An Evolutionary Deep Learning Approach Using Genetic Programming with Convolution Operators for Image Classification

Ying Bi, Bing Xue and Mengjie Zhang
School of Engineering and Computer Science,
Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand
Email:{Ying.Bi, Bing.Xue, Mengjie.Zhang}@ecs.vuw.ac.nz

***Abstract*—Evolutionary deep learning (EDL) as a hot topic in recent years aims at using evolutionary computation (EC) techniques to address existing issues in deep learning. Most existing work focuses on employing EC methods for evolving hyper-parameters, deep structures or weights for neural networks (NNs). Genetic programming (GP) as an EC method is able to achieve deep learning due to the characteristics of its representation. However, many current GP-based EDL methods are limited to binary image classification. This paper proposed a new GP-based EDL method with convolution operators (COGP) for feature learning on binary and multi-class image classification. A novel flexible program structure is developed to allow COGP to evolve solutions with deep or shallow structures. Associated with the program structure, a new function set and a new terminal set are developed in COGP. The experimental results on six different image classification data sets of varying difficulty demonstrated that COGP achieved significantly better performance in most comparisons with 11 effectively competitive methods. The visualisation of the best program further revealed the high interpretability of the solutions found by COGP.**

I. Introduction

Deep learning is a hot research topic in recent decades. Many deep models/methods have been developed and showed promising results in very difficult tasks such as image classification, object detection and natural language processing [1]. Deep learning methods are representation learning methods, where the representation of data with multiple levels of abstraction is learned using multiple processing layers [1, 2]. In other words, the success of deep learning methods owns to three main characteristics: sufficient model complexity, layer-by-layer preprocessing and feature transformation [3, 4]. In general, the three characteristics encourage the development of different types of deep learning methods. However, most existing deep learning methods are neural network (NN)-based methods. The NN-based methods have their own limitations, such as providing black-box solutions, requiring a large number of training instances and computing resources.

Computer vision is one of the main areas that deep learning methods have achieved big success. Image classification is a fundamental task in computer vision with many real-world applications. Image classification is the task of classifying different images into predefined groups based on the content in the images. Due to image variations, such as occlusion, rotation, scale, and deformation variations, image classification remains a challenging task.

In recent years, convolutional neural networks (CNNs) have achieved significant success in image classification. For example, deep CNNs have achieved the best results on the famous ImageNet large scale visual recognition challenge (ILSVRC) from 2012 to 2017. Despite the promise, deep CNNs have several limitations. Firstly, deep CNNs provide black-box-like solutions, which are often very hard to explain and interpret. Therefore, many efforts have been devoted to visualising the features learned by CNNs, such as in [5]. However, due to a large number of learned features, these methods are still uninterpretable [6]. Secondly, developing a new deep CNN needs rich domain knowledge to construct the architecture and set the important parameters [7]. Generally, the model complexity of deep CNNs is determined by experts who develop them. Although many existing well-developed models can be easily used, they might not be effective for other domains/problems or for problems with limited data.

Evolutionary deep learning (EDL) is a research field aiming at using evolutionary computation (EC) techniques to address existing issues in deep learning. Evolutionary computation is a subfield under artificial intelligence and has gained much attention in recent decades. Existing EDL methods can be broadly classified into two groups: NN-based EDL methods and genetic programming (GP)-based EDL methods [4]. The methods in the first category use evolutionary algorithms, including particle swarm optimisation (PSO), genetic algorithms (GAs) and differential evolution (DE), to evolve deep NNs, including the architecture, hyper-parameters and weights of NNs [7–10]. Because NNs have different types of models, such as CNNs, autoencoder and recurrent NN (RNNs), this category has a large range with many existing works. The second category aims at using GP to achieve deep learning as GP is able to automatically evolve models/solutions with sufficient complexity for specific problems. The other characteristics of deep learning can be easily found in GP-based solutions, especially on image-related tasks. Currently, in this category, we only find GP-based methods. Perhaps other EC methods can achieve this in the near future.

GP as an EC technique aims at automatically evolving computer programs from predefined primitive sets to solve

particular problems using the principle of biological evolution and natural selection [11]. Well-known for the good global search ability, flexible representation and high interpretability, GP has achieved promising results in many problems, including symbolic regression, classification and image analysis [12].

GP-based EDL methods have been developed in recent years, where most of them tackled with image classification tasks [13–19]. Traditionally, the process of dealing with image classification contains several steps: preprocessing, feature extraction, feature selection, and classification. One type of GP-based EDL methods deals with these several steps in one single program, which means that the raw image is fed to a GP program and the class label is returned [13–16, 20]. Another type of GP-based EDL methods learns multiple features from raw pixels and uses a traditional classification algorithm for classification [17, 21–24]. Both these two types are able to automatically evolve computer programs that transform non-meaningful raw pixels into discriminative features for classification through many different processes, including feature extraction, feature construction, filtering, and pooling.

Commonly used GP method has a tree-based representation, which is very flexible and is able to find solutions with suitable structure and complexity for particular problems. The tree-based solutions are often evolved by selecting functions from the function set to build the internal or root nodes and selecting terminals from the terminal set to build the leaf nodes and often provide good interpretability. However, the potential of GP for deep learning has not been extensively explored.

There are many functions or operations, such as convolution, pooling and activation, that have been used in CNNs and showed promise in image classification. These functions can be applied and developed as GP functions to allow GP to automatically evolve the combinations of them for effective feature learning and image classification. To well cooperate with these functions, a new program structure is needed.

