhang, and Will N. Browne. 2012. Multi-objective particle swarm optimization (PSO) for feature selection. In *Proceedings of the Genetic Evolutionary Computation Conference.* 81–88.
[208] Bing Xue, Mengjie Zhang, Will N. Browne, and Xin Yao. 2015. A survey on evolutionary computation approaches to feature selection. *IEEE Trans. Evol. Comput.* 20, 4 (2015), 606–626.
[209] Bing Xue, Mengjie Zhang, Yan Dai, and Will N. Browne. 2013. PSO for feature construction and binary classification. In *Proceedings of the Genetic Evolutionary Computation Conference.* 137–144.
[210] Shangshang Yang, Ye Tian, Cheng He, Xingyi Zhang, Kay Chen Tan, and Yaochu Jin. 2021. A gradient-guided evolutionary approach to training deep neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 33, 9 (2021), 4861–4875.

[211] Ziqing Yang, Yiming Cui, Xin Yao, and Shijin Wang. 2022. Gradient-based intra-attention pruning on pre-trained language models. *arXiv preprint arXiv:2212.07634* (2022).
[212] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Boxin Shi, and Chao Xu. 2020. CARS: Continuous evolution for efficient neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1826–1835.
[213] Quanming Yao, Mengshuo Wang, Yuqiang Chen, Wenyuan Dai, and Yu-Feng Li. 2018. Taking human out of learning applications: A survey on automated machine learning. *arXiv preprint arXiv:1810.13306* (2018).
[214] Xin Yao. 1993. A review of evolutionary artificial neural networks. *Int. J. Intell. Syst.* 8, 4 (1993), 539–567.
[215] Xin Yao and Yong Liu. 1996. Ensemble structure of evolutionary artificial neural networks. In *Proceedings of the Genetic Evolutionary Computation Conference*. 659–664.
[216] Guo Yu, Yaochu Jin, and Markus Olhofer. 2019. Benchmark problems and performance indicators for search of knee points in multiobjective optimization. *IEEE Trans. Cybern.* 50, 8 (2019), 3531–3544.
[217] Guo Yu, Yaochu Jin, and Markus Olhofer. 2020. A multiobjective evolutionary algorithm for finding knee regions using two localized dominance relationships. *IEEE Trans. Evol. Comput.* 25, 1 (2020), 145–158.
[218] Guo Yu, Yaochu Jin, Markus Olhofer, Qiqi Liu, and Wenli Du. 2021. Solution set augmentation for knee identification in multiobjective decision analysis. *IEEE Trans. Cybern.* 53, 4 (2021), 2480–2493.
[219] Guo Yu, Lianbo Ma, Yaochu Jin, Wenli Du, Qiqi Liu, and Hengmin Zhang. 2022. A survey on knee-oriented multiobjective evolutionary optimization. *IEEE Trans. Evol. Comput.* 26, 6 (2022), 1452–1472.
[220] Hualong Yu, Guochang Gu, Haibo Liu, Jing Shen, and Jing Zhao. 2009. A modified ant colony optimization algorithm for tumor marker gene selection. *Genom., Proteom. Bioinform.* 7, 4 (2009), 200–208.
[221] Yang Yu. 2018. Towards sample efficient reinforcement learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 5739–5743.
[222] Emigdio Z.-Flores, Leonardo Trujillo, Pierrick Legrand, and Frédérique Faïta-Aïnseba. 2020. EEG feature extraction using genetic programming for the classification of mental states. *Algorithms* 13, 9 (2020), 221.
[223] Zhi-Hui Zhan, Jian-Yu Li, and Jun Zhang. 2022. Evolutionary deep learning: A survey. *Neurocomputing* 483 (2022), 42–58.
[224] Byoung-Tak Zhang and Heinz Mühlenbein. 1995. Balancing accuracy and parsimony in genetic programming. *Evol. Comput.* 3, 1 (1995), 17–38.
