# Learning and Sharing: A Multitask Genetic Programming Approach to Image Feature Learning

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

***Abstract*—Using evolutionary computation algorithms to solve multiple tasks with knowledge sharing is a promising approach. Image feature learning can be considered as a multitask learning problem because different tasks may have a similar feature space. Genetic programming (GP) has been successfully applied to image feature learning for classification. However, most of the existing GP methods solve one task, independently, using sufficient training data. No multitask GP method has been developed for image feature learning. Therefore, this paper develops a multitask GP approach to image feature learning for classification with limited training data. Owing to the flexible representation of GP, a new knowledge sharing mechanism based on a new individual representation is developed to allow GP to automatically learn what to share across two tasks and to improve its learning performance. The shared knowledge is encoded as a common tree, which can represent the common/general features of two tasks. With the new individual representation, each task is solved using the features extracted from a common tree and a task-specific tree representing task-specific features. To find the best common and task-specific trees, a new evolutionary search process and fitness functions are developed. The performance of the new approach is examined on six multitask learning problems of 12 image classification datasets with limited training data and compared with 17 competitive methods. Experimental results show that the new approach outperforms these comparison methods in almost all the comparisons. Further analysis reveals that the new approach learns simple yet effective common trees with high effectiveness and transferability.**

***Index Terms*—Multitask Learning; Knowledge Sharing; Genetic Programming; Feature Learning; Image Classification**

## I. Introduction

Evolutionary multitask learning has become an increasingly popular topic in evolutionary computation (EC) and has been successfully applied to solve many problems such as function optimisation [1, 2, 3], combinatorial optimisation [4, 5], complex engineering design [1], and multiobjective optimisation [6, 7]. Evolutionary multitask learning [1] aims to use EC techniques to simultaneously solve multiple tasks, where the knowledge learned from different tasks can be implicitly or explicitly shared/transferred during the evolutionary process. Compared with traditional evolutionary single-task learning methods, evolutionary multitask learning methods often have faster convergence speed to find better solutions [1, 8]. However, the potential of evolutionary multitask learning has not been comprehensively investigated in image feature learning.

Image feature learning is the task of automatically learning informative features from images to solve a task, such as image classification [9]. It is an essential step to image analysis, which has a wide range of real-world applications in many fields, including medicine, robotics, remote sensing, and security [9, 10, 11]. However, image feature learning is difficult because of the high variations across images and a large search space. The task becomes even more challenging when only a small number of training instances are available to evaluate the performance of the learned features during the learning process. A small number of training instances often lead to poor generalisation performance [12].

Many feature learning methods have been developed for image classification [9, 13]. Most of them are based on convolutional neural networks (CNNs) [12, 13]. However, most CNNs have limitations, e.g., requiring a large number of training instances and having poor interpretability. Except for CNNs, EC-based methods have also been developed for feature learning in image classification. The most commonly used method is genetic programming (GP)-based method [9, 14]. Unlike CNNs, GP can use existing image-related operators as internal nodes to automatically evolve tree-like solutions with potentially good interpretability, such as in [15, 16]. Therefore, this paper uses GP to achieve image feature learning.

Image feature learning can be considered as a multitask learning problem because similar or related tasks may have a similar or common feature space. For example, to classify different texture image datasets, texture features, such as extracted by local binary patterns (LBP), are desirable. It is possible to simultaneously learn LBP-like features for different texture image classification tasks. Existing work has empirically shown the effectiveness of multitask feature learning with different assumptions on the task relatedness, such as in [17, 18]. However, most of them are based on sparse representation learning that uses well-extracted features, not raw image data. Therefore, this study investigates to simultaneously learn features from raw images for different classification tasks. There could be multiple ways to achieve it. However, this has never been investigated using EC techniques.

Existing EC methods have shown their potential of simultaneously solving two tasks and achieved better learning performance and/or faster convergence speed than solving those tasks individually [1, 2, 3]. In evolutionary multi-

Manuscript received XXX; revised XXX and 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 Evolutionary Computation Research Group, 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.

task learning, to avoid negative knowledge transfer/sharing across multiple tasks is important to improve the learning performance. Furthermore, image feature learning has two phases, i.e., the training/learning phase and the test phase. The training/learning phase can be achieved via the evolutionary process, while the test phase is after the evolutionary process. The test phase examines the performance of the learned features or models on unseen data, which is the generalisation performance. The learned model may memorize all the training instances and achieve high training performance, but have poor generalisation performance. This is known as overfitting, which is a common and open issue in machine learning. This is quite different from most optimisation problems solved by most of existing evolutionary multitask learning methods. Therefore, it is necessary to consider how to improve the generalisation performance when addressing multitask image feature learning using EC methods.