A. Goals

The aim of this paper is to develop a new GP-based EDL method for feature learning and image classification. Inspired by the convolution and pooling operations employed in CNNs, the COGP method will use these operations as functions. A new program structure, a new function set and a new terminal set will be developed to allow COGP to learn discriminative features for classification. The performance of COGP will be examined on a number of image classification data sets of varying difficulty and compared with that of state-of-the-art algorithms. Specifically, this paper will answer the following research questions.

1) How are the convolution and pooling operations achieved in COGP?
2) How are the new functions and new terminals integrated into a single COGP program?
3) Can COGP achieve better performance than state-of-the-art methods?
4) Can the solutions of COGP be easy to understand and interpret?

II. Related Work

This section reviews and discusses recent work on existing EDL methods for feature learning and image classification. The existing methods can be broadly classified into NN-based EDL methods and GP-based EDL methods.

A. NN-based EDL Methods

Using EC techniques for evolving NN is not a new topic, which has been investigated in the 1980s and some early works have been reviewed in [25]. In 2002, the well-known method, NeuroEvolution of Augmenting Topologies (NEAT) [26], was developed to use GAs for evolving the NN architectures and weights. NEAT has been extended to hyperNEAT in [26], where a composition pattern producing network was used as the coding strategy. But the early work is not evolving deep methods. Recently, EC methods have applied to evolve deep models. Iba [9] reviewed several different methods to evolve deep NNs, including NEAT, Genetic CNNs, hierarchical feature constructing using GP, and differentiable pattern-producing network (DPPSN).

As the architecture and weights connection of CNNs are different from traditional NNs, several EC methods have recently been developed to evolving deep CNNs for image classification. Suganuma et al. [27] applied Cartesian GP to evolve CNNs (CGP-CNN) with convolutional block (ConvBlock) and with residual block (ResBlock) for image classification. CGP-CNN automatically selects hyper-parameters for each block and finds the combinations/connections of different modules of CNNs. However, this method requires large computational resources and takes a few weeks to run the experiments. Sun et al. [7] developed an EvoCNN method for image classification, where a new GA method with a flexible gene encoding strategy was proposed to evolve the architectures and connection weights of CNNs. Based on the new representation, new selection, crossover and mutation operators have also been developed in EvoCNN. Wang et al. [28] proposed an IPPSO method for evolving the architecture and the hyper-parameters for CNNs on image classification.

Besides supervised learning methods, EC algorithms have been applied for evolving unsupervised learning methods based on NNs. Sun et al. [8] applied GA to evolve unsupervised deep NNs, i.e., autoencoder and Restricted Boltzmann machine (RBM), to learn meaningful representation for image classification. The comparisons between PSO and grid search on the hyper-parameter optimisation for autoencoder have been conducted in [10]. The experimental results have showed the effectiveness and efficiency of PSO on this optimisation task. Sun et al. [29] developed a PSO method for evolving deep convolutional autoencoder for image classification. However, these aforementioned methods often require extensive computational resources to run the experiments and generate models with millions of parameters.

B. GP-based EDL Methods

Many GP-based EDL methods have been developed for image-related tasks. To the best of our knowledge, the first

GP-based EDL method is the multi-tier GP (known as 3TGP) method [13], which has an image filtering tier, an aggregation tier and a classification tier to learn high-level features and construct classifiers for image classification. The image filtering tier has several filtering functions such as mean, max and min to process the input image to obtain feature maps. The aggregation tier performs region detection from the feature map and extracts a feature from each detected region. The classification tier constructs the extracted features by the aggregation tier to a high-level feature for classification. The classification decision is based on the final output of each GP program using a zero-threshold. Naturally, each 3TGP program is a classifier for assigning a class label to each input image when dealing with binary image classification.

Al-Sahaf et al. [14] developed a two-tier GP (2TGP) method by removing the filtering tier of 3TGP. The aggregation tier of 2TGP can detect regions with different shapes and sizes, such as line, column, circle, and rectangle. Obviously, 2TGP is much faster than 3TGP. Many variants of 2TGP have been developed to extract high-level features rather than the pixel statistics in the aggregation tier. Lensen et al. [20] proposed a HoG+GP method based on the framework of 2TGP, where the histogram of gradient (HOG) descriptor was developed as functions in the aggregation tier to extract high-level features. Similarly, Bi et al. [15, 16] presented new GP methods using different filtering functions, feature extraction functions and classification functions. Each program of the two methods could perform region detection, feature extraction, feature construction, and image classification, simultaneously. Evans et al. [6] developed a ConvGP method for image classification. The program structure of ConvGP is similar to 3TGP, where the image filtering tier is replaced with a convolution layer. The convolution layer uses convolution functions and pooling functions to obtain meaningful features. However, these aforementioned methods only produce one high-level feature and are only effective for binary image classification.

GP-base EDL methods have also been developed to automatically learn multiple features for image classification, including multi-class classification. Shao et al. [17] proposed a multi-objective GP (MOGP) method for feature learning and image classification. The program structure of MOGP has an input layer, a filtering layer, a pooling layer, a concatenation layer, and an output layer. The program of MOGP performs the corresponding process to each input image using the evolved functions and generates a number of features as output. The features were fed to principal component analysis (PCA) for dimensionality reduction and then fed to a linear support vector machine (SVM) for classification.

