Genetic Programming with Image-Related Operators and A Flexible Program Structure for Feature Learning in Image Classification

Ying Bi, *Student Member, IEEE*, Bing Xue, *Member, IEEE*, and Mengjie Zhang, *Fellow, IEEE,*

***Abstract*—Feature extraction is essential for solving image classification by transforming low-level pixel values into high-level features. However, extracting effective features from images is challenging due to high variations across images in scale, rotation, illumination, and background. Existing methods often have a fixed model complexity and require domain expertise. Genetic programming with a flexible representation can find the best solution without the use of domain knowledge. This paper proposes a new genetic programming-based approach to automatically learning informative features for different image classification tasks. In the new approach, a number of image-related operators, including filters, pooling operators and feature extraction methods, are employed as functions. A flexible program structure is developed to integrate different functions and terminals into a single tree/solution. The new approach can evolve solutions of variable depths to extract various numbers and types of features from the images. The new approach is examined on 12 different image classification tasks of varying difficulty and compared with a large number of effective algorithms. The results show that the new approach achieves better classification performance than most benchmark methods. The analysis of the evolved programs/solutions and the visualisation of the learned features provide deep insights on the proposed approach.**

***Index Terms*—Genetic Programming; Feature Learning; Image Classification; Representation; Evolutionary Computation.**

I. Introduction

IMAGE classification aims to assign each image in the dataset with a class label from a set of predefined class labels based on the content in the image. Image classification is an important task in computer vision and machine learning with many real-world applications, such as medical diagnosis, self-driving and biological identification [1, 2, 3, 4]. However, many factors, including high image variations and high dimensionality of image data, make this task very challenging.

To solve image classification effectively, feature extraction is critical, where low-level pixel values are transformed into high-level features. However, feature extraction is challenging because of high inter-class and intra-class variations across images in scale, rotation, background, and lighting conditions. The desired features should contain invariant and discriminative information of the images so that the discrimination of different classes and the similarity of the same class can be preserved. These features could help build effective classifiers for classification [5]. Many methods have been developed to extract features from images [6], such as local binary patterns (LBP) [7], histogram of oriented gradients (HOG) [8] and scale-invariant feature transform (SIFT) [9]. These features are known as hand-crafted features [10], which are typically effective for particular tasks. For example, LBP features are effective for texture classification. When applied to new tasks with unknown types of images, these methods may not be effective [2]. Therefore, feature learning methods have been proposed to automatically learn features from images for classification [2, 11]. With a learning process, the features can be optimised or fine-tuned to achieve good performance. e.g., the maximum classification accuracy on the training set [11]. Compared with the hand-crafted features, the learned features are often more effective for a wide range of image classification [2, 12]. However, existing feature learning methods such as convolutional neural networks (CNNs) have limitations, e.g., requiring rich domain expertise to design the model for solving a specific task and requiring a large amount of computational resources and training data. Therefore, it is necessary to develop new feature learning methods to overcome these limitations for image classification.

Genetic programming (GP) is an evolutionary computation (EC) technique and can automatically evolve computer programs to solve problems using the principles of biological evolution and natural selection [13]. GP is well known for its flexible representation, good global search ability and high interpretability of the evolved solutions [14, 15, 16]. GP has achieved promising results in many tasks, such as symbolic regression, classification, scheduling, and image analysis [17]. Compared with other EC techniques, GP has a more flexible representation, i.e., tree-based representation [13, 18]. In tree-based GP, each individual is represented by a tree, where the root and internal nodes are functions/operators, and the leaf nodes are terminals [19]. Tree-based GP can evolve solutions without predefined structures. In other words, it can find solutions/models of various depths, i.e., shallow models for easy tasks and complex models for difficult tasks.

GP has been applied to image feature learning and achieved promising results [4]. In these methods, several image-related operators, e.g., Gaussian filter, mean filter, min filter, Laplacian

This work was supported in part by the Marsden Fund of New Zealand Government under Contracts VUW1509 and VUW1615, the Science for Technological Innovation Challenge (SfTI) fund under grant E3603/2903, the University Research Fund at Victoria University of Wellington grant number 223805/3986, MBIE Data Science SSIF Fund under the contract RTVU1914, and National Natural Science Foundation of China (NSFC) under Grant 61876169. This work of Ying Bi was supported by the China Scholarship Council (CSC)/Victoria University Scholarship.

The authours are with School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand (E-mail: Ying.Bi@ecs.vuw.ac.nz; Bing.Xue@ecs.vuw.ac.nz; Mengjie.Zhang@ecs.vuw.ac.nz).

filter, and Sobel filter, have been employed as GP functions to evolve solutions that extract high-level image features [2, 20]. Besides, other image-related operators, e.g., existing feature extraction methods, can be employed as GP functions to form the internal nodes of GP trees. However, this has not been extensively investigated. To effectively use image-related operators in GP, a program structure is typically needed to integrate different functions and terminals into a single tree. Several GP methods with different program structures, e.g., multi-tier [21], two-tier [22] and multi-layer [2], have been developed. But these program structures are often restricted in a certain way and cannot be used when new image-related operators are introduced in GP. More importantly, most of these program structures are fixed [20, 21, 22, 23], and they extract features in a predefined procedure, which may limit their performance. Furthermore, most GP-based methods only address binary image classification [20, 21, 22, 24]. In these methods, the output of a GP tree from an image is a floating-point number, which is often compared with a predefined threshold to determine which class the image belongs to. For example, if the output is smaller than zero, the image belongs to the negative class. Otherwise, the image belongs to the positive class. However, most image classification tasks are multi-class classification so that these methods cannot be directly used. In addition, multi-class classification often involves more image variations and separations and is often more difficult. Therefore, this paper proposes a new GP approach, by addressing these limitations, to feature learning for image classification, including multi-class classification.

The goal of this paper is to develop a new GP-based approach with image-related operators and a flexible program structure to feature learning for different image classification tasks. The new approach is called FGP in short. A flexible program structure, a new function set including image-related operators, and a new terminal set will be developed in FGP. The new approach will be evaluated on 12 benchmark datasets of varying difficulty and compared with a large number of state-of-the-art methods. Further analysis will be conducted to provide an in-depth understanding of the new approach.

The characteristics of FGP can be summarised into the following four aspects:

1) FGP can automatically evolve solutions/trees of variable depths to extract features from the images. The complexity of the FGP solutions for different tasks can be different, which is more flexible than a predefined model complexity, such as in CNNs.
2) FGP can produce various types and numbers of features from images. The produced features may be generated through filtering, pooling, or feature extraction using different image-related operators. The features can be from filtering/pooling, or from feature extraction, or from filtering/pooling and feature extraction simultaneously, where current GP-based methods cannot achieve this.
3) FGP can be easily applied to different image classification tasks to achieve good classification performance. The experimental results on 12 benchmark datasets of varying difficulty show that FGP can achieve better performance than a number of effective algorithms.
4) The solutions evolved by FGP can be visualised to understand what image-related operators are used in the trees/solutions and what types of features are produced by the trees. The visualisation of the learned features provides further insights on the FGP approach.

II. Background and Related Work

This section introduces the basics of commonly used image-related operators. Then it discusses existing methods for image classification, including traditional methods, NN-based methods and GP-based methods. The current limitations of these methods are summarised.

A. Image-Related Operators

1) Per-pixel Transformation: An image can be denoted as $P_{M\times N}$ if the image is gray-scale. Per-pixel transformation represents that each $p_{m,n}$ in P is changed to $x_{m,n}$ to form a new array $X_{M\times N}$. Many image-related operators can achieve this, such as filtering or convolution.

Filtering is often used in image preprocessing such as using a Gaussian filter to reduce the noise [6]. In this process, a filter kernel $F_{(2a+1)\times(2b+1)}$ with $(2a+1)\times(2b+1)$ weights are employed to convolve the image P. The two-dimensional discrete convolution process is denoted in Eq. 1. Commonly used filters are Gaussian filter, derivatives of Gaussian, Laplacian filter, Laplacian of Gaussian (LoG) filter, Gabor filter, and Sobel edge detector [6].

$$x_{m,n} = \sum_{f_1=-a}^{a}\sum_{f_2=-b}^{b} p_{m-f_1,n-f_2}F_{f_1,f_2} \tag{1}$$

2) Image Feature Descriptors: Feature description transforms an image into a set of features. Well-known methods are LBP [7], HOG [8], SIFT [9], and Gabor features [3]. LBP is a simple but effective method for texture description. LBP compares each central pixel in a small window with its neighbour pixels to obtain a binary code. Then each central pixel is changed to a decimal number according to the binary code and predefined weights. In general, the histogram of the LBP image is extracted as features. More details and other versions of LBP can be seen in [7]. HOG is well known for object shape and appearance description [8]. The main idea of HOG is to extract locally normalised histograms of gradient magnitudes and orientations in each overlapping block as features. The SIFT method [9] is a popular method to describe local features as it detects keypoints from the image. From each detected keypoint, it extracts 128 histogram features of gradient magnitude and direction. Without keypoint detection, a dense SIFT method was developed for feature extraction to reduce the computational complexity [25].

B. Related Work on Image Classification

1) Traditional Methods: Traditional methods use a feature description method to extract features from images and feed them into classification algorithms for classification. Chapelle et al. [26] developed a heavy-tailed RBF's kernel function in support vector machines (SVMs) for image classification based on histogram features. The results showed that the classification performance can be improved by using the developed kernel function in SVMs. Sergyan [27] extracted simple histogram features and employed *k*-nearest neighbour (KNN) for image classification, which achieved good performance. Bosch et al. [28] developed a method to automatically detect regions of interest and extract shape and appearance features from the regions for object classification. However, most of these methods cannot effectively solve other image classification tasks since they use simple hand-crafted features.

