Dual-Tree Genetic Programming for Few-Shot Image Classification

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

***Abstract*—Few-shot image classification is an important but challenging task due to high variations across images and a small number of training instances. A learning system often has poor generalisation performance due to the lack of sufficient training data. Genetic programming (GP) has been successfully applied to image classification and achieved promising performance. This paper proposes a GP-based approach with a dual-tree representation and a new fitness function to automatically learn image features for few-shot image classification. The dual-tree representation allows the proposed approach to have better search ability and learn richer features than a single-tree representation when the number of training instances is very small. The fitness function based on the classification accuracy and the distances of the training instances to the class centroids aims to improve the generalisation performance. The proposed approach can deal with different types of few-shot image classification tasks with various numbers of classes and different image sizes. The results show that the proposed approach achieves significantly better performance than a large number of state-of-the-art methods on nine 3-shot and 5-shot image classification datasets. Further analysis shows the effectiveness of the new components of the proposed approach, its good searchability, and the high interpretability of the evolved solutions.**

***Index Terms*—Genetic Programming; Representation; Fitness Evaluation; Few-Shot Learning; Image Classification**

I. Introduction

Recently, few-shot learning (FSL) has gained increasingly more attention and become a popular topic in artificial intelligence and machine learning. FSL aims to learn from a small number of training instances to solve a task [1]. When the number of training instances is one in each class, the task is known as one-shot learning. FSL can reduce human effort and expense/cost of collecting and/or labelling a large number of training instances. Furthermore, for some domains, such as medicine, biology and remote sensing, it is often difficult to obtain a large set of labelled training instances because of privacy, safety or other issues. FSL can provide solutions to the machine learning problems in these domains by only using a small number of training instances [1].

Manuscript received XXX; revised XXX; accepted XXX. This work was supported in part by the Marsden Fund of New Zealand Government under Contracts 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.

Few-shot image classification (FSIC) is a typical task in FSL that aims to classify images into a number of predefined groups according to their contents in the images using a very small number of training instances [1], [2]. FSIC can be applied to deal with tasks such as biological identification and medical diagnosis that typically have a small number of labelled images. Generally, images are sampled under different environments and have high variations. A few training instances with limited information cannot effectively represent the class distribution [2]. Thus, it is challenging to build effective models/classifiers for FSIC using a very small number of training instances.

Most current image classification methods have several limitations to achieve FSIC. The dominant image classification methods, i.e., deep convolutional neural networks (CNNs), often have millions of parameters, e.g., AlexNet has 138 million parameters [3]. These deep models typically require a large number of instances to train [4]. Therefore, they may not be effective for FSIC. More importantly, existing methods cannot be well generalised to the (unseen) test set when the number of training instances is small. In other words, the generalisation performance of these methods are often poor due to the lack of training data.

Existing methods for FSIC focus on two mainstreams to address the aforementioned difficulties. The first one aims to generate more images using different rules and add these images to the small training set as data augmentation. For example, new images can be generated by simple rules such as rotation, flipping, blurring, and cropping [3] or learned representations and embeddings, such as by Auto-Encoders [5] and generative adversarial networks (GANs) [6]. The second one focuses on transfer learning, i.e., transferring learned models and parameters from the domain with a large number of training instances to the FSIC domain [7], [8]. Based on this, many methods have been developed, including multitask learning [9], embedding learning [10], [11], and meta-learning [12]. However, these methods still have limitations. It is straightforward to generate more data to enlarge the training set so that the normal image classification methods can be applied to FSIC. However, the generated data only bring limited information to the training set [1]. It is also possible that the generated data harm the classification performance due to the distribution gap between the real data and the generated (estimated) data. For the second type of methods, it is necessary to find suitable and similar source domain datasets for effective knowledge transfer. Knowledge transfer could help when the number of instances is very small. However, in some real-world scenarios, it is not always possible to find source datasets with sufficient training instances that are similar to

the FSIC task. More importantly, transfer learning itself is difficult, requiring more investigations on what to transfer, how to transfer and when to transfer. Therefore, this study focuses on solving FSIC without using data augmentation and transfer learning, i.e., follows a traditional yet simple classification procedure that only uses the training set of the FSIC task.

Genetic programming (GP) is an evolutionary computation (EC) technique. It automatically evolves computer programs to solve a problem [13]. GP has been applied to many optimisation and learning tasks, including feature construction, classification, regression, and clustering [14], [15], [16], [17]. GP has also been used to automatically learn effective features for image classification and achieved promising results [18], [19]. Unlike CNNs, GP-based methods typically evolve trees/programs consisting of functions to extract features for image classification. A few GP-based methods have been developed to learn from a small number of training instances and showed their potential ability for FSIC [20], [21], [22], [23]. However, these works have only addressed texture classification and two-class image classification. The potential ability of GP for FSIC has not been extensively investigated on other types of image classification tasks.

The overall goal of this paper is to develop a GP-based approach for FSIC. It is known that poor generalisation performance is a common issue when the number of training instances is very small. To address this issue, a new fitness function is developed to allow the proposed approach to find effective soluitons for FSIC. In addition, a dual-tree representation is employed in the proposed approach to improve its learning performance. The proposed approach is termed as DTGPN, indicating Dual-Tree GP with a New fitness function.

The main contributions are summarised as follows.

- A DTGPN approach is developed to automatically evolve programs/trees with potentially high interpretability for FSIC. The proposed approach is able to solve many-class and few-class FSIC problems without using data augmentation and transfer learning, which is different from most of existing neural network (NN)-based FSIC methods [1]. This is the first time to apply GP to solve many different FSIC tasks. The proposed approach achieves significantly better performance than a large number of competitive benchmark methods on different FSIC tasks.
- A dual-tree representation is employed in DTGPN to improve its search ability and learning performance. The dual-tree representation allows DTGPM to explore a larger potential area and find a better solution that extracts richer features than a single-tree representation used in many existing methods [24], [25], [26]. DTGPN with this representation achieves better learning and generalisation performance than that with a single-tree representation when the training set is very small.
- A new fitness function is developed to improve the generalisation performance of DTGPN without adding too much computational cost. By optimising the fitness function, DTGPN is able to learn features and classifiers with potentially high generalisation ability. The results show that the fitness function can improve the generalisation performance of DTGPN.

II. Background and Related Work

A. Few-Shot Image Classification (FSIC)

FSIC is a representative task in FSL. FSIC aims to solve image classification tasks when the number of training instances is very small. An image classification task can be denoted as T with a dataset $\mathcal{D} = \{\mathcal{D}_{train},\ \mathcal{D}_{test}\}$, consisting of a training set and a test set. The training set is $\mathcal{D}_{train} = \{(\boldsymbol{x_0}, y_0), (\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_n}, y_n)\}$ and the test set is $\mathcal{D}_{test} = \{\boldsymbol{x_0}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_m}\}$. $\boldsymbol{x} \in \mathbb{R}^{M \times L}$ or $\mathbb{R}^{M \times L \times 3}$ denotes an image with a size of $M \times L$ (i.e. a gray-scale image) or $M \times L \times 3$ (i.e. a colour image with three channels). $y \in \mathbb{Z}$ denotes the class label. n indicates the number of training images and m indicates the number of testing images. A FSIC task is often described as a C-way-K-shot classification task, where C indicates the number of classes and K indicates the number of instances per class in the training set. Thus, the total number of training instances is $n = C \times K$.

B. Genetic Programming (GP)

GP can automatically evolve solutions in a form of programs/trees to solve a problem without requiring domain knowledge. In tree-based GP, a solution is represented by a tree, consisting of internal nodes and leaf nodes. The internal nodes are often constructed by functions or operators, such as arithmetic functions and logical functions. The leaf nodes are often constructed by terminals, i.e., variables/features and ephemeral random constants. Figure 1 shows two example GP trees. The left one is a classic GP tree that can be reformulated as $((x_1 + x_2)/x_3) \times (x_1 + 0.9)$. The right one is an example GP tree for image feature extraction. This tree can be reformulated as $Root(O2(O1(Image)), O3(Image, Image))$, where $Root$, $O1$, $O2$, and $O3$ are some functions or operators dealing with the input image ($Image$). The tree-based representation allows GP to automatically evolve variable-length/depth solutions consisting of different types of functions to solve a task.

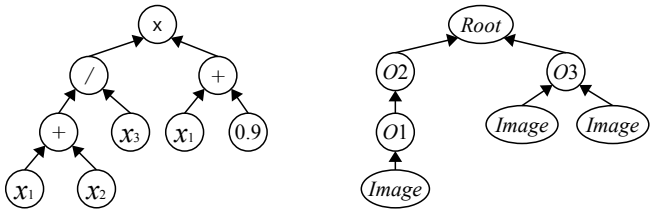


Fig. 1. A classic GP tree (left) and an example GP tree for image feature extraction (right).

C. Related Work

1) Image Classification: Traditional methods typically use manually extracted features to solve image classification [14]. In recent years, CNNs, which can automatically learn informative features from images, have been widely applied to image classification and achieved promising results [3], [27]. However, these methods have a number of limitations, e.g., requiring rich expertise to design CNN architectures, extensive computational resources and a large number of training

instances. Therefore, many methods have been employed to automatically construct CNNs for image classification. Chen et al. [28] developed genetic algorithms (GAs) with a variable-length encoding and new genetic operators to automatically evolve convolutional variational autoencoders for image classification. This method achieved better performance than nine variants of autoencoders on three well-known datasets. Lu et al. [29] proposed a multiobjective GA method to search for neural architectures with simultaneously optimising the objectives of the classification performance and the floating point operations. This method achieved superior performance than the other methods on four image datasets. More related work on image classification can be found in [30], [16].

2) GP for Image Classification: Bi et al. [26] proposed a GP method with image descriptors to automatically learn global and/or local features for image classification. This method achieved better performance than many baseline methods, including CNN-based methods and the methods using manually extracted features, on eight different image datasets. Shao et al. [25] proposed a multi-objective GP method to automatically learn features for image classification by simultaneously maximising classification performance and minimising the tree size. This method achieved better classification performance on four datasets than a number of traditional methods and NN-based methods. Bi et al. [31] developed an evolutionary deep learning method using GP with convolution operators to automatically learn image features for classification. This method described image features using convolution and pooling layers, and achieved better performance than simple CNNs and other traditional methods. These existing methods showed the effectiveness of GP in image classification [31], [25], [26]. However, these methods may have poor generalisation on FSIC since they uses a large number of training instances during the evolutionary process.

Several GP methods have been developed for image classification using a small number of training instances. Al-Sahaf et al. [23] proposed an one-shot GP method and a compound-GP method for image classification with one or a few training instances per class. The one-shot GP method uses two images per class to learn a similarity measure. The compound-GP method detects different regions of images, where local binary pattern features can be extracted for classification. However, these two methods have only been applied to two-class image classification. Al-Sahaf et al. [21] proposed a GP method to describe rotation-invariant features for texture image classification using a small number of training instances. This method uses a similar way as LBP to describe texture features based on the learned descriptors. This method achieved better classification performance than LBP and its variants. In [20], a GP method was developed to learn a variable number of features for texture classification. This method achieved better classification performance than several GP-based methods and other methods using handcrafted features. However, these two methods were developed for texture feature description, which may not be effective for classifying images that do not have rich texture features.