GP has been successfully applied to feature learning for image classification [9, 10, 19]. GP uses a tree-based variable-length representation and can automatically evolve trees/models that extract effective features from images for classification. Existing work on GP-based feature learning has achieved promising classification performance and potential interpretability [9, 15]. However, those existing methods often solve each task individually. To the best of our knowledge, there has not been any multitask GP method for image feature learning. Furthermore, most of existing GP-based methods evaluate the learned features using a *sufficient* number of training instances. However, training data is not always sufficient such as in the medicine and security domains, and/or need large manual effort to label. It is necessary to investigate whether GP can learn effective image features using limited training data.

The overall goal of this paper is to further explore the capability of GP by developing a multitask GP approach to feature learning for image classification with limited training data. The new approach is termed as KSMTGP, indicating MultiTask GP with a new Knowledge Sharing mechanism. To avoid negative knowledge transfer/sharing and make the best use of the flexible representation of GP, a new explicit knowledge sharing mechanism and a new individual representation will be developed the new approach. The main idea is to use KSMTGP to learn a common representation across two tasks and two task-specific representations in the form of trees. Each task will be solved by using two GP trees, i.e., a common tree and a task-specific tree. To achieve this, a new evolutionary search process will be developed in KSMTGP to search for the best common and task-specific trees with variable lengths. The performance of KSMTGP will be evaluated on 12 image classification datasets with limited training data, i.e., six multitask feature learning problems, and compared with 17 competitive methods to show its effectiveness.

The contributions of this paper are summarised as follows.

1) A new approach called KSMTGP is developed to achieve multitask feature learning for image classification. KSMTGP can achieve *explicit* knowledge sharing by automatically evolving a common tree with a variable length across two tasks and using this common tree as a part of the solution to image feature learning. The new approach is able to automatically determine what to share via learning and improves its performance by sharing. More importantly, owning to the flexible presentation of GP, the shared knowledge, i.e., the common tree, can be variable lengths and shapes.
2) A multi-tree representation is proposed in KSMTGP to achieve explicit knowledge sharing. The multi-tree representation includes three trees, i.e., a common tree and two task-specific trees. Each task is solved by using two trees, i.e., the common tree and its corresponding task-specific tree. The common tree represents the common knowledge discovered from the two tasks, while the task-specific trees represent the knowledge specifically learned for solving each single task.
3) A new evolutionary search process is developed to search for the best common tree and the best task-specific trees for the two tasks. Similar to co-evolution, it searches for the best common tree and selects the best common tree coupled with a task-specific tree. This design narrows the search space and can find the best individuals of two trees for each single task effectively.
4) Different fitness functions are employed for evaluating the performance of the common trees and the performance of the solutions on each task. The fitness function based on the classification accuracy of the two tasks and the tree size allows the new approach to find simple yet effective common trees across two tasks. The fitness function only based on the classification accuracy of a single task allows the new approach to find the best individual of the two trees for each single task.
5) The proposed KSMTGP approach is able to achieve better generalisation performance than two single-task GP methods, the multifactorial GP method, and 14 non-GP-based competitive methods on 12 image classification datasets with limited training data.

## II. Background and Related Work

### A. Multitask Image Feature Learning

Evolutionary multitask learning [1] is to simultaneously solve multiple problems using EC techniques with knowledge sharing. An evolutionary multitask learning problem has multiple tasks, denoted as $\{T_1,\ T_2,\ \ldots,\ T_K\}$, where $K$ indicates the number of total tasks and $T_k$ indicates the $k$th task. Each task $T_k$ has a search space of $X_k$ and an objective function $f_k$. A multitask learning problem is to find the optimal solution $x_k^*$ for each task, concurrently, which is denoted as

$$\{x_1^*, \ldots, x_K^*\} = \{\operatorname{argmin} f_1(x_1), \ldots, \operatorname{argmin} f_K(x_K)\}. \quad (1)$$