2) Neural Network (NN)-based Methods: In recent decades, NN-based methods, e.g., auto-encoders (AEs), CNNs, deep belief networks (DBNs), and deep Boltzmann machines (DBMs), have obtained promising results in image classification. Among these methods, CNN is the most commonly used method for image classification. Qian and Zhang [29] developed a feedforward convolutional conceptor neural network (FCCNN) by integrating components of CNN, principal component analysis (PCA), binary thresholding (BT) and non-temporal conceptor classifiers. The performance of FCCNN has been examined on the MNIST variant datasets. Li and Gong [30] proposed a self-paced CNN (SPCN) by assigning weights to the training samples during the learning process to enhance the learning robustness of CNN. The method has gained better performance than a number of state-of-the-art algorithms on six benchmark datasets. Bruna and Mallat [31] proposed an invariant scattering convolution network (ScatNet) for image classification by cascading wavelet transforms and modulus pooling operators. This method has achieved better performance than a Gaussian kernel SVM and a generative PCA classifier on digital recognition and texture classification. Based on the concept of PCA, Chan et al. [32] developed the famous PCA network (PCANet) for feature learning, where a set of filters generated by PCA were used in the convolutional layer, and binary hashing and blockwise were employed to obtain the final output features. PCANet has shown promising results in many well-known benchmark datasets, including face recognition and digit recognition. Rifai et al. [33] developed contractive auto-encoders (CAE) by using a new cost function with a well-chosen penalty term. This method has been examined on seven datasets and compared with other types of AE, including three-layers stack AE (SAE-3) and three-layers denoising AE with binary masking noise (DAE-b-3). As variants of NNs, DBN and DBM have also been applied by Larochelle et al. [34] to digit recognition. However, NNs have limitations such as requiring a large number of computing resources and training instances and requiring rich domain knowledge to design the models.

3) GP-based Methods: GP has been applied to solve image classification by simultaneously performing feature extraction and feature construction. The multiple subtasks of dealing with image classification, including region detection, feature extraction, feature construction, and image classification, can be integrated into a single GP tree using strongly typed GP (STGP) [35]. The final output of a GP tree is a number/feature, which can be used for classification using a predefined threshold. To the best of our knowledge, the first method is the multi-tier GP (known as 3TGP) proposed by Atkins et al. [21], where high-level features are learned through an image filtering tier, an aggregation tier and a classification tier. Al-Sahaf et al. [22] improved 3TGP by simplifying the structure to an aggregation tier and a classification tier (2TGP) to perform region detection, feature extraction, feature construction, and image classification, simultaneously. Bi et al. [20] developed a multi-layer GP method (MLGP) with the utilisation of image-related operators to extract and construct high-level features for image classification. These methods have achieved good performance on binary image classification, but their performance has not been investigated on multi-class image classification.

Several GP methods have been developed to learn multiple features from images and a traditional classification algorithm have been employed for classification. Al-Sahaf et al. [36] proposed a GP-based method to automatically produce a set of features using an LBP-similar manner for texture image classification. However, the number of features generated by this method is fixed. To improve this, a dynamic method was developed in [12] to learn a dynamic number of image features for texture classification. In this method, a *root* function that accepts a flexible number of child nodes was developed. However, these two methods are for texture description and their performance has not been examined on other types of datasets. Shao et al. [2] proposed a multi-objective GP (MOGP) method with an input layer, a filtering layer, a pooling layer, and a concatenation layer to transform images into high-level features. This method has achieved better performance than the methods using hand-crafted features and simple CNNs on four different datasets. But this method has a fixed program structure to produce features from filtering and pooling, which could not produce invariant features to achieve good performance on difficult datasets. Bi et al. [37] developed a GP method to automatically and simultaneously learn features and evolve ensembles for image classification. This method built an ensemble of classifiers to solve image classification. But this method has shown inferior performance on large image classification datasets. Bi et al. [38] proposed a GP algorithm to simultaneously learn features and evolve ensemble for image classification. This method uses classification algorithms and image-related operators to evolve ensembles of classifiers for classification. This method have achieved promising results on several datasets. But this method focused on ensemble learning and the models/solutions formed by a number of various classifiers are complex and difficult to explain.

In summary, although many GP-based methods have been developed for image classification [2, 36], most methods are only for binary image classification due to easy implementation and the nature of the GP program's output [1, 20, 21, 22, 24]. However, most image classification tasks are multi-classification, which requires different algorithm designs to solve it. The GP methods in [2, 12, 36, 39, 40] have shown promising results on multi-class or binary image classification by learning multiple features. However, these methods have their limitations, such as have a fixed program structure and are only for texture classification. Due to the current limitations, the potential of GP in feature learning has not been extensively investigated. Moreover, very few GP-based methods have been examined on large benchmark datasets and compared with state-of-the-art algorithms for image classification. Therefore, this paper proposes a new GP approach to feature learning with a flexible program structure and image-related operators for different types of image classification tasks, including multi-class classification of large datasets.

III. The Proposed Approach

In this section, the proposed approach with a flexible program structure, named as FGP in short, is described in detail, including the algorithm overview, the flexible program structure, the function set, and the terminal set.

A. Algorithm Overview

The framework of the FGP approach is outlined in Algorithm 1. FGP starts with population initialisation, where N (population size) individuals/trees are randomly generated using a commonly used tree generation method: *ramped half-and-half*. Each FGP individual/tree is built by selecting functions from the new function set to construct internal/root nodes and selecting terminals from the new terminal set to construct the leaf nodes. Each individual is then evaluated through the fitness evaluation process to have a fitness value. After fitness evaluation, the best individual and a developed hash table, $Cache\_Table$, are updated. The $Cache\_Table$ is used to avoid evaluating the individuals that have been evaluated in past generations. More details about $Cache\_Table$ will be introduced in the following paragraph. At each generation, the selection method and three genetic operators, i.e., *subtree crossover, subtree mutation* and *elitism*, are used to generate a new population to replace the current one. The evolutionary process is terminated when the maximum number of generations is reached. Finally, the best individual is returned.

Algorithm 1: Framework of FGP

```
Input  : $X\_train$: the training images; $Y\_train$: the labels of
         the training images.
Output : $Best\_Individual$: the best individual.
$Cache\_Table \leftarrow \emptyset$;
$P_0 \leftarrow$ Initialise the population using the ramped half-and-half
  method according to the new program structure, the new function
  and terminal sets;
Evaluate $P_0$ using Algorithm 2;
Update $Best\_Individual$ and $Cache\_Table$;
$g \leftarrow 0$;
while $g < G$ do
    $I \leftarrow$ The best individuals of $P_g$ using elitism operator;
    $S \leftarrow$ Individuals selected from $P_g$ using tournament selection;
    $O_{g+1} \leftarrow$ Offspring generated from $S$ using subtree crossover
      and subtree mutation operators;
    Evaluate the fitness of each individual $p$ in $O_{g+1}$ using
      Algorithm 2;
    $P_{g+1} \leftarrow O_{g+1} \cup I$;
    Update $Best\_Individual$ and $Cache\_Table$;
    $g \leftarrow g + 1$
end
Return $Best\_Individual$.
```

A new fitness evaluation process is developed in FGP to evaluate each individual, as described in Algorithm 2. On image data, GP is often known as a computationally expensive method, especially when the number of instances is large. To avoid evaluating the same subtrees, subtree caching strategy has been developed in GP on image data [41]. Inspired by this, a hash table $Cache\_Table$ is employed in FGP to store individuals and their fitness values so that the repeated individual can be directly assigned a fitness value without evaluation using the training data. To balance the search time in $Cache\_Table$ and the evaluation time of an individual, only N_c best individuals and the previous population (P_g) are stored in $Cache\_Table$. With the $Cache\_Table$, the new fitness evaluation process is described in Algorithm 2. It starts with checking whether the individual is in the $Cache\_Table$. If the individual is in the $Cache\_Table$, the fitness value is directly assigned to the individual. Otherwise, the individual is evaluated on a training set using a linear SVM. The linear SVM is chosen because it is commonly used for image classification [2]. In this process, the FGP individual transforms each image in X_train to a number of features to form $X\_features$. Then $X\_features$ is normalised using the min-max normalization method and fed into a linear SVM using the stratified k-fold cross-validation method. The stratified k-fold cross-validation method splits the dataset ($X\_features$ and class labels) into k folds by preserving the class ratio. Each time $k - 1$ folds are used to build an SVM classifier and the classifier is tested on the remaining one fold. The average test accuracy of the k folds is set as the fitness value to the individual. In FGP, k is set to be 5 instead of 10 used in [2] to reduce the computational cost of the fitness evaluation.

Algorithm 2: Fitness Evaluation

```
Input  : $Cache\_Table$: the hash table to store evaluated
         individuals and their fitness values; $X\_train$: the training
         images; $Y\_train$: the labels of the training images; $p$: the
         individual to be evaluated.
Output : The fitness value for $p$: $f(p)$.
if $p$ in $Cache\_Table$ then
    $f(p) \leftarrow$ the fitness value of $p$ in $Cache\_Table$;
else
    Use $p$ to transform $X\_train$ into features $X\_features$;
    Normalise $X\_features$ using the min-max normalisation
      method;
    Feed normalised $X\_features$ and $Y\_train$ into a linear SVM
      using stratified $k$-fold cross-validation;
    $f(p) \leftarrow$ average test accuracy of $k$ folds
end
Return $f(p)$.
```

Besides the above feature learning process, the main properties of FGP that lead to its success on feature learning and difference from other GP-based methods are the flexible program structure, the new function set and the new terminal set. The following subsections will introduce these properties.