3) Few-Shot Image Classification: Many methods have been developed for image classification with a small or limited number of training instances. The most straightforward method is to generate more training images to enlarge the training set so that a normal image classification method can be applied to. Krizhevsky et al. [3] used translation, horizontal reflection and pixel value modification to generate more images for deep model training. The results showed that the generalisation performance of deep models could be improved by using a large training set. However, this method could only introduce limited information for model training when there are only a few unique training images. Antoniou et al. [6] proposed a data augmentation GAN (DAGAN) method to automatically learn images by considering wide variations. The images generated by DAGAN are added to the small training set and the classification performance of the classifier can be increased. Koch et al. [10] proposed convolutional Siamese NN (net) for FSIC. Convolutional Siamese net uses two CNNs with shared weights to learn a function that distinguishes whether two input images are similar or not. It is one of the most popular NNs for FSIC. It achieved better classification performance than other methods on one-shot image classification datasets. Meta-learning [12] has also been applied to FSIC. It uses a learning-to-learn scheme to learn a meta-learner using a collection of related FSIC tasks and transfers the learned model to the target FSIC task. The meta-learner is expected to improve the generalisation performance on the FSIC task.

To sum up, although many FSIC methods have been developed, there are different limitations. Most existing methods are based on NNs, which have a large number of trainable parameters, need rich expertise to design the CNN architectures and large computation resources [32]. The data-generation-based methods need efforts to design better rules or methods to learn rules to generate new images that can improve the generalisation performance. The meta-learning-based FSIC methods need to find a large number of similar FSIC tasks to train an effective meta-learner, which makes the whole process complex and the learned model difficult to understand and explain. Therefore, it is worth developing new methods for FSIC without using data augmentation and transfer learning, which could be more applicable and practical in many real-world scenarios.

The existing works show the potential of GP for FSIC, but they focus on very limited applications, i.e., texture classification and two-class image classification. This study will explore the potential of GP for FSIC without the use of data augmentation and transfer learning. It is known that poor generalisation performance is a common issue in the methods for FSIC. To address this, we will develop a GP approach with a new fitness function to achieve high generalisation performance in different FSIC tasks.

III. The Proposed Approach

This section describes the DTGPN approach for FSIC, including the dual-tree representation, the new fitness function, the fitness evaluation process, and the overall algorithm.

A. Dual-Tree Representation

GP with a multi-tree representation (i.e., an individual has more than one tree) can be found in the literature, such as using

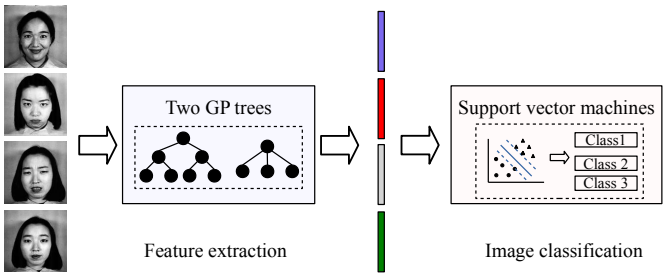


Fig. 2. The image classification process using two GP trees and support vector machines (SVMs).

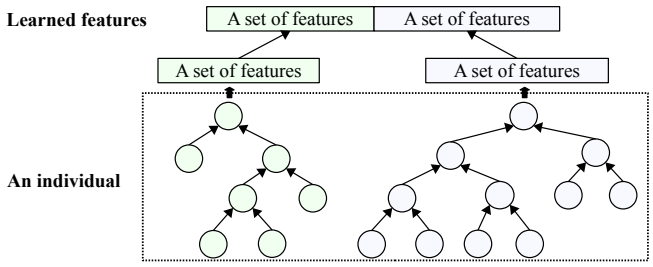


Fig. 3. The dual-tree representation of DTGPN. Each individual is represented by two trees. The features described by each tree are combined to form a feature vector for classification.

multi-tree GP to construct multiple features for classification in [33]. GP with a multi-tree representation often has better search ability and achieve better results than that with a single-tree representation. In this paper, GP with a dual-tree representation is proposed to learn features for FSIC. The dual-tree representation is shown in Fig. 3. In this representation, each individual is represented by two GP trees. Each tree can produce a variable number of features from an input image using different function nodes. The features produced by the two trees are concatenated to form a feature vector as the learned features for few-shot classification. We expect this to learn rich and informative features that cover more possible image variations and achieve good performance when the number of training instances is very small.

A recently proposed tree structure [26] is used in DTGPN to represent each tree in an individual. The tree structure allows GP to automatically learn various types of image features, i.e., a combination of different global and/or local features [26]. However, this tree structure increases the tree depth to have more functions that extract rich combinations of features, which is not effective for search. Increasing the tree width is an effective way to find more possible combinations of features. Therefore, we use a dual-tree representation in DTGPN to increase the search ability.

1) Tree Structure: The tree structure of DTGPN is based on strongly-typed GP (STGP) [34]. It consists of a region detection layer, a feature extraction layer and a feature concatenation layer. The region detection layer is to detect small but important regions from the large input image. The feature extraction layer uses five representative image descriptors to extract features from the detected regions or the input image. The feature concatenation layer is to concatenate the extracted features to form a feature vector. Three example trees are illustrated in Fig. 4 to show this structure. The region detection and feature extraction layers have a tree depth of one, while the feature concatenation layer has a flexible tree depth. To produce more features, it is necessary to increase the tree depth of the feature concatenation layer. However, the tree depth of such a higher-level layer is not easy to be increased during the search process based on the mutation and crossover operations due to the type and tree depth constraints in GP. To alleviate the search pressure, a dual-tree representation is employed to search for rich features by increasing the width of the tree.

2) Terminal Set: The terminal set is comprised of the $Image$, X, Y, S, W, and H terminals. The $Image$ terminal denotes the input image, which is an array normalised dividing by 255. The other terminals are ephemeral random constants of GP and represent the parameters of the region detection functions. Their details are listed as follows.

- $Image$: a $M \times L$ image with values in the range of $[0,\ 1]$
- X: the horizontal coordinate of the top-left point of the detected region in the image. It is an integer in the range of $[0,\ \ M-20]$
- Y: the vertical coordinate of the top-left point of the detected region in the image. It is an integer in the range of $[0,\ L-20]$
- S: the size of a detected square region. It is an integer in the range of $[20,\ 50]$
- W: the width of a detected rectangle region. It is an integer in the range of $[20,\ 50]$
- H: the height of a detected rectangle region. It is an integer in the range of $[20,\ 50]$

3) Function Set: The function set has two region detection functions (i.e., $Region\_S$ and $Region\_R$), five feature extraction functions in the global and local scenarios (i.e., G_DIF, G_Hist, G_SIFT, G_HOG, G_LBP, L_DIF, L_Hist, L_SIFT, L_HOG, and L_LBP) and two feature concatenation functions (i.e., $FeaCon2$ and $FeaCon3$). The **region detection** functions, $Region\_S$, and $Region\_R$, detect square and rectangle regions from the large input image, respectively. The region detected by $Region\_S$ is $Image[X : min(M,\ X+S),\ Y\ :\ min(L,\ Y+S)]$ and the region detected by $Region\_R$ is $Image[X : min(M,\ X+W),\ Y : min(L,\ Y+H)]$. Note that $Region\_S$ is a special case of $Region\_R$ and the use of $Region\_S$ allows DTGPN to easily detect square regions, since only using $Region\_R$ typically result in rectangle regions. The **feature extraction** functions are based on five representative image descriptors, i.e., DIF [35], Histogram (Hist), SIFT [36], HOG [37], and LBP [38]. These descriptors are developed into the global (G) and local (L) scenarios to extract global features from the whole image and local features from the detected region, respectively. Each of these functions takes an image or region as input and returns a set of features. The numbers of features produced by these functions are different. More details of them can be seen in [26]. The two **feature concatenation** functions can concatenate features extracted from different regions/images into a feature vector. These two functions can concatenate two or three feature vectors into a feature vector.

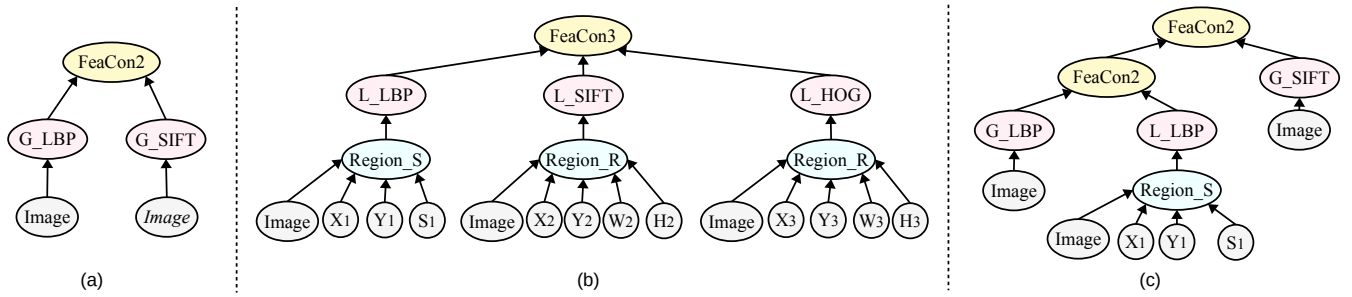


Fig. 4. Three example trees/programs to show the tree structure. These trees are able to extract (a) the combination of global features, (b) the combination of local features and (c) the combination of global and local features.

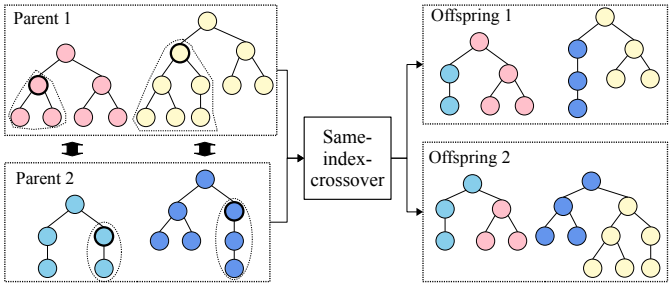


Fig. 5. The same-index-crossover operation.

Note that the region detection functions themselves are sensitive to the position changes. However, the learning process allows DTGPN to filter out the regions that are not informative and select the regions that contain informative features. This may allow the learned features from those regions are not so sensitive to translation. The representation enables DTGPN to learn features from the whole image and/or features from the detected region. Specifically, DTGPN can extract global features from the whole image and does not necessarily have the region detection functions in the trees, which means that the learned features can be translationally invariant.

