Genetic Programming-Based Discriminative Feature Learning for Low-Quality Image Classification

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

***Abstract*—Being able to learn discriminative features from low-quality images raises much attention recently due to their wide applications ranging from autonomous driving to safety surveillance. However, this task is difficult due to high variations across images, such as scale, rotation, illumination, and viewpoint, and distortions in images, such as blur, low contrast and noise. Image preprocessing could improve the quality of the images, but it often requires human intervention and domain knowledge. Genetic programming (GP) with a flexible representation can automatically perform image preprocessing and feature extraction without human intervention. Therefore, this study proposes a new evolutionary learning approach using GP (EFLGP) to learn discriminative features from images with blur, low contrast, and noise for classification. In the proposed approach, we develop a new program structure (individual representation), a new function set and a new terminal set. With these new designs, EFLGP can detect small regions from a large input low-quality image, select image operators to process the regions or detect features from the small regions, and output a flexible number of discriminative features. A set of commonly used image preprocessing operators are employed as functions in EFLGP to allow it to search for solutions that can effectively handle low-quality image data. The performance of EFLGP is comprehensively investigated on eight datasets of varying difficulty under the original (clean), blur, low contrast, and noise scenarios, and compared with a large number of benchmark methods using hand-crafted features and deep features. The experimental results show that EFLGP achieves significantly better or similar results in most comparisons. The results also reveal that EFLGP is more invariant than the benchmark methods to blur, low contrast and noise.**

***Index Terms*—Genetic Programming; Feature Learning; Representation; Low-quality Image; Classification.**

I. Introduction

IMAGE classification is an important task in machine learning and computer vision with a wide range of applications, e.g., face images, hyperspectral images, medical images, and vehicle images [1, 2, 3, 4]. Image classification is the task of assigning class labels to images according to the content in the images. Low-quality image classification is a task of classifying low-quality images. The quality of images may be degraded during the processes of image acquiring, storage and transmission [5]. For example, using an out-of-focus camera or sensor may lead to obtaining blurring images. Typically, the images with noise, blur, low brightness/contrast, and low resolution are known as low-quality images [5]. Low-quality image classification raises much attention in recent years due to their wide applications ranging from autonomous driving to safety surveillance [6]. However, this task is difficult due to distortions, such as blur, noise and low contrast, and high variations across images, such as scale, rotation, illumination, deformation, and viewpoint variations. Many effective methods, including deep learning methods, have been developed for image classification and achieved promising results in recent years [7]. But their performances often degrade on images with noise, blur or low contrast [5, 6, 8]. It is noted that deep learning methods can obtain satisfactory results on low-resolution images, such as on the MNIST dataset, in which the image size is 28×28 [9]. This study, however, focuses on the classification of noisy, blurring, or low-contrast images.

Typically, algorithms of image classification can be used for classifying low-quality images. A traditional image classification procedure often includes the processes of image preprocessing, feature extraction and image classification [10]. Image preprocessing, including image denoising/smoothing, brightness correction or pixel transformation, can be employed to improve the quality of the image [11]. However, when the types and the levels of noise, blur or contrast are unknown, it is difficult to manually perform image preprocessing. Feature extraction is an essential step for image classification. Commonly used feature extraction methods include gray-level co-occurrence matrix (GLCM) [12], local binary patterns (LBP) [13], histogram of oriented gradients (HOG) [14], and scale-invariant feature transform (SIFT) [15]. These image features are often manually extracted and effective for particular tasks. For example, LBP features are effective for texture classification. Instead of manually extracting features, feature learning methods can automatically learn informative features from images for classification [7]. Feature learning methods are more adaptive for different tasks and often achieve better classification results than using manually extracted features [7]. However, many feature learning methods are neural network (NN)-based methods, which often need a large number of training data and have fixed model complexity [16, 17]. In addition, very few feature learning methods can handle images with noise, blur or low contrast. Several methods have been developed for simultaneous image preprocessing and feature learning, such as in [6, 18]. But these methods

This work was supported in part by the Marsden Fund of New Zealand Government under Contracts VUW1509, VUW1615, VUW1913 and VUW1914, 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.

The authours are with School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (e-mail: Ying.Bi@ecs.vuw.ac.nz; Bing.Xue@ecs.vuw.ac.nz; Mengjie.Zhang@ecs.vuw.ac.nz).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

focus on limited types and levels of quality degradations and are examined on limited datasets. To this end, this study develops a new approach based on genetic programming (GP) for simultaneous image preprocessing and feature learning from images with various levels of noise, blur or low contrast.

GP [19] is an evolutionary computation (EC) technique and searches for the best solution using a population of individuals/solutions via an evolutionary process. GP is able to automatically evolve computer programs to solve problems without the assumption of the solution structure [19, 20]. With powerful search ability, GP has been successfully applied to many tasks, e.g., symbolic regression, classification, scheduling, and feature learning [16]. GP has a tree-based variable-length individual representation, which allows it to evolve solutions of variable depths to solve a problem. In feature learning, the flexible representation enables GP to employ image operators for feature extraction in many possible ways [16, 21, 22]. Specifically, GP may find shallow/simple solutions for easy tasks and deep/complex solutions for difficult tasks, which is a limitation of many other techniques including convolutional neural networks (CNNs).

Compared with the CNNs, which are well-developed and popular techniques [7], GP is an emerging technique for feature learning. Several GP-based methods have been developed for feature learning, such as [21, 22, 23, 24]. These methods have achieved promising results in different image classification tasks. However, very few GP-based feature learning methods have been developed for classifying images with noise, blur or low contrast. It is possible that the performances of existing GP-based methods degrade on such low-quality images, especially without image preprocessing.

Therefore, this study develops a GP-based feature learning approach for low-quality image classification, specifically on noisy, blurring or low-contrast images. To achieve this, a set of image preprocessing operators are employed as functions to form the internal nodes of GP trees. These operators include the well-known mean filter, the max filter, the Gaussian-based filters, and the Laplacian filter. These operators allow the proposed approach to evolving solutions that can handle the image distortions by smoothing images, adjusting the contrast of images or detecting salient features. With these operators, effective features can be easily learned from the low-quality images. A new program structure, a new function set and a new terminal set are developed in the proposed approach, which is termed as EFLGP in short. The main contributions of this paper are summarised as follows.

1) A new function set and a new terminal set are developed for the EFLGP approach. The new function set has a number of image-related operators, e.g. the mean filter, the Sobel filter, the Gabor filter, and the Gaussian filter, which can perform image preprocessing, smoothing or edge detection. The operators can help preprocess an image and extract important features from the image. Note that in some operators, the parameters are developed as ephemeral random constants (terminals), which can be automatically selected during the evolutionary process.
2) A new program structure (individual representation) is developed for the EFLGP approach to allow it to evolve solutions of variable depths. The new representation allows EFLGP to detect small regions from the large input image, select image operators to process the regions, and use feature extractors to extract effective features with variable lengths. The output of an EFLGP solution is a feature vector, where the dimensionality of the feature vector is automatically determined by the learning process.
3) Extensive experiments are conducted to investigate the effectiveness of the EFLGP approach. The performance of EFLGP is examined on eight different image datasets under the original, blur, low contrast, and noise scenarios. The results on the total 32 datasets show that EFLGP achieves better performance than many hand-crafted features, CNNs and deep features in most comparisons. Furthermore, the analysis of the solutions found by EFLGP shows high interpretability.

II. Background and Related Work

A. Low-Quality Image Classification

Low-quality image classification is the task of classifying images with low contrast, low resolution, noise, or blur. The quality of images may be degraded in the processes of image sampling, storage and transmission [5]. Algorithms of image classification can be used for classifying low-quality images. But recent work has shown that the classification performances of many algorithms degrade on low-quality images. Zhou et al. [5] reviewed two traditional methods and two CNN-based methods for face detection on images with noise, blur, or low contrast. The results showed that the methods using hand-crafted and deep-learning-based features are sensitive to these low-quality images. Hosseini et al. [25] investigated the performance of Cloud Vision API, which was introduced by Google for classifying images, on noisy images. The results revealed that adding enough noise to an image, the class label predicted by API can be completely different. Albukhanajer et al. [26] addressed feature extraction in the noisy images by developing a Pareto-based evolutionary multiobjective algorithm to optimise the functionals in the trace transform.

A simple method to improve the classification performance on low-quality image data is to preprocess images before feature extraction and classification. Image preprocessing can be employed to improve the quality of the image. Image preprocessing often includes image denoising/smoothing, brightness correction or pixel transformation [11]. da Costa et al. [8] investigated the impact of different types of noise on the classification performance using LBP and HOG features and the use of denoising methods for performance improvement. The results showed that noise has a negative effect on classification performance and denoising methods can improve the classification performance when the type of noise in the training and test images is the same. However, performing image preprocessing such as denoising needs human intervention and domain knowledge. In many images, the types and levels of noise, blur or contrast are often unknown, which makes the image preprocessing difficult.

Image preprocessing operations have been automatically performed by being integrated into the feature learning process. Diamond et al. [6] developed differentiable denoising, deblurring and classification architecture based on CNN for classifying noisy and blurring images. Cai et al. [18] developed resolution-aware deep CNN, which have convolutional super-resolution layers and normal convolutional classification layers, for fine-grained classification with low-resolution images. Although these methods have achieved promising results in particular types of low-quality images, their performances could be further improved. In addition, these methods addressed limited types and levels of low-quality images, e.g., one type of low-quality images in [18]. Therefore, it is necessary to develop a new approach to simultaneously performing image preprocessing and feature learning for low-quality image classification that involves various levels of blur, low contrast or noise.

