Genetic Programming-Based Evolutionary Deep Learning for Data-Efficient Image Classification

Ying Bi\textsuperscript{\textregistered}, Member, IEEE, Bing Xue\textsuperscript{\textregistered}, Senior Member, IEEE, and Mengjie Zhang\textsuperscript{\textregistered}, Fellow, IEEE

Abstract—Data-efficient image classification is a challenging task that aims to solve image classification using small training data. Neural network-based deep learning methods are effective for image classification, but they typically require large-scale training data and have major limitations, such as requiring expertise to design network architectures and having poor interpretability. Evolutionary deep learning (EDL) is a recent hot topic that combines evolutionary computation with deep learning. However, most EDL methods focus on evolving architectures of neural networks, which still suffer from limitations such as poor interpretability. To address this, this article proposes a new genetic programming-based EDL approach to data-efficient image classification. The new approach can automatically evolve variable-length models using many important operators from both image and classification domains. It can learn different types of image features from color or grayscale images, and construct effective and diverse ensembles for image classification. A flexible multilayer representation enables the new approach to automatically construct shallow or deep models/trees for different tasks and perform effective transformations on the input data via multiple internal nodes. The new approach is applied to solve five image classification tasks with different training set sizes. The results show that it achieves a better performance in most cases than deep learning methods for data-efficient image classification. A deep analysis shows that the new approach has good convergence and evolves models with high interpretability, different lengths/sizes/shapes, and good transferability.

Index Terms—Deep learning, evolutionary computation (EC), evolutionary deep learning (EDL), genetic programming (GP), image classification, small data.

I. INTRODUCTION

Image classification tasks have a wide range of applications, such as detecting cancer from x-ray images, identifying a face from a set of photographs, and classifying fish species from underwater images [1], [2], [3], [4]. Although image classification has been investigated for decades, it remains a challenging task due to many factors, e.g., high interclass similarity and intraclass dissimilarity, image distortion, lack of sufficient training data, and requirement of domain knowledge. In recent years, data-efficient image classification aimed at using small data to perform effective image classification has attracted much attention. It can reduce the reliance on a large number of training data, therefore reducing the cost and effort for data collection, labeling, prepossessing, and storage. Data-efficient image classification is also important for many applications in medicine, remote sensing, and biology, where the labeled data are not easy to obtain due to the cost, privacy and security.

Deep learning is a hot research area with many successful applications [5]. The goal of deep learning is to automatically learn or discover multiple levels of abstraction from data (i.e., representation learning) that are effective for a particular task [6]. These methods often include three main characteristics, i.e., sufficient model complexity, layer-by-layer data processing, and feature transformation [7]. Deep learning methods include both neural-network (NN)-based methods, such as convolutional NNs (CNNs) [8] and non-NN-based methods, such as deep forest [7] and PCANet [9]. In recent years, deep CNNs are the dominant approach to image classification [8]. However, these NN-based methods have major limitations, such as requiring expensive computing devices to run, a large number of training data to train, and rich expertise to design the architectures [7]. Another important disadvantage of deep NNs is poor interpretability due to the large number of parameters and very deep structures in the model [7]. To address these limitations, it is worth inventing new non-NN-based deep learning methods, which can not only maintain the diversity of the artificial intelligence/machine learning/computational intelligence research community but also bring new ideas to this community.

Evolutionary deep learning (EDL) is a new research field that aims to use evolutionary computation (EC) techniques to evolve or optimize deep models [10], [11]. The advantages, i.e., population-based beam search, nondifferential objective functions, ease of cooperating with domain knowledge, robustness to dynamic changes, etc., have enabled EC methods to

1089-778X © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
solve many complex optimization and learning problems in various fields including finance, engineering, security, healthcare, manufacturing, and business [11], [12].

Existing works of EDL can be broadly classified into two categories, i.e., the use of EC methods to automatically optimize deep NNs by searching for architectures, weights, loss functions, etc., and the use of EC methods to automatically evolve/optimise variable-length, relatively deep, non-NN-based models. In the first category, many EC methods, such as genetic algorithms (GAs), genetic programming (GP), particle swarm optimization (PSO), and evolutionary multiobjective optimization (EMO) have been used to automatically evolve different types of deep NNs, e.g., CNNs, Autoencoders, recurrent neural networks (RNNs), and generative adversarial networks (GANs) [10], [13], [14]. However, they often suffer from the limitations of deep NNs, such as requiring a large number of training examples, having “black-box” models with millions of parameters, and being computationally expensive. In the second category, to the best of our knowledge, the main method is GP, which can automatically evolve deep models to solve a problem [11]. According to the definition of deep learning [5], [7], GP can be a deep learning method because it can automatically learn models with advanced operators that are sufficiently complex to perform data/feature transformation to solve tasks. There is also an increasing trend in recent works recognizing GP as a deep learning method [10], [15]. Unlike NN-based deep learning methods, GP-based EDL methods can automatically evolve variable-length models and their coefficients without making any assumptions about the model structure/architecture [16]. Importantly, the models learned by GP often have varying complexity and potentially high interpretability, which are difficult to achieve in NNs. Benefiting from these advantages, GP-based EDL methods have been successfully applied to image classification and showed a promise [17], [18], [19]. However, the potential of GP has not been fully investigated. For tasks with small training data, limited computation resources, and high requirements for model interpretability, where deep NNs have difficulty to address, it is necessary to investigate novel non-NN-based deep learning methods, i.e., GP-based methods.

Existing GP-based EDL methods have limitations in solving image classification, which can be broadly summarized into three aspects. First, most existing methods have only been examined on datasets with grayscale images. Real-world problems may have images with various numbers of channels such as RGB channels, so it is necessary to develop a new GP method applicable to these different images. Second, most of the existing GP-based methods have rarely been applied to large-scale image classification datasets, such as SVHN, CIFAR10, CIFAR100, and ImageNet. Third, the potential of GP with different operators to automatically learn models has not been fully investigated. The flexible representation allows GP to automatically build models using different types of simple and complex operators from different domains as functions (internal nodes) [20]. But there is still a lot of research potential in developing new representations, including GP tree structures, the function, and terminal sets.

Therefore, this article focuses on developing a new GP-based EDL approach to data-efficient image classification, where current NN methods cannot provide satisfactory performance due to the requirement of large training data. To achieve this, we will develop a new GP-based EDL approach (i.e., EDLGP) with a new model/solution representation, a new function set and a new terminal set, enabling it to automatically learn features from grayscale and/or color images, select classification algorithms, and build effective ensembles for image classification. We will examine the performance of the new approach on well-known image classification datasets including CIFAR10, Fashion_MNIST, and SVHN and two face image datasets under the scenario of small training data. EDLGP will be compared with a large number of existing competitive methods including CNNs of varying architectures and complexity. To highlight the potential of GP-based EDL methods, we perform comprehensive comparisons between EDLGP and CNNs for image classification. In addition, a deep analysis of convergence behavior, tree/model size, model interpretability, and transferability is conducted.

The main characteristics of the new EDLGP approach are summarized as follows.

1) The EDLGP approach can evolve variable-length tree-based symbolic models, achieving promising classification performance in the data-efficient scenario. These goals are difficult to address using the current popular NN-based methods due to the requirement of a large number of training images. The design of automatically evolving ensembles further enhance generalization, which is a key issue when the training set is small.

2) Compared with the existing methods, the EDLGP approach has a flexible multilayer model representation to automatically evolve shallow or deep models for different image classification tasks. With this representation, EDLGP can deal with grayscale and/or color images, extract image features via different processes, such as image filtering and complex feature extraction, perform classification by using cascading and ensembles with high diversity, and automatically select parameters for image operators and classification algorithms. Unlike NN-based methods, EDLGP evolves models with small numbers of parameters, which are easy to learn and rely less on the scale of the data.

3) The EDLGP approach can evolve models with high interpretability and transferability. This is important for providing insights into the problems being solved and also for future model reuse and development.

II. RELATED WORK

This section reviews related works on EDL including automatically evolving CNNs and GP for image classification and related works on data-efficient image classification. The limitations of these works are summarized.