B. Flexible Program Structure

A flexible program structure is developed in FGP to integrate functions and terminals into a single tree. The development of the new program structure is based on three motivations. First, in state-of-the-art GP methods on feature learning in [2, 39], the program structure has two main components, the filtering layer and the pooling layer, connecting in a bottom-up manner. This program structure is fixed, which may not

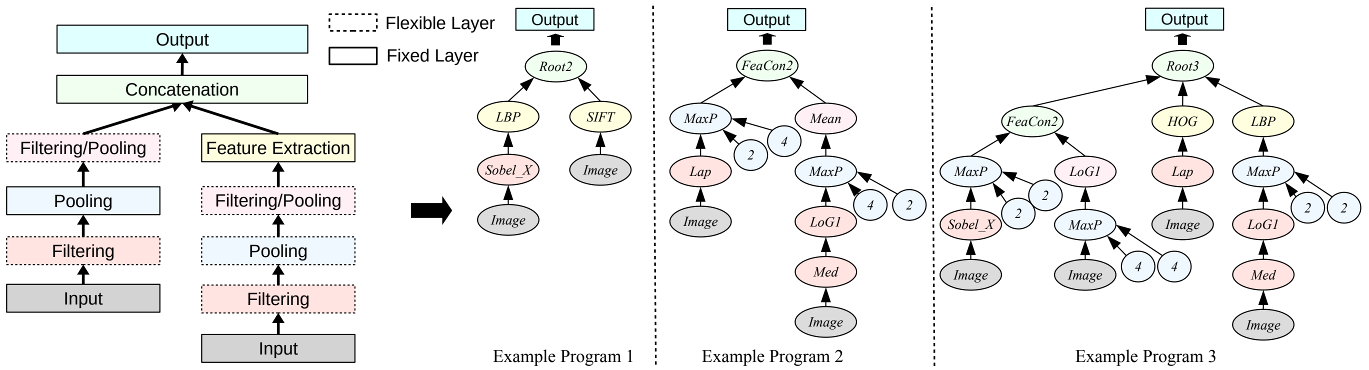


Fig. 1. The program structure of the FGP approach (left) and three typical example programs that can be evolved by FGP (right).

be effective for learning invariant features as learned by CNN using multiple layers' transformations. To this end, we relax this constraint to design a more flexible program structure, which allows FGP to evolve programs with multiple layers of filtering and pooling. Second, integrating feature extraction functions into GP for feature learning has achieved promising results in [40]. However, this method has a limitation of fixed tree depth, and the learned features may not be invariant to noise without filtering/denoising process. To address this, a flexible filtering/pooling layer is added between the input layer and the feature extraction layer. Third, hybrid features/representations, i.e., combining features extracted from filtering/pooling and features extracted from traditional feature extraction methods, have been seen in CNN-based methods with promising performance [42]. But no existing GP methods can achieve this. Therefore, the flexible program structure of FGP can combine the two different types of features to produce hybrid features.

The program structure of FGP is based on STGP [35], which has types constraint on functions (input types and output types) and terminals (output types). In STGP, each function can only use particular functions or terminals as child nodes, where its input types must be the same as the output types of its child nodes. Based on STGP, a program structure is developed in FGP to integrate functions and terminals of different types into trees. The new program structure and three typical example programs are shown in Fig. 1. The new program structure has several different layers, i.e., an input layer, filtering layers, pooling layers, a feature extraction layer, a concatenation layer, and an output layer. The input layer feeds the image and ephemeral random constants into the FGP system. The filtering layer performs filtering operations or other operations on the image. The pooling layer conducts max-pooling to the image with size reduction, which is in contrast to that in [2, 39]. The feature extraction layer extracts features from the image using several well-known feature extraction methods. The concatenation layer concatenates/combines features from different processes, i.e., filtering/pooling and feature extraction into a feature vector to form the output of the FGP system.

More importantly, as shown in Fig. 1, the layers circled with dash line are flexible, indicating that they may be in an FGP program. These flexible layers allow the FGP program to have multiple filtering and pooling layers to extract features, which are similar to those in CNNs. The layers that are circled with the line are fixed layers to make sure that there are feature transformations from the input to the output. Using this flexible program structure, three typically different types of features can be produced by FGP. The first is to produce the combined features from the feature extraction process as the *Example Program 1* shown in Fig. 1. The second is to produce the combined features from the filtering and/or pooling processes as the *Example Program 2* shown in Fig. 1. The third is to produce the combined features from the feature extraction process and the filtering and pooling processes, as shown in the *Example Program 3* in Fig. 1.

This program structure allows FGP to evolve shallow trees that contain a few functions or to evolve deep trees with multiple layers of pooling and/or filtering. With this program structure, FGP can produce various types and numbers of features, which are flexible for solving different image classification tasks. Associated with this program structure, a number of functions and several terminals are employed in FGP, which will be described in the following subsections.

C. Function Set

Many operators and methods have been developed for feature detection and description. These operators can give insights on what type of features are detected and why they are effective. Therefore, a set of well-known image-related operators is used in the function set of FGP. Based on the program structure, these functions are classified into filtering functions, pooling functions, feature extraction functions, and feature concatenation functions.

1) ***Filtering Functions****:* There are 19 functions employed in the filtering layer of FGP, as listed in Table I. The Gau function takes an image and standard deviation σ as inputs and returns an image convolved by a Gaussian kernel. $GauD$ has three parameters, i.e., standard deviation σ, o_1 and o_2. The parameters o_1 and o_2 represent orders of the derivative along the X and Y axis, respectively. The $Gabor$ filter is generated by a Gabor wavelet function. It has parameters θ and f, which indicate the orientation of the kernel and the wavelength ($\lambda = 1/f$) of the sinusoid function in the Gabor wavelet function. The Lap function is generated by discretising and approximating the Laplacian operator, and it can detect the flat areas or edges. The $LoG1$ and $LoG2$ functions convolve the Laplacian filter with the Gaussian function, which can reduce

TABLE I
FILTERING FUNCTIONS

Function	Input	Output	Function Description
Gau	1 image, σ	1 image	Gaussian filter with standard deviation σ
$GauD$	1 image, σ, o_1, o_2	1 image	Derivatives of Gaussian filter
$Gabor$	1 image, θ, f	1 image	Gabor filter with θ orientation and f frequency ($\frac{1}{\lambda}$)
Lap	1 image	1 image	Laplacian filter
$LoG1$	1 image	1 image	Laplacian of Gaussian filter with $\sigma = 1$
$LoG2$	1 image	1 image	Laplacian of Gaussian filter with $\sigma = 2$
$Sobel$	1 image	1 image	Sobel edge detector
$SobelX$	1 image	1 image	Sobel filter along the X axis
$SobelY$	1 image	1 image	Sobel filter along the Y axis
Med	1 image	1 image	3×3 median filter
$Mean$	1 image	1 image	3×3 mean filter
Min	1 image	1 image	3×3 min filter
Max	1 image	1 image	3×3 max filter
$LBP\text{-}F$	1 image	1 image	Return LBP image
$HOG\text{-}F$	1 image	1 image	Return HOG image
$W\text{-}Add$	2 images, n_1, n_2	1 image	Add two weighted images
$W\text{-}Sub$	2 images, n_1, n_2	1 image	Subtract two weighted images
$ReLU$	1 image	1 image	The rectified linear unit
$Sqrt$	1 image	1 image	Sqrt an image

noise in the image. The standard deviation of the Gaussian function in $LoG1$ and $LoG2$ is set as 1 and 2, respectively. Among these filters, the Gau, Med and $Mean$ filters are often employed for image denoising and smoothing. The filters, including $GauD, Lap, LoG1, LoG2, Sobel, SobelX$, and $SobelY$, can detect edges or flat areas in the image. The kernel sizes for the $Mean$, Med, Min, and Max functions are 3×3, which is a commonly used kernel size. The kernel sizes for the other filters are based on their parameters or default settings. For example, the kernel sizes of the Gau and $GauD$ functions are related to their parameter σ; and the kernel size of the $Sobel$, $SobelX$ and $SobelY$ functions is 3×3.

Besides the above filters, the $LBP\text{-}F$, $HOG\text{-}F$, $W\text{-}Add$, $W\text{-}Sub$, $ReLU$, and $Sqrt$ functions listed in Table I are also employed. The $LBP\text{-}F$ and $HOG\text{-}F$ functions return a LBP image and a HOG image for an input image, respectively. The two functions produce high-level feature maps that may be informative. The $W\text{-}Add$ and $W\text{-}Sub$ functions are used to add or subtract two weighted images with different or the same sizes, where the weights are n_1 and n_2. In the case where the sizes of the two images are not the same, $W\text{-}Add$ and $W\text{-}Sub$ overlap the image pixels at coordinates (0, 0), cut the exceeding part of the larger images, and perform the add or subtract operation. $ReLU$ is the rectified linear unit. $Sqrt$ calculates the square root of each pixel value in an image and is protected by returning 1 if the pixel value is negative. $ReLU$ and $Sqrt$ can rescale the input image by transforming the pixel values from the negative to non-negative.

*2) **Pooling Functions**:* Commonly used pooling functions are max-pooling and average-pooling. Max-pooling returns the maximum value of each sliding window, while average-pooling returns mean value of the sliding window. The important features, e.g., edges, can be extracted by the max-pooling function but may be smoothed by the average-pooling function. Therefore, only max-pooling (simplified as $MaxP$) function is employed. The $MaxP$ function takes three arguments as inputs, i.e., an image and the kernel sizes, k_1, k_2, and returns a smaller image. The $MaxP$ function in FGP not only extracts important features but also reduces the dimensionality of the features. Note that k_1 and k_2 are two parameters of $MaxP$ and are used as two ephemeral random constants of FGP. The values of k_1 and k_2 are randomly selected from a predefined range at the initialisation step and can be mutated by mutation operator during the evolutionary process.

*3) **Feature Extraction Functions**:* The feature description methods introduced in Section II-B1 can be employed as GP functions to extract informative features. To reduce the search space of FGP, the three most commonly used methods, i.e., HOG, LBP, and SIFT, are employed for feature extraction. Table II lists the details of these functions.

TABLE II
FEATURE EXTRACTION FUNCTIONS

Function	Input	Output	Description
$SIFT$	1 Image	1 Vector	SIFT descriptor. 128 features are extracted from the image [25]
LBP	1 Image	1 Vector	LBP descriptor. It extracts 59 uniform LBP histogram features. In the LBP method, the radius is 1.5 and the number of neighbours is 8 [7]
HOG	1 Image	1 Vector	HOG descriptor. In HOG, the orientation is 9, the cell size is 8×8 and the block size is 3×3 [8]. The mean value of each 4×4 grid is extracted from an HOG image

*4) **Concatenation Functions**:* To concatenate features produced by different functions, five different concatenation functions ($Root2$, $Root3$, $Root4$, $FeaCon2$, and $FeaCon3$) are employed and developed. The descriptions of these functions are listed in Table III. Each concatenation function can be used as the root node of a program tree or a child node of another concatenation function. This means that the tree depth of the concatenation layer is flexible. With these functions, FGP trees can output various numbers of features from an input image.