B. Image Features

Image features are important for solving image classification. Well-known hand-crafted features include histogram, domain-independent feature (DIF) [27], GLCM [12], LBP [13], Gabor features [28], HOG [14], SIFT [29], and others [30]. The histogram and DIF features are simple and domain-independent features by calculating the statistics of the images or small regions [27]. The GLCM and LBP features are known for texture analysis. GLCM calculates the occurrences of adjacent grey levels according to the predefined distance and orientation and extracts the statistics such as contrast, correlation and entropy as features [12, 31]. The LBP features represent local patterns calculated from a binary code and a predefined weight vector. The binary code is obtained by comparing a pixel value with its neighbours. Many LBP variants have been developed, e.g., uniform LBP, to handle image variations, e.g., rotation [13]. Gabor features are extracted from the convolve images by a set of Gabor bank (wavelets) filters with different orientations and scales. The Gabor feature can capture saliency information [28]. The HOG and SIFT features [14, 15] are similar by calculating the histogram of gradient orientations. Different from HOG, SIFT performs keypoint detection and produces 128 features from a keypoint. A dense SIFT method has been developed to describe features without keypoint detection to reduce computational complexity [29]. Other methods such as speeded up robust features (SURF), binary robust invariant scalable keypoints (BRISK) and gradient location-orientation histogram (GLOH) have also been developed for image description [30].

Image features can also be automatically learned from images. A commonly used method is CNN, which learns deep features through multiple layers of non-linear transformation [9]. The typical operations in CNNs are convolution and pooling. In CNNs, the weights of the convolutional filters are often optimised using backpropagation to obtain effective features for classification. In recent years, many variations of CNNs have been developed to learn features, such as AlexNet, GoogLeNet, VGGNet, ResNet, and DenseNet [7]. However, to train such a deep model often requires a large number of computing resources and training instances [32]. To avoid training the deep models from scratch, pre-trained CNNs with weights are often employed for solving new image classification tasks. Liang [32] used the pre-trained VGG-16 and proposed a new approach to further optimise the pre-trained AlexNet. Cen and Wang [33] developed a subspace decomposition-based estimation using deep features extracted from ResNet on ImageNet of original occlusion-free images for classification. However, this may lead to the waste of computing resources since some tasks do not need such a deep model to solve. Instead of using NNs with a fixed representation (model complexity), this study employs GP with a flexible representation for feature learning.

C. GP-based Feature Learning for Image classification

Besides CNNs, GP has also been applied to learn features for image classification [16, 34]. A GP solution often includes a set of functions/operators (typically image operators), which transform an input image into features. The flexible representation allows GP to evolve solutions of variable depths in many possible ways for feature learning in image classification. Atkins et al. [35] proposed a multi-tier GP method with an image filtering tier, an aggregation tier and a classification tier to learn features from the input image for classification. Lensen et al. [36] designed a HoG+GP method to simultaneously detect regions, extract HOG features from the detected regions and construct a high-level feature for binary classification. Bi et al. [37] presented a multi-layer GP method with simultaneous region detection, feature extraction, feature construction, and image classification. In [38], the use of the Gaussian-based filters in GP was investigated for image classification. However, these methods have only been examined on binary image classification tasks. Shao et al. [22] proposed a multi-objective GP (MOGP) method to learn holistic features for multi-class image classification. This method achieved promising results in four different datasets. But it produced a high-dimensional feature vector from an input image, which needs additional dimensionality reduction. Price and Anderson [39] proposed a GP method (GOOFeD) for image descriptor learning based on a set of image transform operators, e.g., Canny, hough circle and Harris corner detector. Al-Sahaf et al. [23] proposed a GP-criptor^{ri} method to learn rotation-invariant texture features from pixel statistics for texture classification. However, GP-criptor^{ri} learned a fixed number of features. A dynamic GP approach [40] was developed to learn a flexible number of texture features, which was more effective than GP-criptor^{ri}. However, these methods have only been examined on very limited types of image classification tasks, i.e., texture image classification. Bi et al. [41] developed a GP approach based on the current image descriptors to automatically learn global and local features from images for classification. An improved method can be found in [24] that learned global and local features in a more flexible way for binary and multi-class image classification.

Although GP-based feature learning methods have achieved promising results in various image classification tasks, very few of them have been typically developed for classifying

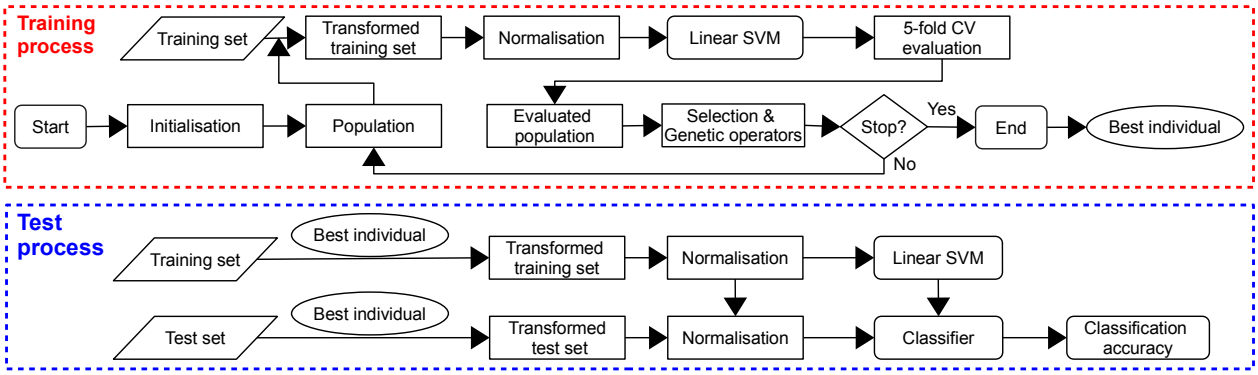


Fig. 1. The overall algorithm.

images with blur, low contrast or noise. It is possible that the performances of these methods degrade on such low-quality images. Motivated by this, we develop a new GP-based approach (i.e., EFLGP) to automatically learning effective features for classifying images with blur, low contrast or noise. To achieve this, a new program structure, a new function set and a new terminal set are developed in the new GP-based approach, which will be introduced in the following section.

III. Proposed Approach

In this section, the proposed EFLGP approach is described. It starts with the description of the overall algorithm, followed by the new program structure, the new function set and the new terminal set.

A. Overall Algorithm

The proposed EFLGP approach aims to learn a set of discriminative features for effective image classification. The overall algorithm of EFLGP for image classification is shown in Fig. 1. It has a training process and a testing process, where the best GP individual/program/tree is learned from a training set and is then tested on a test set. In the training process, EFLGP generates a number of individuals/programs, where each individual is an image descriptor. The training set, having a number of labelled training images, is transformed by a GP individual/tree and then normalised using the min-max normalisation method. The normalised training set is fed into a linear SVM to perform classification. The reasons of using the linear SVM method are that it is commonly used for image classification [42] and it is less expensive compared with SVM with other kernel functions such as radial basis function. To increase the generalisation ability, the stratified 5-fold cross-validation (5-fold CV) method is employed to evaluate each GP tree. The fitness function of EFLGP is the average classification accuracy of the 5 folds. During the evolutionary process, the GP system uses genetic operators, e.g. *elitism, mutation* and *crossover*, to update the population and search for the best solution. When reaching the maximum number of generations, the evolutionary process terminates and the best GP tree is returned.

In the test process, the best individual is employed to transform images of the training and test sets into feature vectors. Then the transformed training and test sets are normalised using the min-max normalisation method. Note the normalisation for the test set is based on the training set. The normalised training set is used to train a linear SVM classifier, which is tested on the normalised test set. The test classification accuracy is reported.

B. Program Structure

The proposed EFLGP approach is based on strongly typed GP (STGP) [43] with a tree-based representation. In STGP, it is necessary to specify an input type and an output type for each function and an output type for each terminal. Each function in STGP only takes the functions or terminals, whose output type is the same as its input type, as the children nodes. To integrate different types of functions and terminals into a single tree, a new program structure is developed in EFLGP. Fig. 2 shows the new program structure and an example program that can be evolved by EFLGP.

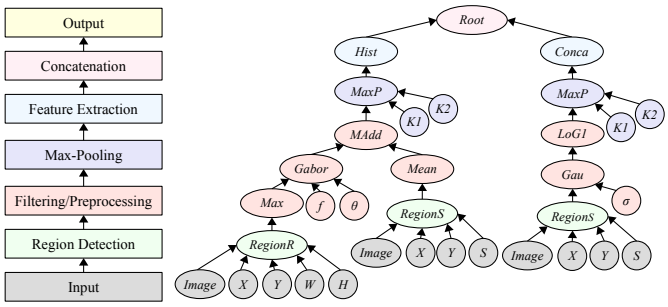


Fig. 2. The program structure and an example program/tree of the EFLGP approach.

Different from those program structures of existing methods [22, 38], the new program structure has seven layers, i.e., input, region detection, filtering/preprocessing, max-pooling, feature extraction, concatenation, and output. The input layer represents the inputs to a GP tree, such as an image and the parameters for the image operators. The region detection layer detects small regions from a large input image. The filtering/preprocessing layer uses a number of image preprocessing operators/filters to process the detected regions to obtain better regions or features. The max-pooling layer samples

the input image to contain salient information. The feature extraction layer extracts features from the processed regions. The concatenation layer combines two feature vectors into a vector and the output layer produces the final feature vector for classification. It is noteworthy that the filtering/preprocessing layer has many image preprocessing operators/filters, such as the mean filter, the max filter, the Gaussian-based filters, the Sobel filters, and the Laplacian filters, that can be used to process the detected regions to improve the quality of the low-quality images.

With the new program structure, a GP tree can be constructed as that in the left part of Fig. 2. In this example program, the leaf nodes are constructed using the terminals in the terminal set of EFLGP. They are $Image$, X, Y, W, H, S, $K1$, $K2$, σ, θ, and f, which will be described in the next subsection. The internal nodes of the example tree are constructed using the functions from the function set of EFLGP. Based on the program structure, these functions are region detection functions, filtering/preprocessing functions, max-pooling functions, feature extraction functions, and feature concatenation functions. For example, in Fig. 2, the region detection functions are $RegionR$ and $RegionS$, the filtering/preprocessing functions are $Mean$, Max, $Sobel$, Gau, $LoG1$, $Sobel$, $Gabor$, the max-pooling function is $MaxP$, the feature extraction functions are $Hist$ and $Conca$, and the feature concatenation function is $Root2$. More details of these functions will be described in next subsection.