A. Evolutionary Deep Learning

1) NN-Based EDL Methods: The methods that fall into this category typically use EC techniques to optimize deep NNs
for image classification by searching for architectures, weights, hyperparameters, etc. A detailed review of existing work can be found in [10], [13], and [14]. Sun et al. [21] proposed an evolutionary algorithm to automatically search for CNN architectures and connection weights for image classification. In [22], a GA-based method was proposed to automatically design block-based CNN architectures for image classification. Lu et al. [23] developed an EMO-based method (i.e., NSGA-II) to automatically search CNN architectures while optimizing multiple objectives. Zhu and Jin [24] proposed an EC-based neural architecture search (NAS) method under the real-time federated learning framework. The computational and communication costs are significantly reduced by using this method. Zhang et al. [25] developed a method based on sampled training and node inheritance to improve the computational efficiency of EC-based NAS for image classification.

2) GP-Based EDL Methods: Unlike NNs, GP can use a flexible tree-like symbolic representation to automatically evolve trees/models with different lengths to meet the needs of problem solving. For many problems, including image classification, GP can automatically discover the representation of the data by evolving models of appropriate complexity. Based on the definition of deep learning [5], [7], GP can be a deep learning method.

An early review on GP-based EDL methods is presented in [11], which mentions typical GP methods, such as 3-tier GP and multilayer GP for image classification. These methods can automatically evolve models, extract features from raw images and perform classification, similarly to CNNs. Rodriguez-Coayahuitl et al. [26] developed a GP autoencoder with structured layers to achieve image representation learning and introduced the concept of deep GP. Shao et al. [27] proposed a GP-based method that can use different filters and image operators to achieve feature learning for image classification. This method achieved better performance than CNNs on several datasets. Bi et al. [17] presented a series of GP-based methods, such as FLGP [28], FGP [20], and IEGP [29], for representation learning and image classification, where the evolved GP trees are constructed via multiple functional layers.

B. Data-Efficient Image Classification

Data-efficient image classification focuses on solving image classification with a small number of available training data, which has become an increasingly important topic in recent years. Bruintjes et al. [30] and Lengyel et al. [31] proposed the first and second well-known challenges on data-efficient deep learning, which aims to develop new deep learning methods using small training data to solve computer vision tasks including image classification, object detection, instance segmentation, action recognition, and reidentification. In [30] and [31], two important rules of solving data-efficient image classification are proposed, i.e., 1) models can only be trained from scratch using the training set of the task, 2) other data, transfer learning and model pretraining are prohibited.

Arora et al. [32] proposed a convolutional neural tangent kernel (CNTK) method for classifying CIFAR10 with 10 to 640 training images. The results showed that the CNTK methods with several layers can beat ResNet. Brigato and Iocchi [33] investigated different CNNs of varying complexity for image classification on small data. The results showed that dropout can improve the generalization of CNNs. Bi et al. [34] proposed a multiobjective GP method to simultaneously optimize classification performance and a distance measure to improve the generalization of the classification system using a small training set.

Barz and Denzler [35] proposed the well-known Cosine loss for training deep NNs using small datasets. By maximizing the cosine similarity between the outputs of the NNs and the one-hot vectors of the true class, this loss function gained better results than the commonly used cross-entropy loss on several image datasets. Brigato et al. [36] proposed eight data-efficient image classification benchmarks, including object classification, medical image classification, and satellite image classification. Several existing methods were tested on these benchmarks. The results showed that parameter tuning in terms of batch size, learning rate, and weight decay can improve the performance of the existing methods on the benchmark datasets. Sun et al. [37] proposed a visual inductive priors framework with a new neural network architecture for data-efficient image classification. A loss function based on the positive class classification loss and the intraclass compactness loss was developed to improve the generalization. This method ranked first in the first data-efficient image classification challenge [30]. Zhao and Wen [38] proposed a method with a two-stage manner to train a teacher network and a student network for data-efficient image classification. This method ranked second in the challenge [30].

C. Summary

Existing work shows the potential of GP for data-efficient image classification [18], [39], [40]. However, all these methods focus on relatively simple tasks and use grayscale images. Furthermore, very few GP-based methods have been tested on commonly used datasets, such as CIFAR10. The comparison between GP-based EDL methods and NN-based deep learning methods is not comprehensive enough to show the advantages of GP in solving data-efficient image classification. To address these limitations and further explore the potential of GP-based EDL methods, this article proposes a new data-efficient image classification approach based on GP and conducts comprehensive comparisons between the new approach and deep CNNs to demonstrate its effectiveness.

III. PROPOSED APPROACH

This section introduce the proposed EDLGP approach to image classification, including the overall algorithm, the individual representation, genetic operators, and fitness function. To highlight the advantages of the new approach, a detailed comparison between it and deep CNNs is presented.

A. Overall Algorithm

An image classification task often requires a training set and a test set. The training set is denoted as $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^m$
and the test set is denoted as \( D_{\text{test}} = \{ x_j \}_{j=1}^n \), where \( x \in \mathbb{R}^{G \times W \times H} \) denotes the images with a size of \( W \times H \) and \( G \) channels, and \( y \in \mathbb{Z} \) denotes the class label of the image (the total number of classes is \( C \)). The number of training images is \( m \) and the number of testing images is \( n \). The task is to use \( D_{\text{train}} \) to build a model \( g(x) \) that can effectively predict \( y \) for testing/unseen images \( D_{\text{test}} \).

The overall process of EDLGP for image classification is shown in Fig. 1. The input of the system is the training set \( D_{\text{train}} \) and the output is the best GP tree/model \( g(x) \). EDLGP starts by randomly initializing a number of tree-based models according to the predefined representation, function set, and terminal set. The population is evaluated using the fitness function, where each model/tree/individual is assigned a fitness value, i.e., the classification accuracy of the training set. During the evolutionary process, promising individuals are selected using a selection method and new individuals are generated using genetic operators, i.e., subtree crossover and subtree mutation. A new population is created by copying a proportion of individuals with the highest fitness values and by generating new individuals using the crossover and mutation operations. The process of population generation, fitness evaluation, and selection is repeated until reaching a predefined termination criterion. After the evolutionary process, the best GP tree/model is returned. The best tree is represented by \( g^*(x) \) and is used to classify images in the test set.

The optimization process of the EDLGP approach can be defined as follows:

\[
g^*(x) = \text{argmax}_{g \in \mathcal{F}, \; \ell \in \mathcal{T}} L(g(x), \; D_{\text{train}})
\]

where \( g(x) \) represents a GP tree/model and \( L \) represents the objective function. \( \mathcal{F} \) denotes the functions and \( \mathcal{T} \) denotes the terminals, which are employed to construct GP models/trees. The best model is learned by using EDLGP with a goal of maximizing \( L \) using the training set \( D_{\text{train}} \). Note that \( L \) can be a loss function as that in NNs to be minimized.

To find the best model \( g^*(x) \), a new representation, a new function set, and a new terminal set are developed in EDLGP. These new components allow EDLGP to automatically evolve variable-length models that extract informative image features and build effective ensembles of classifiers for classification.

### B. Components of EDLGP

The EDLGP approach has a new model representation/encoding, making it different from the existing GP methods [11], [17]. Furthermore, EDLGP uses genetic operators to create new populations of trees and a fitness function for evaluating the new models. These components are introduced as follows.

1) **New Model Representation:** The EDLGP approach uses a tree-based structure based on strongly typed GP [41] to represent the model for image classification. The tree structure and two example trees that can be evolved by EDLGP are shown in Fig. 2. Specifically, the tree structure consists of nine concept layers, i.e., input, image filtering, feature extraction, concatenation, classification and cascade, concatenation, classification, summation, and output. The input layer represents the inputs of a GP tree, such as raw images and the parameters of operators. The image filtering layer uses a set of commonly used image filters to process the input image. The feature extraction layer contains well-known image descriptors that can generate a set of features from images. The concatenation layers aim to concatenate the features generated from the previous layer to generate a more comprehensive feature set. The classification and cascade layer performs classification and cascade the predicted class labels or weights to the input feature vector to form a stronger feature set. This idea was borrowed from cascade learning [42]. The features generated by the classification and cascade layer can be further concatenated by a concatenation layer. The classification layer performs classification on the image using the features from its child nodes. The summation layer adds the predicted “probabilities” from all its child nodes/classifiers for each class. The
output layer performs the majority voting to make a prediction on which class the image belongs to, i.e., the class label.