Agapitos et al. [18] developed a greedy layer-wise GP-based EDL method for handwritten digits recognition. This method has a filter bank layer, a transformation layer, an average pooling layer, and a classification layer. Specifically, the filter bank layer has a collection of filters to convolve the input image. Similar to this method, Suganuma et al. [19] developed a new method by using more layers with filters for feature construction. The method in [19] has two stages of feature construction, where the first stage uses a combination of image processing filters and the second stage uses a combination of evolved filters. The structures of these two methods are very similar to the architecture of CNNs, but their performance has not been compared with CNNs.

Rodriguez-Coayahuitl [30] defined GP for representation learning and proposed GP autoencoder for unsupervised representation learning for image classification. Similar to autoencoder, a GP encoding forest was used for encoding and a GP decoding forest was used for decoding, which means each GP individual contains two forests with a number of trees. This method has been examined on three image classification data sets and showed promising results.

In summary, the above work on EDL has showed promising results in image classification. However, most EDL methods have their own limitations. It is necessary to develop new effective and interpretable EDL methods. The GP-based EDL methods have showed promising results and high interpretability in image classification. However, most GP-based EDL methods only deal with binary image classification [13, 15, 16, 20, 31]. Therefore, this paper develops a new GP-based EDL method for binary and multi-class image classification. To achieve this, a new flexible program structure, a new function set and a new terminal set are developed in the new method.

III. The Proposed Approach

This section describes the proposed COGP approach in detail, including the overall algorithm, the new program structure, the new function set, and the new terminal set.

A. Overall Algorithm

The COGP method starts with randomly generating a number of initial GP trees based on the new program structure, the new function set and the new terminal set. Then each tree is evaluated using a training set and a linear support vector machine (SVM). The evaluation process starts with using the GP tree to transform each image in the training set into features. The transformed training set is then normalised using the min-max normalisation method [24] and is fed into a linear SVM to perform classification. To evaluate each tree and to improve the generalisation ability, 5-fold cross-validation is used for SVM on the training set [24]. The mean accuracy of the 5 folds is obtained and employed as the fitness value of the GP tree. After the fitness evaluation, the whole GP population is updated using the selection method and the genetic operators, i.e., *elitism, crossover* and *mutation*. Then the population is evaluated again. The overall process proceed until reaching the maximum number of generations. Finally, the COGP system returns the best individual.

B. Program Structure

To deal with multiple tasks in a single GP program, it is necessary to design the program structure. COGP uses a novel program structure, which is developed based on strongly typed GP (STGP) [32]. Compared with traditional GP, STGP gives an input type and an output type for each function and an

output type for each terminal. STGP also uses a tree-based representation. But the output types of the children nodes of each function in STGP must be the same as the input types of the functions.

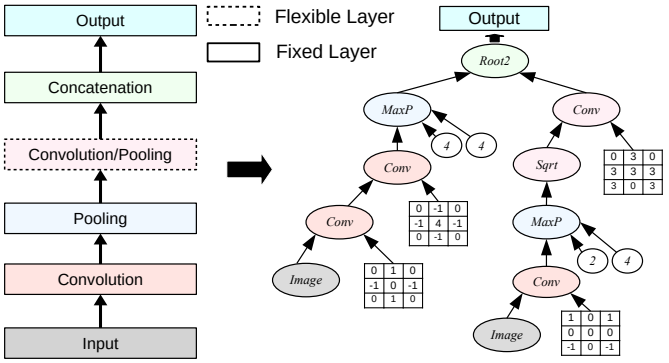

Fig. 1. The program structure of COGP and two example programs that can be evolved by COGP.

The new program structure of COGP is shown in Fig. 1, where the right part of the figure shows an example program that may be evolved by COGP. The program structure of COGP has several layers, including an input layer, a convolution layer, a pooling layer, a convolution/pooling layer, a concatenation layer, and an output layer. These layers are connected using a bottom-up manner as shown in Fig. 1. Different layers have various functions for different purposes, which will be introduced in the following subsection. The input layer takes the image, which will be classified, as input. The convolution layer mainly performs convolution operation to the image and also has several important functions for rescaling, activation or enhancement. The pooling layer conducts max-pooling to the input image, where one important purpose is for dimensionality reduction. The convolution/pooling layer has the same functionality as the convolution and pooling layers. The concatenation layer is to concatenate images or vectors into one vector and the output layer returns the output. Obviously, the input for each COGP program is an image and the final output is a feature vector with particular dimensions. Owing to the flexible representation of GP and the functions used in the concatenation layer, the dimension of the output features is flexible and not known in advance.

One characteristic of the new program structure is that it has flexible layers and fixed layers. As shown in Fig. 1, the output, convolution, pooling, concatenation, and input layers are fixed, while the convolution/pooling layer is flexible. The fixed layers make sure there exist feature transformation and dimensionality reduction from raw pixels to features. The flexible layer allows the COGP programs to have multiple convolution and/or pooling layers and it may be evolved or not be evolved. As shown in Fig. 1, the left branch of the example program does not have the convolution/pooling layer, while the right branch has the convolution/pooling layer. This is the main difference from existing GP-based EDL methods in terms of the program structure, such as in [6, 17].

Note that for each layer except for the input and output layers, the tree depth is flexible, which means that each layer may have several functions in the evolved COGP programs. Owing to the flexibility, the new program structure allows COGP to evolve simple programs for easy tasks and complex programs with multiple convolutional operators and pooling operators for difficult problems.