In the proposed EFLGP approach, the tree depths for the region detection and feature extraction layers are fixed. The tree depths of the image filtering/preprocessing, max-pooling and feature concatenation layers are flexible and automatically evolved by the mutation and crossover operators during the evolutionary process. This means that an evolved GP program tree can be shallow or deep, which depends on the problem being tackled. One assumption behind it is that when the problem is easy, a simple program tree is sufficient to represent the solution and when the problem is difficult, a complex program tree is expected to be formed to solve it.

C. Function Set

The new function set consists of five different types of functions, i.e., region detection functions, filtering/preprocessing functions, max-pooling function, feature extraction functions, and concatenation functions. These functions are summarised in Tables I and II.

TABLE I
FUNCTION SET OF EFLGP

Function type	Functions
Region Detection	$RegionR$, $RegionS$
Filtering/Preprocessing	Gau, $GauD$, $Gabor$, Lap, $LoG1$, $LoG2$, $Sobel$, $SobelX$, $SobelY$, Med, $Mean$, Min, Max, $MAdd$, $MSub$, $ReLU$, $Sqrt$, Abs
Max-Pooling	$MaxP$
Feature Extraction	$Hist$, $Conca$
Concatenation	$Root$

Region Detection Functions: The $RegionR$ function takes five arguments, i.e, $Image$, X, Y, W, and H, as inputs and returns a small rectangle region. The $RegionS$ function takes four arguments, i.e, $Image$, X, Y, and S, as inputs and returns a small square region. The $Image$ argument is the input image, where the regions are detected from. The width and height of the image are $Image_{width}$ and $Image_{height}$. The X and Y arguments indicate the horizontal and vertical coordinates of the top-left point of the detected region in the input image (the coordination of the top-left point of the image is (0, 0)). The W and H (or S) arguments determine the width and the height (or size) of the detected region. With these arguments, the $RegionR$ function returns a small region: $Image[X : min(X+W,\ Image_{width}),\ Y : min(Y+H,\ Image_{height})]$. The $RegionS$ function returns a small region: $Image[X : min(X+S,\ Image_{width}),\ Y : min(Y+S,\ Image_{height})]$. It is noted that these two functions can reduce the dimensionality of the produced features by selecting the most effective regions from a large input image. An example is shown in Fig. 3 to illustrate region detection. The two region detection functions form the internal nodes of GP trees. During the evolutionary learning process, the region detection operators and their parameter values can be changed or modified via crossover and mutation in order to detect better regions for feature description.

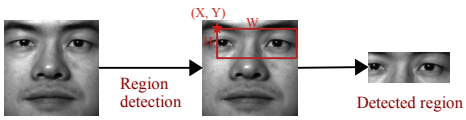


Fig. 3. Example to illustrate region detection.

Filtering/Preprocessing Functions: These functions can change the pixel values of an image by performing filtering, preprocessing or arithmetic operations. To show the effect of these filtering/preprocessing functions, an example image and the images after the corresponding functions are shown in Fig. 4. The Gau function is widely used for image smoothing, which performs filtering/preprocessing using a filter generated by a two-dimensional Gaussian function. The standard deviation of the Gaussian function σ is a terminal of EFLGP. The $GauD$ function can find edges in the image by performing the filtering using the derivatives of the Gaussian filter. The order of the derivative along the X axis is o_1 and the order of the derivative along the Y axis is o_2. The $Gabor$ function can capture salient visual information, such as spatial localisation and orientation selectivity [28]. The filter in $Gabor$ is generated from the product of a Gaussian function with a complex harmonic function. The $Gabor$ function takes an image, θ and f as inputs and returns an image. θ is the orientation in radians of the harmonic function and f is the spatial frequency of the harmonic function. They are the most important parameters of Gabor. The Lap function can detect the flat area or the area with significant edges in an image by performing Laplacian filtering to the image. The $LoG1$ and $LoG2$ functions perform Laplacian of Gaussian filtering to the image. The standard deviation of the Gaussian function is 1 in $LoG1$ and 2 in $LoG2$. The results produced by $LoG1$ and $LoG2$ are less sensitive to noise compared with that by Lap. The $SobelX$ and $SobelY$ functions perform edge detection

along with the horizontal or vertical directions, respectively. The $Sobel$ function calculates the gradient magnitude of the results produced by $SobelX$ and $SobelY$. The Med function performs non-linear median filtering and the $Mean$ filter performs linear mean filtering. Both of them can be employed for denoising/smoothing by replacing the pixel value with the median/mean value of its neighbours. The Max and Min functions perform the morphological filtering by replacing the pixel value with the maximum and minimum value of its neighbours, respectively.

TABLE II
FUNCTIONS OF EFLGP

Function	Input	Output	Description
$Root$	2 vectors	1 vector	Concatenate two vectors into a vector
$Hist$	1 image	1 vector	Extract histogram features from small window of the image
$Conca$	1 image	1 vector	Concatenate each rows of the image into a vector
$MaxP$	1 image, K_1, K_2	1 image	Max-pooling with kernel size of $K_1 \times K_2$
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
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	Median filter
$Mean$	1 image	1 image	Mean filter
Min	1 image	1 image	Min filter
Max	1 image	1 image	Max filter
$MAdd$	2 images	1 image	Add two images with different sizes.
$MSub$	2 images	1 image	Subtract two images with different sizes
$ReLU$	1 image	1 image	Return $max(0,\ pixel\ value)$ for each pixel
$Sqrt$	1 image	1 image	Sqrt each pixel value in the image, return 1 if the pixel value is negative or zero.
Abs	1 image	1 image	Return $\lvert pixel\ value \rvert$ for each pixel
$RegionR$	1 image, X, Y, W, H	1 image	Return a small region of the image
$RegionS$	1 image, X, Y, S	1 image	Return a small square region of the image

The other functions, $MAdd$, $MSub$, $ReLU$, $Sqrt$, and Abs, are employed in the image filtering process to rescale the range of the pixel values of an image or to generate new features. As region detection is performed before image filtering, the $MAdd$ and $MSub$ functions are employed to perform the add and $subtract$ operations on two regions with different sizes. These two functions find the minimum size of the two input regions, cut the two regions according to the minimum sizes and perform Add or $Subtract$ to the two regions. These two functions may generate informative features by enhancing or complementing different regions from the same image. The $ReLU$, $Sqrt$ and Abs functions are used for rescaling the input image and transforming the pixel values from negative to zero or positive, which might enhance the effect of the filtering operations.

Max-Pooling Function: The max-pooling function is commonly used in object recognition and responses robust to the clutter or multiple stimuli in the receptive filed [22]. The $MaxP$ function in EFLGP takes an image and two integers as inputs and returns a smaller image, which can reduce the dimensionality of the feature vector to avoid producing a high-dimensional feature vector. The $MaxP$ function samples the maximum value from a sliding window in the image and replaces this window with the maximum value. The two integers $K1$ and $K2$ represent the kernel size of $MaxP$.

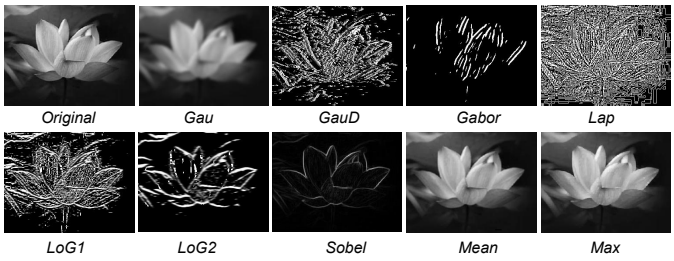


Fig. 4. An example image of lotus flower and the output images convolved by the corresponding filters (the top-left image is the original image).

Feature Extraction Functions: The two feature extraction functions, i.e., $Hist$ and $Conca$, produce different features from the input image. The $Hist$ function extracts 10 histogram features from each 5×5 sliding window of the input image. The final outputs of $Hist$ are a set of histogram features. The $Conca$ function concatenates each row of the images into a vector. This function does not perform any feature transformation but directly uses the pixel values of the image since the input images for $Hist$ and $Conca$ can be processed by different filtering/preprocessing and pooling functions. Note that the children node of the $Hist$ and $Conca$ functions is the $MaxP$ function.

Concatenation Function: The concatenation function $Root$ concatenates two vectors into a vector. This function can form the root node of the GP program by using two of the $Hist$, $Conca$ and $Root$ as its children nodes. With this function, variable numbers of features can be combined to form the output features for image classification.

D. Terminal Set

Terminals can be used to form the leaf nodes of GP trees. The details of these terminals are listed in Table III. They are $Image$, X, Y, S, W, H, $K1$, $K2$, θ, f, σ, o_1, and o_2. The $Image$ terminal represents the input image, which is normalised by dividing 255. The X, Y, S, W, and H terminals indicate parameters of the $RegionR$ and $RegionS$ functions. The θ and f terminals indicate parameters of the $Gabor$ function. The σ terminal indicates the standard deviation of a Gaussian function, and the o_1 and o_2 terminals represent parameters of the $GauD$ function. The value ranges of these parameters are listed in Table III. The values of these terminals are randomly initialised according to their ranges and can be changed by mutation operator during the evolutionary process.

IV. EXPERIMENT DESIGN

A large number of experiments are conducted to evaluate the performance of the proposed EFLGP approach on different image classification tasks of varying difficulty under the original, blur, low contrast, and noise scenarios. To show the effectiveness of the proposed EFLGP approach, a set of benchmark methods are employed for comparisons.

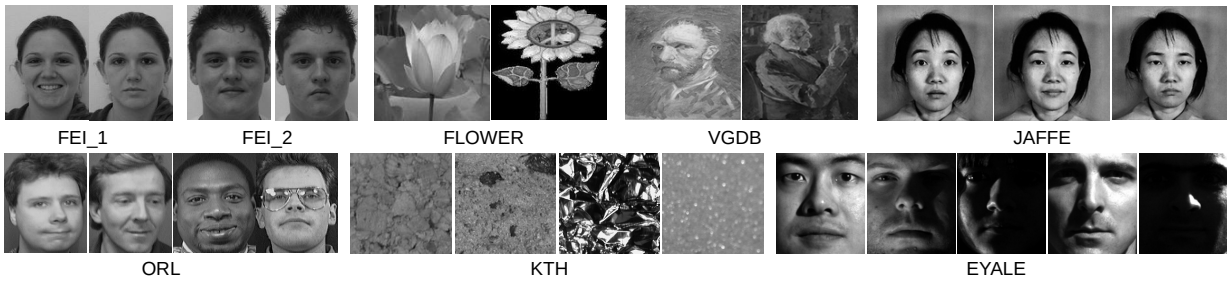


Fig. 5. Example images from the JAFFE, YALE, FEI_1, FEI_2, SCENE, TEXTURE, FLOWER and VGDB datasets.