An image classification task often has a training set $\mathcal{D}_{train}$ and a test set $\mathcal{D}_{test}$. The training set has a number of labelled images, i.e., $\mathcal{D}_{train} = \{(x_0, y_0), \ldots, (x_n, y_n)\}$. The test set has a number of unlabelled images, i.e., $\mathcal{D}_{test} = \{z_0, \ldots, z_m\}$. $x \in \mathbb{R}^{w \times h \times g}$ and $z \in \mathbb{R}^{w \times h \times g}$ denote a grey scale ($g = 1$) or colour ($g = 3$) image with a size of $w \times h$. $y \in \mathbb{R}$ denotes the class label. An image feature learning task is to find the

optimal feature set transformed via a function/model $\Phi$ that can maximise a performance measure $\mathcal{L}$ on $\mathcal{D}_{train}$ as

$$\Phi^* = \text{argmax}\, \mathcal{L}(\Phi, \mathcal{D}_{train}). \quad (2)$$

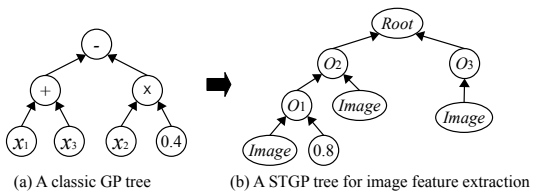
In this study, the problem of multitask image feature learning to image classification is to simultaneously find multiple optimal feature sets transformed by different functions/models, which can be denoted as

$$\{\Phi_1^*, \ldots, \Phi_K^*\} = \{\text{argmax}\, \mathcal{L}_1(\Phi_1, \mathcal{D}_{train}^1), \ldots, \text{argmax}\, \mathcal{L}_K(\Phi_K, \mathcal{D}_{train}^K)\}. \quad (3)$$

Typically, a feature learning problem has two phases, training/learning and test phases. The test phase is after the learning process. If $\{\Phi_1^*, \ldots, \Phi_K^*\}$ are obtained, they will be applied to classify images in the test sets, i.e., $\{\mathcal{D}_{test}^1, \ldots, \mathcal{D}_{test}^K\}$, respectively. The classification performance on the test set is used to show the generalisation performance. For an image classification task, various methods can be used to find the optimal $\Phi$. This paper uses GP as the main method because of its flexible representation, which will be introduced in the following subsection. This paper aims to further explore the capability of GP for feature learning in image classification.

### B. Genetic Programming

GP can automatically evolve computer programs to solve problems. Typically, GP uses a tree-based representation with variable lengths. An example GP tree is shown in Fig. 1(a), where the functions $+$, $-$ and $\times$ form the internal nodes and the features/variables/constants $x_1$, $x_2$, $x_3$, and $0.4$ form the leaf nodes. This example tree can be formulated as $(x_1+x_3)-(x_2\times 0.4)$. This tree representation is often used for regression, classification, feature construction, scheduling, and planning tasks [10]. For some other tasks such as image analysis [11], a different tree-based representation and some domain-specific operators are often used. Figure 1(b) shows an example GP tree with several specific operators $Root$, $O_1$, $O_2$ and $O_3$ and terminals $Image$ and $0.8$. This example tree can be formulated as $Root(O_2(O_1(Image,\ 0.8), Image),\ O_3(Image))$. This representation allows GP to use many domain-specific operators to evolve solutions with variable depths/lengths to solve complex tasks. Examples of such GP trees to solve image classification and fault diagnosis can be found in [20, 21]




(a) A classic GP tree (b) A STGP tree for image feature extraction

Fig. 1. Example GP trees.

### C. Related Work

*1) GP for Image Feature Learning:* Al-Sahaf et al. [22] proposed a GP-based method to detect features from keypoints and generate texture features in a way similar to LBP for texture image classification. This method achieved better performance than the other methods using different manually extracted texture features. This method has been further extended in [23] by developing a GP method with a new representation to generate a dynamic number of features for texture classification. This method has improved the classification performance. Bi et al. [21] proposed a FGP method with many image-related operators to automatically learn various types and numbers of features for image classification. This method achieved better classification performance than many methods, including traditional methods using manually extracted features and CNN-based methods, on different image classification tasks. Bi et al. [14] proposed a FLGP method with existing well-developed image descriptors to automatically learn global and/or local features for image classification. This method achieved significantly better performance than traditional methods using manually extracted features. In [24], a divide-and-conquer GP method was developed to use multiple small populations to build an ensemble for image classification. These aforementioned GP methods focus on how effective image features are learned and generated by developing various GP representations. Most of them tackled a limited type of tasks and none of them has been applied to simultaneously solve two image classification tasks.