The tree structure of EDLGP consists of flexible layers and fixed layers with a good balance between the model complexity necessary for dealing with an image classification task and the flexibility of the model growing in depth. Specifically, the input, feature extraction, classification, summation, and output layers are fixed, which follows a commonly used manner for building an ensemble for image classification. These layers must appear in each GP tree/model, but the functions used in those layers vary with trees/models. The flexible layers are the image filtering layer, the concatenation layer, and the classification and cascade layer, which are optional for constructing GP trees/models. These flexible layers allow EDLGP to generate more effective features by using different numbers of functions as internal nodes in the GP trees. This is important for dealing with difficult image classification tasks that require sufficiently complex data transformation to generate informative features.

There are four main characteristics of the new model representation. First, it allows EDLGP to evolve shallow and deep models for different tasks. A simple model may only have a few internal and leaf nodes, similar to the example tree shown on the middle part of Fig. 2. A deep model may be constructed by a large number of internal nodes, as the example tree shown on the right part of Fig. 2. Second, this representation allows EDLGP to construct ensembles of classifiers/ensembles for image classification, which enhances the effectiveness of the models when the number of training images is small. Third, this representation allows the constructed model to automatically learn effective image features with different processes, i.e., image filtering, feature extraction based on commonly used image descriptors, and cascading after classification, which are common in computer vision and machine learning. Fourth, EDLGP can evolve ensembles with high diversity for image classification. The ensembles include cascade classifiers and classifiers built using features extracted by the corresponding subtree. Their diversity ensures the effectiveness of the constructed ensembles. These characteristics make EDLGP significantly different from the existing GP-based methods for image classification.

2) Functions and Terminals: The functions represent the internal and root nodes of the new model representation. A simple model may only have a few internal and leaf nodes, similar to the example tree shown on the middle part of Fig. 2. A deep model may be constructed by a large number of internal nodes, as the example tree shown on the right part of Fig. 2. Second, this representation allows EDLGP to evolve ensembles of classifiers/ensembles for image classification, which enhances the effectiveness of the models when the number of training images is small. Third, this representation allows the constructed model to automatically learn effective image features with different processes, i.e., image filtering, feature extraction based on commonly used image descriptors, and cascading after classification, which are common in computer vision and machine learning. Fourth, EDLGP can evolve ensembles with high diversity for image classification. The ensembles include cascade classifiers and classifiers built using features extracted by the corresponding subtree. Their diversity ensures the effectiveness of the constructed ensembles. These characteristics make EDLGP significantly different from the existing GP-based methods for image classification.

Table I: Terminals

| Terminal | Description |
|----------|-------------|
| Red, Blue, Green | The red, blue, and green channels of the colour image. Each of them is a 2D array with values in range [0, 1] |
| Gray | The gray-scale image, which is a 2D array with the values in range [0, 1] |
| t | The number of trees in RF, ERF, CC_RF, and CC_ERF. It is an integer in range [50, 100] with a step of 50 |
| d | The maximal tree depth in RF, ERF, CC_RF, and CC_ERF. It is an integer in range [10, 100] with a step of 10 |
| f | The frequency of Gabor and Gabor_FE. Its value is in range $[\pi/8, \pi/2]$ with a step of $\pi/2\sqrt{2}$ |
| $\theta$ | The orientation of Gabor. Its value is in range $[0, \frac{\pi}{4}]$ with a step of $\frac{\pi}{4}$ |
| $o_1, o_2$ | The derivative orders of GauD and GauD_FE. It is an integer in range [0, 2] |
| $\sigma$ | The standard deviation of Gau and Gau_FE. It is an integer in range [1, 3] |
TABLE II

| Function  | Description                                      |
|-----------|--------------------------------------------------|
| Mean      | Perform 3 × 3 mean filtering                      |
| Median    | Perform 3 × 3 median filtering                    |
| Min       | Perform 3 × 3 minimum filtering                   |
| Max       | Perform 3 × 3 maximum filtering                   |
| Gau       | Perform Gaussian filtering and the standard deviation is σ |
| GauD      | Generate a new image by calculating derivatives of Gaussian filter with standard deviation σ and orders α1 and α2 |
| Lap       | Perform Laplacian filtering                       |
| LoG1      | Perform Laplacian of Gaussian filtering, and the standard deviation is 1 |
| LoG2      | Perform Laplacian of Gaussian filtering, and the standard deviation is 2 |
| Sobel     | Perform 3 × 3 Sobel filtering to the input image  |
| Gabor     | Perform Gabor filtering, where the orientation is θ and the frequency is f |
| LBP_P     | Generate a LBP image                              |
| HOG_F     | Generate an HOG image                              |
| Sqrt      | Return sqrt root value of each value of the input image. Return 1 if the value is negative |
| ReLU      | Perform the operation using rectified linear unit on the input image |
| Add_MaxP  | Perform 2 × 2 max-pooling on two input images or the smaller image and add the two images |
| Sub_MaxP  | Perform 2 × 2 max-pooling on two input images or the smaller image and subtract the two images |

Fig. 3. Illustration of image filtering and the images obtained by applying some functions.

also include the operators that can process an image, such as HOG_F, LBP_F, Sqrt, and ReLU. In addition, the functions include Add_MaxP and Sub_MaxP, which take two images as inputs, perform 2 × 2 max pooling to the images, and add and subtract the two small images to generate a new image, respectively. If the size of the two input images is not the same, the two functions will perform max pooling only on the small image to make the size the same before sum or subtraction. This is possible since the size of images can only be reduced by using the 2 × 2 max-pooling operation in the function set. By using these different operators, we expect to generate more effective features from images. Table II lists all functions and the process is illustrated in Fig. 3.

The feature extraction layer contains 11 functions listed in Table III, i.e., cascade random forest (CC_RF), cascade extremely randomized trees (CC_ERF), cascade logistic regression (CC_LR), and cascade support vector machine (CC_SVM), which perform classification and concatenate the predicted class probabilities with the input features to form the output features, as shown in Fig. 6. This follows the concept of cascade ensemble learning [7]. The prediction of an instance is a C-dimensional vector denoting the probabilities for a C-class problem. If the number of features is fn, the number of output features from these functions will be fn + C. Unlike CC_RF, CC_ERF, and CC_LR, which are soft classifiers, CC_SVM is a hard classifier so that the prediction vector is binary (discrete) rather than continuous. To build effective classifiers, the main parameters of CC_RF and CC_ERF, i.e., the number of trees and the maximal tree depth, are set as terminals and their values can be automatically selected by EDLGP.

The classification layer contains functions, i.e., RF, ER, LR, and SVM, which take features as inputs and return the predicted class probabilities. The summation layer has three functions, i.e., Sum2, Sum3, and Sum4, which take two, three and four vectors of predicted probabilities as inputs and add them, respectively. Each vector denotes the probabilities of all
313

Fig. 6. Illustration of classification and cascade.

Table IV

| Function | Description |
|----------|-------------|
| CC_RF, | Use the RF, ERF, LR, and SVM methods to perform classification. The outputs of these functions are the concatenation of the input features and the predicted class probabilities |
| CC_ERF, | Random forest classification method. The number of trees is t and the maximal tree depth is d |
| CC_LR, | Extremely randomized trees classification method. The number of trees is t and the maximal tree depth is d |
| CC_SVM | Logistic regression classification method |
| SVM | Support vector machine classification method |
| Sum2/3/4 | Sum 2/3/4 vectors of predicted probabilities |

Fig. 7. Illustration of crossover operation.

classes an instance belongs to. These functions are listed in Table IV. The output layer can make predictions according to the output of GP trees by assigning the class label with the highest probability to the input image.

Besides the above functions/operators, it is possible to use other different functions at the corresponding layer in EDLGP. In other words, EDLGP can be easily extended by using more functions. The representation and the search mechanism allow EDLGP to automatically select the functions and terminals to build trees/models. However, it is suggested to carefully set the function set to keep a good balance between the effectiveness of the potential models and the search space dimension.

3) Genetic Operators: During the evolutionary process, new GP trees are generated using genetic operators, i.e., subtree crossover and subtree mutation. The subtree crossover operator is conducted on two selected trees/parents. It randomly selects two subtrees from the parents and swaps the two subtrees to generate two new trees, as shown in Fig. 7. Note that the two selected subtrees must have the same output types in order to generate two feasible trees. The subtree mutation operator is conducted on one selected tree/parent. It randomly selects a subtree of the parent and replaces the subtree with a randomly generated subtree, as shown in Fig. 8.