TABLE III
TERMINAL SET

Terminal	Type	Description
$Image$	Image	The input grey-scale image (2D array containing pixel values in the range $[0,\ 1]$)
X, Y	Integer	The coordinates of the top-left point of a detected region. They are in the range $[0,\ Image_{width}-20]$ or $[0,\ Image_{height}-20]$
S	Integer	The size of a square region detected by the $RegionS$ function. It is in the range $[20,\ 50]$
W, H	Integer	The width and height of a rectangle region detected by the $RegionR$ function. They are in the range $[20,\ 50]$
$K1$, $K2$	Integer	The kernel size of the $MaxP$ function. They are in the range $[2,\ 10]$ with a step of 2
θ	Float	The orientation of the $Gabor$ filter. It is in the range $[0,\ 7\pi/8]$ with a step of $\pi/8$ [28]
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 $[0,\ 4]$ [28]
σ	Integer	The standard deviation of the Gaussian filter. It is in the range $[1,\ 3]$
o_1, o_2	Integer	The orders of the Gaussian derivatives. They are in the range $[0,\ 2]$

A. Original Datasets

Eight image classification datasets of varying difficulty are employed to examine the performance of the proposed EFLGP approach. They are FEI_1 [44], FEI_1 [44], FLOWER [45], VGDB [46], JAFFE [47], ORL [48], KTH [49], and EYALE [50]. These datasets include facial expression classification (FEI_1, FEI_2, JAFFE), face recognition (ORL, EYALE), texture classification (KTH), object classification (FLOWER), and painting classification (VGDB), which are representative tasks in image classification.

The FEI_1 and FEI_2 datasets contain 200 face images of Brazilian with different appearances, hairstyle and adorns in natural and smile expressions [44]. The VGDB dataset is from the VGDB-2016 dataset for identification of Vincent Van Gogh's paintings [46]. This task is very difficult because all the images are abstract without any particular objects inside. The FLOWER dataset has the *Lotus* and *Sunflower* classes from the Caltech 101 dataset [45], which is an object classification task. The JAFFE dataset contains seven different expressions of ten Japanese women [47]. The ORL dataset [48] is a small dataset of face recognition, which has 40 different classes of faces with open or closed eyes, smiling or nonsmiling, and glasses or nonglasses. The KTH dataset has ten classes of texture images sampled in nine scales with three poses under four lighting conditions. The EYALE dataset [50] is a large dataset, having 2,424 face images of 38 classes. The face images have high variations in illumination conditions and pose.

The FEI_1, FEI_1, FLOWER, VGDB, and KTH datasets are split using a commonly used proportion, i.e., 75% images for training and 25% for testing [51]. The JAFFE dataset uses 20 images per class for training and the remaining images for testing because it only has about 30 images per class. The ORL dataset has 7 images per class for training and 3 images per class for testing. The EYALE dataset is a large dataset so that about 50% images are used for training and the remaining images are used for testing. The information about these datasets, including the image size, the number of classes, the number of images in the training and test sets, are listed in Table IV. Several example images of these datasets are shown in Fig. 5.

TABLE IV
DATASET PROPERTIES

Name	Image size	#Class	Training set	Test set
FEI_1	180 × 130	2	150	50
FEI_2	180 × 130	2	150	50
VGDB	200 × 200	2	247	83
FLOWER	100 × 100	2	112	38
JAFFE	128 × 128	7	140	73
ORL	92 × 112	40	280	120
KTH	100 × 100	10	600	210
EYALE	100 × 100	38	1,209	1,213

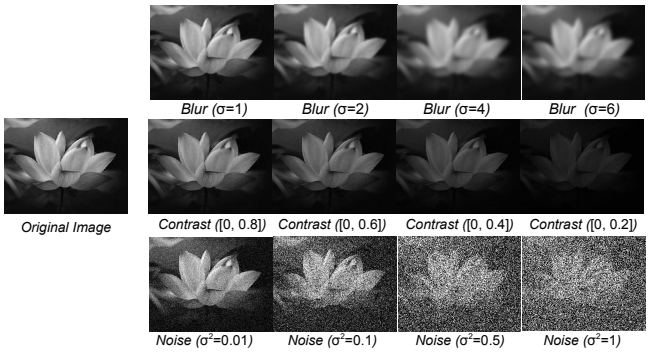


Fig. 6. An example image (left) and the images after corresponding blurring, lowering contrast and adding noise.

It is noted that there are many other datasets with a large number of images that the proposed approach has not been examined on. Different from the deep learning methods, which can be sped up by the graphics processing unit (GPU) imple-

mentation, evolutionary computation techniques including the GP-based methods are currently implemented on the central processing unit (CPU). Since the focus of this paper on low-quality image classification and a large number of experiments are conducted, we only test the proposed EFLGP approach on the datasets with a relatively small number of images because of the high computational cost. It is noteworthy that the experiments on these eight original/clean datasets are conducted to show the effects of the different levels of blur, contrast and noise on the classification performance degradation.

B. Low-Quality Datasets

Low-quality datasets are formed based on these eight datasets and are used in the experiments. For better comparing and analysing the effects of blur, noise and low contrast, we add them separately into the datasets. To increase the difficulty, different levels of blur/low contrast/noise are randomly added to the training and test sets. It is also worthy to investigate the performances of the proposed approach and the baseline methods on classifying images with more than one type of distortions, i.e., more challenging tasks. Due to the page limit, we compare the classification performance of the proposed approach with the baseline methods on two datasets with multiple types and levels of blur/low contrast/noise in the supplementary materials.

Blurring Datasets: Blur image datasets are formed by using Gaussian blur with different standard deviations (σ) to randomly deal with each image in the original dataset. Based on [5], four different values, i.e., 1, 2, 4, and 6, are employed in the Gaussian blur filter. For each image, it has a chance of 25% to be convolved by one of the four filters. Example images are shown in the first row of Fig. 6.

Low-Contrast Datasets: Low-contrast images are obtained by adjusting the contrast/brightness of the image. Four different levels of contrast reduction are utilised to lower the range of the pixel values in an image. The four levels are set as $[0,\ 0.8]$, $[0,\ 0.6]$, $[0,\ 0.4]$, and $[0,\ 0.2]$ according to [5]. For each image, it has 25% proportion to be adjusted by one of the four levels. Example images are shown in the second row of Fig. 6, where distinguishing the object and the background is difficult when the image has the lowest contrast.

Noisy Datasets: Adding Gaussian noise to the original images forms the noisy datasets. Four types of Gaussian noise with different variances, i.e., 0.01, 0.1, 0.5, and 1 [5], are employed and each has a chance of 25% to be added to an image. As example images shown in the third row of Fig. 6, it is obvious that high-level noise makes the object in the image difficult to recognise.

C. Benchmark Methods

To show the effectiveness of EFLGP, 15 effective methods are used for comparisons. These methods include hand-crafted features, two CNNs, and deep features.

Hand-Crafted Features: Seven well-known feature extraction methods are employed to extract features for classification. These methods are DIF [27], Histogram, GLCM [12], Gabor [28], SIFT [15], HOG [14], and LBP [23], which have been introduced in Section II. For Gabor and HOG, which produce a large number of features, the mean value of every small grid is sampled to form the final feature vector [22]. More details of these methods are listed in Table V.

TABLE V
HAND-CRAFTED FEATURES

Method	Description
DIF	Domain independent features [27]
Histogram	256 histogram features based on the pixel values of the image
GLCM	GLCM features [12]. Four different orientations are used and the contrast, dissimilarity, homogeneity, energy, correlation, and angular second moment are extracted from each GLCM
Gabor	Gabor bank features. Forty Gabor filters with eight different orientations at five scales are used [28]. The mean value of each 32×32 grid is extracted to form the features
SIFT	128 SIFT features. The whole image is used as a keypoint [29]
HOG	A HOG image is generated by using the HOG descriptor with the same parameter settings in [14]. The mean value of each 20×10 grid is extracted from the HOG image
LBP	256 LBP histogram features [13]. In LBP, the number of neighbours is set to 8 and the radius is set to 1.5

CNNs: The first CNN method is the well-known LeNet-5 and its main hyper-parameters are the same as those in [52]. The ReLU activation function is employed and the final layer uses the commonly used softmax for classification. The second CNN method has two convolutional layers, one max-pooling and two fully-connected layers. The first convolutional layer contains 32 filters with a kernel size of 3×3 and the second convolutional layer has 64 filters with a kernel size of 3×3. The max-pooling has a kernel size of 4×4 without padding. Dropout is added to the pooling layer and the first fully-connected layer with 0.25 and 0.5 probabilities, respectively, to avoid overfitting [53]. The activation function is ReLU and the softmax is used for classification. The loss function is cross-entropy and the adaptive subgradient method is used to train CNNs [54]. In LeNet-5 and CNN, the batch size is 128 according to [55] and the number of epochs is 500. Early stopping is employed to avoid overfitting. The training will be stopped if the training accuracy reaches 100%.

Deep Features: Five pre-trained deep CNNs are used to extract features from the dataset and the extracted deep features are fed into a linear SVM to perform classification [33, 56]. The deep features are from the pre-trained models of VGG, Xception, InceptionV3, ResNet, and DenseNet, respectively [57]. These models were trained on the famous large-scale dataset, ImageNet [58], which is a 1000-class dataset. We choose these models because they are easily obtained and are commonly used [33, 56]. The deep features are extracted by removing the final or final two layers of these models. Because some datasets in this study are very small, e.g., the ORL dataset only has 6 images per class for training, it is impossible to train these deep models using such a small dataset. Therefore, these pre-trained models are used as feature extractors. We have also chosen three deep models pre-trained on a large face dataset, VGG-Face [59] as feature extractors and compared the performance of the proposed method with those deep models on the eight datasets. Due to the page limit, the results of these three deep models are discussed in the supplementary materials.