*2) Evolutionary Multitask Optimisation and Learning:* Evolutionary multitask optimisation and learning has become an increasingly popular research topic in EC and has been widely applied to solve many tasks. Gupta et al. [1] proposed the early paradigm of evolutionary multitask optimisation, i.e., the multifactorial evolutionary algorithm (MFEA), to solve multiple problems, simultaneously. The main idea of MFEA was to use a unified search space to represent the population, which is split into different groups based on skill factors, where each group deals with a task. The whole population was evolved during the evolutionary process, where genetic materials were implicitly transferred via genetic operators. The results showed that MFEA improved the convergence speed by simultaneously optimising two tasks. In [6], the potential ability of MEFA was further explored by proposing MO-MFEA to simultaneously solve multiple multi-objective optimisation problems with achieving promising results. Feng et al. [2] proposed an EMT framework with explicit knowledge sharing, which was achieved by using a denoising autoencoder. Unlike MFEA, which uses a unified search space for different tasks, EMT used independent individual representation for each task. The results on single and multi-objective multitask optimisation benchmarks showed that EMT was more effective than the single-tasking method and MEFA.

Zhou et al. [25] developed an adaptive knowledge transfer strategy in MEFA, i.e., MEFA-AKT, for single- and multi-objective optimisation. The adaptive knowledge transfer was achieved by changing the crossover operators in MEFA using the information collected from the evolutionary process. The results showed that MEFA-AKT achieved better performance than MEFA. In [4], the computing resources allocation problem in MEFA was investigated and a new strategy was proposed to dynamically allocate the resources to the tasks according to their complexities. Zheng et al. [26] proposed a

self-regulated evolutionary multitask optimisation framework, i.e., SREMTO with the consideration of tasks relatedness. This method used an ability vector to denote the ability of an individual handling each task rather than the factorial rank. The superiority of SREMTO has been verified on two evolutionary multitask optimisation test suites.

Zhong et al. [27] proposed a multifactorial GP method for symbolic regression. This method used a scalable chromosome encoding scheme to represent multiple solutions and the multifactorial framework to search for the best solutions for multiple tasks using a single population. This method achieved promising performance on synthetic and real-world symbolic regression problems under two-task settings. Kattan et al. [28] developed a new multitask GP method to solve more than two tasks simultaneously. This method used new genetic operators to generate populations for multiple tasks with implicit knowledge sharing. The results showed that this method evolved small solutions and were significantly faster on different regression and classification tasks.

*3) Summary:* Existing GP-based methods for multitask learning have achieved promising results in symbolic regression and classification problems. However, the potential ability of multitask GP has not been investigated in feature learning for image classification. On the other hand, many GP-based methods have been developed for feature learning in image classification. But most of them use sufficient training data to achieve feature learning. Existing works have also shown that multitask feature learning may be helpful for improving generalisation when training data is limited [17, 18]. Therefore, to address these limitations, this paper develops a new multitask GP approach to feature learning for image classification with limited training data.

## III. The Proposed Approach

The section describes the main idea of knowledge sharing and how this is achieved in the new KSMTGP approach by developing a new individual representation. Then the evolutionary process, the fitness functions and the tree representation (the baseline method) are presented.

### A. A New Individual Representation for Knowledge Sharing

When applying GP to learn features, a specific tree structure is often employed. The tree structure typically consists of many image-related operators that construct GP trees to extract a set of features from an image, such as in [15, 21]. Using such GP trees, the image classification process is shown in Fig. 2, where support vector machine (SVM) performs classification using the features extracted by GP trees [15, 21].