4) Crossover and Mutation Operators: During the evolutionary process, the same-index-crossover operator and the mutation operator are used to generate new individuals for the next generation. The same-index-crossover operator pairs the trees with the same index of the two selected parents and performs standard single-tree crossover operation on each paired two trees to generate new offspring. An example is shown in Fig. 5 to demonstrate the crossover process. Specifically, it randomly selects crossover points from each two trees and swaps the branches of these two trees to generate two new trees. The mutation operator mutates every single tree in an individual to generate an offspring. For each tree, it randomly selects one subtree and replaces it with a new randomly generated subtree.

B. Fitness Evaluation Using The New Fitness Function

One of the key contributions of this paper is to design a new fitness evaluation method and a new fitness function for DTGPN to learn effective features with high generalisation performance for FSIC. When the number of training instances is very small, a potential issue is overfitting or underfitting, which leads to poor generalisation performance on the unseen (test) set. The GP-based feature learning algorithms often learn a set of features from images and evaluate the performance of these features on a training set. The commonly used fitness function is the classification accuracy, such as in [25], [26], [18]. However, when the number of training instances is very small, the current fitness function can not accurately evaluate the performance of the learned features because of the overfitting or underfitting issue. For example, it may easily reach 100% accuracy on the training set at early generations. To address this, we develop a new fitness function in DTGPN for FSIC. The fitness function is an integrated objective function of the classification accuracy and a distance measure, aiming to improve the generalisation performance without adding too much computational cost. The fitness evaluation process and the fitness function are introduced as follows.

1) Overall Fitness Evaluation Process: The fitness evaluation process is described in Algorithm 1 and the fitness function is defined in Eq. (1). In the fitness evaluation process, only the training set is employed. For a C-way-K-shot image classification task, the training set is $\mathcal{D}_{train} = \{(\boldsymbol{x_i}, y_i)\}_{i=0}^{C\times K}$ ($x \in \mathbb{R}^{M\times L}$, $y \in \mathbb{Z}$), having K training images in each of the C classes, where $M \times L$ denotes the size of the image and C denotes the number of classes. Each image in $\mathcal{D}_{train}$ is transformed into features by two trees of a GP individual. The features extracted by two trees are concatenated to form a feature vector to describe an image (as shown in Fig. 3). The transformed training set is $\mathcal{D}^{tr}_{train} = \{(\boldsymbol{x_i}, y_i)\}_{i=0}^{C\times K}$ ($\boldsymbol{x} \in \mathbb{R}^n$, $y \in \mathbb{Z}$), where n denotes the number of features extracted by the two trees of a GP individual. The min-max normalisation is performed to rescale different value ranges of the extracted features into $[0,\ 1]$. The normalised training set is denoted as $\mathcal{D}^{nrom}_{train} = \{(\boldsymbol{x_i}, y_i)\}_{i=0}^{C\times K}$. Based on $\mathcal{D}^{nrom}_{train}$, SVM classifiers (which is the most commonly used one in GP) are trained and the fitness value is calculated. The SVM classification algorithm with a linear kernel is employed for classification because it often achieves high generalisation performance and has fewer parameters than it with other kernels [25], [26].

The main process to calculate the fitness value follows the step of classification with stratified K-fold cross-validation. Unlike the traditional one, which often uses K=5 or 10, we set the value of K to the number of instances in each class in the training set. For example, K equals 3 for a 3-shot image classification problem. The normalised training set is

Algorithm 1: Fitness Evaluation

Input : $\mathcal{D}_{train}$: the training set of a C-way-K-shot image classification task; p: the individual with two trees to be evaluated.

Output : The fitness value for p: $f(p)$.

for *each image* i *in* $\mathcal{D}_{train}$ **do**
  $\boldsymbol{f_i^{left}}$, $\boldsymbol{f_i^{right}} \leftarrow$ features transfomred by the left and right trees of individual p from image i, respectively;
  $\boldsymbol{f_i} \leftarrow \{\boldsymbol{f_i^{left}}, \boldsymbol{f_i^{right}}\}$;
end
$\mathcal{D}_{train}^{tr} \leftarrow$ the training set represented by the learned features;
$\mathcal{D}_{train}^{norm} \leftarrow$ the training set after min-max normalisation;
Split $\mathcal{D}_{train}^{norm}$ into K folds with each has one instance per class;
for $k = 1; k \leq K; k++$ **do**
  Use the k^{th} fold as the evaluation test set and the remaining $K-1$ folds as the evaluation training set;
  Train SVM classifiers using the evaluation training set and test the classifiers on the evaluation test set;
  $Acc_k \leftarrow$ the accuracy of all the instances in the k^{th} fold;
  Use the trained SVM classifiers to transform $\mathcal{D}_{train}^{norm}$ into the decision space;
  $\boldsymbol{u_c}$ $(1 \leq c \leq C) \leftarrow$ the centroid of class c of the evaluation training set in the deceion space;
  $Dist_k = 0$;
  for *each instance* i *in the evaluation test set* **do**
    $S_i \leftarrow$ the distance between instance i to the centroid of the same class of the evaluation training set;
    $D_i \leftarrow$ the sum of all the distances between instance i to the centroids of the different classes of the evaluation training set;
    $Dist_k = Dist_k + S_i/D_i$;
  end
  $Acc_k - Dist_k \leftarrow$ calculate the fitness value of the k^{th} fold;
end
$f(p) \leftarrow$Average the fitness value by dividing K;
Return $f(p)$.

split into non-overlapping K folds. Specifically, each fold has C instances in C classes, i.e., each class has one instance. Each time one fold with C instances is employed as the evaluation test set and the remaining $K-1$ folds with $C(K-1)$ instances are used as the evaluation training set. The evaluation training set is used to train SVM classifiers. Based on the trained classifier(s), the classification accuracy (Acc_k defined in Eq. 2) and the distance ($Dist_k$ defined in Eq. 3) of the evaluation test set are calculated. Repeating this process K times, K Acc_k and $Dist_k$ values are obtained. The fitness value is then calculated using Eq. (1).

2) The New Fitness Function: The fitness function is an integrated function of the classification accuracy and a new distance measure. The classification accuracy aims to evaluate the classification performance of the learned features. The distance measure aims to increase the generalisation performance of the learned features and the SVM classifiers. The fitness function to be maximised is defined as

$$Fitness = \frac{1}{K}\sum_{k=1}^{K}(Acc_k - Dist_k), \tag{1}$$

where K denotes the number of folds, which equals to the number of instances in the training set. Acc_k denotes the classification accuracy of the k^{th} fold of the evaluation test set and $Dist_k$ denotes the distance measure, which is based on the distances between the instances in the k^{th} fold of the evaluation test set and the class centroids of the evaluation training set. In Eq. (1), the Acc_k is based on (2).

$$Acc = \frac{N_{correct}}{N_{total}}, \tag{2}$$

where $N_{correct}$ denotes the number of correctly classified instances and N_{total} denotes the total number of instances in the evaluation test set. Note that k is omitted for better presentation.

3) The New Distance Measure: The $Dist_k$ is defined in Eq. (3), which is a new distance measure based on the evaluation training set and the evaluation test set. $Dist_k$ measures the distances between the instances in the evaluation test set to the class centroids of the evaluation training set, which is related to not only the performance of the learned features but also the performance of the classifiers. With optimising both parts, the generalisation performance is expected to be improved. This will be discussed in the later paragraphs.

$$Dist = \sum_{c=1}^{C}\frac{\sum_{m=1}^{M}(z_{c,m} - u_{c,m})^2}{\sum_{j\neq c, j=1}^{C}\sum_{m=1}^{M}(z_{j,m} - u_{j,m})^2}, \tag{3}$$

where C denotes the number of classes (and the number of instances in the evaluation test set). $\boldsymbol{z} = \{z_1, \ldots, z_M\}$ is a single value or a vector, denoting an instance in the decision space based on SVM classifiers, which is the sum of the weighted features and intercept. M denotes the number of features in $\boldsymbol{z}$. $\boldsymbol{u_c} = \{u_{c,1}, \ldots, u_{c,M}\}$ denotes the class centroid of the instances in class c.

Unlike the existing distance measure employed in many GP methods, such as [20, 20, 23], which calculate the distance based on the learned feature space, the distance measure $Dist$ is based on the decision space. The decision space is defined as the space that the features transformed by the classifiers. Optimising the distance based on the learned feature space does not necessarily improve the performance because the classification process has feature weighting. Note that this distance measure is suitable for SVM or other classifiers that have feature weighting or transformation. We discussed SVM because it is the most commonly used classifier and is treated as a part of the proposed approach to image classification.

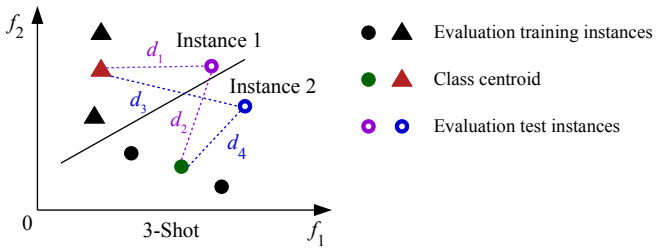


Fig. 6. An example to illustrate the distance between the evaluation test instances and each class centroid in a 2-way-3-shot classification problem. The $Dist$ value of instance 1 is d_1/d_2 and the $Dist$ value of instance 2 to the class centroid is d_4/d_3. The total distance is $d_1/d_2 + d_4/d_3$.

We will give a detailed explanation of how an instance is represented in a decision space based on the SVM classifiers. For a binary classification task, SVM builds a binary classifier expressed by $\{w_0, w_1, \ldots, w_{n-1}, b\}$. For a C-class classification task, SVM builds C binary classifiers expressed by $\{\{w_{1,0}, \ldots, w_{1,n-1}, b_1\}, \ldots, \{w_{C,0}, \ldots, w_{C,n-1}, b_C\}\}$ using the one-versus-rest (OvR) strategy [39]. An instance $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$ is transformed into $\boldsymbol{z}$ using these classifiers. Eq. 4 shows the calculation of $\boldsymbol{z}$ when there is only one classifier, i.e., the task is a *binary* classification task. Eq. 5 shows the calculation of $\boldsymbol{z}$ for a *multi-class* classification task.

$$\boldsymbol{z} = \sum_{i=1}^{n} w_i * x_i + b, \tag{4}$$

$$\boldsymbol{z} = \Big\{ \sum_{i=1}^{n} w_{1,i} * x_i + b_1, \ldots, \sum_{i}^{n} w_{C,i} * x_i + b_C \Big\}, \tag{5}$$

where n denotes the number of features. In the DTGPN approach, the value of n is dynamically changed.

After transforming all the instances into the decision space, the class centroid ($\boldsymbol{u_c}$) of the evaluation training set is calculated according to Eq. (6).

$$u_c = \frac{1}{N_c} \sum_{y_i=c, i=0}^{N_c} z_i. \tag{6}$$

where N_c denotes the number of instances in class c.

The distance measure (in Eq. 3) is a newly designed measure, which aims to minimise the distances between the instances and the centroid of the same class and maximise the distances between the instances and the centroids of the other classes. A simple example is shown in Fig. 6 to further explain how the distance is calculated in the decision space.