D. Parameter Settings

The implementation of the EFLGP approach is based on the *DEAP (Distributed Evolutionary Algorithm in Python)* [60] package, which is a popular evolutionary algorithm package in Python. The parameter settings for EFLGP are based on the commonly used parameter settings of the GP community [21]. The population size is 500 and the maximum number of generations is 50. The crossover rate is 0.8, the mutation rate is 0.19 and the elitism rate is 0.01. The selection method is Tournament selection with size seven. The tree depth is set between 2 and 10. The population generation method is the *ramped half-and-half* method.

The features learned by EFLGP, the hand-crafted features and the deep features are fed into the linear SVM to perform classification, respectively. The linear SVM is implemented in the *scikit-learn* [61] package with its default parameter settings for simplification. Follow the convention of EC, the experiments of the EFLGP approach run 30 independent times on each dataset. To achieve a fair comparison, the benchmark methods also run 30 independent times on each dataset. The mean test accuracy and the standard deviation obtained by these methods are reported.

V. Results and Discussions

This section discusses the results obtained by EFLGP and the baseline methods on the eight datasets of original images and images with different levels of blur, low contrast and noise. The mean accuracy and standard deviation obtained by these methods on the test sets of the eight datasets are listed in Tables VI and VII. Each block of these two tables shows the results on one dataset under the four scenarios. To show the significance of performance improvement, Wilcoxon rank-sum test with a 5% significance level is conducted to compare EFLGP with a baseline method. In Tables VI and VII, the symbols "+", "=" and "–" denote that EFLGP achieves significantly better, similar or significantly worse results than/to the compared method. The final row of each block summaries the results of the significance test on every dataset. The overall summary of the significance test is listed in Table VIII.

A. Results on Original Datasets

The second column of Tables VI and VII lists the classification results of the EFLGP approach and the 14 baseline methods on the original dataset. Compared with using the seven types of hand-craft features, using the features learned by EFLGP achieves significantly better results in 53 comparisons out of the total 56 comparisons on the eight datasets. Compared with the two CNNs, EFLGP achieves significantly better results in 11 comparisons and similar results in 3 comparisons out of the total 16 comparisons. EFLGP achieves significantly better performance in 36 out of the 40 comparisons to the methods using five different types of deep features. Overall, EFLGP achieves significantly better performance in 100 comparisons and similar performance in 6 comparisons out of the total 112 (14×8) comparisons. The results show that EFLGP achieves better classification results than the 14 baseline methods using hand-crafted features,

TABLE VI
Classification accuracy (%) of EFLGP and 14 baseline methods on FEI_1, FEI_2, FLOWER, and VGDB datasets under the four scenarios

	Mean+St.dev	Mean+St.dev	Mean+St.dev	Mean+St.dev
FEI_1	**Original**	**Blur**	**Low contrast**	**Noise**
DIF	61.13±4.97+	62.93±5.89+	59.80±6.46+	52.33±2.97+
Histogram	48.13±3.44+	50.07±1.23+	55.80±5.24+	47.27±2.80+
GLCM	49.67±0.76+	51.40±2.47+	47.60±7.67+	50.07±2.49+
Gabor	71.60±8.01+	72.00±6.93+	74.60±9.38+	71.20±9.86+
SIFT	82.00±0.00+	70.00±0.00+	80.00±0.00+	60.00±0.00+
HOG	94.00±0.00+	88.00±0.00+	92.00±0.00+	49.47±3.10+
LBP	62.47±3.55+	62.40±5.97+	59.33±3.69+	48.60±3.11+
LeNet-5	94.40±1.99+	93.40±0.93+	**95.20±1.86**–	80.20±10.9+
CNN	96.00±1.58=	91.00±1.36+	91.20±2.61+	77.80±14.57+
VGG	50.00±0.00+	50.00±0.00+	50.00±0.00+	50.00±0.00+
Xception	75.73±11.67+	52.67±3.12+	61.07±11.42+	54.00±4.20+
InceptionV3	86.00±3.93+	70.87±11.16+	78.87±5.40+	53.20±3.04+
ResNet	50.00±0.00+	50.00±0.00+	50.00±0.00+	50.00±0.00+
DenseNet	57.53±6.98+	50.00±0.00+	50.60±1.59+	52.60±3.16+
EFLGP	**96.0±1.58**	**95.27±1.7**	93.93±2.32	**87.20±2.91**
Overall	**13+, 1=**	**14+**	**13+, 1–**	**14+**
FEI_2	**Original**	**Blur**	**Low contrast**	**Noise**
DIF	62.80±6.21+	62.40±7.03+	65.20±8.04+	54.67±7.49+
Histogram	50.13±2.57+	51.80±1.32+	45.80±4.41+	46.00±4.98+
GLCM	50.13±0.73+	50.27±3.27+	48.20±3.46+	48.93±4.83+
Gabor	65.67±5.23+	66.47±5.93+	68.67±5.42+	60.60±6.17+
SIFT	78.00±0.00+	64.00±0.00+	82.00±0.00+	48.00±0.00+
HOG	88.00±0.00+	76.00±0.00+	92.00±0.00=	47.73±3.31+
LBP	57.60±3.62+	55.67±3.68+	53.67±3.28+	53.07±5.67+
LeNet-5	90.80±1.86+	86.80±12.64=	86.00±12.34+	72.20±1.69+
CNN	86.20±2.64+	85.40±12.07+	86.80±3.04+	73.00±3.43+
VGG	50.00±0.00+	50.00±0.00+	50.00±0.00+	50.00±0.00+
Xception	69.60±11.58+	51.80±2.64+	53.80±3.17+	53.53±4.83+
InceptionV3	84.40±5.21+	68.47±10.22+	78.53±9.48+	58.33±2.47+
ResNet	50.00±0.00+	50.00±0.00+	50.00±0.00+	50.00±0.00+
DenseNet	59.13±6.23+	50.27±0.69+	51.33±2.80+	52.67±2.37+
EFLGP	**92.60±3.11**	**91.73±1.95**	**92.33±2.93**	**77.73±4.13**
Overall	**14+**	**13+, 1=**	**13+, 1=**	**14+**
FLOWER	**Original**	**Blur**	**Low contrast**	**Noise**
DIF	80.53±6.05+	75.79±6.35+	79.56±6.13+	68.86±8.18+
Histogram	54.82±2.94+	56.05±3.18+	52.37±3.93+	55.70±4.32+
GLCM	58.16±9.66+	50.70±5.73+	54.65±6.17+	44.47±5.80+
Gabor	66.40±8.92+	66.58±11.1+	69.74±8.54+	68.60±5.21+
SIFT	86.84±0.00=	78.95±0.00=	**86.84±0.00**–	76.32±0.00+
HOG	70.52±1.27+	52.02±2.99+	60.27±1.27+	52.89±3.27+
LBP	66.58±2.60+	54.30±2.80+	71.23±4.68+	54.38±4.05+
LeNet-5	84.74±1.60+	**81.58±2.39**–	83.95±2.79=	80.27±1.79=
CNN	**88.15±1.79**=	81.05±1.07=	83.68±3.34=	80.26±1.34=
VGG	50.35±5.34+	49.65±5.34+	51.40±5.16+	51.05±5.24+
Xception	82.02±8.61=	72.89±9.28+	74.74±8.42+	62.46±8.66+
InceptionV3	81.58±0.00+	73.77±4.06+	75.35±3.49+	65.70±0.48+
ResNet	51.75±5.04+	51.05±5.24+	50.35±5.34+	52.45±4.73+
DenseNet	80.18±3.22+	70.88±9.57+	66.05±7.58+	78.07±4.60=
EFLGP	86.75±3.62	80.35±3.95	83.16±3.36	**80.35±3.43**
Overall	**11+, 3=**	**11+, 2=, 1–**	**11+, 2=, 1–**	**11+, 3=**
VGDB	**Original**	**Blur**	**Low contrast**	**Noise**
DIF	55.62±10.42+	57.75±8.81+	50.36±10.25+	50.80±11.92+
Histogram	62.21±0.80+	62.21±0.59=	62.65±0.00+	55.18±8.38+
GLCM	53.33±9.97+	54.82±10.59+	52.25±7.63+	54.38±11.86=
Gabor	56.02±8.42+	54.26±6.80+	56.67±7.32+	54.66±11.00=
SIFT	60.24±0.00+	61.45±0.00+	61.45±0.00+	**66.27±0.00**–
HOG	57.23±0.69+	60.44±1.10+	57.03±2.04+	56.99±6.30+
LBP	**80.56±3.28**–	62.05±8.24=	**80.16±2.60**–	60.12±9.71=
LeNet-5	58.07±4.93+	58.07±2.38+	47.47±12.61+	59.52±2.40=
CNN	61.81±2.05+	63.26±2.07=	58.43±7.93+	60.84±3.21=
VGG	53.37±12.4+	54.22±12.13+	63.53±3.55+	53.37±12.4=
Xception	61.45±8.86=	56.18±9.68+	72.29±0.00–	52.93±10.61+
InceptionV3	69.32±1.21–	**65.90±8.89**–	79.52±0.00–	64.42±0.69–
ResNet	52.53±12.61+	53.37±12.4+	56.87±6.30+	55.06±11.79=
DenseNet	60.48±8.09+	55.26±11.18+	72.29±0.00–	58.88±6.39=
EFLGP	66.47±4.52	63.98±3.80	66.63±4.41	60.72±3.92
Overall	**11+, 1=, 2–**	**10+, 3=, 1–**	**10+, 4–**	**4+, 8=, 2–**

learned features (by LeNet-5 and CNN), and deep features in almost all the comparisons on the original image dataset.