TABLE I
THE FUNCTION SET

Methods	Input	Output	Description
Root1	2 vectors	1 vector	Concatenate two vectors into a vector
Root2	2 images	1 vector	Concatenate two images into a vector
Root3	3 images	1 vector	Concatenate three images into a vector
Root4	4 images	1 vector	Concatenate four images into a vector
Conv	1 image, filter	1 image	Perform convolution to the image using the filter
MaxP	1 image, k_1, k_2	1 image	Perform max-pooling to the image
ZMaxP	1 image, k_1, k_2	1 image	Perform max-pooling to the image and then perform zero-padding to the image
Sub	2 image, n_1, n_2	1 image	Subtract each two weighted image with the same or different sizes
Add	2 image, n_1, n_2	1 image	Add each two weighted image with the same or different sizes
ReLU	1 image	1 image	Return $max(0,\ x)$ for each x in the image
Sqrt	1 image	1 image	Return $\sqrt{x}$ or 1 for each x in the image
Abs	1 image	1 image	Return $\|x\|$ for each x in the image

C. Function Set

All the functions of COGP and their input, output, and description are listed in Table I. The **Root1**, **Root2**, **Root3**, and **Root4** functions are used in the concatenation layer. These functions transform images or vectors to one vector by concatenating all the arrays in images or vectors. The **MaxP** function is employed in the pooling layer, which down-samples an image by returning the maximum value of each sliding window. This function can reduce the image size and keep important information of an image. The **MaxP** function takes an image and the kernel size, k_1 and k_2 as input and returns a smaller image. The convolution layer has the **Conv**, **Sub**, **Add**, **ReLU**, **Sqrt**, and **Abs** functions. The **Conv** function performs convolution operation to the input image by using the evolved **filter**. The convolution operation is to replace the pixel value with the weighted sum of its neighbours' values, where the filter is actually used as the weights. The **Sub** and **Add** functions perform weighted subtract and add operation to two images using the n_1 and n_2 as weights. In case that the two input images may have different sizes, the **Sub** and **Add** functions also perform image cutting to make the two images have the same size. The **ReLU**, **Sqrt** and **Abs** functions are used for rescaling the pixel values of an image by returning non-negative values. The **ReLU** function is known as rectified linear unit (ReLU), which is commonly used as the activation function in CNNs. The **Sqrt** function is protected by returning 1 if the pixel value is negative.

The flexible layer, convolution/pooling, has the functions that are used in the convolution and pooling layers and an additional function **ZMaxP**. The **ZMaxP** function performs max-pooling to an image and adds zero-padding around the image to keep the size of the image. If many **MaxP** functions

are evolved in a branch of a GP tree, the dimension of the image will be reduced quickly. The **ZMaxP** function is used for reserving the size of the image.

D. Terminal Set

All the terminals and their value range and description are shown in Table II. The terminals are $image_{m\times n}$, $filter_{3\times 3}$, $filter_{5\times 5}$, $filter_{7\times 7}$, k_1, k_2, n_1, and n_2. The $image_{m\times n}$ terminal represents the input image with size of $m \times n$. The $filter_{3\times 3}$, $filter_{5\times 5}$ and $filter_{7\times 7}$ terminals are the filters with different sizes, which are used as one of the inputs of the **Conv** function. The values in the filters are randomly initialised based on the range that is listed in Table II and automatically evolved during the evolutionary process. The n_1 and n_2 terminals are the inputs of the **Add** and **Sub** functions, which are randomly sampled from $[0.000, 1.000)$. The k_1 and k_2 terminals mean the kernel size of the **MaxP** and **ZMaxP** functions, where their values are randomly generated from the range initially and automatically evolved during the evolutionary learning process.

TABLE II
TERMINALS

Terminal	Value Range	Description
$image_{m\times n}$	$\{0, 1, 2, ..., 255\}$	Input image with size of $m \times n$
$filter_{3\times 3}$	$\{-5, -4, ..., 5\}$	3×3 filter. It is the children node of **Conv**
$filter_{5\times 5}$	$\{-5, -4, ..., 5\}$	5×5 filter. It is the children node of **Conv**
$filter_{7\times 7}$	$\{-5, -4, ..., 5\}$	7×7 filter. It is the children node of **Conv**
n_1, n_2	$[0.000, 1.000)$	Random numbers and they are children nodes for **Add** and **Sub**
k_1, k_2	$\{2, 4\}$	The kernel size of **MaxP** and **ZMaxP**

IV. Experiment Design

To examine the performance of the proposed COGP method, a series of experiments have been conducted. This section designs the experiments, including data sets, comparative methods and parameter settings.

A. Data Sets

Six different data sets of varying difficulty are employed in the experiments. They are FEI_1 [33], FEI_2 [33], ORL [34], JAFFE [35], VGDB [36], and KTH [37]. The FEI_1 and FEI_2 data sets are binary classification of facial expression. The ORL data set with 40 classes is the face recognition task. The JAFFE data set has images with seven different facial expressions such as happy, sad and fear. The VGDB data set is recognising the paintings drawn by Vincent Van Gogh and it is challenging. The KTH data set is a commonly used texture classification data set. The data set properties, including the size of the image, the number of classes, the number of images in the training and test sets, are listed in Table III. Several example images of each data set are shown in Fig. 2.

Fig. 2. Example images of the six different image data sets.