By minimising the distance defined in Eq. 3, the instances in the same class will be more clustered and the distances of the instances in different classes will be larger. Since these distances are calculated from all the training instances using a K-fold cross validation manner, a good distribution of all the training instances in the decision space can be obtained if the distances are optimised, which can be seen from Fig. 7. With a clear boundary and more clustered instances in each class, the built classifiers could have higher generalisation performance to correctly classify more unseen instances.

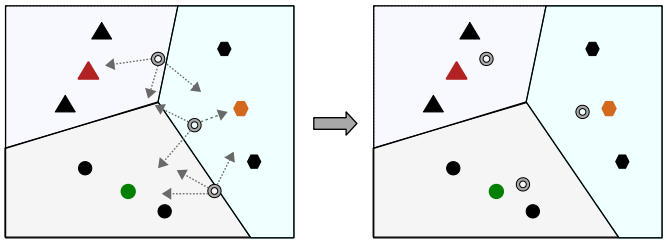


Fig. 7. The original distribution (left) and a good distribution (right) of all the instances in the decision space by maximising the distance measure for a three-class classification problem.

C. DTGPN for FSIC

The overall algorithm of DTGPN for image classification is described in Algorithm 2. DTGPN searches for the best individual of two trees (i.e., $Best\_Ind$) via the evolutionary process using a small number of training instances. After the evolutionary process, the best individual is tested on the test set. In the test process, the training set and the test set are transformed by the two trees of $Best\_Ind$. The transformed training and test sets are normalised. The small training set is used to train linear SVM to obtain classifier(s) and the trained classifier(s) are used to classify the instances in the test set. The classification accuracy of the test set is calculated.

Algorithm 2: DTGPN for FSIC

Input : $\mathcal{D}_{train}$: a small number of training instances; $\mathcal{D}_{test}$: the test set.
Output : The classification accuracy of the test set

```
// Evolutionary learning procedure
```
$P_0 \leftarrow$ Initialise the population, where each individual has two trees;
$g \leftarrow 0$;
while $g < G$ **do**
  Evaluate each individual in P_g using Algorithm 1;
  Update $Best\_Ind$;
  $I_g \leftarrow$ the best individuals selected from P_g using elitism;
  $S_g \leftarrow$ individuals selected from P_g using tournament selection;
  $O_g \leftarrow$ offspring generated from S_g using the same-index-crossover and mutation operators;
  $P_{g+1} \leftarrow O_g \cup I_g$;
  $g \leftarrow g + 1$;
end
$Best\_Ind$ is obtained;
```
// Classification procedure
```
$\mathcal{D}^{tr}_{train}$, $\mathcal{D}^{tr}_{test} \leftarrow$ transformed $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$ with a set of features described by $Best\_Ind$;
$\mathcal{D}^{norm}_{train} \leftarrow$ normalised $\mathcal{D}^{tr}_{train}$ using the min-max normalisation method;
$\mathcal{D}^{norm}_{test} \leftarrow$ normalised $\mathcal{D}^{tr}_{test}$ based on $\mathcal{D}^{tr}_{train}$;
Train SVM classifiers using $\mathcal{D}^{norm}_{train}$;
Use the trained SVM classifiers to classify $\mathcal{D}^{norm}_{test}$ and obtain the classification accuracy.

TABLE I
SUMMARY OF THE DATASETS

Data Sets	#Class	Image Size	Train Set (3-shot)	Test Set (3-shot)	Train Set (5-shot)	Test Set (5-shot)
JAFFE	7	128×128	21	192	35	178
FEI	2	130×180	6	194	10	190
ORL	40	112×92	120	280	200	200
Faces	72	90×100	216	1,224	360	1,080
YaleB	38	90×100	114	2,310	190	2,234
COIL	20	100×100	60	1,380	100	1,340
DSLR	31	100×100	93	405	155	343
KTH	10	100×100	30	780	50	760
Outex	24	128×128	72	4,248	120	4,200

IV. Experiment Design

This section designs the experiments, including the benchmark datasets, baseline methods and parameter settings.

A. Benchmark Datasets

To show the effectiveness of the proposed DTGPN approach on various image classification tasks, nine different image datasets of varying difficulties are employed as benchmark datasets. These datasets are JAFFE [40], FEI [41], ORL [42], Faces [43], YaleB [44], COIL [45], DSLR [46], KTH [47], and Outex [48]. These datasets represent a wide variety of image classification tasks, i.e., facial expression classification (JAFFE and FEI), face recognition (ORL, Faces, and YaleB), object classification (COIL and DSLR), and texture classification (KTH and Outex). These tasks with various types of images, different numbers of classes and various image sizes can comprehensively demonstrate the effectiveness of the proposed DTGPN approach to FSIC.

In the experiments, few-class (2 and 7 classes) and many-class (including 72, 40, 38, and 24 classes) 3-shot and 5-shot image classification tasks are investigated. The training set has three or five images per class. In total, there are 18 benchmark datasets, i.e., nine datasets on two different scenarios. In these datasets, 3 or 5 images are randomly selected from each class to form the training sets and the remaining images form the test sets. The detailed information of these datasets is summarised in Table I. It shows that these datasets have different numbers of classes, image sizes, and numbers of instances for testing. Example images of these datasets are shown in Fig. 8.

Since this paper aims to solve FSIC without the use of data augmentation or transfer learning, only the training set of the dataset is employed for model training and feature learning. Furthermore, this paper is to investigate FSIC with different types of images and different numbers of classes. Based on such considerations, the current nine datasets with different numbers of classes, including many-class FSIC tasks [49], are employed in the experiments.

B. Baseline Methods

A large number of baseline methods are used for comparisons. These baseline methods consist of three main groups.

The first group has one baseline GP method and 12 non-GP-based methods. The baseline GP method is the FLGP method in [26]. The 12 non-GP-based methods are SVM, random forest (RF) [32], k-nearest neighbour (KNN) [20], sparse representation-based classification (SRC) [50], linear discriminant analysis (LDA) [51], SIFT [36], DIF [35], Histogram, HOG [37], LBP [38], LeNet [52] and a five-layer CNN (CNN-5) [31]. The SVM, RF, KNN, SRC, and LDA methods use the raw pixel values for classification. The SIFT, DIF, Histogram, HOG, and LBP methods use the corresponding features as inputs of SVM to perform classification. The LeNet and CNN-5 methods have a smaller number of parameters than very deep CNNs since only a small number of training instances are used. The aim of the experiments is to investigate whether DTGPN can achieve better performance than these commonly used image classification methods.

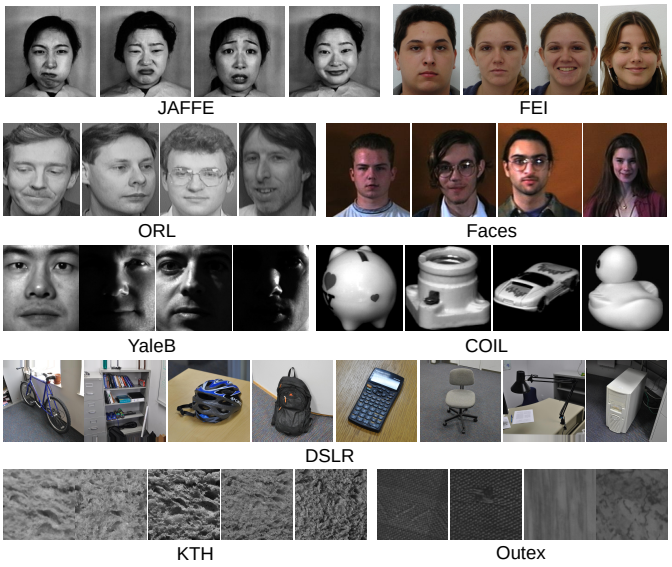


Fig. 8. Example images from the nine image classification datasets.

The second group has the same 12 non-GP-based methods in the first group with using data augmentation. The images in the training set are rotated, flipped, or blurred to generate new images [3]. The size of the training set is enlarged twice for feature learning and model training. The aim of the experiments is to investigate whether DTGPN without data augmentation can achieve better performance than these methods with data augmentation, which is a simple way to address FSIC.

The third group includes five well-known FSIC methods for comparisons. They are Convolutional Siamese Networks [10], Convolutional Siamese Networks with triple loss (Triple Networks for short) [53], Prototypical Networks [11], Matching Networks [54], and Model-Agnostic Meta-Learning (MAML) [55]. The implementations of these five methods are based on Pytorch [1]. We have conducted parameter tuning for Convolutional Siamese Networks and Triple Networks regarding the number of epochs (i.e., 20, 50, 100, 200), the output dimension (i.e., 1, 32, 64, 256, 4096), and weight initialisation

[1]The codes for Siamese Networks and Triple Networks are downloaded from https://github.com/adambielski/siamese-triplet. The codes for the other three methods are downloaded from https://github.com/oscarknagg/few-shot

(with and without the one recommended in [10]). The detailed analysis is presented in the supplementary materials due to the page limit. The best parameters (i.e., 50 epochs, 64-dimension outputs and the weight initialisation method in [10]) are used in the experiments. For the other parameters, we kept the same as the repositories. In the other three networks, we set the number of evaluation episodes to 10, the number of epochs to 40, and the number of episodes each epoch to 100 after empirical search. It is noted that we did not do parameter tuning for the proposed approach, so the comparisons are expected to be fair or at least not to be biased to the proposed approach (making entirely fair comparisons is not easy since these methods are based on NNs, while the proposed approach is an EC method). In these NN methods, we use the training set for learning (including meta-learning), and the test set for calculating classification accuracy. We also used an external large dataset to train the model and use the training set to fine-tune the model to obtain the performance of the test set. However, due to the page limit and the high computational cost, we only compared these methods with the proposed DTGPN approach on several datasets in the supplementary materials. The aim of the experiments is to investigate whether the proposed DTGPN approach can achieves better results than well-known FSIC methods.

C. Parameter Settings

The parameter settings for the four GP-based methods follow the commonly used settings in the GP community [26] [56]. The population size is 100 and the maximal number of generations is 50. The crossover, mutation and elitism rates are 0.8, 0.19 and 0.01, respectively. Tournament selection with size five is employed to select individuals for crossover and mutation. The tree depth is between 2-8. It is noted that the depth of GP trees may be above eight because the type constraints are more important than the tree depth constraint when building GP trees. The *ramped-half-and-half* method is used for generating the initial population.

The parameters settings for the 12 non-GP-based baseline methods are based on the commonly used settings. The number of neighbours is 1 in KNN [20]. SVM uses a commonly used linear kernel because it has few parameters [25]. The number of trees is 500 and the maximal tree depth is 100 in RF [32]. In LeNet and CNN-5, the batch size is set to 20 and the number of epochs is set to 100 [32].

The *DEAP* (Distributed Evolutionary Algorithm in Python) [57] package is used to implement all the GP-based methods. The *scikit-learn* [58] package is used to implement the classification algorithms and the *Keras* [59] package is used to implement the CNNs. The experiments of each algorithm have been executed 30 independent times and the results of the 30 runs are reported. All the algorithms use the training sets to learn features or train classifiers, and the test sets have never been used in these processes.