In terms of different types of datasets, EFLGP performs superiorly on facial expression classification and face classification datasets, i.e., FEI_1, FEI_2, JAFFE, ORL, and EYALE. On these five datasets, EFLGP achieves significantly better or similar performance than any of the baseline methods except for InceptionV3 on ORL. EFLGP can capture discriminative information from face images for effective classification. The new designs of EFLGP in the program structure and the function set allow it to detect minor differences among different expressions using region detection functions and generate informative features using image operators. Consistently, EFLGP performs well on the object dataset, FLOWER, which has images with high variations in rotation, scale, and illumination condition. On FLOWER, EFLGP achieves significantly better or similar performance than any of the baseline methods. The results indicate that the features learned by EFLGP can handle image variations in order to obtain good classification performance. The performance of EFLGP on VGDB is better than 11 methods and worse than two methods, i.e., LBP and InceptionV3. VGDB is a challenging task of understanding the painting style rather than objects. This abstract information is difficult for EFLGP to learn using the region detection functions and the image filtering functions. EFLGP performs worse than LBP, CNN and InceptionV3 and better than the other 11 baseline methods on the KTH dataset, which is a texture classification dataset. EFLGP can learn effective features for classifying texture images, but the features are not as good as the LBP features, the features learned by CNN and the deep features of InceptionV3. To sum up, the detailed analysis indicates that EFLGP is more effective for learning features for facial expression or object classification than texture and painting classification. The reason may be that EFLGP learns local features from the automatically detected regions, which are often more effective for object classification than texture classification.

B. Results on Low-Quality Datasets

In the experiments, the low-quality datasets have images with different levels of blur, low contrast and noise, respectively. The classification results of the eight datasets under the three scenarios are listed in Tables VI and VII. To better show the effects of different factors in classification performance, the mean accuracy (%) of EFLGP and the eight best baseline methods (i.e., SIFT, HOG, LBP, LeNet-5, CNN, Xception, InceptionV3, and DenseNet) are drawn in Fig. 7. It is noted that only eight best baseline methods (three hand-crafted features, two CNNs and three deep features) are drawn in Fig. 7 for simplification and a clear plot.

On the eight blurred datasets, EFLGP achieves significantly better or similar results in 106 comparisons out of the total 112 comparisons. EFLGP achieves significantly better results than any of the baseline methods on the FEI_1, FEI_2, and JAFFE datasets. EFLGP ranks the second on the VGDB, ORL and EYALE datasets and ranks the third on the FLOWER and KTH datasets among all these methods. As a result,

TABLE VII
CLASSIFICATION ACCURACY (%) OF EFLGP AND 14 BASELINE METHODS ON JAFFE, ORL, KTH, AND EYALE DATASETS UNDER THE FOUR SCENARIOS

	Mean+St.dev	Mean+St.dev	Mean+St.dev	Mean+St.dev
JAFFE	**Original**	**Blur**	**Low contrast**	**Noise**
DIF	28.22±8.57+	24.61±7.44+	28.13±1.72+	16.21±2.25+
Histogram	20.28±1.16+	12.38±0.57+	13.93±1.53+	19.04±2.08+
GLCM	15.02±2.23+	16.17±2.40+	14.98±1.29+	16.85±2.67+
Gabor	26.30±6.16+	27.31±5.60+	31.56±1.74+	21.83±5.31+
SIFT	46.58±0.00+	27.40±0.00+	46.58±0.00+	17.81±0.00+
HOG	69.45±2.55+	47.63±1.75+	22.74±0.68+	25.98±2.51+
LBP	21.97±2.26+	16.99±2.09+	19.59±2.13+	18.18±3.01+
LeNet-5	75.75±2.42+	57.76±3.58+	17.99±1.06+	45.57±3.31+
CNN	79.27±1.89=	56.21±2.98+	17.99±1.43+	43.34±4.38+
VGG	14.25±1.17+	13.38±1.64+	15.12±1.99+	13.43±1.81+
Xception	41.37±12.29+	18.72±4.75+	16.39±2.20+	21.19±4.60+
InceptionV3	74.70±2.89+	28.54±7.49+	17.54±1.85+	24.11±1.67+
ResNet	17.44±4.91+	16.03±3.10+	14.84±1.53+	13.75±1.71+
DenseNet	35.94±13.0+	21.23±5.87+	15.80±2.09+	23.52±8.92+
EFLGP	**79.82±2.97**	**66.66±3.42**	**77.17±3.16**	**53.88±3.12**
Overall	**13+, 1=**	**14+**	**14+**	**14+**
ORL	**Original**	**Blur**	**Low contrast**	**Noise**
DIF	77.28±8.43+	76.67±6.12+	41.92±0.66+	24.64±7.88+
Histogram	93.33±0.00+	71.78±0.29+	12.89±0.52+	15.72±1.36+
GLCM	7.33±3.07+	8.28±4.69+	3.03±1.06+	3.39±1.27+
Gabor	16.25±5.97+	17.42±7.04+	34.97±1.6+	10.11±3.79+
SIFT	95.83±0.00+	92.50±0.00+	**95.83±0.00**–	49.17±0.00+
HOG	92.53±0.15+	84.94±0.21+	21.67±0.00+	23.64±1.11+
LBP	94.75±0.62+	56.56±1.56+	92.53±0.56–	10.64±1.57+
LeNet-5	93.36±1.74+	94.75±1.78+	9.97±1.30+	82.19±15.26+
CNN	95.83±0.79+	**96.94±1.12**–	10.05±1.07+	69.06±23.87+
VGG	5.61±2.95+	3.28±1.37+	2.64±0.49+	2.42±0.55+
Xception	49.25±16.94+	10.42±14.72+	4.75±3.70+	8.72±7.68+
InceptionV3	**99.86±0.31**–	40.22±20.38+	6.56±4.45+	36.97±0.71+
ResNet	10.14±8.25+	5.50±3.41+	2.58±0.34+	2.67±0.51+
DenseNet	59.00±21.34+	15.81±24.62+	5.64±3.61+	17.06±9.39+
EFLGP	97.28±1.09	95.86±1.13	90.17±3.45	**86.42±2.41**
Overall	**13+, 1–**	**13+, 1–**	**12+, 2–**	**14+**
KTH	**Original**	**Blur**	**Low contrast**	**Noise**
DIF	27.52±7.44+	25.00±9.20+	32.79±0.83+	17.98±6.59+
Histogram	42.41±2.06+	26.33±3.08+	13.33±0.92+	20.43±1.87+
GLCM	22.51±9.68+	25.08±2.54+	10.65±1.61+	14.54±2.10+
Gabor	16.97±7.58+	15.10±7.12+	27.30±0.97+	14.91±5.54+
SIFT	71.43±0.00+	53.33±0.00+	75.24±0.00–	26.67±0.00+
HOG	44.68±3.16+	37.78±0.88+	13.84±0.72+	15.57±2.52+
LBP	**90.03±1.20**–	51.05±2.72+	**90.48±2.08**–	36.22±3.23+
LeNet-5	72.97±2.46+	**67.25±3.03**–	14.65±0.95+	43.33±2.52=
CNN	81.62±1.64–	65.17±2.78–	16.73±1.80+	**46.92±2.23**–
VGG	15.09±5.01+	11.38±1.89+	9.75±0.92+	12.14±2.89+
Xception	62.19±12.18+	28.64±10.5+	19.05±9.43+	30.73±7.41+
InceptionV3	84.35±3.72–	37.71±11.61+	19.38±7.13+	43.62±2.56–
ResNet	15.06±4.89+	12.33±3.25+	10.82±1.08+	10.11±1.18+
DenseNet	45.54±13.97+	19.60±7.87+	14.35±2.99+	27.78±7.12+
EFLGP	80.60±2.30	61.22±2.35	61.22±4.97	42.63±2.76
Overall	**11+, 3–**	**12+, 2–**	**12+, 2–**	**11+, 1=, 2–**
EYALE	**Original**	**Blur**	**Low contrast**	**Noise**
DIF	8.77±1.74+	7.27±2.13+	14.75±0.32+	4.89±1.74+
Histogram	8.46±0.74+	5.03±0.45+	3.70±0.16+	3.80±0.42+
GLCM	3.55±1.35+	3.01±0.41+	2.88±0.18+	2.84±0.34+
Gabor	4.88±1.28+	5.15±1.30+	9.65±0.38+	3.29±0.70+
SIFT	81.32±0.00+	69.14±0.00+	81.15±0.00+	34.32±0.00+
HOG	78.44±0.24+	61.92±0.21+	19.08±0.15+	15.14±1.45+
LBP	54.49±2.86+	17.51±4.60+	53.73±3.25+	7.89±0.42+
LeNet-5	89.29±2.55+	81.08±1.82+	27.02±29.13+	3.02±0.56+
CNN	98.66±0.27+	**91.04±1.17**–	90.25±3.75+	2.77±0.33+
VGG	3.97±1.76+	3.51±1.19+	2.68±0.17+	2.85±0.39+
Xception	31.77±24.4+	17.13±17.91+	5.20±3.69+	7.35±7.37+
InceptionV3	89.76±0.98+	46.24±17.44+	9.20±7.22+	35.09±2.02+
ResNet	7.10±6.16+	6.23±5.20+	2.92±0.51+	2.57±0.23+
DenseNet	53.34±19.62+	21.29±22.17+	7.53±5.18+	16.7±9.44+
EFLGP	**99.70±0.21**	89.13±1.84	**99.85±0.10**	**70.21±3.54**
Overall	**14+**	**13+, 1–**	**14+**	**14+**