TABLE III
CONCATENATION FUNCTIONS

Function	Input	Output	Description
$RootX$	2/3/4 Vectors	1 Vector	Concatenate vectors to a vector
$FeaConY$	2/3 Images	1 Vector	Convert images to a vector by concatenating each row

D. Terminal Set

The terminal set of FGP contains the input image ($Image$) and the parameters for the functions, i.e., σ, o_1, o_2, θ, f, n_1, n_2, k_1, and k_2. More details of them are listed in Table IV. The $Image$ terminal represents the input image, which is a 2D array and the values in the array are normalised into $[0, 1]$ dividing by 255. The other terminals are ephemeral random constants of FGP and only appear in the trees where the corresponding functions are used. The values of these terminals are randomly selected from a predefined range at the initialisation step and can be modified by the mutation operator during the evolutionary process.

TABLE IV
TERMINAL SET

Terminal	Type	Description
$Image$	Image	The input gray-scale image (2D array containing image pixel values in the range of $[0,\ 1]$)
σ	Integer	The standard deviation of the Gaussian filter. It is randomly initialised in the range of $\{1, 2, 3\}$
o_1, o_2	Integer	The order of the Gaussian derivatives. They are randomly initialised in the range of $\{0, 1, 2\}$
θ	Float	The orientation of the $Gabor$ filter. It is in the range of $[0,\ 7\pi/8]$ with a step of $\pi/8$ [3]
f	Float	The frequency of the $Gabor$ filter. It equals to $\frac{\frac{\pi}{2}}{\sqrt{2}^{v}}$, where v is an integer in the range of $\{0, 1, 2, 3, 4\}$ [3]
n_1, n_2	Float	The parameters for the $W-Add$ and $W-Sub$ functions. They are randomly generated in the range of $[0,\ 1)$
k_1, k_2	Integer	The kernel size of the $MaxP$ function. They are in the range of $\{2, 4\}$

IV. Experiment Design

In this section, the design of the experiments is described, including benchmark datasets, benchmark methods, parameter settings, and test process.

A. Benchmark Datasets

Twelve widely used image classification datasets are employed to examine the performance of the FGP approach. The details of the datasets are listed in Table V. These datasets represent different types of image classification tasks, i.e., facial expression classification (FEI_1 [43] and FEI_2 [43]), face recognition (ORL [44]), texture classification (KTH [45]), scene classification (FS [46]), digit recognition (MB [34], MRD [34], MBR [34], and MBI [34]), and object classification (Rectangle [34], RI [34] and Convex [34]). The images in these datasets are gray-scale or converted to gray-scale images to reduce the computational cost. The example images from the 12 datasets are shown in Fig. 2–4. The main reason for selecting these datasets is that they represent a wide range of classification tasks or different challenges in image classification, such as background change or additional noise. These datasets are very suitable for evaluating the performance of the proposed approach on different types of images. It is noticeable that the MB, MRD, MBR, MBI, Rectangle, RI, and Convex datasets are large datasets with 50000 images for testing. Very few GP-based methods have been tested on these datasets. In addition, the performance of the proposed FGP approach has not been tested on the well-known large datasets such as ImageNet due to the high computational cost.

The FEI_1 and FEI_2 datasets are facial expression classification tasks [43], containing facial images with smiling or natural expressions sampled from 200 different people. The ORL dataset [44] has 40 classes of different facial images and each class only has 10 image, which is very small. The KTH dataset [45] is a texture classification dataset, containing 810 images equally in 10 classes. The images are sampled on nine different scales with three poses under four illumination conditions, which indicates the difficulty of classification. The FS dataset [46] includes 3859 natural scene images belonging to 13 classes, such as the scene of the forest, street, highway, and coast, respectively. FS is a challenging task of understanding the context of the complex scene.

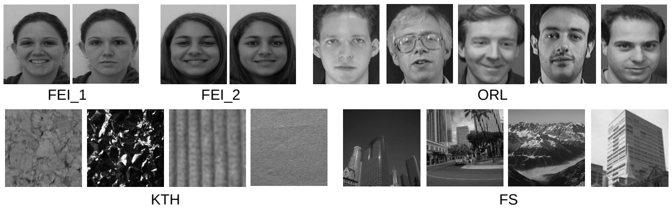


Fig. 2. Example images from the FEI_1, FEI_2, ORL, KTH, and FS benchmark datasets, respectively.

TABLE V
SUMMARY OF THE 12 BENCHMARK DATASETS

No.	Dataset	Image Size	Training Set Size	Test Set Size	#Class
1	FEI_1	60×40	150 (75)	50	2
2	FEI_2	60×40	150 (75)	50	2
3	ORL	50×55	240 (6)	160	40
4	KTH	50×50	480 (48)	330	10
5	FS	55×55	1300 (100)	2559	13
6	MB	28×28	12000	50000	10
7	MRD	28×28	12000	50000	10
8	MBR	28×28	12000	50000	10
9	MBI	28×28	12000	50000	10
10	Rectangle	28×28	1200	50000	2
11	RI	28×28	12000	50000	2
12	Convex	28×28	8000	50000	2

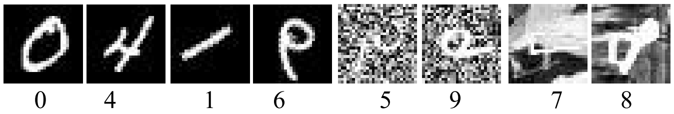


Fig. 3. Example images from the MB, MRD, MBR, and MBI benchmark datasets, respectively. Each dataset has two example images and the corresponding class labels (digits) are under these images.

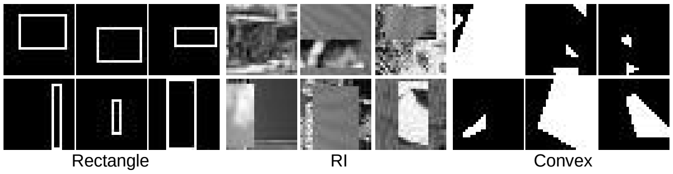


Fig. 4. Example images from the Rectangle, RI and Convex benchmark datasets, respectively. The first row shows the negative class, and the second row shows the positive class.

The datasets 1-5 do not have public training and test sets so that they are split using commonly used proportions, as shown in Table V. In contrast, the MB, MRD, MBR, MBI, Rectangle, RI, and Convex datasets [34] have separated training and test sets [1], which can be directly used in experiments. The MB dataset is a subset of the famous MNIST benchmark dataset.

[1] The training and test sets can be downloaded from http://www.iro.umontreal.ca/ lisa/twiki/bin/view.cgi/Public/PublicDatasets

The training set of MB has 12000 images, and the test set has 50000 images. The MRD, MBR and MBI datasets are variants of MB by adding additional factors of variations, including rotation, random background and image background. MRD contains digit images with rotation by an angle generated uniformly between 0 and 2π. MBR has images with random background, and MBI has images with adding random images as their background. The MRD, MBR and MBI datasets are more difficult than MB due to these additional variations. The Rectangle, RI and Convex datasets are object classification. Rectangle and RI have images with rectangle objects, and the tasks are to recognise whether each rectangle in an image has a larger width or length. RI is more difficult than Rectangle due to an additional random background. The Convex dataset has images with convex or non-convex, i.e., two classes.

B. Benchmark Methods

To show the effectiveness of the proposed FGP approach, a large number of benchmark methods are used for comparisons. Because the datasets 1-5 do not have public training and test sets, we need to split them and run all the experiments of the benchmark methods to make sure that the reported classification results are on the same test sets. For the datasets 6-12, the results of many methods have been reported on the same public test sets. These results can be directly used for comparisons. Therefore, the benchmark methods on the datasets 1-5 are different from those on the datasets 6-12.

On datasets 1-5, 13 different methods are used as benchmark methods. They are the EGP method [37], the IEGP method [38], six commonly used classification algorithms using raw pixels, three SVM methods using different pre-extracted features, and two CNN-based methods with different architectures. The EGP and IEGP methods are able to automatically learn features from images and evolve an ensemble of classifiers for classification [37, 38]. The six commonly used classification algorithms are linear SVMs, KNN, logistic regression (LR), RF, adaptive boosting (AdaBoost), and extremely randomised trees (ERF). These methods use the normalised raw pixel values of images as inputs to train the classifiers. The three SVM methods are LBP+SVM, HOG+SVM and SIFT+SVM [11], which use LBP, HOG, or SIFT features as inputs of SVMs for classification. The LBP, HOG and SIFT features are extracted by the methods described in Table II. The final two benchmark methods are a five-layer CNN (CNN-5) [2] and an eight-layer CNN (CNN-8) [47]. CNNs are well known for image classification so that it is necessary to compare FGP with CNNs.

On datasets 6-12, 20 existing methods are used as benchmark methods. These methods have been reported recently or are representative methods for image classification. The classification results of these 20 methods are collected from the corresponding papers. These methods are SVM+RBF [34], SVM+Poly [34], SAE-3 [33], DAE-b-3 [33], CAE-2 [33], SPAE [48], RBM-3 [33], ScatNet-2 [31, 32], RandNet-2 [32], PCANet-2 (softmax) [32], LDANet-2 [32], NNet [34], SAA-3 [34], DBN-3 [34], FCCNN [29], FCCNN (with BT) [29], SPCN [30], EvoCNN [49], EGP [37], and IEGP [38]. Most of these methods are NN-based methods, which have been introduced in Section II-B2. Note that in several methods, including SVM+RBF, SVM+Poly, NNet, SAA-3, and DBN-3, model selection has been conducted to find the best parameters using a training set and a validation set. Then these methods with the best parameters were trained using the training set and tested on the test set. The EvoCNN method and the IEGP method have achieved the best performance on some of these benchmark datasets. The EvoCNN method is a deep learning method, which uses an evolutionary algorithm to automatically search for the best architectures of CNNs. The IEGP method is an ensemble method for image classification, which includes a number of difference classifiers.