V. Results and Discussions

This section discusses and analyses the results obtained by DTGPN and the baseline methods on the nine datasets under the 3-shot and 5-shot image classification scenarios.

A. Comparisons with Baseline Methods

The mean test accuracy (%) and the standard deviation values of the 30 runs of the DTGPN approach and the baseline methods are listed in Table II. To show the significant improvement of the performance, Wilcoxon rank-sum test with a 5% significance level is employed to compare the DTGPN approach with a baseline method. In Table II, the "+", "−" and "=" symbols denote that DTGPN achieves significantly better, worse or similar performance than/to the compared method. The overall significance results are summarised at the final row of each block of Table II.

Compared with the five methods using raw pixels, i.e., SVM, RF, KNN, SRC, and LDA, it can be found that DTGPN achieves significantly better results in 88 comparisons out of the total 90 comparisons. Importantly, DTGPN achieves significantly better results than these methods on all the datasets of 5-shot image classification. From Table II, we can find that these five methods using raw pixels achieve very low accuracy on most datasets, particularly the FEI, Faces, YaleB, COIL, DSLR, KTH, and Outex datasets. Compared with these methods, the proposed DTGPN approach significantly improves the classification performance by automatically learning effective features for image classification.

Compared with the methods using pre-extracted features, i.e., SIFT, DIF, Histogram, HOG, and LBP, the proposed DTGPN approach achieves significantly better results in all the comparisons under the 3-shot and 5-shot scenarios. The DTGPN approach significantly improves the classification performance on the JAFFE, FEI, YaleB, COIL, DSLR, KTH, and Outex datasets. It can be found that the performance of the methods using these pre-extracted features vary with the datasets. Compared with these methods, DTGPN can learn features that are more effective than these pre-extracted features from a very small number of training instances to achieve better classification performance.

Compared with the two CNN methods, i.e., LeNet and CNN-5, the proposed DTGPN approach achieves significantly better or similar results in all the comparisons. Given a dataset having images with a size of 100×100, LeNet has $3,316,696 + (85 \times C)$ trainable parameters and CNN-5 has $1,992,160 + (129 \times C)$ trainable parameters, where C denotes the number of classes. Thus, LeNet and CNN-5 cannot achieve good performance on these FSIC datasets because of the lack of sufficient training images to effectively train the models with such a large number of parameters. Compared with these two methods, the DTGPN approach only needs to train SVM classifiers with $f + 1$ (when $C = 2$) or $(f + 1) \times C$ trainable parameters (when $C > 2$), where f denotes the number of learned features. Typically, f is significantly smaller than 5,000 (the number of features learned by DTGPN will be analysed in Section VI-C). Compared with these two CNN methods, the number of the trainable parameters of the DTGPN-based image classification approach is significantly smaller. Therefore, DTGPN can achieve significantly better performance than these two CNN methods when the number of training instances is very small.

Compared with FLGP, the proposed DTGPN approach

TABLE II
TEST ACCURACY (%) OF DTGPN AND THE BASELINE METHODS ON THE 3-SHOT AND 5-SHOT IMAGE CLASSIFICATION TASKS. THE "+", "−" AND "=" SYMBOLS DENOTES THAT DTGPN ACHIEVES SIGNIFICANTLY BETTER, WORSE OR SIMILAR PERFORMANCE THAN/TO THE COMPARED METHOD

Method	**Mean±St.dev**	**Mean±St.dev**	**Mean±St.dev**	**Mean±St.dev**	**Mean±St.dev**	**Mean±St.dev**
	JAFFE (3-shot)	**JAFFE (5-shot)**	**FEI (3-shot)**	**FEI (5-shot)**	**ORL (3-shot)**	**ORL (5-shot)**
SVM	31.09±0.28 −	42.21±0.25 +	51.03±0.00 +	52.63±0.00 +	91.07±0.00 +	94.50±0.00 +
RF	21.60±1.36 +	36.84±1.25 +	53.38±1.23 +	54.72±1.49 +	89.29±0.78 +	93.62±1.16 +
KNN	14.58±0.00 +	18.54±0.00 +	48.97±0.00 +	47.89±0.00 +	66.43±0.00 +	83.50±0.00 +
SRC	**32.29±0.00** −	37.64±0.00 +	52.58±0.00 +	50.00±0.00 +	87.86±0.00 +	91.00±0.00 +
LDA	14.58±0.00 +	29.21±0.00 +	52.06±0.00 +	48.95±0.00 +	89.29±0.00 +	93.50±0.00 +
SIFT	20.31±0.00 +	28.09±0.00 +	52.58±0.00 +	52.11±0.00 +	93.21±0.00 +	93.50±0.00 +
DIF	18.75±0.00 +	15.73±0.00 +	52.04±0.09 +	50.53±0.00 +	63.21±0.00 +	69.13±0.22 +
Histogram	17.00±0.77 +	19.36±1.73 +	51.55±0.00 +	47.37±0.00 +	80.92±0.73 +	83.20±0.99 +
HOG	16.20±0.16 +	31.31±0.39 +	56.41±0.71 +	46.09±0.36 +	61.29±0.26 +	66.98±0.09 +
LBP	16.39±1.54 +	21.91±3.60 +	46.17±0.26 +	55.26±0.00 +	83.15±1.56 +	83.55±1.11 +
LeNet	26.27±3.54 =	41.48±1.48 +	52.65±1.81 +	57.00±3.44 +	75.42±3.30 +	82.68±3.44 +
CNN-5	25.71±2.07 =	39.83±1.35 +	50.52±1.26 +	51.46±1.39 +	86.93±1.78 +	92.77±1.54 +
FLGP	23.35±7.18 =	44.49±9.67 +	69.36±12.29 +	**81.53±9.28** =	96.02±0.60 +	98.37±0.59 +
DTGPN	25.16±6.96	**53.52±3.49**	**84.85±6.97**	81.16±4.37	**96.76±0.32**	**99.73±0.29**
Overall	**8+, 3=, 2–**	**13+**	**13+**	**12+, 1=**	**13+**	**13+**
	Faces (3-shot)	**Faces (5-shot)**	**YaleB (3-shot)**	**YaleB (5-shot)**	**COIL (3-shot)**	**COIL (5-shot)**
SVM	64.18±0.05 +	73.09±0.06 +	44.11±0.08 +	82.41±0.03 +	78.26±0.01 +	80.97±0.00 +
RF	74.58±0.48 +	84.63±0.43 +	42.53±0.41 +	84.10±0.37 +	84.44±0.52 +	91.06±0.38 +
KNN	40.85±0.00 +	52.87±0.00 +	8.83±0.00 +	29.45±0.00 +	59.57±0.00 +	63.36±0.00 +
SRC	59.64±0.00 +	64.63±0.00 +	63.07±0.00 +	92.03±0.00 +	73.48±0.00 +	77.61±0.00 +
LDA	75.25±0.00 +	80.46±0.00 +	46.10±0.00 +	82.68±0.00 +	75.29±0.00 +	80.67±0.00 +
SIFT	82.52±0.00 +	89.72±0.00 +	27.10±0.00 +	59.83±0.01 +	79.42±0.00 +	84.25±0.00 +
DIF	42.40±0.00 +	51.20±0.00 +	8.40±0.00 +	14.33±0.00 +	67.17±0.00 +	73.43±0.00 +
Histogram	47.95±4.55 +	53.07±7.92 +	4.12±0.32 +	5.84±0.59 +	60.72±1.74 +	67.60±2.30 +
HOG	20.15±0.14 +	21.53±0.07 +	6.43±0.04 +	8.54±0.04 +	60.98±0.08 +	57.65±0.05 +
LBP	81.19±0.79 +	91.97±0.52 +	22.82±0.52 +	39.78±1.68 +	73.38±1.76 +	81.50±1.10 +
LeNet	60.36±3.35 +	73.27±2.07 +	32.10±2.78 +	69.67±3.92 +	78.87±3.05 +	85.77±2.98 +
CNN-5	64.31±2.09 +	76.16±1.73 +	41.81±1.19 +	77.51±2.19 +	79.74±1.73 +	85.58±2.08 +
FLGP	92.65±1.11 +	96.57±0.77 +	69.92±3.06 +	95.45±2.90 +	84.34±1.28 +	92.98±1.68 +
DTGPN	**93.77±1.03**	**96.97±0.89**	**75.94±2.97**	**97.84±0.92**	**87.97±1.43**	**95.75±1.16**
Overall	**13+**	**13+**	**13+**	**13+**	**13+**	**13+**
	DSLR (3-shot)	**DSLR (5-shot)**	**KTH (3-shot)**	**KTH (5-shot)**	**Outex (3-shot)**	**Outex (5-shot)**
SVM	22.52±0.39 +	28.34±0.29 +	15.24±0.82 +	19.55±0.44 +	5.41±0.12 +	7.25±0.09 +
RF	34.84±1.14 +	41.83±0.99 +	32.82±1.58 +	38.27±1.54 +	31.79±0.38 +	35.30±0.39 +
KNN	13.83±0.00 +	18.66±0.00 +	21.41±0.00 +	18.55±0.00 +	27.45±0.00 +	26.10±0.00 +
SRC	18.27±0.00 +	24.78±0.00 +	19.62±0.00 +	22.50±0.00 +	7.72±0.00 +	7.98±0.00 +
LDA	21.23±0.00 +	25.66±0.00 +	30.26±0.00 +	32.37±0.00 +	32.72±0.00 +	34.45±0.00 +
SIFT	38.52±0.00 +	46.65±0.00 +	39.36±0.00 +	45.53±0.00 +	22.89±0.01 +	22.80±0.01 +
DIF	27.65±0.00 +	28.28±0.00 +	39.10±0.02 +	40.26±0.00 +	23.53±0.02 +	23.81±0.01 +
Histogram	12.22±1.36 +	15.68±1.19 +	39.19±0.73 +	34.57±1.45 +	42.54±7.57 +	47.96±8.13 +
HOG	17.04±0.00 +	17.79±0.05 +	20.13±0.00 +	20.47±0.09 +	8.52±0.00 +	8.54±0.01 +
LBP	25.88±0.88 +	34.10±1.01 +	52.72±2.25 +	63.05±3.45 +	64.19±1.31 +	74.03±0.89 +
LeNet	24.26±3.51 +	32.59±3.29 +	32.95±4.89 +	36.82±5.90 +	44.68±7.58 +	51.74±8.00 +
CNN-5	32.78±2.73 +	38.36±3.42 +	37.73±4.12 +	44.28±3.36 +	35.74±3.97 +	44.21±5.51 +
FLGP	50.07±3.48 =	55.77±2.39 =	58.85±3.70 +	**69.81±2.14** −	73.50±2.24 =	82.29±0.96 =
DTGPN	**50.94±3.24**	**57.26±2.74**	**61.57±3.86**	68.03±1.47	**74.10±1.04**	**82.12±0.78**
Overall	**12+, 1=**	**12+, 1=**	**13+**	**12+, 1–**	**12+, 1=**	**12+, 1=**

achieves significantly better or similar performance in 17 comparisons out of the total 18 comparisons. It can be found that DTGPN achieves better or similar performance than FLGP on all the 3-shot image classification tasks. DTGPN significantly improves the classification performance on the JAFFE (5-shot), FEI (3-shot), YaleB (3-shot), COIL (3-shot), and COIL (5-shot) datasets. Compared with FLGP, the DTGPN approach uses a dual-tree representation and a new fitness function to learn effective features for FSIC. From the results, it is clear that the dual-tree representation and the fitness function significantly improve the classification performance.