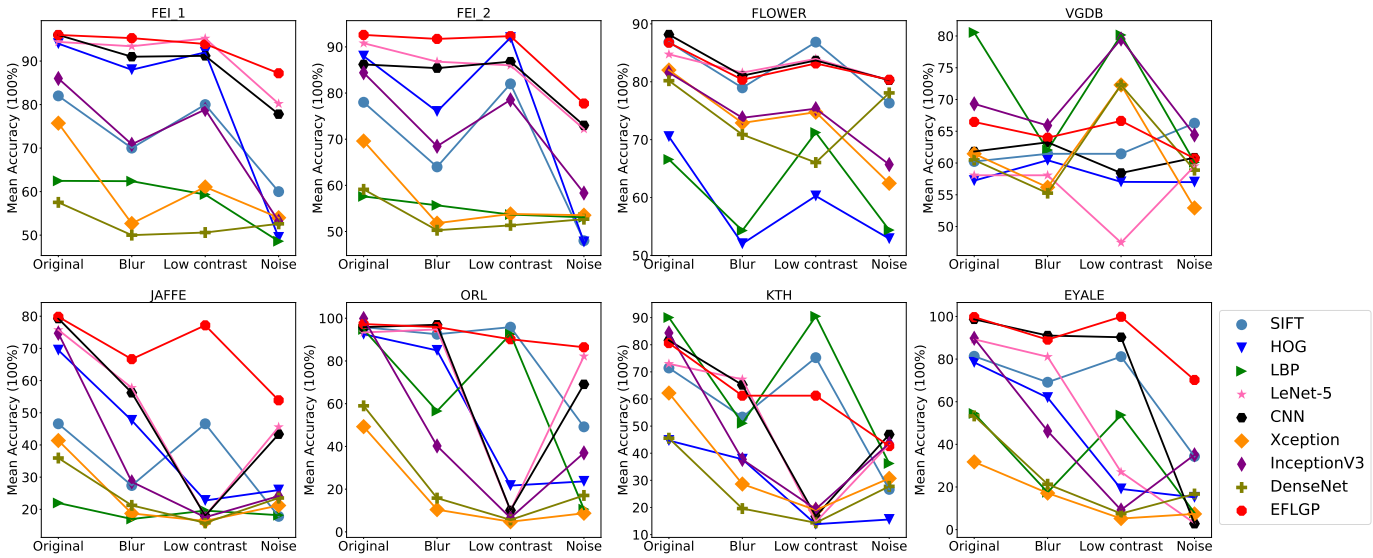


Fig. 7. Mean classification accuracy (%) obtained by the EFLGP approach and eight best baseline methods on the eight datasets under the four scenarios. Each subfigure represents one dataset. The Y axis represents the accuracy and the X axis represents the four different scenarios.

EFLGP is effective for feature learning on the blurred datasets. From Fig. 7, it is clear that some methods such as LBP, HOG, InceptionV3, and SIFT degrade their performances on the blurred datasets. Compared with these methods, the performance of EFLGP is less affected by blurring the images.

On the eight low-contrast datasets, EFLGP obtains significantly better or similar results in 102 comparisons. Specifically, EFLGP achieves better results than any of the benchmark methods on FEI_2, JAFFE and EYALE. It is noticeable that the mean accuracy achieved by EFLGP (77.17%) is much higher than that by the benchmark methods (46.58%) on JAFFE. From Fig. 7, it can be found that the LBP and SIFT methods are less affected by lowering the contrast of images than the other benchmark methods. The reason is that the LBP and SIFT features are invariant to illumination changes. Compared with LBP and SIFT, EFLGP is less affected by adjusting the contrast on more datasets except for KTH. The features learned by EFLGP are less effective for KTH compared with LBP so that adjusting the contrast may increase the difficulty of EFLGP to learn effective features. As a result, on the low-contrast datasets, EFLGP can achieve comparable and stable performance than most benchmark methods.

On the noisy datasets, EFLGP achieves better or similar results in 108 comparisons out of the total 112 comparisons. Specifically, EFLGP achieves better results than any of the benchmark methods on the FEI_1, FEI_2, FLOWER, JAFFE, ORL, and EYALE datasets. Adding noise in images significantly degrades the classification performances of these methods (including EFLGP). However, EFLGP still achieves better results than most of these methods, which indicates that EFLGP is less affected by noise compared with these benchmark methods. This pattern can be easily found from Fig. 7. The performances of HOG, SIFT, LBP, Xception, and Inception degrade significantly on most of the noisy datasets. Compared with these methods, EFLGP can still achieve better accuracy on the noisy datasets.

TABLE VIII
SUMMARY OF SIGNIFICANCE TEST

	Original	Blur	Low contrast	Noise
Significantly better (+)	100	100	99	96
Similar (=)	6	6	3	12
Significantly worse (–)	6	6	10	4

The comparisons show that EFLGP achieves better results than most benchmark methods on these (8×3=24) datasets of different levels of blur, low contrast and noise. From Fig. 7, it is obvious that the performances of these methods (including EFLGP) are affected by blurring, decreasing contrast and adding noise in images. Compared with these benchmark methods, EFLGP is less affected by blur, low contrast and noise, which indicates that EFLGP is effective for feature learning on low-quality image data. It is also noticeable that some of the deep features achieve very low accuracy on the datasets. For example, VGG and ResNet achieve less than 10% accuracy on ORL, because the datasets are small and the numbers of features produced by VGG and ResNet are very large. The results from supplementary materials show that the proposed approach can also achieve better performance than the compared methods when the images have two or three types of distortions, i.e., blur, low contrast and noise.

To sum up, EFLGP is an effective approach for feature learning on low-quality image data. Compared with the 14 benchmark methods, EFLGP achieves significantly better or similar results in 422 comparisons out of the total 448 (14×8) comparisons. The results demonstrate that the features learned by EFLGP are more effective for facial expression and face classification than for painting and texture classification. The results confirm that classifying images with blur, low contrast or noise is more difficult than classifying the original (clean) images. The majority of the methods degrade their performances when the images are blurred, or have low contrast or

noise. The results show that EFLGP is less affected by these factors, which indicates that the goal of this study has been successfully achieved.

VI. Further Analysis

In this section, the example trees/programs evolved by EFLGP are deeply analysed to show the interpretability and to demonstrate how features are extracted and why EFLGP achieves good performance.

An Example Tree/Program on the Original FEI_1 Dataset: An example tree found by EFLGP on the original FEI_1 dataset is shown in Fig. 8. This tree achieves 99.33% classification accuracy on the training set and 98% accuracy on the test set. As shown in the figure, this tree detects a square region and a rectangle region from the input image, employs operators (i.e., Max, Lap, $SobelY$, and $MaxP$) to deal with these regions, and uses $Conca$ to produce 157 features. The square region captures the left middle face and the rectangle region captures the middle area of the face. This is consistent with the fact that the mouth area is discriminative in the face with the *Happy* and *Natural* expressions. These minor differences in the face images are captured by the two region detection functions in the example tree. At the left branch of the example tree, the Max filter and the $MaxP$ function with a kernel size of 10×2 are used to extract the maximum pixel values/features from the detected region. This branch produces 115 (5×23) features. The right branch of the example tree has operators/filters, such as $SobelY$ and Lap, to extract edge features from the detected regions. The $MaxP$ function with a kernel size of 8×8 is employed in the tree to sample the region from 49×43 to 7×6. This branch produces 42 features. In total, there are 157 features produced by the example tree from a 180×130 image.

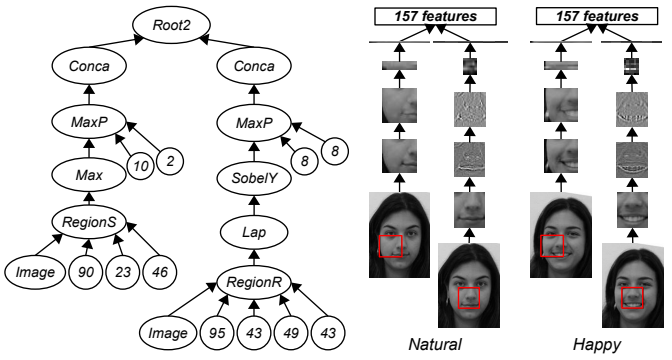


Fig. 8. An example tree (left) evolved by EFLGP on the **original** FEI_1 dataset and two examples to show how it extracts features on the *Natural* (left) image and the *Happy* (right) image. Note that the images are rescaled for better visualisation.

An Example Tree/Program on the Noisy FEI_1 Dataset: An example tree on the noisy FEI_1 dataset is employed to deeply analyse the reason why EFLGP obtains good performance under different scenarios. The tree is shown in Fig. 9, which achieves 94% accuracy on the training set and 92% accuracy on the test set. This tree has two branches split from the top node and has three region detection functions, which detect three smaller regions from the input 180×130 image. The three small regions are different from the two regions detected by the tree on the original dataset in Fig. 8. As shown in Fig. 10, the images have different types of noise so that the discriminative areas, such as the mouth area, might be significantly affected and become less discriminative. In this example tree, the Gau and $Gabor$ functions are employed to deal with the three detected regions. Especially, the Gau function can denoise the images and the $Gabor$ function can describe effective features from noisy images. From Fig. 10, it is clear that the regions after the corresponding operations become smoother and the patterns are clearer. For example, the second region after employing Gau with standard deviation $\sigma = 4$ from the *Happy* class contains less black colour than that from the *Natural* class. In total, this tree produces 186 features from an input image, where 138 (23×6) features are from the left branch and 48 (4×12) features are from the right branch.

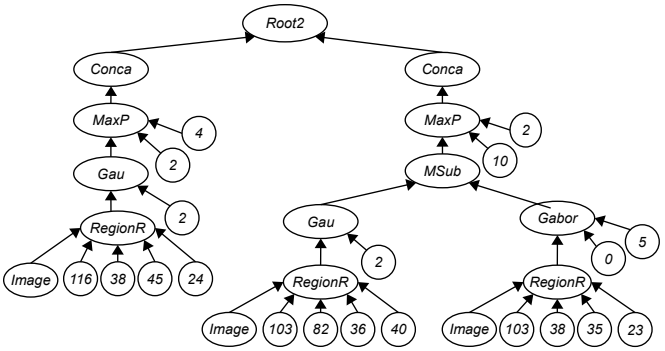


Fig. 9. An example tree evolved by EFLGP on the **noisy** FEI_1 dataset.