[225] Haoling Zhang, Chao-Han Huck Yang, Hector Zenil, Narsis Aftab Kiani, Yue Shen, and Jesper N. Tegner. 2020. Evolving neural networks through a reverse encoding tree. In *Proceedings of the IEEE Congress on Evolutionary Computation*. 1–10.
[226] Jiawei Zhang and Fisher B. Gouza. 2018. GADAM: Genetic-evolutionary ADAM for deep neural network optimization. *arXiv preprint arXiv:1805.07500* (2018).
[227] Mengjie Zhang. 2018. Evolutionary deep learning for image analysis. Retrieved from https://ieeetv.ieee.org/mengjie--zhang--evolutionary--deep--learning--for--image--analysis.
[228] Mengjie Zhang and Stefano Cagnoni. 2020. Evolutionary computation and evolutionary deep learning for image analysis, signal processing and pattern recognition. In *Proceedings of the Genetic Evolutionary Computation Conference*. 1221–1257.
[229] Mengjie Zhang and Will Smart. 2004. Genetic programming with gradient descent search for multiclass object classification. In *Proceedings of the European Conference on Genetic Programming*. 399–408.
[230] Yong Zhang, Dun-wei Gong, Xiao-yan Sun, and Yi-nan Guo. 2017. A PSO-based multi-objective multi-label feature selection method in classification. *Sci. Rep.* 7, 1 (2017), 1–12.
[231] Yang Zhang and Peter I. Rockett. 2011. A generic optimising feature extraction method using multiobjective genetic programming. *Appl. Soft Comput.* 11, 1 (2011), 1087–1097.
[232] Qijun Zhao, David Zhang, and Hongtao Lu. 2006. A direct evolutionary feature extraction algorithm for classifying high dimensional data. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 561–566.
[233] Qijun Zhao, David Dian Zhang, Lei Zhang, and Hongtao Lu. 2009. Evolutionary discriminant feature extraction with application to face recognition. *EURASIP J. Adv. Signal Process.* 2009 (2009), 1–12.
[234] Tianwen Zhao, Qijun Zhao, Hongtao Lu, and David Dian Zhang. 2007. Bagging evolutionary feature extraction algorithm for classification. In *Proceedings of the International Conference on Natural Computation*. 540–545.
[235] Xun Zhou, A. K. Qin, Maoguo Gong, and Kay Chen Tan. 2021. A survey on evolutionary construction of deep neural networks. *IEEE Trans. Evol. Comput.* 25, 5 (2021), 894–912.
[236] Yao Zhou, Bing Hu, Xianglei Yuan, Kaide Huang, Zhang Yi, and Gary G. Yen. 2023. Multi-objective evolutionary generative adversarial network compression for image translation. *IEEE Trans. Evol. Comput.* (2023). Early Access. DOI: https://doi.org/10.1109/TEVC.2023.3261135
[237] Yao Zhou, Gary G. Yen, and Zhang Yi. 2020. Evolutionary compression of deep neural networks for biomedical image segmentation. *IEEE Trans. Neural Netw. Learn. Syst.* 31, 8 (2020), 2916–2929.

[238] Yao Zhou, Gary G. Yen, and Zhang Yi. 2021. Evolutionary shallowing deep neural networks at block levels. *IEEE Trans. Neural Netw. Learn. Syst.* 33, 9 (2021), 4635–4647.

[239] Yao Zhou, Gary G. Yen, and Zhang Yi. 2021. A knee-guided evolutionary algorithm for compressing deep neural networks. *IEEE Trans. Cybern.* 51, 3 (2021), 1626–1638.

[240] Hui Zhu, Zhulin An, Chuanguang Yang, Kaiqiang Xu, and Yongjun Xu. 2019. EENA: Efficient evolution of neural architecture. In *Proceedings of the IEEE International Conference on Computer Vision*. 1891–1899.

[241] Hangyu Zhu and Yaochu Jin. 2022. Real-time federated evolutionary neural architecture search. *IEEE Trans. Evol. Comput.* 26, 2 (2022), 364–378.

[242] Barret Zoph and Quoc V. Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).

Received 21 August 2022; revised 6 April 2023; accepted 31 May 2023