B. Comparisons with the Methods with Data Augmentation

The classification results of the baseline methods with data augmentation and the DTGPN approach are listed in Table I in the supplementary materials due to the page limit. The summary of the comparisons in terms of the significance test is listed in Table IV. Compared with these baseline methods with the use of data augmentation, the proposed DTGPN approach achieves significantly better performance in 210 comparisons out of the total 216 comparisons. Only on JAFFE (3-shot), DTGPN achieves better performance than KNN, SIFT, DIF, Histogram, HOG, and LBP, similar performance to RF, and worse performance than SVM, SRC, LDA, LeNet, and CNN-5. On the remaining 17 datasets, DTGPN achieves significantly better performance than any of the baseline methods. The results show that DTGPN is better than the baseline methods with data augmentation on these different FSIC datasets.

The results in Table I of the supplementary materials show that the use of data augmentation can help improve the

TABLE III
TEST ACCURACY (%) OF POPULAR FSIC METHODS AND DTGPN ON THESE 18 DATASETS. THE "+", "−" AND "=" SYMBOLS INDICATE DTGPN ACHIEVES SIGNIFICANTLY BETTER, WORSE OR SIMILAR PERFORMANCE THAN/TO THE COMPARED METHOD

	Siamese Networks	**Triplet Networks**	**Prototypical Networks**	**Matching Networks**	**MAML**	**DTGPN**
Dataset	**Mean±St.dev**	**Mean±St.dev**	**Mean±St.dev**	**Mean±St.dev**	**Mean±St.dev**	**Mean±St.dev**
JAFFE (3-shot)	18.11±2.77 +	16.60±3.17 +	66.09±1.46 –	66.08±1.22 –	42.70±9.96 –	25.16±6.96
JAFFE (5-shot)	31.50±3.17 +	22.15±4.26 +	65.76±1.13 –	66.29±1.41 –	29.73±7.73 +	53.52±3.49
FEI (3-shot)	55.24±4.15 +	58.35±8.61 +	74.26±2.42 +	74.82±2.66 +	64.84±11.85 +	84.85±6.97
FEI (5-shot)	50.56±4.68 +	54.70±8.09 +	74.48±1.93 +	74.60±3.02 +	67.83±10.93 +	81.16±4.37
ORL (3-shot)	70.10±4.43 +	56.49±5.09 +	63.74±0.48 +	62.54±1.45 +	62.33±0.83 +	96.76±0.32
ORL (5-shot)	74.30±5.85 +	64.97±5.70 +	63.72±0.53 +	62.46±1.08 +	62.56±0.59 +	99.73±0.29
Faces (3-shot)	61.15±2.38 +	52.36±4.60 +	63.28±0.40 +	28.03±4.22 +	61.20±1.17 +	93.77±1.03
Faces (5-shot)	67.81±2.56 +	60.90±4.98 +	63.51±0.50 +	25.61±6.48 +	61.93±0.59 +	96.97±0.89
YaleB (3-shot)	16.28±1.53 +	6.52±0.80 +	63.62±0.51 +	51.65±3.20 +	17.61±12.33 +	75.94±2.97
YaleB (5-shot)	43.19±2.38 +	11.59±2.07 +	63.64±0.47 +	52.14±3.87 +	32.99±17.40+	97.84±0.92
COIL (3-shot)	72.23±2.19 +	63.13±4.72 +	63.94±0.75 +	64.18±0.66 +	59.69±0.96 +	87.97±1.43
COIL (5-shot)	80.88±1.69 +	69.61±4.14 +	64.03±0.74 +	64.12±0.82 +	60.18±0.75 +	95.75±1.16
DSLR (3-shot)	22.28±2.30 +	19.61±2.55 +	63.78±0.54 –	30.13±5.56 +	56.15±4.29 –	50.94±3.24
DSLR (5-shot)	27.64±2.64 +	28.06±3.97 +	63.74±0.74 –	29.40±5.64 +	55.16±4.63 =	57.26±2.74
KTH (3-shot)	42.62±3.57 +	38.12±3.45 +	64.65±0.92 –	49.69±5.00 +	41.64±1.92 +	61.57±3.86
KTH (5-shot)	48.38±3.05 +	39.93±3.21 +	64.98±0.91 +	64.77±1.09 +	41.45±1.94 +	68.03±1.47
Outex (3-shot)	51.82±3.90 +	53.37±4.47 +	63.99±0.63 +	46.75±3.11 +	42.47±4.88 +	74.10±1.04
Outex (5-shot)	58.63± 5.51 +	58.94±5.77 +	64.17±0.80 +	47.47±3.36 +	50.06±2.45 +	82.12±0.78
Overall	18+	18+	13+, 5–	16+, 2–	15+, 1=, 2–	

TABLE IV
SUMMARY OF COMPARISONS BETWEEN DTGPN WITH 12 METHODS WITH DATA AUGMENTATION IN TERMS OF THE SIGNIFICANCE TEST ON THE 18 FSIC DATASETS

JAFFE (3-shot)	6+, 1=, 5–	JAFFE (5-shot)	12+
FEI (3-shot)	12+	FEI (5-shot)	12+
ORL (3-shot)	12+	ORL (5-shot)	12+
Faces (3-shot)	12+	Faces (5-shot)	12+
YaleB (3-shot)	12+	YaleB (6-shot)	12+
COIL (3-shot)	12+	COIL (5-shot)	12+
DSLR (3-shot)	12+	DSLR (5-shot)	12+
Outex (3-shot)	12+	Outex (5-shot)	12+

classification performance of these different baseline methods on some datasets. For example, on the object classification datasets, i.e., COIL and DSLR, the classification performance of some methods such as RF and SRC is improved. However, comparing the results with data augmentation with those in Table II, it can be found that using data augmentation does not necessarily improve the classification performance on these datasets with a small number of training instances. Data augmentation can only introduce limited information into the training set so that the classification performance may not be improved. In some cases, using data augmentation even decreases the classification performance, which indicates that data augmentation harms the quality of the training set. This shows the ineffectiveness of data augmentation in FSIC.

C. Comparisons with State-of-the-art FSIC Methods

Table III lists the classification results obtained by the five well-known FSIC methods (i.e., Siamese Networks, Triplet Networks, Prototypical Networks, Matching Networks, and MAML) on the 18 datasets. The final row of the table summarises the results of the significance tests. Due to the page limit, the comparisons with some of these methods using an external large dataset for model pre-training are discussed in the supplementary materials.

The results show that DTGPN achieves significantly better performance in 80 comparisons out of the 90 comparisons. Specifically, DTGPN achieves significantly better performance than Siamese Networks and Triplet Networks on all these 18 datasets. Compared with the Prototypical Networks, DTGPN achieves significantly better performance on 13 datasets and worse performance on 5 datasets, i.e., the JAFFE (3-shot and 5-shot), DSLR (3-shot and 5-shot), and KTH (3-shot) datasets. On JAFFE (3-shot and 5-shot), DTGPN achieves worse performance than Matching Networks. Similarly, DTGPN achieves worse performance than MAML on JAFFE (3-shot) and DSLR (3-shot and 5-shot). Although DTGPN is worse in the above cases, it significantly improves the classification accuracy on most of the remaining datasets. For instance, it improves the accuracy of over 25% on ORL (3-shot and 5-shot), over 32% on Faces (3-shot), over 15% on COIL (3-shot and 5-shot), and over 10% on FEI (3-shot) and Outex (3-shot and 5-shot). Overall, the results suggest that the DTGPN approach achieves significantly better performance than these five well-known few-shot learning methods in most comparisons.

D. Summary

To sum up, the proposed DTGPN approach achieves significantly better or similar performance in almost all the comparisons. The results show that DTGPN achieves significantly better performance than the baseline GP method (i.e. FLGP), which indicates the effectiveness of the new representation and the new fitness function. The results show that DTGPN achieves significantly better performance than 12 non-GP-based baseline methods with and without data augmentation on these 18 FSIC datasets, including different types of images and different numbers of classes. Compared with five popular FSIC methods (i.e., Siamese Networks, Triplet Networks,

TABLE V
TEST ACCURACY (%) OF DTGPN, GPN (SINGLE-TREE GP WITH THE NEW FITNESS FUNCTION) AND DTGP (DUAL-TREE GP WITHOUT THE NEW FITNESS FUNCTION) ON THESE 18 FSIC DATASETS

	GPN	DTGP	DTGPN
Dataset	**Mean±St.dev**	**Mean±St.dev**	**Mean±St.dev**
JAFFE (3-shot)	21.60±4.53 =	**26.25±7.60** =	25.16±6.96
JAFFE (5-shot)	53.45±3.85 =	50.58±4.86 +	**53.52±3.49**
FEI (3-shot)	84.17±12.08 =	63.71±9.13 +	**84.85±6.97**
FEI (5-shot)	80.47±9.01 =	**81.37±8.43** =	81.16±4.37
ORL (3-shot)	96.44±0.51 +	96.06±0.78 +	**96.76±0.32**
ORL (5-shot)	99.72±0.28 =	98.75±0.50 +	**99.73±0.29**
Faces (3-shot)	93.32±0.78 +	92.82±0.95 +	**93.77±1.03**
Faces (5-shot)	96.40±0.96 +	**97.29±0.71** =	96.97±0.89
YaleB (3-shot)	70.89±2.52 +	75.69±3.10 =	**75.94±2.97**
YaleB (5-shot)	96.98±1.07 +	96.96±1.05 +	**97.84±0.92**
COIL (3-shot)	86.54±1.70 +	84.65±2.57 +	**87.97±1.43**
COIL (5-shot)	95.37±1.37 =	93.54±1.96 +	**95.75±1.16**
DSLR (3-shot)	47.96±2.65 +	50.78±2.39 =	**50.94±3.24**
DSLR (5-shot)	56.50±1.44 =	**57.60±2.35** =	57.26±2.74
KTH (3-shot)	60.24±3.86 =	56.92±3.84 +	**61.57±3.86**
KTH (5-shot)	**69.47±2.66** –	67.39±2.96 =	68.03±1.47
Outex (3-shot)	73.71±1.28 =	73.90±2.12 =	**74.10±1.04**
Outex (5-shot)	**82.99±0.98** –	82.46±1.10 =	82.12±0.78
Overall	7+, 9=, 2–	9+, 9=	

Prototypical Networks, Matching Networks, and MAML), the proposed approach achieves significantly better performance in most comparisons. The results show the effectiveness of DTGPN on these FSIC tasks. Overall, DTGPN can be easily applied to different FSIC tasks with different numbers of classes to achieve good performance.