TABLE III
DATA SET PROPERTIES

Name	Image Size	#Class	Training set	Test set
FEI_1	60×40	2	150	50
FEI_2	60×40	2	150	50
ORL	50×55	40	240	160
JAFFE	32×32	7	140	73
VGDB	50×50	2	247	83
KTH	50×50	10	480	330

B. Comparative Methods

Ideally, the performance of COGP is compared with existing GP-based COGP methods that have been mentioned in Section II-B. However, most of these methods learn one high-level feature from each image and are only suitable for binary classification. Some methods have showed their deficiency in dealing with difficult tasks, especially when comparing them with CNNs, such as in [6]. Therefore, to show the effectiveness of COGP, we compare it with existing effective non-GP methods, i.e., traditional classification algorithms using raw pixels or pre-extracted features and CNNs.

1) Traditional classification algorithms using raw pixels: Five well-known classification algorithms, i.e., SVM, k-nearest neighbour (KNN), logistic regression (LR), random forest (RF), and AdaBoost have been employed in this category for comparisons. SVM uses a linear kernel according to [17] and its parameters are the default ones in *scikit-learn* [38]. The number of neighbours in KNN is 1. In RF and AdaBoost, the number of trees is 500 and the maximum tree depth is 100 according to [3]. The parameter settings for LR are the default ones in *scikit-learn* [38], where the parameter C is 1.

2) Traditional classification algorithms using pre-extracted features: In this category, four commonly used feature extraction methods are employed to extract different features from the image and SVM is used for classification. The four methods are local binary pattern (LBP), uniform local binary pattern (uLBP), histogram of gradients (HOG), and scale-invariant feature transform (SIFT) [23]. The LBP method extracts 256 histogram features from each LBP image. The uLBP method extracts 59 histogram features from each uLBP image, which are invariant to rotation. The HOG method refers to [39], which generates a high-dimensional feature vector from each image. For example, it produces 2400 features for each image in the FEI_2 data set. The SIFT method is dense SIFT [40], where each input image is considered as a keypoint and a feature vector is generated from each image. In these methods, SVM is employed for classification in order to keep consistent with the proposed COGP method.

3) CNNs: Since the data sets in our experiments are small and the well-known deep CNN models such as ResNet and DenseNet may not be efficient and effective. Two CNNs with different architectures are manfully crafted for comparisons. The first has five layers (simplified as CNN-5), i.e., two convolutional layers, one max-pooling layer and two fully-connected layers. The second has eight layers (simplified as CNN-8), i.e., four convolutional layers, two max-pooling layers and two fully-connected layers. Dropout is used after

the pooling layer and the first fully-connected layer with 0.25 and 0.5 probabilities, respectively, to avoid overfitting [41]. The activation function in CNN-5 and CNN-8 is the popular ReLU function and the loss function is the cross-entropy. The batch size is 128 and the number of epochs is 500 because the benchmark data sets are not large.

C. Parameter Settings and Test Process

Parameter settings for COGP are listed in Table IV. Because of the usage of the filter and the Conv function, a larger mutation rate allows the values of the filter to change more. Other parameter settings are commonly used settings for GP. The implementation of COGP is based on the *DEAP* (*Distributed Evolutionary Algorithm Package*) package [42].

TABLE IV
GP RUN TIME PARAMETERS

Parameter	Value	Parameter	Value
Generations	50	Crossover rate	0.5
Population size	500	Mutation rate	0.49
Population generation	Ramped half-and-half	Elitism rate	0.01
Selection type	Tournament (size=7)	Tree depth	2–8

The test process of COGP starts with using the best individual learned from the training process to transform each image in the training set and the test set to features. The transformed training and test sets are then normalised using the min-max normalisation. The normalised training set is used to train a linear SVM and the trained SVM classifier is tested on the normalised test set. Finally, the classification accuracy of the test set is reported.

The experiments of all the methods run 30 times independently. The final results of *the test set* are reported.

V. RESULTS AND DISCUSSIONS

This section discusses the experimental results obtained by COGP and all the comparable methods on the six data sets.

Table V lists the maximum accuracy (Max), mean accuracy and standard deviation (Mean±St.dev) of COGP and the comparable methods on the six data sets. Each block in Table V shows the results of one data set and the best maximum and mean accuracy is highlighted in bold. The Wilcoxon rank-sum test with a 5% level is used to compare COGP with a comparable method to show the significance of performance improvement. The symbols "–", "+" or "=" in Table V denote whether COGP is significantly worse, better than or similar to a comparable method. The final row of each block summarises the overall results of the significance test on each data set.

Compared with the five traditional methods using raw pixels, i.e., SVM, KNN, LR, RF, and AdaBoost, the COGP method achieves significantly better performance in 25 comparisons out of 30 (5×6) comparisons. COGP outperforms SVM, KNN and AdaBoost significantly on the six data sets. Especially, the comparisons between COGP and SVM show that the features learned by COGP based the new program structure, the new function set and the new terminals are more effective and discriminative than raw pixels for image classification. It is not entirely fair to compare COGP with RF as RF is an ensemble method. But the comparisons further show the effectiveness of COGP on feature learning and image classification as RF only achieves better performance than COGP on FEI_1 and VGDB. The overall comparisons demonstrate that COGP learns effective features from raw pixels to improve the classification accuracy.