C. Parameter Settings

The parameter settings for FGP are based on the commonly used settings in the community of GP [50]. In FGP, the maximum number of generations G is 50 and the population size N is 500. The crossover rate Pc is 0.8, the mutation rate Pm is 0.19, and the elitism rate Pe is 0.01. The selection method is the tournament selection with size 7. The tree depth is between 2-6 at the initialisation step, and the maximum tree depth is 8. Note that in FGP, which is based on STGP, the type constraint is more important than the depth constraint. Therefore, a tree may have a depth of over eight. As a new parameter, Nc, the number of individuals stored in the $Cache\_Table$ is set to $6*N$ (N for the previous population and $5*N$ for the best individuals at the past generations) based on the assumption that $6*N$ is efficient and effective. In general, the value of Nc can be any number but a too big one may lead to a long searching time in $Cache\_Table$, and a too small one only stores very limited individuals, which makes the $Cache\_Table$ not very useful. Note that the parameter settings for FGP are the same on the 12 different datasets for generality, although performing parameter tuning for FGP could further improve its performance on these datasets. The analysis of the parameter settings for FGP is presented in the supplementary materials due to the page limit.

The parameter settings for the six classification algorithms SVM, KNN, LR, RF, AdaBoost, and ERF refer to [51, 52]. In KNN, the number of nearest neighbours is set to 1 [12]. In SVM and LR, the penalty parameter C is set to 1 [51]. In RF, ERF and AdaBoost, the number of trees is set to 500, and the maximum tree depth is set to 100 [52]. In CNN-5 and CNN-8, the commonly used ReLU function is used as the activation function and softmax is used for classification [47]. To avoid overfitting, dropout is added after the pooling layer and the first fully connected layer with 0.25 and 0.5 probabilities, respectively [53]. The maximum number of epochs is set to 500, and the batch size is set to 128, which is commonly used.

The implementation of FGP is based on the DEAP (*Distributed Evolutionary Algorithm in Python*) [54] package. The implementations of the classification algorithms are based on the scikit-learn [55] package and the implementations of CNNs are based on Keras [47]. The experiments of FGP on each dataset conduct 30 independent runs to avoid the experimental bias, which follows the conventions of the EC communities.

Each of the benchmark methods has been run 30 times on datasets 1-5 to obtain the classification results.

D. Test Process

The test procedure of the best individual/tree found by FGP is shown in Fig. 5. In the process, the training set and the test set are used (note that only the training sets are employed during the evolutionary process). The best FGP individual/tree is used to transform the training and test images into features. Then the transformed training and test sets are normalised using the min-max normalisation method to scale the features [56]. Note that the normalisation of the test set is based on the minimal and maximum values of each feature in the training set. The normalised training set is used to train a linear SVM classifier. The trained classifier is tested on the normalised test set to obtain the classification error rate.

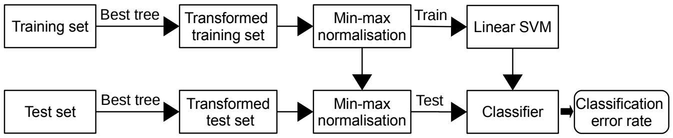


Fig. 5. The test procedure of the best individual/tree found by FGP on the test set.

V. Results and Discussions

In this section, the experimental results of FGP on the 12 benchmark datasets are analysed and compared with that of a large number of benchmark methods.

A. Classification Results on Datasets 1-5

The classification results on datasets 1-5, i.e., FEI_1, FEI_2, ORL, KTH, and FS, are listed in Table VI. The results are the minimal classification error rate (Min), the average classification error rate of 30 runs and the standard deviation (Mean±St.dev). To show the significance of performance improvement, the Wilcoxon rank-sum test with a 95% significance interval is used to compare FGP with a benchmark method. In Table VI, the symbols "+" and "–" indicate that FGP achieves significantly better and worse results than the compared method. The symbol "=" denotes that FGP achieves similar results to the compared method. In Table VI, the best error rate and the average error rate on each dataset are highlighted in bold. The final row of each bock in the table summaries the overall results of the significance test.

From Table VI, it can be found that FGP achieves significantly better performance in 52 comparisons out of the 65 comparisons. More importantly, FGP achieves significantly better or similar results than any of the 13 benchmark methods on the ORL and KTH datasets, which are the face recognition and texture classification tasks. On the FEI_1 and FEI_2 datasets, which are facial expression classification tasks, FGP is significantly better than seven methods on FEI_1 and than nine methods on FEI_2. On the ORL dataset, FGP not only obtains the minimal error rate but also achieves the best mean error rate among all the methods. The ORL dataset is a small dataset with six training images per class. The results indicate that FGP is more effective than IEGP when learning from a small number of training images. On the KTH dataset, FGP achieves the best minimal error rate and slightly worse mean error rate (0.36% higher) than the best error rate obtained by IEGP. On the FS dataset, which is a difficult dataset, FGP achieves better results than any of the 13 benchmark methods except for IEGP. IEGP is an ensemble method using multiple classifiers for classification. Compared with IEGP, FGP only uses one classifier so that its performance may be limited on some difficult datasets. The experimental results show that FGP is very effective for dealing with different types of image classification tasks.

Compared with SVM, KNN, LR, RF, AdaBoost, and ERF, which use raw pixels for classification, FGP is more effective by automatically learning a number of high-level features for classification of different datasets. The results show that feature extraction is more important for texture and scene classification since FGP achieves better results than any of these methods on scene and texture datasets. Comparing the results obtained by FGP with that by LBP+SVM, HOG+SVM and SIFT+SVM, it is clear that the features learned by FGP are more effective than the LBP, SIFT and HOG features for image classification, especially for texture classification and scene classification. This shows that automatically learning features is more effective than manually extracting features for image classification. Feature extraction methods often require domain expertise, while feature learning methods do not. There are two advantages: effectiveness and no domain knowledge requirement, of FGP as a feature learning method in contrast to traditional feature extraction methods. Compared with CNN-5 and CNN-8, the FGP approach achieves comparable or significantly better performance on the five datasets. Compared with EGP and IEGP, which use ensembles to solve image classification, FGP uses a single classifier but achieves similar or slightly worse performance. As a result, FGP is an effective approach to learning informative features for different types of image classification tasks.

B. Classification Results on Datasets 6-12

On datasets 6-12, 20 baseline methods with published results are used for comparisons. Note that some of the 20 methods have not been examined on the Rectangle, RI, and Convex datasets so that there are 17 benchmark methods on Rectangle, 16 benchmark methods on RI and 12 benchmark methods on Convex. Table VII lists the classification error rates (%) of FGP and 20 benchmark methods. Each column of Table VII shows the results on one dataset and the minimal error rate is highlighted in bold. The results of FGP, including the minimal error rate (best), the mean error rate (mean) and the standard deviation (std), are listed at the bottom of Table VII. Since the benchmark methods only have the best classification results, we compare the FGP approach with them using the best error rate. The symbol "+" in the table denotes that FGP is better than the compared method in terms of the best error rate. The final row of Table VII summaries the ranking results of FGP among all the methods on each dataset.

From Table VII, it can be found that FGP achieves a smaller error rate than any of the benchmark methods on two

TABLE VI
CLASSIFICATION ERROR RATES (%) OBTAINED BY THE FGP APPROACH AND THE BENCHMARK METHODS ON DATASETS 1-5

	FEI_1		**FEI_2**		**ORL**		**KTH**		**FS**	
Methods	Min	Mean±St.dev	Min	Mean±St.dev	Min	Mean±St.dev	Min	Mean±St.dev	Min	Mean±St.dev
SVM	10.00	10.00±0.00+	12.00	12.00±0.00+	5.62	5.62±0.00+	53.03	55.41±2.83+	79.37	79.71±0.15+
KNN	68.00	68.00±0.00+	92.00	92.00±0.00+	5.62	5.62±0.00+	65.76	65.76±0.00+	75.65	75.65±0.00+
LR	8.00	8.00±0.00+	12.00	12.00±0.00+	6.25	6.25±0.00+	51.21	51.21±0.00+	76.51	76.51±0.00+
RF	2.00	**2.93±1.01**–	10.00	10.80±1.13+	6.88	7.67±0.63+	40.00	42.19±0.83+	62.64	63.47±0.49+
AdaBoost	20.00	21.33±1.32+	20.00	24.00±3.44+	40.62	47.73±4.00+	62.12	66.56±1.37+	82.53	86.96±1.47+
ERF	6.00	6.73±0.98+	8.00	9.40±0.93+	2.50	3.29±0.59+	38.48	40.17±0.86+	62.06	62.85±0.36+
LBP+SVM	34.00	43.27±3.66+	32.00	37.47±3.52+	12.50	12.58±0.21+	21.21	26.71±4.18+	50.21	66.73±8.90+
HOG+SVM	4.00	4.00±0.00–	18.00	18.00±0.00+	8.75	8.75±0.00+	42.73	44.04±0.64+	87.89	92.09±2.47+
SIFT+SVM	44.00	44.00±0.00+	38.00	38.00±0.00+	6.25	6.25±0.00+	34.24	34.24±0.00+	39.08	39.08±0.00+
CNN-5	2.00	4.60±1.30=	2.00	4.73±1.62–	3.12	4.71±1.06+	14.24	17.44±1.87+	49.86	51.97±1.16+
CNN-8	2.00	4.67±1.32=	4.00	9.07±1.87=	5.00	6.96±1.09+	23.64	28.37±3.18+	50.84	53.21±1.01+
EGP [37]	**0.00**	3.80±2.02=	**0.00**	**1.93±1.67**–	0.62	2.56±1.24+	12.12	22.47±5.08+	32.83	38.93±2.87+
IEGP [38]	**0.00**	3.33±2.59–	**0.00**	3.79±3.73–	**0.00**	1.71±0.98=	1.51	**3.57±5.08=**	**7.46**	**10.37±1.50**–
FGP	2.00	5.53±2.67	4.00	8.67±3.36	**0.00**	**1.37±1.04**	**1.21**	3.93±1.13	25.52	29.41±1.74
Overall		7+, 3=, 3–		9+, 1=, 3–		12+, 1=		12+, 1=		12+, 1–