VI. FURTHER ANALYSIS

This section further analyses the performance of DTGPN and its new components, i.e., the dual-tree representation and the fitness function. Besides, the number of learned features and the example trees evolved by DTGPN are analysed. Due to the page limit, the analysis of the parameter sensitivity, convergence behaviour, running and testing time is presented in the supplementary materials.

A. Effectiveness of the Dual-Tree Representation

The effectiveness of the dual-tree representation in DTGPN is analysed by comparing its performance with a single-tree GP approach with the same fitness function (i.e., GPN for short). The test results of DTGPN and GPN are listed in Table V. The "+", "−" and "=" symbols indicate that DTGPN achieves significantly better, worse or similar performance than/to GPN. The results show that DTGPN achieves significantly better results in seven comparisons than and similar results in nine comparisons to GPN. It shows that the GP method with a dual-tree representation can significantly improve the classification performance. Compared with a single-tree representation, a dual-tree representation can improve the search ability of GP to finding better solutions, which is also found from the analysis of the convergence behaviour in the supplementary materials. The analysis confirms the effectiveness of the dual-tree representation in DTGPN.

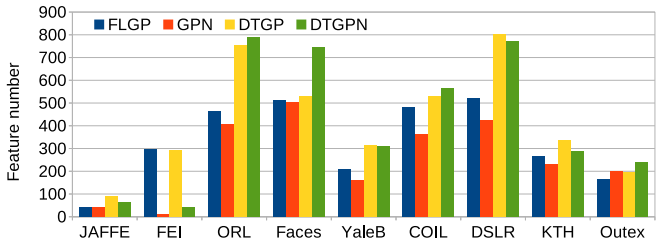


Fig. 9. Number of features learned by FLGP, GPN, DTGP, and DTGPN on the nine datasets of 3-shot image classification.

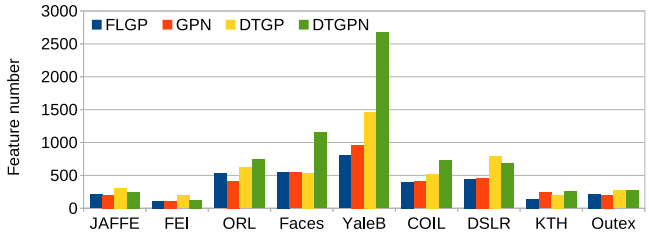


Fig. 10. Number of features learned by FLGP, GPN, DTGP, and DTGPN on the nine datasets of 5-shot image classification.

B. Effectiveness of The New Fitness Function

The effectiveness of the new fitness function is analysed by comparing it with the fitness function of using only the classification accuracy in DTGPN. The compared method is termed as DTGP. The fitness function of the classification accuracy is selected because it is straightforward and commonly used [25, 26]. Table V lists the test results of DTGPN and DTGP. Compared with DTGP, DTGPN achieves significantly better or similar results in all the comparisons. The fitness function in DTGPN optimises not only the classification accuracy but also the distances between the training instances to the class centroids. It can provide more accurate information of the goodness of the learned features and the SVM classifiers trained using these features. From the results, it is clear that the fitness function can effectively improve the classification performance on the test set (i.e., the generalisation performance). This analysis confirms the effectiveness of the fitness function in the DTGPN approach.

C. Number of the Learned Features

The number of the features learned by DTGPN is compared with that obtained by FLGP, GPN and DTGPN on 3-shot image classification tasks in Fig. 9 and 5-shot image classification tasks in Fig. 10. The feature number in these two figures indicates the average value of the 30 runs. From these two figures, it can be found that the GP methods with a dual-tree representation (i.e., DTGP and DTGPN) learn a larger number of features than the GP methods with a single-tree representation (i.e., GPN and FLGP) using the same fitness function. Having one more tree, more possible combinations of the features can be learned by DTGP and DTGPN. Furthermore, a large number of features may cover more variations of the representing images, which can improve the generalisation performance of the classification system based on them. Thus,

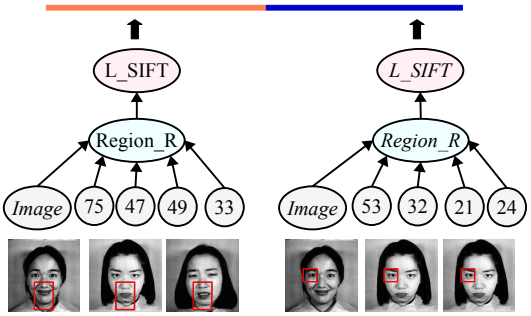


Fig. 11. Example trees evolved by DTGPN on the JAFFE (5-shot) dataset.

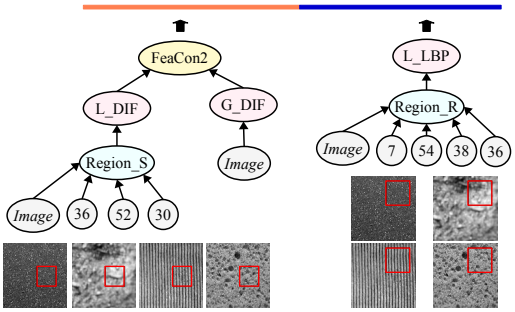


Fig. 12. Example trees evolved by DTGPN on the KTH (3-shot) dataset.

DTGPN achieves better classification performance than GPN in most comparisons. Figures 9 and 10 also show that the GP methods with the proposed fitness function (i.e., GPN) learn a smaller number of features than the GP methods with the fitness function of the classification accuracy (i.e., FLGP) on most of the datasets, particularly on 3-shot image classification tasks. This suggests that the proposed fitness function can lead to learning a small number of features when the number of training instances is small. The proposed fitness function has a component of a distance measure based on the decision space. The distance measure may prefer the SVM classifiers with fewer parameters, indicating that a small number of features are needed. Comparing DTGPN with the other three GP methods, it is clear that DTGPN learns an appropriate number of features from different datasets to achieve good generalisation performance by having the dual-tree representation and the fitness function. On a few datasets, such as Faces (5-shot) and EYALE (5-shot), DTGPN learns a large number of features and achieves the best classification performance. On most of the other datasets, the number of features learned by DTGPN is not the biggest one or the smallest one. This shows that DTGPN can automatically balance the training accuracy and the classifier complexity by learning a reasonable number of features, but this needs to be further investigated in the future.

D. Example Trees Evolved by DTGPN

1) Example Trees on JAFFE (5-shot): An example individual of two trees evolved by DTGPN is visualised in Fig. 11. This example trees achieve 58.43% accuracy on the JAFFE (5-shot) dataset. The two trees detect rectangle regions from the face images and extract 128 SIFT features from the detected regions, respectively. The left tree can achieve 53.93% accuracy on this dataset using 128 SIFT features and the right tree can achieve 46.63% accuracy on this dataset. A combination of the 256 SIFT features extracted by these two trees achieves better classification performance. From the face images in Fig. 11, it can be found that the two detected regions contain discriminative information of the images with different facial expressions. From Table II, it can be found that the SIFT features are more effective than the LBP, DIF and Histogram features for classifying this dataset. Therefore, it is reasonable that the example trees have the L_SIFT functions to extract features from these two detected regions.

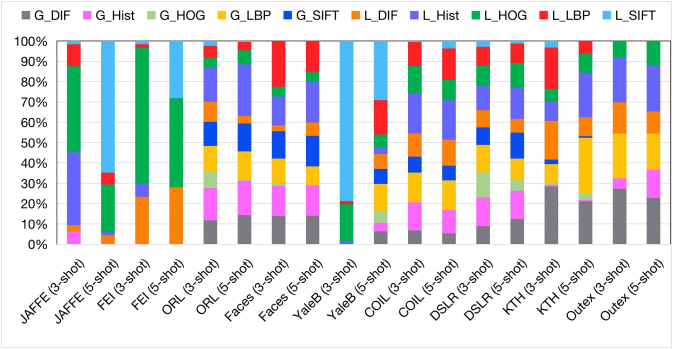


Fig. 13. The frequency (normalised to 100%) of the feature extraction functions in the trees evolved by DTGPN on the 18 datasets.

2) Example Trees on KTH (3-shot): Two example trees evolved by DTGPN on KTH (3-shot) are shown in Fig. 12. The left tree extracts DIF features from a detected square region and the whole image. These features can achieve 42.56% test accuracy on this dataset. The right tree extracts LBP features from a detected rectangle region, which achieves 45.26% test accuracy. A combination of these features achieves 61.03% test accuracy, which is better than many baseline methods. The analysis show that DTGPN can find good combination of features to achieve high classification performance.

3) Function Frequency Analysis of the GP Trees: The frequency (normalised to 100%) of the feature extraction functions in the trees evolved by DTGPN of the 30 runs on the 18 datasets are shown in Fig. 13. It can be found that the frequencies of these functions vary with datasets. On JAFFE (3-shot) and FEI (3-shot), the L_HOG function (extracting local HOG features) is frequently used. On JAFFE (5-shot) and YaleB (3-shot), the L_SIFT function (extracting local SIFT features) is frequently used. These functions extract different image features to achieve various performance on these datasets. DTGPN can evolve trees having these functions to extract effective features to achieve good performance when the number of training instance is very small. By analysing these frequencies, it is clear to understand what features, e.g., global features, local features, LBP features, HOG features, and SIFT features, are effective for which dataset.

To sum up, the analysis verifies the effectiveness of the new components, i.e., the dual-tree representation and the fitness function, in the proposed DTGPN approach. The example trees evolved by DTGPN show potentially high interpretability by

providing insights into what and why the features are effective for an image classification task.

VII. Conclusions

The goal of this paper was to develop a new GP-based approach to FSIC. This goal was achieved by proposing the DTGPN approach with a dual-tree representation and a fitness function and examining its performance on different types of FSIC tasks with various numbers of classes. A dual-tree representation allows DTGPN to better search for the solution space to find solutions that extract rich features for effective classification. To improve the generalisation performance, a fitness function was proposed to optimise the classification accuracy and the distances of the training instances to the class centroids. The performance of DTGPN was examined on nine datasets of varying difficulties under both the 3-shot and 5-shot scenarios. The results showed that DTGPN achieved significantly better classification performance than the baseline methods, the baseline methods with data augmentation, and the five well-known few-shot learning methods. The results showed that DTGPN is an effective approach to few-shot image classification.

With the new designs, the DTGPN approach was able to achieve good classification performance by automatically learning features and training SVM classifiers using a very small number of training instances. Further analysis showed that DTGPN with a dual-tree representation had better search ability and learned a relatively larger number of features to achieve better classification performance than that with a single-tree representation. The analysis also showed that the proposed fitness function improved the generalisation performance of DTGPN. The analysis on the example trees evolved by DTGPN showed the potentially high interpretability.