The KSMTGP approach is developed to simultaneously search for the best individuals for the two image feature learning tasks. To achieve this, a new individual representation is developed with the idea of knowledge sharing, as shown in Fig. 3. It is encoded by three trees, i.e., the left (red) tree indicates a task-specific tree for task 1, the middle (yellow) tree indicates the common tree shared by the two tasks, and the right (blue) tree indicates a task-specific tree for task 2. In other words, the features of the images for each task are

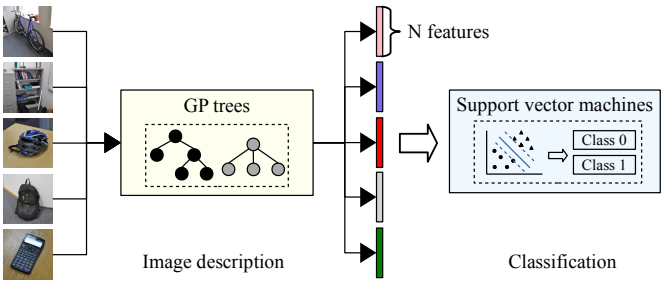



Fig. 2. A general process of GP-based feature learning for image classification. The GP tree is able to transform images into $N(N > 1)$ features, which are fed into a classification algorithm such as SVM for classification.

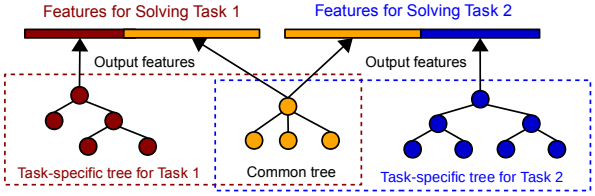



Fig. 3. Individual representation of KSMTGP. Each task is solved by using a task-specific tree and the common tree. Each tree can return a flexible number of features from an image using a specific tree structure proposed in [21].

the combination of features generated using the common tree and the task-specific tree. The task-specific trees and the common trees are automatically learned through the evolutionary learning process of KSMTGP. This design determines when to transfer and how to transfer, which are two typical questions in transfer learning. Specifically, the knowledge transfer/sharing happens when using the task-specific tree and the common tree to solve every single task. The common tree is automatically learned from the two tasks using the new evolutionary process and used to extract features from images when solving each single task, which answers the question of how to transfer.

Such a design is derived from the concept/assumption that two similar or related image classification tasks may have/share common features. This concept/assumption can be seen from many other methods for multitask feature learning [17, 29] or the applications of pretrained CNN models as feature extractors to solve other image classification tasks [30]. In KSMTGP, the common tree is employed to describe common/general features across two tasks. The common features often improve the generalisation performance, particularly when the training data are limited [17]. The common tree, having a variable-length and shape, can be automatically evolved through the evolutionary process with the use of fitness function and genetic operators. Two important characteristics are that the number of features described by the common tree varies with the tasks and the common tree can return features from the given images of different sizes. These two characteristics enable the high adaptability of KSMTGP on various image classification tasks.

Since the common tree represents the knowledge learned from the two tasks, which could be more general but less discriminative. On extreme case is that the two tasks are not related, leading to a bad common tree. To improve the

discriminability of the features on each task, a task-specific tree is employed in KSMTGP to represent task-specific features. Two similar or related tasks may need not only common features but also task-specific features [31]. The combination of common and task-specific features is expected to improve the learning performance on each task. Therefore, each task is solved by using two trees, i.e., a common tree and a task-specific tree. The features extracted by these two trees are combined for classification, as shown in Figures 2 and 3.

One important point is that the search space of finding the common tree and the task-specific tree is fixed and the same when apply KSMTGP to solve different feature learning tasks, which is different from most existing evolutionary multitask learning methods [1]. In KSMTGP, the search space is typically related to the program structure, the function set, and the terminal set. Once these components are defined/set, they can be kept the same when applying KSMTGP to different image classification tasks. Owing to the flexible variable-length representation, KSMTGP can learn different trees with various shapes, sizes and nodes for different tasks, although the search space is the same.