4) Fitness Evaluation: A fitness function is used in the fitness evaluation process to evaluate the fitness of the GP trees/models to the tasks. For a classification task, the most commonly used fitness function is the classification accuracy to be maximized, which is defined as

\[ \mathcal{L} = \frac{N_{\text{correct}}}{N_{\text{total}}} \times 100\% \]

where \( N_{\text{correct}} \) denotes the number of correctly classified instances and \( N_{\text{total}} \) denotes the total number of instances in the training set. The fitness function is calculated using k-fold cross-validation on the training set. In EDLGP, the value of \( K \) is set to \( \min(3, nc) \), where \( nc \) denotes the number of instances in the smallest class of the training set. If \( nc = 1 \), the fitness function will be the training accuracy. Note that this fitness function is effective for balanced classification. For unbalanced classification, other measures such as balanced accuracy can be used as the fitness function for EDLGP.

C. Comparisons Between EDLGP and CNNs

The proposed EDLGP approach can automatically learn features and evolve ensembles for image classification. To highlight the characteristics of EDLGP, this section provides detailed analysis and comparisons between EDLGP (as a typical example of GP-based EDL methods) and CNNs, which is a dominant deep learning method for image classification.

The main similarities are summarized as follows.
1) CNNs and EDLGP can automatically learn features and perform classification from raw images.
2) CNNs and EDLGP are learning algorithms that learn models of many layers, which can perform complex data transformation to achieve good performance.
3) CNNs and EDLGP use image operations, such as convolution and pooling to learn features from raw images. The main differences are summarized as follows.
1) EDLGP uses a variable-length tree-based structure to represent a model, while CNNs typically uses a fixed-length layerwise structure to represent a model. The flexible representation allows EDLGP to automatically learn models with different shapes, sizes, and depths. In contrast, CNNs need to predefine the model structure and length before training.
2) EDLGP is open to incorporate domain knowledge in a way of adding functions/operators to the corresponding layers, which cannot be easily achieved in CNNs.
3) EDLGP has fewer model parameters and running parameters than CNNs. Detailed comparisons in terms of the running parameters are listed in Table V. The number of CNN model parameters depends on the architecture configuration, which is often much larger than that of the automatically evolved EDLGP models.
4) The models evolved by EDLGP have potentially higher interpretability than CNN models. The EDLGP model is composed of functions/operators from the image and
TABLE V

COMPARISONS OF RUNNING PARAMETERS OF EDLGP AND CNNs

| Convolutional neural networks | Evolutionary deep learning based on genetic programming |
|-------------------------------|--------------------------------------------------------|
| Type of activation functions: | Type of selection:                                      |
| ReLU, tanh, sigmoid, etc.     | Tournament selection or Roulette wheel selection       |
| Architecture configurations:  | Architecture configurations:                            |
| No. hidden layers             | Function set                                           |
| No. convolutional layers      | Terminal set                                           |
| No. pooling layers            | Optimisation configurations:                           |
| No. other layers              | No. generations                                        |
| How to connect these layers   | Population size                                        |
| No. features maps/pixels      | Probability of elitism/reproduction                    |
| Kernel sizes                  | Probability of mutation                                |
| Optimisation configurations:  | Probability of crossover                               |
| Learning rate                 | minimal and maximal tree size                          |
| Dropout: 0.25/0.50            | selection size                                         |
| Momentum                      | Tree generation method: full, grow, ramped-half-and-half |
| L1/L2 weight regularisation   |                                                        |
| penalty                       |                                                        |
| Weight initialisation: uniform, Xavier Glorot, etc. |                                                        |
| Batch size: 32/64/128         |                                                        |

The CNN model is composed of layers of a huge number of parameters, which are not straightforward to explain.

5) The feature extraction and classification processes of CNNs and EDLGP are different. CNNs usually extract and construct features via a number of convolutional and pooling layers with learned kernels, and perform classification using softmax. EDLGP extracts features using image operators with predefined kernels and uses a traditional classification algorithm to perform classification. This makes CNNs and EDLGP applicable and effective for image classification under different scenarios. CNNs requires sufficient training data to learn effective feature maps and subsampling maps in these predefined layers to achieve promising performance, i.e., suitable for large-scale image classification. The EDLGP approach is more effective than CNNs when the training data is small, i.e., data-efficient image classification.

6) The CNN methods can greatly benefit from running on GPUs, while EDLGP would not at the current stage.

IV. EXPERIMENT DESIGN

This section describes the experiment design, including datasets, the comparison methods, and parameter settings.

A. Image Classification Datasets

To evaluate the effectiveness of the new approach, five different datasets are used, which are CIFAR10 [46], CIFAR10 [36], Fashion_MNIST (FMNIST in short) [47], SVHN [48], ORL [49], and Extended Yale B [50]. The CIFAR10, FMNIST, and SVHN datasets are object classification tasks. The ORL and Extended Yale B datasets are face classification tasks. The example images of these datasets are shown in Figs. 9 and 10. Table VI lists the detailed information of these datasets.

The CIFAR10 dataset is composed of 50,000 training images and 10,000 testing images. The images are color and of size $32 \times 32$. The CIFAR10 dataset is a data-efficient benchmark and is a variant of CIFAR10 by removing redundant testing images from the original test set of CIFAR10. The Fashion_MNIST dataset has 60,000 training images and 10,000 testing images in grayscale. The image size is $28 \times 28$. The SVHN dataset contains 73,257 color images for training and 26,032 color images for testing. The image size is $32 \times 32$. The ORL dataset consists of 400 facial images from 40 people, i.e., ten images per person. The Extended Yale B dataset has 2,424 facial images in 38 classes.

The EDLGP approach is examined on data-efficient image classification problems. A small number of training images are used in the experiments. For the CIFAR10, Fashion_MNIST and SVHN datasets, 10, 20, 40, 80, 160, 320, and 1,280 images are randomly selected from each class in the original training set to form a small training set used in the experiments, following the settings in [33]. The small datasets are called sCIFAR10-X, sFMNIST-X, and sSVHN-X, where X denotes the number of training images in each class. To comprehensively show the performance, we also test EDLGP using very small training sets of CIFAR10, i.e., 1, 2, 4, 8, 16, 32, 64, and 128 images per class, following the settings in [32]. In addition, we also test the proposed approach on CIFAR10.
with 100, 500, 600, and 1000 training images in order to compare with other existing methods. For the cifAR10 dataset, the training set size is 500 and the test set size is 10,000. In all these above settings, the original test sets are used for testing. For ORL, 2, 3, 4, and 5 images are randomly selected to form the training sets and the remaining images are used for testing in the experiments [51], respectively. For Extended Yale B, 15, 20, 25, and 30 images are randomly selected for training and the rest are used for testing [42], respectively. As a result, a large number of experiments are conducted to comprehensively demonstrate the performance of EDLGP in comparison with different deep learning methods.

B. Comparison Methods

We compare EDLGP with existing methods, including CNNs with different architectures, state-of-the-art methods, and non-NN-based deep learning methods, which are very effective methods on the corresponding datasets.

1) On sCIFAR10, sFMNIST, and sSVHN, the comparison methods are CNN-lc [33], CNN-mc [33], CNN-hc [33], and ResNet-20 [52]. CNN-lc, CNN-mc, and CNN-hc denote different CNN methods with low, middle, and high complexity used for image classification. Different Dropout rates are investigated in these methods and the details can be found in [33].

2) On few-shot sCIFAR10, the reference methods are ResNet [52], Harmonic Networks [53], and CNTK networks with 5, 8, 11, and 14 layers, respectively, [32].

3) On CIFAR10, the reference methods include VGG [54], WRN 16-8 [55], Scat+i-WRN [56], CAE [57], and EvoVAE [58].

4) On cifAR10, the reference methods are data-efficient methods, including the cross-entropy baseline, deep hybrid networks [56], OLÉ [59], Grad-t2 Penalty [60], Cosine Loss [35], Cosine Loss + Cross-Entropy [35], Full Convolution [61], Dual Selective Kernel Networks [37], and T-vMF Similarity [62].