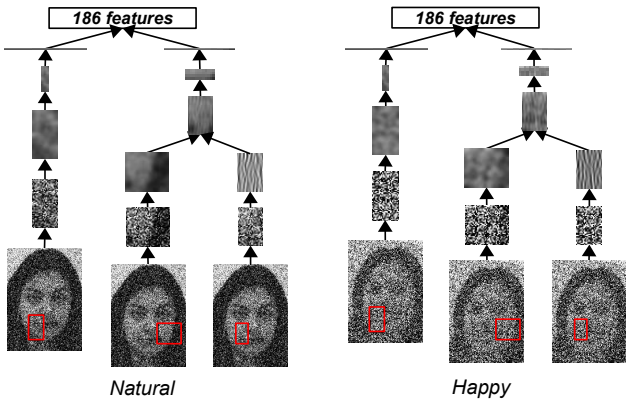


Fig. 10. Two examples to show how the tree in Fig 9 extracts features on the *Natural* (left) images and the *Happy* (right) images. Note that the images are rescaled for better visualisation.

To sum up, the analysis shows that the solutions of EFLGP have high interpretability. The analysis demonstrates that EFLGP can detect small discriminative regions from the large input image, employ image filters/operators to deal with the detected regions and produce a set of effective features. On the noisy dataset, EFLGP can evolve the operators/filters to deal with noise to obtain effective features. This is the reason why EFLGP is more stable and effective for feature learning on low-quality image data.

VII. Conclusions

The goal of this paper was to develop a new GP-based feature learning approach to learning features for classifying images with blur, low contrast or noise. This goal has been successfully achieved by developing the EFLGP approach and examining it on eight different datasets of original, blurred, low-contrast, and noisy images. A new program structure, a new function set including a set of image (preprocessing) operators and a new terminal set were developed in EFLGP. With these designs, EFLGP can evolve solutions that perform image preprocessing and feature extraction in a single tree. Specifically, it detects small regions from the large input image, evolves image operators/filters to learn features and produces a feature vector with dynamic length. The performance of EFLGP was extensively investigated on eight datasets of varying difficulty under four different scenarios, i.e., original, blur, low contrast, and noise. The results demonstrated that the proposed EFLGP approach achieved significantly better or similar results in most comparisons. The results showed that EFLGP was less affected by these factors compared with the 14 benchmark methods. Further analysis showed the high interpretability of the solutions evolved by EFLGP.

Low-quality image classification can be a challenging task. In the future, more advanced techniques or operators, such as image quality assessment method [62] and data argumentation, can be investigated to improve the performance of low-quality image classification tasks.

References

[1] S. A. Thomas, Y. Jin, J. Bunch, and I. S. Gilmore, "Enhancing classification of mass spectrometry imaging data with deep neural networks," in *Proc. IEEE SSCI*, 2017, pp. 1–8.

[2] W. A. Albukhanajer, Y. Jin, and J. A. Briffa, "Neural network ensembles for image identification using pareto-optimal features," in *Proc. IEEE CEC*, 2014, pp. 89–96.

[3] Z. Zhong, J. Li, D. A. Clausi, and A. Wong, "Generative adversarial networks and conditional random fields for hyperspectral image classification," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 3318–3329, 2019.

[4] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–22, 2020, DOI: 10.1109/TITS.2019.2962338.

[5] Y. Zhou, D. Liu, and T. Huang, "Survey of face detection on low-quality images," in *Proc. 13th IEEE Inter. Conf. Auto. Face & Gest. Recog.*, 2018, pp. 769–773.

[6] S. Diamond, V. Sitzmann, S. Boyd, G. Wetzstein, and F. Heide, "Dirty pixels: Optimizing image classification architectures for raw sensor data," *arXiv preprint arXiv:1701.06487*, 2017.

[7] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[8] G. B. P. da Costa, W. A. Contato, T. S. Nazare, J. E. Neto, and M. Ponti, "An empirical study on the effects of different types of noise in image classification tasks," *arXiv preprint arXiv:1609.02781*, 2016.

[9] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[10] S. B. Park, J. W. Lee, and S. K. Kim, "Content-based image classification using a neural network," *Pattern Recognit. Lett.*, vol. 25, no. 3, pp. 287–300, 2004.

[11] M. Sonka, V. Hlavac, and R. Boyle, *Image pre-processing*. Boston, MA: Springer US, 1993, pp. 56–111. [Online]. Available: https://doi.org/10.1007/978-1-4899-3216-7_4

[12] R. M. Haralick, K. Shanmugam *et al.*, "Textural features for image classification," *IEEE Trans. Syst. Man Cybern.*, no. 6, pp. 610–621, 1973.

[13] 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.

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

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

[16] 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.

[17] Y. Bi, B. Xue, and M. Zhang, "Evolving deep forest with automatic feature extraction for image classification using genetic programming," in *Proc. PPSN*. Springer, 2020, pp. 3–18.

[18] D. Cai, K. Chen, Y. Qian, and J.-K. Kämäräinen, "Convolutional low-resolution fine-grained classification," *Pattern Recognit. Lett.*, vol. 119, pp. 166–171, 2019.

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

[20] 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).

[21] Y. Bi, B. Xue, and M. Zhang, "Genetic programming with image-related operators and a flexible program structure for feature learning to image classification," *IEEE Trans. Evol. Comput.*, 2020. DOI:10.1109/TEVC.2020.3002229.

[22] 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.

[23] 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.

[24] Y. Bi, B. Xue, and M. Zhang, "An effective feature learning approach using genetic programming with image descriptors for image classification [research frontier]," *IEEE Comput. Intell. Mag.*, vol. 15, no. 2, pp. 65–77, 2020.

[25] H. Hosseini, B. Xiao, and R. Poovendran, "Google's cloud vision api is not robust to noise," in *Proc. IEEE ICMLA*, 2017, pp. 101–105.

[26] 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.

[27] M. Zhang, V. B. Ciesielski, and P. Andreae, "A domain-independent window approach to multiclass object detection using genetic programming," *EURASIP J. Adv. Sig. Pr.*, vol. 2003, no. 8, pp. 841–859, 2003.

[28] D. Tao, X. Li, X. Wu, and S. J. Maybank, "General tensor discriminant analysis and gabor features for gait recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 10, pp. 1700–1715, 2007.

[29] 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.

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

[31] C. Ryan, J. Fitzgerald, K. Krawiec, and D. Medernach, "Image classification with genetic programming: Building a stage 1 computer aided detector for breast cancer," in *Handbook of Genetic Programming Applications*. Springer, 2015, pp. 245–287.

[32] S. D. Liang, "Optimization for deep convolutional neural networks: How slim can it go?" *IEEE Trans. Emerg. Topics Comput.*, vol. 4, no. 2, pp. 171–179, 2018.

[33] F. Cen and G. Wang, "Boosting occluded image classification via subspace decomposition-based estimation of deep features," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 3409 – 3422, 2020.

[34] A. Khan, A. S. Qureshi, M. Hussain, M. Y. Hamza *et al.*, "A recent survey on the applications of genetic programming in image processing," *arXiv preprint arXiv:1901.07387*, 2019.

[35] 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.

[36] 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.

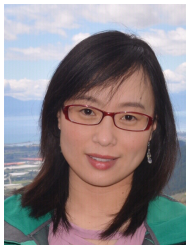
[37] Y. Bi, B. Xue, and M. Zhang, "An automatic feature extraction approach to image classification using genetic programming," in *Proc. Int. Conf. Appl. Eov. Comput.*, 2018, pp. 421–438.

[38] ——, "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.
[39] S. R. Price and D. T. Anderson, "Genetic programming for image feature descriptor learning," in *Proc. IEEE CEC*, 2017, pp. 854–860.
[40] 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.
[41] 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.
[42] D. Tao, X. Tang, X. Li, and X. Wu, "Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 7, pp. 1088–1099, 2006.
[43] D. J. Montana, "Strongly typed genetic programming," *Evol. Comput.*, vol. 3, no. 2, pp. 199–230, 1995.
[44] C. E. Thomaz, "Fei face database," *online: http://fei.edu.br/˜cet/facedatabase.html*, Mar 2012.
[45] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Comput. Vis. Image Und.*, vol. 106, no. 1, pp. 59–70, 2007.
[46] G. Folego, O. Gomes, and A. Rocha, "From impressionism to expressionism: Automatically identifying van gogh's paintings," in *Proc. IEEE Int. Conf. Image Process.*, 2016, pp. 141–145.
[47] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets," in *Third IEEE Int. Conf. Auto. Face Gest. Recog.*, 1998, pp. 200–205.
[48] 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.
[49] 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.
[50] K.-C. Lee, J. Ho, and D. J. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 5, pp. 684–698, 2005.
[51] X. Ji, Y. Cui, H. Wang, L. Teng, L. Wang, and L. Wang, "Semisupervised hyperspectral image classification using spatial-spectral information and landscape features," *IEEE Access*, vol. 7, pp. 146 675–146 692, 2019.
[52] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[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] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Jul 2011.
[55] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Adv. Neural Inf. Process Syst.*, 2012, pp. 1097–1105.
[56] A. Kumar, J. Kim, D. Lyndon, M. Fulham, and D. Feng, "An ensemble of fine-tuned convolutional neural networks for medical image classification," *IEEE J. Biomed. Health Inform.*, vol. 21, no. 1, pp. 31–40, 2016.
[57] F. Chollet *et al.*, "Keras," https://keras.io, 2015.
[58] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE CVPR*, 2009, pp. 248–255.
[59] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proc. BMVC*, 2015, pp. 1–6.
[60] 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.
[61] 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.
[62] H. Zhu, L. Li, J. Wu, W. Dong, and S. Guangming, "Metaiqa: Deep meta-learning for no-reference image quality assessment," in *Proc. IEEE CVPR*, 2020, pp. 14 143–14 152.

Ying Bi (M'17) received the B.Sc.degree in 2013 from Wuhan Polytechnic University, Hubei, China, the M.Sc. degree in 2016 from Shenzhen University, Shenzhen, China, and the PhD degree in 2020 from Victoria University of Wellington, New Zealand.

She is currently a post-doctoral research fellow with the School of Engineering and Computer Science, Victoria University of Wellington. Her current research interests include evolutionary computation, computer vision, and machine learning. She has published over 30 papers in this field, including top journal and conference papers. She is a member of the IEEE Computational Intelligence Society and has been severing as reviewers for top international journals and conferences.



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 a 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 on 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 has 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.