This paper is the first work to comprehensively investigate the effectiveness of GP for FSIC without the use of data augmentation and transfer learning. The overall process of the proposed approach follows a standard learning and classification process, which is easy to understand. However, it is different from most existing FSIC algorithms that are mainly based on NNs or use NNs as the backbone. This makes it hard to verify the performance of the proposed approach on well-known benchmark datasets such as Omniglot and miniImageNet and compare it with other FSIC algorithms. In the future, we will develop new GP approaches and apply them to the well-known FSIC benchmark datasets. Secondly, the region detection operators in the proposed approach are not position-invariant. In the future, we will also develop new translationally invariant region detection operators in GP for feature extraction and learning.

References

[1] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–34, 2020.

[2] D. Das and C. G. Lee, "A two-stage approach to few-shot learning for image recognition," *IEEE Trans. Image Process.*, vol. 29, pp. 3336–3350, 2019.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NeurIPS*, 2012, pp. 1097–1105.

[4] Z.-H. Zhou, "Learnware: on the future of machine learning." *Frontiers Comput. Sci.*, vol. 10, no. 4, pp. 589–590, 2016.

[5] E. Schwartz, L. Karlinsky, J. Shtok *et al.*, "Delta-encoder: an effective sample synthesis method for few-shot object recognition."

[6] A. Antoniou, A. Storkey, and H. Edwards, "Data augmentation generative adversarial networks," *arXiv preprint arXiv:1711.04340*, 2017.

[7] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, p. 60, 2019.

[8] Q. Sun, Y. Liu, Z. Chen, T.-S. Chua, and B. Schiele, "Meta-transfer learning through hard tasks," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020, DOI: 10.1109/TPAMI.2020.3018506.

[9] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Trans. Knowl. Data Eng.*, 2021, DOI: 10.1109/TKDE.2021.3070203.

[10] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proc. ICML Deep Learning Workshop*, vol. 2, 2015, pp. 1–8.

[11] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," *Proc. NeurIPs*, pp. 4080–4090, 2017.

[12] T. Schaul and J. Schmidhuber, "Metalearning," *Scholarpedia*, vol. 5, no. 6, p. 4650, 2010.

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

[14] 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.

[15] 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.

[16] Y. Bi, B. Xue, and M. Zhang, *Genetic Programming for Image Classification: An Automated Approach to Image Classification*. Springer International Publishing, XXVIII, 258pp, 2021, DOI: https://doi.org/10.1007/978-3-030-65927-1.

[17] B. Peng, S. Wan *et al.*, "Automatic feature extraction and construction using genetic programming for rotating machine fault diagnosis," *IEEE Trans. Cybern.*, 2020. DOI: 10.1109/TCYB.2020.3032945.

[18] Y. Bi *et al.*, "Genetic programming with image-related operators and a flexible program structure for feature learning to image classification," *IEEE Trans. Evol. Comput.*, vol. 25, no. 1, pp. 87–101, 2021.

[19] Y. Bi, B. Xue, and M. Zhang, "A divide-and-conquer genetic programming algorithm with ensembles for image classification," *IEEE Trans. Evol. Comput.*, 2021, DOI: 10.1109/TEVC.2021.3082112.

[20] 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.

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

[22] Y. Bi, B. Xue, and M. Zhang, "Learning and sharing: A multitask genetic programming approach to image feature learning," *IEEE Trans. Evol. Comput.*, 2021, DOI: 10.1109/TEVC.2021.3097043.

[23] H. Al-Sahaf, M. Zhang, and M. Johnston, "Binary image classification: A genetic programming approach to the problem of limited training instances," *Evol. Comput.*, vol. 24, no. 1, pp. 143–182, 2016.

[24] Y. Bi, B. Xue, and M. Zhang, "Multi-objective genetic programming for feature learning in face recognition," *Appl. Soft Comput.*, vol. 103, 2021, DOI:https://doi.org/10.1016/j.asoc.2021.107152.

[25] 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.

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

[27] 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.

[28] X. Chen, Y. Sun, M. Zhang, and D. Peng, "Evolving deep convolutional variational autoencoders for image classification," *IEEE Trans. Evol. Comput.*, 2020, DOI: DOI: 10.1109/TEVC.2020.3047220.

[29] Z. Lu, I. Whalen, Y. Dhebar *et al.*, "Multi-objective evolutionary design of deep convolutional neural networks for image classification," *IEEE Trans. Evol. Comput.*, vol. 25, no. 2, pp. 277–291, 2021.

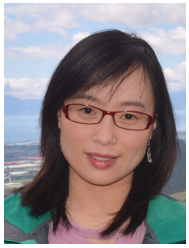
[30] X. Zhou, A. Qin, M. Gong, and K. C. Tan, "A survey on evolutionary construction of deep neural networks," *IEEE Trans. Evol. Comput.*, 2021, DOI: 10.1109/TEVC.2021.3079985.

[31] Y. Bi, B. Xue, and M. Zhang, "An evolutionary deep learning approach using genetic programming with convolution operators for image clas-

sification," in *Proc. IEEE CEC*, 2019, pp. 3197–3204.
[32] Z.-H. Zhou and J. Feng, "Deep forest," *Natl. Sci. Rev.*, vol. 6, no. 1, pp. 74–86, 2018.
[33] Q. U. Ain, H. Al-Sahaf, B. Xue, and M. Zhang, "Generating knowledge-guided discriminative features using genetic programming for melanoma detection," *IEEE Trans. Emerg. Topics Comput.*, 2020, DOI: 10.1109/TETCI.2020.2983426.
[34] D. J. Montana, "Strongly typed genetic programming," *Evol. Comput.*, vol. 3, no. 2, pp. 199–230, 1995.
[35] 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.
[36] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
[37] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE CVPR*, vol. 1, 2005, pp. 886–893.
[38] T. Ojala, M. Pietikainen *et al.*, "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.
[39] J.-H. Hong and S.-B. Cho, "A probabilistic multi-class strategy of one-vs.-rest support vector machines for cancer classification," *Neurocomputing*, vol. 71, no. 16-18, pp. 3275–3281, 2008.
[40] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets," in *Proc. IEEE AFGR*, 1998, pp. 200–205.
[41] C. E. Thomaz, "Fei face database," *online: http://fei.edu.br/˜cet/facedatabase.html*, 2012.
[42] 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.
[43] L. Spacek, "Face recognition data," *University of Essex. UK. Computer Vision Science Research Projects, http://cswww.essex.ac.uk/mv/allfaces/index.html*, 2012.
[44] 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.
[45] S. A. Nene, S. K. Nayar, H. Murase *et al.*, "Columbia object image library (coil-100)," 1996.
[46] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proc. ECCV*. Springer, 2010, pp. 213–226.
[47] 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.
[48] T. Ojala *et al.*, "Outex-new framework for empirical evaluation of texture analysis algorithms," in *Object recognition supported by user interaction for service robots*, vol. 1, 2002, pp. 701–706.
[49] L. Liu, T. Zhou, G. Long, J. Jiang, and C. Zhang, "Many-class few-shot learning on multi-granularity class hierarchy," *IEEE Trans. Knowl. Data Eng.*, 2020, DOI: 10.1109/TKDE.2020.3004939.
[50] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, 2008.
[51] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 7, pp. 711–720, 1997.
[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] F. Schroff *et al.*, "Facenet: A unified embedding for face recognition and clustering," in *Proc. IEEE CVPR*, 2015, pp. 815–823.
[54] O. Vinyals, C. Blundell *et al.*, "Matching networks for one shot learning," *arXiv preprint arXiv:1606.04080*, 2016.
[55] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. ICML*, 2017, pp. 1126–1135.
[56] Y. Bi, B. Xue, and M. Zhang, "Genetic programming with a new representation to automatically learn features and evolve ensembles for image classification," *IEEE Trans. Cybern.*, vol. 51, no. 4, pp. 1769–1783, 2021.
[57] F.-A. Fortin, F.-M. De Rainville *et al.*, "DEAP: Evolutionary algorithms made easy," *J. Mach. Learn. Res.*, vol. 13, no. Jul, pp. 2171–2175, 2012.
[58] F. Pedregosa, G. Varoquaux, and et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
[59] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

Ying Bi (M'17) received the Ph.D. degree in 2020 from Victoria University of Wellington (VUW), New Zealand. She is currently a Post-Doctoral Research Fellow and Project Coordinator with the School of Engineering and Computer Science, VUW. Her research focuses mainly on computer vision, machine learning, evolutionary computation, classification, feature learning, and transfer learning. She has published an authored book and over 30 papers in fully refereed journals and conferences in computer vision and evolutionary computation.

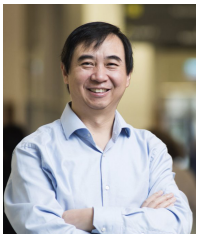
Dr Bi is a member of IEEE Computational Intelligence Society (CIS). She has been serving as an organizing committee member of IEEE CEC 2019 and Australasian AI 2018, an organiser of a workshop in ICDM 2021, a special session in IEEE SSCI 2021 and a special session in IDEAL 2021, and a program committee member of over ten international conferences including IJCAI, GECCO, IEEE CEC, and IEEE SSCI. She is serving as a reviewer of over ten international journals including IEEE Transactions on Evolutionary Computation and IEEE Computational Intelligence Magazine.



Bing Xue (M'10-SM'21) received the B.Sc. degree from the Henan University of Economics and Law, Zhengzhou, China, in 2007, the M.Sc. degree in management from Shenzhen University, Shenzhen, China, in 2010, and the Ph.D. degree in computer science in 2014 at VUW, New Zealand.

She is currently a Professor in Computer Science, and Program Director of Science in the School of Engineering and Computer Science at VUW. She has over 300 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 NNs, image analysis, transfer learning, multi-objective machine learning.

Dr Xue is currently the Chair of IEEE CIS Task Force on Transfer Learning & Transfer Optimization, and Vice-Chair of IEEE CIS Evolutionary Computation Technical Committee, Editor of IEEE CIS Newsletter, Vice-Chair of IEEE Task Force on Evolutionary Feature Selection and Construction and IEEE CIS Task Force on Evolutionary Deep Learning and Applications. She has also served as associate editor of several international journals, such as IEEE Computational Intelligence Magazine, IEEE Transactions on Evolutionary Computation and IEEE Transactions on Artificial Intelligence.



Mengjie Zhang (M'04-SM'10-F'19) received the B.E. and M.E. degrees from Artificial Intelligence Research 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, VUW. His current research interests include evolutionary computation, particularly genetic programming and particle swarm optimization with application areas of image analysis, multi-objective optimization, feature selection and reduction, job shop scheduling, and transfer learning. He has published over 700 papers in refereed international journals and conferences.

Prof. Zhang is a Fellow of Royal Society of New Zealand, a Fellow of IEEE, an IEEE CIS Distinguished Lecturer, and have been a Panel member of the Marsden Fund (New Zealand Government Funding). He was the chair of the IEEE CIS Intelligent Systems and Applications Technical Committee, the 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.