5) On ORL, the reference methods are Eigenfaces based on PCA [63], Fisherfaces based on LDA [64], Laplacianfaces [65], neighborhood preserving embedding (NPE) [66], marginal Fisher analysis (MFA) [67], CNN [51], and deep metric learning based on CNN and k nearest neighbor classification (KCNN) [51].

6) On Extended Yale B, the reference methods are collaborative representation classifier (CRC) [68], SRC [69], corentropy-based sparse representation (CESR) [70], robust sparse coding (RSC) [71], half-quadratic with the additive form (HQA) [72], half-quadratic with the multiplicative form (HQM) [72], NMR [73], robust matrix regression (RMR) [74], Fisher discrimination dictionary learning (FDDL) [75], low-rank shared dictionary learning (LRSDL) [76], DCM based on NMR (DCM(N)) [42], and DCM based on SRC (DCM(S)) [42].

The parameter settings of these methods refer to the corresponding references. The other experimental settings refer to [33], [42], and [51]. The aforementioned methods are used for comparisons because they are effective methods on these datasets using small training sets.

C. Parameter Settings

The parameter settings for the EDLGP approach follow the commonly used settings in the GP community [17], [77]. The maximal number of generations is 50 and the population size is 100. The crossover rate is 0.5, the mutation rate is 0.49, and the elitism rate is 0.01. The selection method is tournament selection with size 5. The population initialization method is ramped-half-and-half. The initial tree depth is 2–10. The maximal tree depth is 10. Note that these parameter values for EDLGP can be fine-tuned on each dataset for optimal settings. However, our study aims to investigate a more general approach that can achieve a good performance using a common setting. In the experiments, EDLGP is executed ten times independently on each dataset using different random seeds because of the stochastic nature and the high computation cost.

V. RESULTS AND DISCUSSION

This section compares the performance of the EDLGP approach with the reference methods on different datasets of various numbers of training images. This section also deeply analyses convergence behaviors of EDLGP and interpretability of the evolved models/trees. Due to the page limit, the analysis of transferability of GP models/trees is presented in the supplementary materials.

A. Classification Accuracy

Overall Performance: In total, there are 45 cases/experiments (i.e., 3 × 8 on sCIFAR10, sFMNIST and sSVHN, 8 on few-shot sCIFAR10, 4 on CIFAR10, 1 on cifAR10, and 2 × 4 on ORL and Extended Yale B) to compare the performance EDLGP with different CNNs and existing methods. All the test results are listed in Tables VII–XII. In total, EDLGP achieves better mean accuracy in over 20 cases among all the existing methods, indicating that EDLGP is effective on these different image classification tasks by automatically evolving variable-length/depth models. Note that the reference methods are CNNs or other effective methods (such as Eigenfaces, KCNN, DCM, and RSC) and their results are from the corresponding references. Compared with these methods, EDLGP achieves a better performance on the ORL and Extended Yale B datasets in all scenarios. On sCIFAR10, sFMNIST, sSVHN, and few-shot CIFAR10, EDLGP achieves better or competitive performance when the training set is small, and slightly worse performance when the size of the training set is large. More importantly, EDLGP achieves a very high maximal accuracy in most cases. Compared with state-of-the-art data-efficient deep learning methods, EDLGP is slightly worse on CIFAR10, which is reasonable since these reference methods are very mature methods with many effective designs. Overall, EDLGP is promising for data-efficient image classification, particularly if the training set is very small.

Results on sCIFAR10, sFMNIST, and sSVHN: The classification accuracies on the test sets are listed in Table VII.
TABLE VII
MEAN CLASSIFICATION ACCURACY (%) OF CNNs OF VARYING COMPLEXITY AND CLASSIFICATION ACCURACY (%) OF EDLGP ON SMALL CIFAR10, FASHION-MNIST, AND SVHN DATASETS OF VARYING TRAINING SET SIZES. NOTE THAT THE REFERENCES HAVE ONLY REPORTED THE AVERAGED ACCURACY OF CNNs

| Model          | Dropout | sCIFAR10-10 | sCIFAR10-20 | sCIFAR10-40 | sCIFAR10-80 | sCIFAR10-160 | sCIFAR10-320 | sCIFAR10-640 | sCIFAR10-1280 |
|----------------|---------|-------------|-------------|-------------|-------------|--------------|--------------|--------------|---------------|
| CNN-lc         | 0.0     | 27.1        | 32.4        | 36.1        | 41.8        | 45.3         | 50.1         | 54.1         | 59.4          |
| CNN-lc         | 0.4     | 29.7        | 33.8        | 38.2        | 43.5        | 47.1         | 52.8         | 57.2         | 61.7          |
| CNN-lc         | 0.7     | 29.7        | 34.9        | 40.0        | 44.9        | 49.4         | 53.9         | 58.1         | 62.3          |
| CNN-mc         | 0.0     | 28.5        | 34.3        | 38.8        | 43.0        | 48.6         | 53.7         | 58.8         | 63.3          |
| CNN-mc         | 0.4     | 29.7        | 34.9        | 39.9        | 45.4        | 50.9         | 55.8         | 61.0         | 66.2          |
| CNN-mc         | 0.7     | 31.5        | 36.2        | 41.3        | 47.1        | 51.9         | 57.7         | 62.8         | 67.6          |
| CNN-he         | 0.0     | 30.1        | 34.2        | 39.1        | 44.7        | 50.6         | 56.1         | 61.2         | 65.9          |
| CNN-he         | 0.4     | 31.7        | 36.1        | 40.8        | 46.5        | 52.0         | 58.0         | 63.5         | 68.8          |
| CNN-he         | 0.7     | 31.9        | 37.0        | 42.5        | 48.1        | 53.9         | 59.5         | 64.8         | 69.8          |
| ResNet-20      | -       | 23.3        | 29.0        | 31.9        | 38.5        | 44.7         | 51.3         | 62.3         | 71.5          |
| EDLGP(best)    | -       | 35.8        | 43.2        | 48.1        | 58.2        | 60.6         | 64.8         | 64.3         | -             |
| EDLGP(mean)    | -       | 31.7±3.8    | 37.7±4.0    | 45.7±1.6    | 49.3±1.8    | 54.3±2.5     | 58.7±1.5     | 61.6±2.0     | 61.1±3.4      |

Note that the test sets are the original one of CIFAR10, Fashion-MNIST, and SVHN and the results of these reference methods are from [33]. The results of CNNs are the mean accuracy of ten runs. The standard deviation value has not been reported. On sCIFAR10, EDLGP achieves the best mean accuracy among all 11 methods including ResNet-20 in four cases out of eight cases and results that are only worse than CNN-hc or ResNet-20 in most of the remaining cases. It should be noted that EDLGP achieves very high maximal accuracy in most cases. Accuracy is much higher than the best one of all reference methods on sCIFAR10, e.g., 6.2% higher (43.2% versus 37.0%) on sCIFAR10–20, 5.5% (48.1% versus 42.6%) on sCIFAR10–40, 4.2% (52.3% versus 48.1%) on sCIFAR10–80, and 4.3% (58.2% versus 53.9%) on sCIFAR10–160. The results further demonstrate that EDLGP is very effective for data-efficient image classification. On Fashion-MNIST, EDLGP achieves the best results among all 11 methods including ResNet-20 in five cases out of nine cases and results that are only worse than CNN-hc or ResNet-20 in most of the remaining cases. Accuracy is much higher than the best one of all reference methods on Fashion-MNIST, e.g., 4.9% higher (77.4% versus 72.5%) on Fashion-MNIST–10, 4.7% (82.1% versus 77.4%) on Fashion-MNIST–20, 4.6% (83.5% versus 78.9%) on Fashion-MNIST–40, 4.4% (85.9% versus 81.5%) on Fashion-MNIST–80, and 4.3% (88.6% versus 84.3%) on Fashion-MNIST–160. The results further demonstrate that EDLGP is very effective for data-efficient image classification.
on sFMNIST-20 and sFMNIST-40. In the remaining cases, EDLGP achieves slightly worse accuracy than the best mean accuracy. The best accuracy of EDLGP is very close to the best mean accuracy of all the compared methods, i.e., worse by 0.1% on sFMNIST-10, 1.5% on sFMNIST-160, 0.6% on sFMNIST-640, and 1.5% on sFMNIST-1280. The results show that EDLGP is effective for classifying the sFMNIST dataset.