TABLE V
CLASSIFICATION ACCURACY(%) OF COGP AND ALL THE COMPETITIVE METHODS ON THE SIX DATA SETS

Methods	Max	Mean±St.dev	Max	Mean±St.dev
Data Set	**FEI_1**		**FEI_2**	
SVM	90.00	90.00±0.00+	88.00	88.00±0.00+
KNN	32.00	32.00±0.00+	8.00	8.00±0.00+
LR	92.00	92.00±0.00+	88.00	88.00±0.00+
RF	**98.00**	**97.07±1.01**–	90.00	89.20±1.13=
AdaBoost	80.00	78.67±1.32+	80.00	76.00±3.44+
uLBP+SVM	66.00	56.73±3.66+	68.00	62.53±3.52+
LBP+SVM	68.00	64.60±1.83+	74.00	69.80±0.00+
HOG+SVM	96.00	96.00±0.00=	82.00	82.00±0.00+
SIFT+SVM	56.00	56.00±0.00+	62.00	62.00±0.00+
CNN-5	**98.00**	95.40±1.30=	**98.00**	**95.27±1.62**–
CNN-8	**98.00**	95.33±1.32=	96.00	90.93±1.87=
COGP	**98.00**	94.60±2.74	96.00	90.33±3.86
	Overall	**7+, 3=, 1–**	**Overall**	**8+, 2=, 1–**
Data Set	**ORL**		**JAFFE**	
SVM	94.38	94.38±0.00+	93.94	**91.06±0.73**–
KNN	94.38	94.38±0.00+	71.21	71.21±0.00+
LR	93.75	93.75±0.00+	89.39	89.39±0.00–
RF	93.12	92.33±0.63+	75.76	72.48±1.99+
AdaBoost	59.38	52.27±4.00+	53.03	47.93±2.68+
uLBP+SVM	87.50	87.42±0.21+	31.82	26.87±3.30+
LBP+SVM	88.12	87.52±0.20+	33.33	28.84±2.05+
HOG+SVM	91.25	91.25±0.00+	81.82	80.30±0.40+
SIFT+SVM	93.75	93.75±0.00+	33.33	33.33±0.00+
CNN-5	96.88	95.29±1.06+	**95.45**	90.96±2.68–
CNN-8	95.00	93.04±1.09+	90.91	84.54±4.33=
COGP	**98.12**	**96.46±1.11**	92.42	86.41±3.14
	Overall	**11+**	**Overall**	**7+, 1=, 3–**
Data Set	**VGDB**		**KTH**	
SVM	55.42	52.33±1.37+	46.97	44.59±2.83+
KNN	59.04	59.04±0.00+	34.24	34.24±0.00+
LR	56.63	56.63±0.00+	48.79	48.79±0.00+
RF	72.29	68.15±2.44–	60.00	57.81±0.83+
AdaBoost	66.27	60.28±3.33+	37.88	33.44±1.37+
uLBP+SVM	72.29	67.19±3.59–	78.79	73.29±4.18+
LBP+SVM	74.70	**73.25±1.07**–	83.64	**82.71±0.51**–
HOG+SVM	54.22	51.81±1.10+	57.27	55.96±0.64+
SIFT+SVM	65.06	65.06±0.00–	65.76	65.76±0.00+
CNN-5	65.06	60.24±2.33=	**85.76**	82.56±1.87–
CNN-8	65.06	56.67±5.43+	76.36	71.63±3.18+
COGP	**75.90**	62.05±6.58	83.94	78.49±2.15
	Overall	**6+, 1=, 4–**	**Overall**	**9+, 2–**

Compare with uLBP+SVM, LBP+SVM, HOG+SVM, and SIFT+SVM, the COGP method achieves significantly better or similar performance in 20 comparisons of the total 24 (4×6) comparisons. On the FEI_1, FEI_2, ORL, and JAFFE data sets, COGP is better than the four methods. This shows that the features learned by COGP are more powerful and discriminative than the commonly used hand-crafted features for classifying face and facial expression images. On the VGDB and KTH data sets, COGP is more effective for achieving the maximum accuracy than the four methods but less effective in terms of the mean accuracy. On the two data sets, the LBP features are more effective than the features learned by COGP. This illustrates that texture features are difficult to be learned

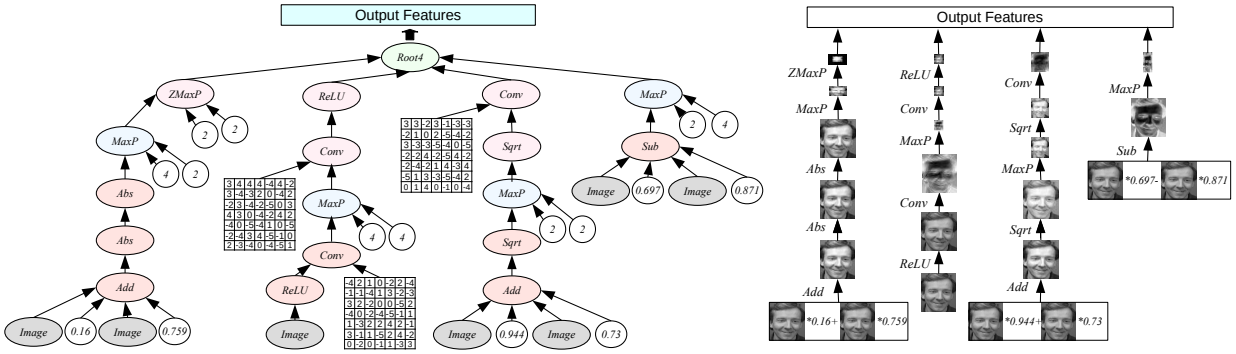

Fig. 3. An example program evolved by COGP on the ORL data set. This program has achieved 99.75% accuracy on the training set and 98.12% accuracy on the test set. The right part of this figure shows the process of transforming an example image into features using the evolved functions.

by using convolutional and pooling operations. In summary, COGP learns effective features, which are more powerful and discriminative than the popular hand-crafted features for image classification, especially on face images.