TABLE VII
CLASSIFICATION ERROR RATES (%) OF FGP AND BENCHMARK METHODS ON DATASETS 6-12

Methods	**MB**	**MRD**	**MBR**	**MBI**	**Rectangle**	**RI**	**Convex**
SVM+RBF [34]	3.03(+)	11.11(+)	14.58(+)	22.61(+)	2.15 (+)	24.04(+)	19.13(+)
SVM+Poly [34]	3.69(+)	15.42(+)	16.62(+)	24.01(+)	2.15(+)	24.05(+)	19.82(+)
SAE-3 [33]	3.46(+)	10.30(+)	11.28(+)	23.00(+)	2.14(+)	24.05(+)	–
DAE-b-3 [33]	2.84(+)	9.53(+)	10.30(+)	16.68(+)	1.99(+)	21.59(+)	–
CAE-2 [33]	2.48(+)	9.66(+)	10.90(+)	15.50(+)	1.21(+)	21.54(+)	–
SPAE [48]	3.32(+)	10.26(+)	9.01(+)	13.24(+)	–	–	–
RBM-3 [33]	3.11(+)	10.30(+)	6.73(+)	16.31(+)	2.60(+)	22.50(+)	–
ScatNet-2 [31, 32]	1.27(+)	7.48(+)	12.30(+)	18.40(+)	0.01(+)	8.02(+)	6.50(+)
RandNet-2 [32]	1.25(+)	8.47(+)	13.47(+)	11.65(+)	0.09(+)	17.00(+)	5.45(+)
PCANet-2 (softmax) [32]	1.40(+)	8.52(+)	6.85(+)	11.55(+)	0.49(+)	13.39(+)	4.19(+)
LDANet-2 [32]	**1.05**	7.52(+)	6.81(+)	12.42(+)	0.14(+)	16.20(+)	7.22(+)
NNet [34]	4.69(+)	18.11(+)	20.04(+)	27.41(+)	7.16(+)	33.20(+)	32.25(+)
SAA-3 [34]	3.46(+)	10.30(+)	11.28(+)	23.00(+)	2.41(+)	24.05(+)	18.41(+)
DBN-3 [34]	3.11(+)	10.30(+)	6.73(+)	16.31(+)	2.60(+)	22.50(+)	18.63(+)
FCCNN [29]	2.43(+)	8.91(+)	6.45	13.23(+)	–	–	–
FCCNN (with BT) [29]	2.68(+)	9.59(+)	6.97(+)	10.80(+)	–	–	–
SPCN [30]	1.82(+)	9.81(+)	5.84	9.55(+)	0.19(+)	10.60(+)	–
EvoCNN (best) [49]	1.18	**5.22**	**2.80**	**4.53**	0.01(+)	**5.03**	4.82(+)
EGP (best) [37]	2.81(+)	–	–	–	0.09(+)	–	6.03(+)
IEGP (best) [38]	1.18	5.72	6.41	10.59(+)	**0.00**	5.12	1.74(+)
FGP (best)	1.18	7.37	6.54	7.48	**0.00**	6.10	**1.54**
FGP (mean)	1.30	8.44	7.34	10.35	0.12	7.34	1.84
FGP (std)	0.06	0.6	0.42	1.41	0.11	0.61	0.19
Rank	2/21	3/20	5/20	**2/20**	**1/18**	**3/17**	**1/13**

datasets, i.e., Rectangle and Convex. Note that these datasets have been used by many effective methods (such as the deep learning method EvoCNN and the ensemble method IEGP) so that even 1% improvement in error rate is very difficult to achieve. On MB, the FGP approach achieves 1.18% error rate, which is better than any of the 20 benchmark methods except for LDANet-2. The LDANet-2 method achieves 1.05% error rate on MB, which is slightly better than FGP's 1.18% error rate. Although FGP is worse than LDANet-2 on MB, it is better than LDANet-2 on the other six datasets. The MRD, MBR and MBI datasets are three variants of MB obtained by adding additional factors to make it more difficult. On MRD, FGP achieves an error rate of 7.37%, which is better than 17 benchmark methods and worse than EvoCNN and IEGP. On MBR, FGP ranks fifth among all the benchmark methods. On MBI, FGP achieves an error rate of 7.48%, which is better than 19 methods and only worse than EvoCNN. On the Rectangle and Convex datasets, FGP achieves better results than any of the benchmark methods. It is noticeable that FGP finds the perfect solution on Rectangle, which the CNN-based or NN-based methods such as SPCN, NNet, SAE-3, and EvoCNN are not able to find. RI is an extension of the Rectangle dataset and it is more difficult. On RI, most methods perform worse with respect to Rectangle, e.g., LDANet-2 obtains 16.20% error rate, SPCN obtains 10.60% error rate, and NNet obtains 33.20% error rate. FGP obtains an error rate of 6.10%. On the Convex dataset, FGP achieves the best error rate of 1.54%, which is better than that of any benchmark method.

Compared with EvoCNN, which is a state-of-the-art deep learning algorithm, FGP achieves better or the same error rates on the MB, Rectangle and Convex datasets. FGP is a non-neural network-based algorithm, while EvoCNN is a CNN-based algorithm, where an evolutionary algorithm is used to search for the best architecture. EvoCNN was designed to find a more complex CNN-based solution for image classification so that it can achieve better performance on the other four difficult datasets. Compared with EvoCNN, FGP has a smaller search space and can find simpler solutions with several image-related operators. These advantages enable FGP to be an alternative to effective feature learning.

Compared with IEGP, which is an effective ensemble method, FGP achieves better or the same performance on four datasets and worse performance on three datasets. FGP only uses a single classifier for classification, while IEGP constructs an ensemble of accurate and diverse classifiers for classification. The results confirm the effectiveness of the features learned by FGP, since the latter can achieve comparable performance even if it uses a single classifier instead of an ensemble. Compared with IEGP, FGP is much faster in training and testing, as discussed in the supplementary materials. Besides, the solutions evolved by FGP are easy to explain, while the solutions of IEGP include a number of classifiers and are difficult to explain.

The comparisons demonstrate that FGP achieves results that are better or comparable to the 20 existing effective algorithms for object classification. Compared with the state-of-the-art deep learning and ensemble methods, FGP achieves better or similar results on several datasets. These results indicates that the features learned by FGP are effective for image classification. FGP has a flexible program structure and a function set including many image-related operators, which enables it to various types and numbers of effective features for image classification.

VI. Further Analysis

This section further analyses the FGP approach to provide insights on why it achieves better results. First, the evolved example programs/solutions of FGP are analysed to understand what features are learned. Second, the datasets with the learned features are visualised and compared with the original raw pixels. This provides insights on how the hidden structures of the datasets are changed by the solutions of FGP.

A. Evolved Programs/Solutions

1) An Example Program on FEI_1: An example program evolved by FGP on the FEI_1 dataset is visualised in Fig. 6. To show how the program extracts features, two example images from the natural and smile classes are used for visualisation, as shown in Fig. 7. The example program has filtering and pooling functions to describe the features from the input image. The edge filters, i.e., $SobelY$ and $GauD$, are used as nodes in the two branches of the example program. These operators can extract edges from the images before applying the pooling operators. The two example images only have a difference in facial expressions. From Fig. 7, it is clear that using different filters can obtain informative features that enlarge the difference between classes. Finally, the example program with 24 nodes produces 750 features from an input 60×40 image, i.e., 600 features by the left branch and 150 features by the right branch.

2) Example Programs on Rectangle: Three example programs of FGP on the Rectangle dataset are visualised in Fig. 8. The three programs achieve 100% accuracy on both the training and test sets, respectively. In contrast to the example program in Fig. 6, which describes features using filtering and pooling functions, the three example programs generate features using feature extraction, filtering and pooling functions. The three example programs extract SIFT and LBP features from the input image or the images after the filtering functions such as Med, $LoG1$, Gau, and $LoG2$. Since the Rectangle dataset has images with rotation and scale variations, the features that are invariant to these variations are more discriminative than the other types of features. Therefore, the LBP and SIFT features are extracted using the example programs. It is noticeable that the LBP and $SIFT$ functions/nodes in the three example programs have different child nodes, which indicate that each LBP or $SIFT$ function extracts different features.

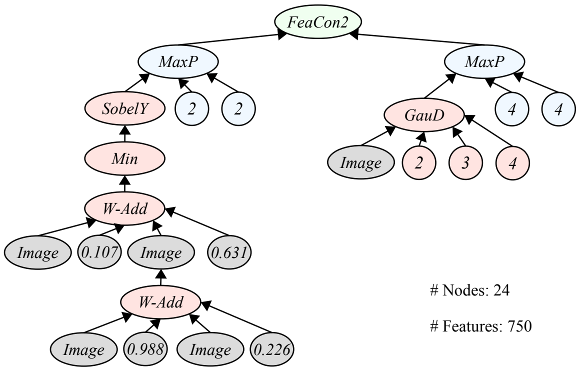


Fig. 6. Example program evolved by FGP on the **FEI_1** dataset. It achieves 98% accuracy on both the training and test sets.

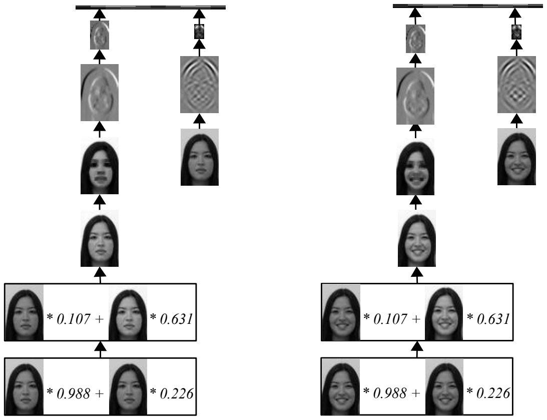


Fig. 7. Features produced by the example program in Fig. 6 on two images from different classes.