On sSVHN, EDLGP achieves the best results on sSVHN-10, sSVHN-20 and sSVHN-40. The maximal accuracy obtained by EDLGP is much higher than all reference methods, i.e., 27.9% higher on sSVHN-10, 19.5% on sSVHN-20, and 8.1% on sSVHN-40. With the increasing of training data, some CNNs and ResNet-20 methods tend to achieve better classification accuracy and the accuracy gap between EDLGP and the best CNN becomes smaller. On sSVHN-640 and sSVHN-1280, EDLGP achieves worse results than most reference methods but the difference is not very big. From the results in Table VII, we can see that EDLGP is typically more effective than CNNs with low complexity but less accurate than CNNs with high complexity, particularly when the training set becomes large. CNNs with high complexity contain more parameters, therefore, a large number of training data are needed to effectively train the models. Unlike CNNs, EDLGP can be a good alternative when the training set is small because it is significantly less data intensive and can achieve a good performance. EDLGP automatically evolves tree-based models of functions/operators that have significantly fewer parameters and does not require a large number of training images.

Results on Few-Shot sCIFAR10: Table VIII shows the results on few-shot sCIFAR10 with 1, 2, 4, 8, 16, 32, 64, and 128 training images per class, respectively. The mean accuracy of these methods except for Harmonic Networks is reported. EDLGP achieves better or similar performance than five methods except for Harmonic Networks in most cases, i.e., better on sCIFAR10–16, sCIFAR10–32, sCIFAR10–64, and sCIFAR10–128, and a similar performance in the remaining cases. Harmonic Networks is the best data-efficient deep learning method, achieving a better performance than several well-known methods [53]. Compared with Harmonic Networks, EDLGP achieves a better performance in four cases. Particularly, EDLGP is more effective than Harmonic Networks when the training set is extremely small. Harmonic Networks are more effective when the training set is large.

To sum up, in the few-shot settings, both the performance of CNN and EDLGP increase with the number of training images. EDLGP is less sensitive to the dataset size and is still very competitive when the training set is extremely small.

Results on CIFAR10 and Comparisons With State-of-the-Art Deep Learning Methods: EDLGP is compared with state-of-the-art deep learning methods on CIFAR10 of different training set sizes. The results are shown in Table IX. EDLGP achieves a better performance than VGG and WRN 16-8 in the cases using 100 and 500 training images and worse performance in the case using 1000 training images. This shows that EDLGP is more effective than VGG and WRN 16-8 using a small training set. EDLGP achieves a better performance than CAE and EvoCAE in all four cases, showing that EDLGP is more effective than autoencoder-based methods under the data-efficient scenario. EDLGP is worse than Scat+WRN in all cases, but the gap in the mean accuracy is small. There is still possibility for GP to achieve a better performance than CNNs in the future under the data-efficient scenario.

Results on a Data-Efficient Benchmark ciFAR10 and Comparisons With Major Data-Efficient Deep Learning Methods: EDLGP is compared with all the existing data-efficient deep learning methods on the ciFAIR10 dataset. The classification accuracy of these methods is presented in Table X. Clearly, EDLGP achieves worse performance than these compared methods on ciFAIR10, i.e., about 7%–14% less accurate. EDLGP is perhaps more effective when the training set is extremely small, while this dataset uses 500 training images. Another point is that EDLGP is an EC algorithm
achieving deep learning that is a relatively new paradigm, while these DNN-based methods are very mature methods with many efforts devoted to the algorithm designs in terms of neural network architectures, loss functions, training strategies, etc. Therefore, it is not surprising that EDLGP cannot outperform these methods at the current stage.

Results on ORL and Extended Yale B: On the face datasets, EDLGP achieves higher maximum and average accuracies than any of the reference methods in all scenarios. Specifically, EDLGP improves the average accuracy by 8.8% on ORL-2, 10.3% on ORL-3, 6.7% on ORL-4, and 4.7% on ORL-5. Furthermore, compared with these seven methods, EDLGP has a smaller standard deviation, which means EDLGP is more stable. On Extended Yale B, EDLGP achieves over 99% accuracy in all scenarios. Among all the reference methods, EDLGP achieves the best accuracy on Extended Yale B, which means that EDLGP is more accurate than the other 12 methods. EDLGP is effective for face image classification using few training images.

To summarize, EDLGP achieves a promising performance in image classification with small training data. Unlike those reference methods, EDLGP can simultaneously and automatically search for the best feature extraction methods and ensemble models to achieve effective classification. The learned EDLGP models have fewer parameters than CNNs so that they can be better trained on small training data and achieve higher classification accuracy. A large number of comparisons show that EDLGP is more data efficient than standard CNNs and other well-known methods on different types of image classification tasks. However, the performance of EDLGP is worse than state-of-the-art deep learning methods. This is reasonable since those methods are very mature methods with many advanced designs, while the proposed EDLGP method can be further improved by designing effective fitness functions, representations and search operators. This article will be a starting point of showing the potential of GP-based methods under the data-efficient image classification scenario. In the future, we hope more advanced GP-based methods can be investigated and explored for data-efficient image classification.

B. Running and Classification Time

We take the few-shot sCIFAR10-X (X ranges from 1 to 128) dataset as a typical example to analyze the running and classification time of EDLGP. The running time of EDLGP on the other datasets is analyzed in the supplementary materials due to the page limit. Fig. 11 shows the average running time and classification time of EDLGP on a single CPU. The running time of EDLGP is less than 10 h when the number of training images per class is smaller than 16. The running time of EDLGP increases gradually with the number of training images. On sCIFAR10–128, it uses more than 90 h to complete the evolutionary learning process. The running time of EDLGP is clearly longer than that of CNNs, although we did not directly compare them. EDLGP is a population-based search algorithm currently implemented using the DEAP package running on CPUs. In contrast, CNNs can benefit from a fast computing facility—GPU, thus, they use a shorter running time. The running time of EDLGP can be accelerated by using a GPU implementation or running it in parallel on multiple CPUs.

The classification/test time of EDLGP on sCIFAR10–X is fast, i.e., less than 4 min in all scenarios. In the test process, EDLGP trains the model found via evolution to classify 10 000 images. The GP model complexity is the main factor that affects the classification time. From Fig. 13, EDLGP may learn a more complex model on sCIFAR10–64 and sCIFAR10–128 than the other scenarios, as it uses a longer classification time. The analysis of the model size/complexity will be conducted in the following section. Overall, EDLGP needs a reasonably long running time to complete the evolutionary process but uses a short classification time.

C. Convergence Behaviors

We take the ORL and Extended Yale B datasets as examples to show the convergence behavior of EDLGP. Note that its convergence behavior on other datasets shows similar patterns. Fig. 12 shows that EDLGP has good search ability and can converge to a high fitness value (i.e., accuracy on the training set) after 50 generations. Using different training set sizes, EDLGP achieves different fitness values during evolution. Specifically, more training data corresponds to higher fitness values of EDLGP on these two datasets. This is because a small training set may lead to underfitting while increasing the training set size can improve the fitness value (the accuracy). To sum up, EDLGP has good search ability and convergence, and can find the best model through evolution.
D. Interpretability of Trees/Models

We analyze model size/complexity and visualize data to show the high interpretability of models learned by EDLGP.

Tree/Model Size: The average tree size of EDLGP on sCIFAR10, ORL, and Extended Yale B is shown in Fig. 13. On sCIFAR10 with 1–128 training images, the average tree size ranges from 42.6 to 73.8. EDLGP tends to gradually increase its tree size with the number of training images on sCIFAR10, which is a difficult task. On the ORL and Extended Yale B datasets, the average tree size ranges from 35.5 to 62. In case 1 (two training images on ORL and 15 training images on Extended Yale B), EDLGP has a small initial fitness value (as shown in Fig. 12) and seems to improve the tree quality by gradually increasing the tree size. When using more training images on these two datasets (i.e., in cases 3 and 4), EDLGP reaches a higher fitness value in initial generations and finds relatively smaller trees than that in case 1.