Compared with CNN-5 and CNN-8, the COGP method achieves significantly better or similar performance in 8 comparisons out of the total 12 (2×6) comparisons. COGP achieves better performance than CNN-8 on the six data sets and achieves better performance than or similar performance to CNN-5 on three data sets. Surprisingly, CNN-8 has more layers and filters but is less accurate than CNN-5. This may because an increase of layers and filters makes the model more difficult to train. Compared with CNN-5 and CNN-8, the selection of convolution and pooling in COGP is automatically determined during the evolutionary learning process. This may be more effective for learning a suitable model that match the problem and easier to train and interpret. However, GP does not have an effective operator for optimising the weights (constant parameters) of the filters, which might limit its performance. Overall, the comparisons show the potential of GP on feature learning and image classification.

In summary, COGP achieves significantly better or similar performance in 55 comparisons out of the total 66 (11×6) comparisons. Importantly, COGP outperforms significantly than all the 11 methods on ORL. COGP obtains the maximum accuracy of the 11 competitive methods on FEI_1, ORL and VGDB. Accordingly, COGP is an effective and promising approach for feature learning and image classification.

VI. Further Analysis

The COGP method has achieved the best results on the ORL data set compared with all the benchmark methods. Therefore, an example program on this data set is employed for further analysis to show how it extract features. An example program is visualised in Fig. 3, where the right part of this figure shows an example image and the features obtained by the corresponding functions in the example program. This example program has achieved 99.75% accuracy on the training set and 98.12% accuracy on the test set. Splitting from the root node, this program has four branches, where each branch generates different features using different functions as shown in the right part of Fig. 3. The first branch (from left to right) enhances the image by using the **Add** functions and down-samples the image using the **MaxP** function. The second branch describes image features in a way that is very similar to CNNs by using **ReLU**, **Conv** and **MaxP** functions. Two **Conv** functions with $filter_{7\times7}$ are evolved in this branch to describe features. As we can see from Fig. 3, the face image is changed after the first **Conv** function. Similarly, the third branch and the fourth branch also generate salient features, which are discriminative for classification.

However, further analysing this program reveals that it has redundant functions in the branches, such as the two **Abs** functions in the first branch. They do not contribute to change the input image when all the pixel values are non-negative. The problem may lead to future work on the investigation of how to simplify the evolved GP trees under the type constraint.

VII. Conclusions

The goal of this paper was to develop a new GP-based EDL method for feature learning and image classification, which has been successfully achieved by proposing the COGP method with a new program structure, a new function set and a new terminal set. COGP has been examined on six binary or multi-class image classification data sets of varying difficulty and compared with 11 commonly used methods. The experimental results showed that COGP achieved significantly better performance in most comparisons. Further analysis revealed the good interpretability of the evolved programs.

The COGP method is an example to show the potential of the GP-based EDL methods in feature learning and image classification. COGP provides tree-based solutions with suitable model complexity and comparable performance. However, due to the flexible representation and the search mechanism of GP, the potential of GP on feature learning has not been extensively investigated. Future work might focus on three aspects. Firstly, it is interesting to develop new GP-based EDL methods with deep structures for complex image classification, such as the well-known large scale image data sets CIFAR10 and CIFAR100. Secondly, powerful search operators such as local search operators might be needed for the GP-based EDL methods to improve computation complexity and search efficiency, where concise and effective solutions are able to be found. Finally, the GP-based EDL methods have been

investigated on supervised feature learning while seldom on unsupervised feature learning. It is desirable to develop new GP-based EDL methods for unsupervised feature learning.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[2] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[3] Z.-H. Zhou and J. Feng, "Deep forest: Towards an alternative to deep neural networks," *arXiv preprint arXiv:1702.08835*, 2017.

[4] M. Zhang and S. Cagnoni, "Evolutionary computation and evolutionary deep learning for image analysis, signal processing and pattern recognition," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2018, pp. 1221–1257.

[5] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision*. Springer, 2014, pp. 818–833.

[6] B. Evans, H. Al-Sahaf, B. Xue, and M. Zhang, "Evolutionary deep learning: A genetic programming approach to image classification," in *IEEE Congress on Evolutionary Computation*, 2018, pp. 1–6.

[7] Y. Sun, B. Xue, and M. Zhang, "Evolving deep convolutional neural networks for image classification," *arXiv preprint arXiv:1710.10741*, 2017.

[8] Y. Sun, G. G. Yen, and Z. Yi, "Evolving unsupervised deep neural networks for learning meaningful representations," *IEEE Transactions on Evolutionary Computation*, 2018,DOI: 10.1109/ICASSP.2017.7952676.

[9] H. Iba, "Evolutionary approach to deep learning," in *Evolutionary Approach to Machine Learning and Deep Neural Networks*. Springer, 2018, pp. 77–104.