By analysing the example programs on FEI_1 and Rectangle, it is clear what types of features can be extracted by FGP and how the features are extracted. Owning to the flexible program structure and the new function set, FGP evolves programs that are able to extract various numbers and types of features for effective classification. Moreover, FGP can find multiple optimal solutions of variable depths for a task.

B. Data Visualisation

A popular visualisation method, t-distributed stochastic neighbour embedding (t-SNE) [57], is employed to visualise the features learned by FGP. The t-SNE method is a nonlinear dimension reduction technique by mapping high-dimensional data into two- or three-dimensional data. The resulting low-dimensional data can be easily visualised in a scatter plot,

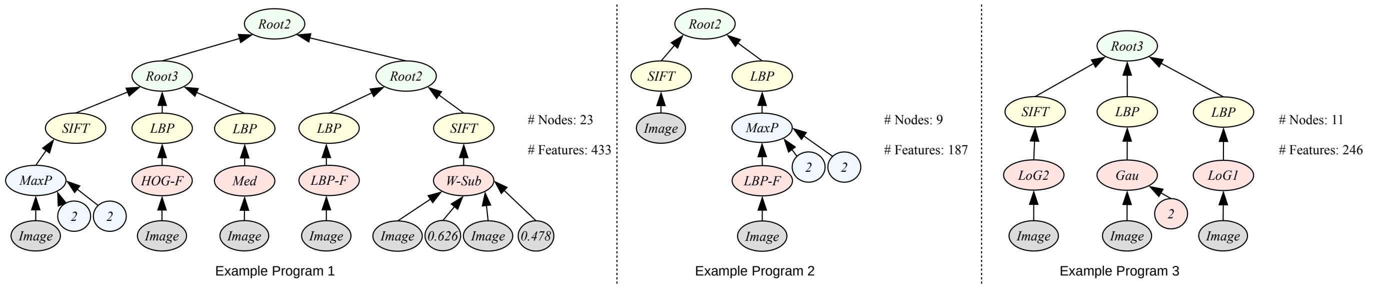
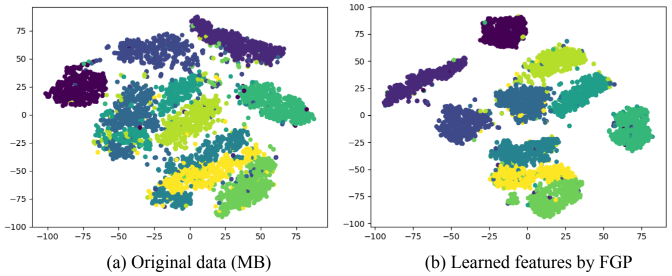


Fig. 8. Example programs evolved by FGP on the **Rectangle** dataset. They achieve 100% accuracy on both the training and test sets. The three programs learn 433, 187 and 246 features, respectively. Meanwhile, their tree sizes (the number of nodes) are 23, 9 and 11, respectively.

which shows how well the similarities within each class is preserved. Compared with other visualisation methods, t-SNE produces better visualisation results [57].

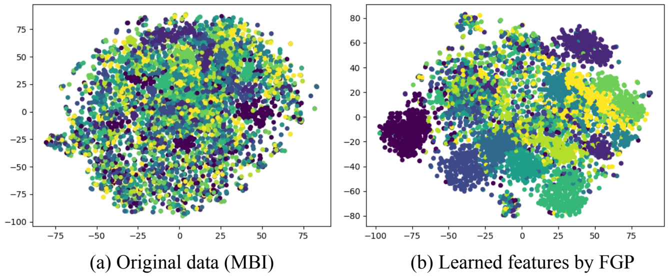
Three large datasets, i.e., MB, MBI and Rectangle, are used to run the experiments for visualisation. To reduce the computational cost, each experiment randomly selects 5,000 images from each dataset for visualisation. To show how the hidden structure of the data is changed by the FGP program, the original data are visualised for comparisons. The comparisons with the other features: LBP, HOG and SIFT features, are shown in the supplementary materials due to the page limit. The visualisation results are plotted in a scatter, and the class label of each point is used to give a specific colour to the point in the plot. The parameter settings for t-SNE are the same as those in [57].



(a) Original data (MB)

(b) Learned features by FGP

Fig. 9. The visualisation results of the ten classes from the **MB** dataset (each colour represents one class). The left figure shows the original subset of MB with raw pixels. The right figure shows the transformed subset of MB with the learned features by FGP.



(a) Original data (MBI)

(b) Learned features by FGP

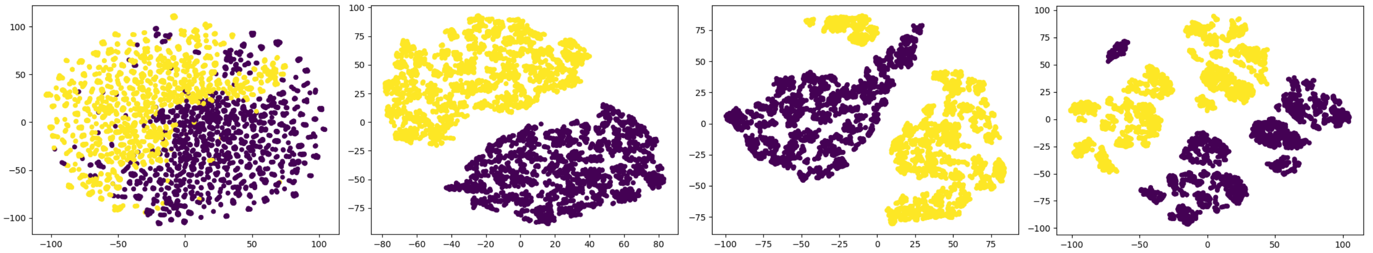
Fig. 10. The visualisation results of the ten classes from the **MBI** dataset (each colour represents one class). The left figure shows the original subset of MBI with raw pixels. The right figure shows the transformed subset of MBI with the learned features by FGP.

Fig. 9 shows that the ten classes of the original MB data and the transformed MB data with the learned features by FGP are well clustered after 1000 iterations using t-SNE. In Fig. 9(a), there are still some points being clustered into wrong classes as each cluster of points contains other points with different colours. In contrast, Fig. 9(b) shows clearer clusters of the data transformed by an example program of FGP. This figure shows that fewer points have been clustered into wrong classes, and each cluster is clearer than that in Fig. 9(a). The visualisation results of the original MBI data (Fig. 10(a)) show a high mixture of different classes as it is very difficult to distinguish a cluster from the scatter plot. The reason is that the images of MBI are noisy, which makes the visualisation of MBI more difficult than that of MB. The visualisation of the transformed MBI data (Fig. 10(b)) has a clearer plot than that of the original data. It can be observed that several clusters of points exist in Fig. 9 even some points are not well clustered. Comparing the visualisation results in Fig. 9 with the results in Fig. 10, it is obvious that the hidden structure of the MBI data is more complex than that of the MB data so that MBI is more difficult than MB. The results reveal that the evolved programs of FGP transform the original data into a space that the new data can be easily clustered by t-SNE, and the hidden structure can be well captured.

The visualisation of the Rectangle dataset is simpler than that of MB and MBI, as Rectangle only has two classes. Fig. 11 shows the visualisation results of the original data and the learned features by FGP using three different evolved programs, respectively. Fig. 11(a) has a clear scatter plot, where the two classes are shown in two different colours. However, it is obvious that many points are clustered into the wrong class in Fig. 11(a). The scatter plots are clearer in Fig. 11(b)-(d) than that in Fig. 11(a). It is noticeable that all the points are clustered into the correct classes by t-SNE using the transformed data (1-3). The visualisation results confirm the search ability and superiority of FGP on finding the optimal solutions to transform the data into a new feature space where the new data can be easily classified.

VII. Conclusions

The goal of this paper was to develop a new GP-based approach with image-related operators to feature learning for different types of image classification tasks. This goal has been successfully achieved by developing the FGP approach with a flexible program structure, a new function set and a new terminal set, and examining it on 12 different image classification datasets of varying difficulty. The proposed FGP approach can evolve solutions of variable depths for a target task. The solutions of FGP can produce various numbers and



(a) Original data (Rectangle)

(b) Learned features by FGP

(c) Learned features by FGP

(d) Learned features by FGP

Fig. 11. The visualisation results of the **Rectangle** dataset (each colour represents one class). The left figure shows the original subset of Rectangle with raw pixels. The other three figures show the transformed subsets of Rectangle with the learned features by FGP using the example programs in Fig. 8, respectively.

types of features from raw images. The experimental results showed that FGP achieved significantly better performance than the 12 commonly used methods on five different types of image classification datasets. Furthermore, the performance of FGP has been examined on seven large datasets, including MNIST variants. The experimental results showed that FGP achieved better classification performance than the existing methods to which it has been compared. To conclude, FGP is an effective and promising approach to feature learning for different types of image classification tasks.

In addition to the encouraging results achieved by FGP, further analysis provided more insights on why it achieves good performance. The solutions found by FGP can be easily visualised as trees to show how and what features are extracted. The visualisation technique, *t*-SNE, was employed to further understand the features learned by FGP in comparison to raw pixel values. The results revealed that the FGP solutions transform raw pixel values of images into a new feature space so that each class can be effectively distinguished.

The FGP approach is an example of showing the potential of GP-based methods for feature learning in image classification. In the future, the scalability of GP-based methods for large-scale datasets will be investigated. To achieve this, other effective approaches, such as surrogate models, may be needed to improve the computational cost of GP on large-scale datasets.

References

[1] A. R. Burks and W. F. Punch, “Genetic programming for tuberculosis screening from raw x-ray images,” in *Proc. GECCO*. ACM, 2018, pp. 1214–1221.

[2] 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, 2014.

[3] C. Liu and H. Wechsler, “Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition,” *IEEE Trans. Image Process.*, vol. 11, no. 4, pp. 467–476, 2002.

[4] Y. Bi, B. Xue, and M. Zhang, “A survey on genetic programming to image analysis,” *J. Zhengzhou Uni. (Eng. Sci.)*, vol. 39, no. 06, pp. 3–13, 2018.