Tree/Model Visualization: Fig. 14 shows an example tree of EDLGP on sCIFAR10–80. This tree is an ensemble of four ERF ensemble classifiers. Each branch can build one ERF classifier using different features generated from the corresponding child nodes and different parameter settings (i.e., number of decision trees and maximal tree depth). It processes images using Min, Sub_MaxP, Gabor, Sqrt, and HOG_F operators on the input image in the gray, red, blue, and green channels, and extracts features using SIFT, LBP, Gau_FE, Conca, HOG_FE, and Sobel_FE operators. CIFAR10 is a complex object classification dataset so that different types of features need to be extracted to improve the classification performance. By using such a model, EDLGP achieves 52.26% test accuracy, which is the best accuracy among all the methods on sCIFAR10 using only 80 training images per class. More importantly, the size of this tree is much smaller than in the case of the CNNs.

To further demonstrate the interpretability of models/trees evolved by EDLGP, the small tree in Fig. 15 is analyzed. It achieves 86.7% test accuracy on sFMNIST-640. In that figure, we use T-SNE [78] to visualize the data (i.e., features) generated by each internal node of the tree using T-SNE [78]. Each plot is drawn using 100 testing images per class of sFMNIST and each color represents one class. The functions at each node modify the input images/features significantly. The data generated by the high-level nodes are more clustered. At the output node, the data generated from the Sum2 node are better clustered, which can reveal why this model can achieve high classification accuracy.

To sum up, EDLGP evolves trees with different lengths, shapes, and depths on different datasets. A single GP tree is a model that can perform image feature extraction and classification using ensembles of classifiers. The functions/internal nodes of GP trees can make important transformations on the data to generate better ones. The GP trees are easy to visualize and some insights can be gained from them.

E. Summary

The following observations on EDLGP as a typical example of GP-based EDL methods are summarized.

1) EDLGP can achieve a better or competitive performance than the compared methods in most cases on CIFAR10, Fashion_MNIST, SVHN, ORL, and Extended Yale B with small training set. This shows that EDLGP is an effective approach to data-efficient image classification.
2) EDLGP shows good convergence and can find trees with different shapes, sizes, and depths on different datasets. Compared with CNN-based image classification methods, EDLGP does not require that one manually tune/design/determine the model architectures and/or coefficients/parameters.

3) EDLGP can evolve small and easily interpretable trees and achieve high accuracy. The evolved EDLGP trees are composed of image and classification domain operators, which are interpretable and provide more insights into the tasks. This is not easy to achieve using CNNs with many parameters.

4) The trees evolved by EDLGP show high transferability, facilitating model reuse, and development in the future.

VI. CONCLUSION

This article proposed a GP-based EDL method, i.e., EDLGP, for automatically evolving variable-length models for image classification on small training data. A new representation was developed that includes a multilayer tree structure, a function set, and a terminal set, enabling EDLGP to build models to perform feature extraction, concatenation, classification, and cascade, and ensemble construction, automatically and simultaneously. The EDLGP approach has shown a great potential in solving data-efficient image classification by achieving a better or competitive performance than many state-of-the-art methods. A detailed analysis showed that the models learned by EDLGP have good convergence, high interpretability, and good transferability. Compared with the existing popular CNN-based image classification methods, the EDLGP approach as a GP-based EDL approach has a number of advantages: it does not need any expensive GPU devices to run and rich domain expertise to design/tune model architectures/coefficients; it is data efficient; finally, it can evolve variable-length models with high interpretability and transferability.

This article is a starting point of the superiority of GP in comparisons with CNN-based methods for data-efficient image classification. The potential of GP-based EDL methods can be further explored in terms of developing effective representations, fitness measures, and search mechanisms in the future. Furthermore, the transferability of the models/trees evolved by GP is also a good point to be deeply investigated.