[10] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "An experimental study on hyper-parameter optimization for stacked auto-encoders," in *IEEE Congress on Evolutionary Computation*, 2018, pp. 1–8.

[11] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT press, Cambridge, 1992, vol. 1.

[12] Y. Bi, B. Xue, and M. Zhang, "A survey on genetic programming to image analysis," *Journal of Zhengzhou University (Engineering Science)*, vol. 39, no. 06, pp. 3–13, 2018.

[13] D. Atkins, K. Neshatian, and M. Zhang, "A domain independent genetic programming approach to automatic feature extraction for image classification," in *IEEE Congress on Evolutionary Computation*, 2011, pp. 238–245.

[14] H. Al-Sahaf, A. Song, K. Neshatian, and M. Zhang, "Two-tier genetic programming: Towards raw pixel-based image classification," *Expert Systems with Applications*, vol. 39, no. 16, pp. 12 291–12 301, 2012.

[15] Y. Bi, M. Zhang, and B. Xue, "An automatic region detection and processing approach in genetic programming for binary image classification," in *International Conference on Image and Vision Computing New Zealand*. IEEE, 2017, Doi: 10.1109/IVCNZ.2017.8402469.

[16] Y. Bi, B. Xue, and M. Zhang, "An automatic feature extraction approach to image classification using genetic programming," in *International Conference on the Applications of Evolutionary Computation*. Springer, 2018, pp. 421–438.

[17] L. Shao, L. Liu, and X. Li, "Feature learning for image classification via multiobjective genetic programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 7, pp. 1359–1371, 2014.

[18] A. Agapitos, M. O'Neill, M. Nicolau, D. Fagan, A. Kattan, A. Brabazon, and K. Curran, "Deep evolution of image representations for handwritten digit recognition," in *IEEE Congress on Evolutionary Computation*, 2015, pp. 2452–2459.

[19] M. Suganuma, D. Tsuchiya, S. Shirakawa, and T. Nagao, "Hierarchical feature construction for image classification using genetic programming," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2016, pp. 001 423–001 428.

[20] A. Lensen, H. Al-Sahaf, M. Zhang, and B. Xue, "Genetic programming for region detection, feature extraction, feature construction and classification in image data," in *European Conference on Genetic Programming*. Springer, 2016, pp. 51–67.

[21] H. Al-Sahaf, A. Al-Sahaf, B. Xue, M. Johnston, and M. Zhang, "Automatically evolving rotation-invariant texture image descriptors by genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 1, pp. 83–101, 2017.

[22] H. Al-Sahaf, M. Zhang, A. Al-Sahaf, and M. Johnston, "Keypoints detection and feature extraction: A dynamic genetic programming approach for evolving rotation-invariant texture image descriptors," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 6, pp. 825 – 844, 2017.

[23] Y. Bi, B. Xue, and M. Zhang, "Genetic programming for automatic global and local feature extraction to image classification," in *IEEE Congress on Evolutionary Computation*, 2018, pp. 1–8.

[24] ——, "A gaussian filter-based feature learning approach using genetic programming to image classification," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2018, pp. 251–257.

[25] X. Yao, "A review of evolutionary artificial neural networks," *International Journal of Intelligent Systems*, vol. 8, no. 4, pp. 539–567, 1993.

[26] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[27] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2017, pp. 497–504.

[28] B. Wang, Y. Sun, B. Xue, and M. Zhang, "Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification," in *IEEE Congress on Evolutionary Computation*, 2018, pp. 1–8.

[29] Y. Sun, B. Xue, and M. Zhang, "A particle swarm optimization-based flexible convolutional auto-encoder for image classification," *arXiv preprint arXiv:1712.05042*, 2017.

[30] L. Rodriguez-Coayahuitl, A. Morales-Reyes, and H. J. Escalante, "Structurally layered representation learning: towards deep learning through genetic programming," in *European Conference on Genetic Programming*. Springer, 2018, pp. 271–288.

[31] H. Al-Sahaf, A. Song, K. Neshatian, and M. Zhang, "Extracting image features for classification by two-tier genetic programming," in *IEEE Congress on Evolutionary Computation*, 2012, DOI: 10.1109/CEC.2012.6256412.

[32] D. J. Montana, "Strongly typed genetic programming," *Evolutionary Computation*, vol. 3, no. 2, pp. 199–230, 1995.

[33] C. E. Thomaz, "Fei face database," *online: http://fei.edu.br/˜cet/facedatabase.html*, 2012.

[34] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proceedings of the Second IEEE Workshop on Applications of Computer Vision*, 1994, pp. 138–142.

[35] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets," in *Third IEEE International Conference on Automatic Face and Gesture Recognition*, 1998, pp. 200–205.

[36] G. Folego, O. Gomes, and A. Rocha, "From impressionism to expressionism: Automatically identifying van gogh's paintings," in *IEEE International Conference on Image Processing*, 2016, pp. 141–145.

[37] P. Mallikarjuna, A. T. Targhi, M. Fritz, E. Hayman, B. Caputo, and J.-O. Eklundh, "The kth-tips2 database," *Computational Vision and Active Perception Laboratory, Stockholm, Sweden*, 2006.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[39] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 886–893.

[40] Y. Jia and T. Darrell, "Heavy-tailed distances for gradient based image descriptors," in *Advances in Neural Information Processing Systems*, 2011, pp. 397–405.

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[42] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, jul 2012.