[5] W. A. Albukhanajer, J. A. Briffa, and Y. Jin, “Evolutionary multiobjective image feature extraction in the presence of noise,” *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1757–1768, 2015.

[6] A. I. Awad and M. Hassaballah, “Image feature detectors and descriptors,” *Stud. Comput. Intell.*, Springer International Publishing, Cham. 2016.

[7] 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, 2002.

[8] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proc. IEEE CVPR*, vol. 1, 2005, pp. 886–893.

[9] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[10] L. Nanni, S. Ghidoni, and S. Brahnam, “Handcrafted vs. non-handcrafted features for computer vision classification,” *Pattern Recognit.*, vol. 71, pp. 158–172, 2017.

[11] Y. Bi, B. Xue, and M. Zhang, “A gaussian filter-based feature learning approach using genetic programming to image classification,” in *Proc. Austra. Joint Conf. Art. Intell.* Springer, 2018, pp. 251–257.

[12] 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 Trans. Evol. Comput.*, vol. 21, no. 6, pp. 825 – 844, 2017.

[13] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT press, Cambridge, 1992.

[14] A. Agapitos, R. Loughran, M. Nicolau, S. Lucas, M. O’Neill, and A. Brabazon, “A survey of statistical machine learning elements in genetic programming,” *IEEE Trans. Evol. Comput.*, vol. 23, no. 6, pp. 1029–1048, 2019.

[15] K. Kim, R. B. McKay, and N. X. Hoai, “Recursion-based biases in stochastic grammar model genetic programming,” *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 81–95, 2016.

[16] Q. N. Huynh, S. Chand, H. K. Singh, and T. Ray, “Genetic programming with mixed-integer linear programming-based library search,” *IEEE Trans. Evol. Comput.*, vol. 22, no. 5, pp. 733–747, 2018.

[17] H. Al-Sahaf, Y. Bi, Q. Chen, A. Lensen, Y. Mei, Y. Sun, B. Tran, B. Xue, and M. Zhang, “A survey on evolutionary machine learning,” *J. Roy. Soc. New Zeal.*, vol. 49, no. 2, pp. 205–228, 2019.

[18] R. Poli, W. B. Langdon, and N. F. McPhee, *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008, (With contributions by J. R. Koza).

[19] E. Kieffer, G. Danoy, M. R. Brust, P. Bouvry, and A. Nagih, “Tackling large-scale and combinatorial bi-level problems with a genetic programming hyper-heuristic,” *IEEE Trans. Evol. Comput.*, pp. 1–13, 2019.

[20] Y. Bi, B. Xue, and M. Zhang, “An automatic feature extraction approach to image classification using genetic programming,” in *Proc. Int. Conf. Appl. Eov. Comput.*, 2018, pp. 421–438.

[21] D. Atkins, K. Neshatian, and M. Zhang, “A domain independent genetic programming approach to automatic feature extraction for image classification,” in *Proc. IEEE CEC*, 2011, pp. 238–245.

[22] H. Al-Sahaf, A. Song, K. Neshatian, and M. Zhang, “Two-tier genetic programming: Towards raw pixel-based image classification,” *Expert Syst. Appl.*, vol. 39, no. 16, pp. 12 291–12 301, 2012.

[23] M. Suganuma, D. Tsuchiya, S. Shirakawa, and T. Nagao, “Hierarchical feature construction for image classification using genetic programming,” in *Proc. IEEE Sys. Man Cybern.*, 2016, pp. 001 423–001 428.

[24] 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 *Proc. EuroGP*. Springer, 2016, pp. 51–67.

[25] 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.

[26] O. Chapelle, P. Haffner, and V. N. Vapnik, “Support vector machines for histogram-based image classification,” *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1055–1064, 1999.

[27] S. Sergyan, “Color histogram features based image classification in content-based image retrieval systems,” in *Proc. 6th Int. Sym. Appl. Mach. Int. Inform.* IEEE, 2008, pp. 221–224.

[28] A. Bosch, A. Zisserman, and X. Munoz, “Image classification using random forests and ferns,” in *Proc. IEEE ICCV*, 2007, pp. 1–8.

[29] G. Qian and L. Zhang, "A simple feedforward convolutional conceptor neural network for classification," *Appl. Soft Comput.*, vol. 70, pp. 1034–1041, 2018.

[30] H. Li and M. Gong, "Self-paced convolutional neural networks," in *Proc. IJCAI*, 2017, pp. 2110–2116.

[31] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872–1886, 2013.

[32] 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, no. 12, pp. 5017–5032, 2015.

[33] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proc. ICML.* Omnipress, 2011, pp. 833–840.

[34] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proc. ICML.* ACM, 2007, pp. 473–480.

[35] D. J. Montana, "Strongly typed genetic programming," *Evol. Comput.*, vol. 3, no. 2, pp. 199–230, 1995.

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

[37] Y. Bi, B. Xue, and M. Zhang, "An automated ensemble learning framework using genetic programming for image classification," in *Proc. GECCO.* ACM, 2019, pp. 365–373.

[38] ——, "Genetic programming with a new representation to automatically learn features and evolve ensembles for image classification," *IEEE Trans. Cybern.*, 2020. DOI: 10.1109/TCYB.2020.2964566.

[39] L. Liu, L. Shao, X. Li, and K. Lu, "Learning spatio-temporal representations for action recognition: A genetic programming approach," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 158–170, 2016.

[40] Y. Bi, B. Xue, and M. Zhang, "Genetic programming for automatic global and local feature extraction to image classification," in *Proc. IEEE CEC*, 2018, pp. 1–8.

[41] M. E. Roberts, "The effectiveness of cost based subtree caching mechanisms in typed genetic programming for image segmentation," in *Proc. Workshop Appl. Evol. Comput.* Springer, 2003, pp. 444–454.

[42] L. Cai, J. Zhu, H. Zeng, J. Chen, C. Cai, and K.-K. Ma, "Hog-assisted deep feature learning for pedestrian gender recognition," *J. Franklin Inst.*, vol. 355, no. 4, pp. 1991–2008, 2018.

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

[44] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proc. Sec. IEEE Workshop Appl. Comput. Vis.*, 1994, pp. 138–142.

[45] 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*, pp. 1–10, 2006.

[46] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *Proc. IEEE CVPR*, vol. 2, 2005, pp. 524–531.

[47] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[48] T. Yu, C. Guo, L. Wang, S. Xiang, and C. Pan, "Self-paced autoencoder," *IEEE Signal Proc. Let.*, vol. 25, no. 7, pp. 1054–1058, 2018.

[49] 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. 1–14, 2019.

[50] M. Iqbal, B. Xue, H. Al-Sahaf, and M. Zhang, "Cross-domain reuse of extracted knowledge in genetic programming for image classification," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 569–587, 2017.

[51] S. Young, T. Abdou, and A. Bener, "Deep super learner: A deep ensemble for classification problems," in *Proc. 31st Can. Conf. Art. Intell.* Springer, 2018, pp. 84–95.

[52] Z.-H. Zhou and J. Feng, "Deep forest," *Natl. Sci. Rev.*, vol. 6, no. 1, pp. 74–86, 2018.

[53] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[54] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *J. Mach. Learn. Res.*, vol. 13, no. Jul, pp. 2171–2175, 2012.

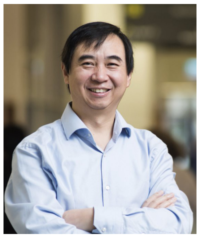
[55] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830,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

[56] R. Herbrich and T. Graepel, "A pac-bayesian margin bound for linear classifiers," *IEEE Trans. Inf. Theory*, vol. 48, no. 12, pp. 3140–3150, 2002.

[57] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov 2008.

Ying Bi (S'17) is pursuing the Ph.D degree in computer science with the School of Engineering and Computer Science, Victoria University of Wellington (VUW), Wellington, New Zealand. She has over 20 journal and conference papers and her current research interests include evolutionary computation, computer vision and machine learning. She is a member of the IEEE Computational Intelligence Society and has been severing as reviewers for top international journals and conferences in this field.

Bing Xue (M'10) 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 PhD degree in computer science in 2014 at Victoria University of Wellington (VUW), New Zealand. She is currently an Associate Professor and Program Director of Science in School of Engineering and Computer Science at VUW. She has over 200 papers published in fully refereed international journals and conferences and her research focuses mainly on evolutionary computation, machine learning, classification, symbolic regression, feature selection, evolving deep neural networks, image analysis, transfer learning, multi-objective machine learning. Dr Xue is currently the Chair of IEEE Computational Intelligence Society (CIS) Data Mining and Big Data Analytics Technical Committee, and Vice-Chair of IEEE Task Force on Evolutionary Feature Selection and Construction, Vice-Chair of IEEE CIS Task Force on Transfer Learning & Transfer Optimization, and of IEEE CIS Task Force on Evolutionary Deep Learning and Applications. She is also served as associate editor of several international journals, such as IEEE Computational Intelligence Magazine and IEEE Transactions of Evolutionary Computation.



Mengjie Zhang (M'04-SM'10-F'19) received the B.E. and M.E. degrees from Artificial Intelligence Re- search Center, Agricultural University of Hebei, Hebei, China, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 1989, 1992, and 2000, respectively. He is currently Professor of Computer Science, Head of the Evolutionary Computation Research Group, and the Associate Dean (Research and Innovation) in the Faculty of Engineering. His current research interests include evolutionary computation, particularly genetic programming, particle swarm optimization, and learning classifier systems with application areas of image analysis, multi-objective optimization, feature selection and reduction, job shop scheduling, and transfer learning. He has published over 500 research papers in refereed international journals and conferences. Prof. Zhang is a Fellow of Royal Society of New Zealand and have been a Panel member of the Marsden Fund (New Zealand Government Funding), a Fellow of IEEE, and a member of ACM. He was the chair of the IEEE CIS Intelligent Systems and Applications Technical Committee, and chair for the IEEE CIS Emergent Technologies Technical Committee and the 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, a vice-chair of the 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.