REFERENCES

[1] J. Schindelin et al., “Fiji: An open-source platform for biological-image analysis,” Nat. Methods, vol. 9, no. 7, pp. 676–682, 2012.
[2] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri, “NIH image to ImageJ: 25 years of image analysis,” Nat. Methods, vol. 9, no. 7, pp. 671–675, 2012.
[3] D. Lu and Q. Weng, “A survey of image classification methods and techniques for improving classification performance,” Int. J. Remote Sens., vol. 28, no. 5, pp. 823–870, 2007.
[4] Y. Bi, B. Xue, P. Mesejo, S. Cagnoni, and M. Zhang, “A survey on evolutionary computation for computer vision and image analysis: Past, present, and future trends,” 2022, arXiv:2209.06399.
[5] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.
[6] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[7] Z.-H. Zhou and J. Feng, “Deep forest,” Nat. Sci. Rev., vol. 6, no. 1, pp. 74–86, Oct. 2018.
[8] L. Alzubiadli et al., “Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions,” J. Big Data, vol. 8, no. 1, pp. 1–74, 2021.
[9] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, “PCANet: A simple deep learning baseline for image classification,” IEEE Trans. Image Process., vol. 24, pp. 5017–5032, 2015.
[10] Z.-H. Zhan, J.-Y. Li, and J. Zhang, “Evolutionary deep learning: A survey,” Neurocomputing, vol. 483, pp. 42–58, Apr. 2022.
[11] H. Al-Sahaf et al., “A survey on evolutionary machine learning,” J. Royal Soc. New Zealand, vol. 49, no. 2, pp. 205–228, 2019.
[12] T. Bäck, D. B. Fogel, and Z. Michalewicz, Handbook of Evolutionary Computation, vol. 97, Bristol, U.K.: IOP Publ., 1997, p. B1.
[13] X. Zhou, A. K. Qin, M. Gong, and K. C. Tan, “A survey on evolutionary construction of deep neural networks,” IEEE Trans. Evol. Comput., vol. 25, no. 3, pp. 894–912, Oct. 2021.
[14] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, “A survey on evolutionary neural architecture search,” IEEE Trans. Neural Netw. Learn. Syst., early access, Aug. 6, 2021, doi: 10.1109/TNNLS.2021.3100554.
[15] A. D. Martínez et al., “Lights and shadows in evolutionary deep learning: Taxonomy, critical methodological analysis, cases of study, learned lessons, recommendations and challenges,” Inf. Fusion, vol. 67, pp. 161–194, Mar. 2021.
[16] W. B. Langdon and R. Poli, Foundations of Genetic Programming, Heidelberg, Germany: Springer, 2013.
[17] Y. Bi, B. Xue, and M. Zhang, Genetic Programming for Image Classification: An Automated Approach to Feature Learning, Cham, Switzerland: Springer, 2021.
[18] Y. Bi, B. Xue, and M. Zhang, “Dual-tree genetic programming for few-shot image classification,” IEEE Trans. Evol. Comput., vol. 26, no. 3, pp. 555–569, Jun. 2022.
[19] Y. Bi, B. Xue, and M. Zhang, “Multi-objective genetic programming for feature learning in face recognition,” Appl. Soft Comput., vol. 103, May 2021, Art. no. 107152.
[20] Y. Bi, B. Xue, and M. Zhang, “Genetic programming with image-related operators and a flexible program structure for feature learning in image classification,” IEEE Trans. Evol. Comput., vol. 25, no. 1, pp. 87–101, Feb. 2021.
[21] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, “Evolving deep convolutional neural networks for image classification,” IEEE Trans. Evol. Comput., vol. 24, no. 2, pp. 394–407, Apr. 2020.
[22] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, “Completely automated CNN architecture design based on blocks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 31, no. 4, pp. 1242–1254, Apr. 2020.
[23] Z. Lu et al., “Multiobjective evolutionary design of deep convolutional neural networks for image classification,” IEEE Trans. Evol. Comput., vol. 25, no. 2, pp. 277–291, Apr. 2021.
[24] H. Zhu and Y. Jin, “Real-time federated evolutionary neural architecture search,” IEEE Trans. Evol. Comput., vol. 26, no. 2, pp. 364–378, Apr. 2022.
[25] H. Zhang, Y. Jin, R. Cheng, and K. Hao, “Efficient evolutionary search of attention convolutional networks via sampled training and node inheritance,” IEEE Trans. Evol. Comput., vol. 25, no. 2, pp. 371–385, Apr. 2021.
[26] L. Rodríguez-Coayahuitl, A. Morales-Reyes, and H. J. Escalante, “Structurally layered representation learning: Towards deep learning through genetic programming,” in Proc. EuroGP, 2018, pp. 271–288.
[27] L. Shao, L. Liu, and X. Li, “Feature learning for image classification via multiobjective genetic programming,” IEEE Trans. Neural Netw. Learn. Syst., vol. 25, no. 7, pp. 1359–1371, Jul. 2014.
[28] Y. Bi, B. Xue, and M. Zhang, “An effective feature learning approach using genetic programming with image descriptors for image classification,” IEEE Comput. Intell. Mag., vol. 15, no. 2, pp. 65–77, May 2020.
[29] Y. Bi, B. Xue, and M. Zhang, “Genetic programming with a new representation to automatically learn features and evolve ensembles for image classification,” IEEE Trans. Cybern., vol. 51, no. 4, pp. 1769–1783, Apr. 2021.
[30] R.-J. Bruintjes, A. Lengyel, M. B. Rios, O. S. Kayhan, and J. van Gemert, “VIPriors 1: Visual inductive priors for data-efficient deep learning challenges,” 2021, arXiv:2103.03768.
[31] A. Lengyel et al., “VIPriors 2: Visual inductive priors for data-efficient deep learning challenges,” 2022, arXiv:2201.08625.
[32] S. Arora, S. S. Du, Z. Li, R. Salakhutdinov, R. Wang, and D. Yu, “Harnessing the power of infinitely wide deep nets on small-data tasks,” in Proc. ICLR, 2019, pp. 1–13.
[33] L. Brigato and L. Iocchi, “A close look at deep learning with small data,” in Proc. IEEE ICPR, 2021, pp. 2490–2497.
[34] Y. Bi, B. Xue, and M. Zhang, “Using a small number of training instances in genetic programming for face image classification,” Inf. Sci., vol. 593, pp. 488–504, May 2022.
[35] B. Barz and J. Denzler, “Deep learning on small datasets without pre-training using cosine loss,” in Proc. IEEE/CVF WCACV, 2020, pp. 1371–1380.
[36] L. Brigato, B. Barz, L. Iocchi, and J. Denzler, “Tune it or don’t use it: Benchmarking data-efficient image classification,” in Proc. IEEE ICCV, 2021, pp. 1071–1080.
[37] P. Sun, X. Jin, W. Su, Y. He, H. Xue, and Q. Lu, “A visual inductive priors framework for data-efficient image classification,” in Proc. ECCV, 2020, pp. 511–527.
[38] B. Zhao and X. Wen, “Distilling visual priors from self-supervised learning,” in Proc. ECCV, 2020, pp. 422–429.
[39] H. Al-Sahaf, M. Zhang, and M. Johnston, “Binary image classification: A genetic programming approach to the problem of limited training instances,” Evol. Comput., vol. 24, no. 1, pp. 143–182, 2016.
[40] Y. Bi, B. Xue, and M. Zhang, “Learning and sharing: A multitask genetic programming approach to image feature learning,” IEEE Trans. Evol. Comput., vol. 26, no. 2, pp. 218–232, Apr. 2022.
[41] D. J. Montana, “Strongly typed genetic programming,” Evol. Comput., vol. 3, no. 2, pp. 199–230, Jun. 1995.
[42] L. Zhang, J. Liu, B. Zhang, D. Zhang, and C. Zhu, “Deep cascade model-based face recognition: When deep-layered learning meets small data,” IEEE Trans. Image Process., vol. 29, pp. 1016–1029, 2020.
[43] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in Proc. IEEE CVPR, vol. 1, 2005, pp. 886–893.
[44] T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 7, pp. 971–987, Jul. 2002.
[45] A. Vedaldi and B. Fulkerson, “VLFeat: An open and portable library of computer vision algorithms,” in Proc. 18th ACM Int. Conf. Multimedia, 2010, pp. 1469–1472.
[46] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” 2009.
[47] H. Xia, K. Rasul, and R. Vollgraf, “Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms,” 2017, arXiv:1708.07747.
[48] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, Reading Digits in Natural Images with Unsupervised Feature Learning, Stanford Univ., Stanford, CA, USA, 2011.
[49] F. S. Samaria and A. C. Harter, “Parameterisation of a stochastic model for human face identification,” in Proc. IEEE Workshop Appl. Comput. J. Mach., vol. 33, no. 8, pp. 1561–1576, Aug. 2011.
[50] K.-C. Lee, J. Ho, and D. J. Kriegman, “Acquiring linear subspaces for face recognition under variable lighting,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 5, pp. 684–698, May 2005.
[51] T. Liao, Z. Lei, T. Zhu, S. Zeng, Y. Li, and C. Yuan, “Deep metric learning for K nearest neighbour classification,” IEEE Trans. Knowl. Data Eng., early access, Jun. 17, 2021, doi: 10.1109/TKDE.2021.3090275.
[52] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE CVPR, 2016, pp. 770–778.
[53] M. Ulicny, V. A. Krylov, and R. Dahyot, “Harmonic networks with limited training samples,” in Proc. EuroGIP, 2019, pp. 1–5.
[54] J. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, arXiv:1409.1556.
[55] S. Zagoruyko and N. Komodakis, “Wide residual networks,” 2016, arXiv:1605.07146.
[56] E. Oyallon et al., “Scattering networks for hybrid representation learning,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 9, pp. 2208–2221, Sep. 2019.
[57] J. Mauci, U. Mele, D. Circean, and J. Schmidhuber, “Stacked convolutional auto-encoders for hierarchical feature extraction,” in Proc. ICANN, 2011, pp. 52–59.
[58] X. Chen, Y. Sun, M. Zhang, and D. Peng, “Evolving deep convolutional variational autoencoders for image classification,” IEEE Trans. Evol. Comput., vol. 25, no. 5, pp. 815–829, Oct. 2021.
[59] J. Lezama, Q. Qiu, P. Musé, and G. Sapiro, “OLE: Orthogonal low-rank embedding and play geometric loss for deep learning,” in Proc. IEEE CVPR, 2018, pp. 8109–8118.
[60] A. Bietti, G. Mialon, D. Chen, and J. Mairal, “A kernel perspective for regularizing deep neural networks,” in Proc. ICML, 2019, pp. 664–674.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Bing Xue (Senior Member, IEEE) received the B.Sc. degree from the Henan University of Economics and Law, Zhengzhou, China, in 2007, the M.Sc. degree in management from Shenzhen University, Shenzhen, China, in 2010, and the Ph.D. degree in computer science from the Victoria University of Wellington (VUW), Wellington, New Zealand, in 2014. She is currently a Professor of Artificial Intelligence, and a Deputy Head of School with the School of Engineering and Computer Science, VUW. She has over 300 papers published in fully refereed international journals and conferences. Her research focuses mainly on evolutionary computation, machine learning, classification, symbolic regression, feature selection, evolving deep NNs, image analysis, transfer learning, and multiobjective machine learning.

Dr. Xue is currently the Chair of the IEEE CIS Evolutionary Computation Technical Committee and IEEE CIS Task Force on Evolutionary Deep Learning and Applications, and the Editor of the IEEE CIS NEWSLETTER. She has also served as an Associate Editor for several international journals, such as the *IEEE Computational Intelligence Magazine*, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, and ACM Transactions on Evolutionary Learning and Optimization.

Mengjie Zhang (Fellow, IEEE) received the B.E. and M.E. degrees from the Artificial Intelligence Research Center, Agricultural University of Hebei, Baoding, Hebei, China, in 1989 and 1992, respectively, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 2000. He is currently a Professor of Computer Science, the Head of the Evolutionary Computation Research Group, and the Associate Dean (Research and Innovation) with the Faculty of Engineering, Victoria University of Wellington, Wellington, New Zealand. He has published over 700 papers in refereed international journals and conferences. His current research interests include evolutionary computation, particularly genetic programming and PSO with application areas of image analysis, multiobjective optimization, feature selection and reduction, job-shop scheduling, and transfer learning.

Prof. Zhang was the Chair of the IEEE CIS Intelligent Systems and Applications Technical Committee, IEEE CIS Emergent Technologies Technical Committee, and Evolutionary Computation Technical Committee, and a member of the IEEE CIS Award Committee. He is a Vice-Chair of the IEEE CIS Task Force on Evolutionary Feature Selection and Construction and Task Force on Evolutionary Computer Vision and Image Processing, and the Founding Chair of the IEEE Computational Intelligence Chapter in New Zealand. He is also a Committee Member of the IEEE NZ Central Section. He is a Fellow of the Royal Society of New Zealand, an IEEE CIS Distinguished Lecturer, and have been a Panel Member of the Marsden Fund (New Zealand Government Funding).