### B. Evolutionary Process

To effectively search for the best common and task-specific trees, KSMTGP uses a new evolutionary search process similar to that of co-evolution by using multiple populations. Co-evolutionary algorithms have been widely used and achieved promising performance since they have more interactions between individuals [32]. KSMTGP uses a population to search for the common tree and selects the best common tree to be combined with the task-specific trees at each generation. Based on the best common tree, two 2-tree populations are employed to search for the best trees for each task. These processes are conducted at each generation so that more possible combinations of these trees can be obtained and the performance on the tasks can be gradually and collaboratively improved. The overall process of KSMTGP is shown in Algorithm 1. The inputs of the KSMTGP system are the training sets of two image classification tasks, i.e., $\mathcal{D}^1_{train}$ and $\mathcal{D}^2_{train}$.

The overall process of KSMTGP starts with randomly initialising three populations, i.e., one common population and two task-specific populations for the two tasks. The common population aims to learn common knowledge from these two tasks, while the task-specific population aims to learn task-specific knowledge from each of these two tasks. This common population will be evaluated using a fitness function. Then the best tree of the common population is selected as the common knowledge learned from these two tasks. The best common tree is combined with these two task-specific populations to form two 2-tree populations. Each individual of these two 2-tree populations has one tree from the task-specific population and one tree from the common population. The two 2-tree populations will be evaluated using the fitness function based on the training sets of these two tasks, respectively.

The evolutionary process of these three populations is the same as that of the traditional GP method. The common population uses selection, subtree crossover and subtree mutation operators to generate a new population. These two task-specific populations use elitism, selection, subtree crossover, and subtree mutation operators to generate new populations. Note that the common population generation does not have the elitism operator because only the best tree is selected to be combined with the task-specific population. So it is unnecessary to have the elitism operator. At each generation, the best individual of two trees for each task is updated. After the evolutionary process, the best individual of two trees for each task is returned.

**Algorithm 1:** KSMTGP

**Input :** $\mathcal{D}^1_{train}$, $\mathcal{D}^2_{train}$: the training sets of tasks 1 and 2.
**Output :** $Best\_Ind_1$: the best individual for task 1;
$Best\_Ind_2$: the best individual for task 2.

$P_0^{com} \leftarrow$ initialise a common population of trees for the two tasks;
$P_0^1, P_0^2 \leftarrow$ initialise two task-specific populations for the two tasks;
$g \leftarrow 0$;
**while** $g \leq G$ **do**
  Evaluate common population $P_g^{com}$ according to **Algorithm** 2;
  $ct_{best} \leftarrow$ select the best common tree from $P_g^{com}$;
  $(P_g^1,\ ct_{best}), (P_g^2,\ ct_{best}) \leftarrow$ form two 2-tree populations for the two tasks;
  Evaluate $(P_g^1,\ ct_{best})$ and $(P_g^2,\ ct_{best})$ on $\mathcal{D}^1_{train}$ and $\mathcal{D}^2_{train}$, respectively, according to **Algorithm** 3;
  Update $Best\_Ind_1$ and $Best\_Ind_2$;
  $P_{g+1}^{com} \leftarrow$ offspring generated from $P_g^{com}$ using selection, subtree crossover and subtree mutation;
  $P_{g+1}^1 \leftarrow$ offspring generated from $P_g^1$ using elitism, selection, subtree crossover and subtree mutation;
  $P_{g+1}^2 \leftarrow$ offspring generated from $P_g^2$ using elitism, selection, subtree crossover and subtree mutation;
  $(P_{g+1}^1,\ ct_{best}), (P_{g+1}^2,\ ct_{best}) \leftarrow$ form two multi-tree populations for the two tasks;
  $g \leftarrow g + 1$;
**end**
Return $Best\_Ind_1$ and $Best\_Ind_2$.

### C. Fitness Evaluation

*1) Fitness Evaluation for Common Population:* The common population aims to learn common and general knowledge across two tasks. A better common tree should not only achieve good performance on two tasks but also have less complexity. Existing work on GP has shown that a simple model often has better generalisation performance than a complex model [33]. However, measuring the complexity of a GP tree with many image-related operators as internal nodes is not easy. For simplicity, the tree size is used as an indicator of the tree complexity. To find the best common tree with a small tree size, an integrated fitness function is employed for common population evaluation. The new fitness function to be maximised is defined in Eq. (4). The overall fitness evaluation process is described in Algorithm 2.

$$f_{com} = \frac{1}{2}(acc_1 + acc_2) - size \tag{4}$$

$$acc = \frac{N_{correct}}{N_{total}} * 100\% \tag{5}$$

where $acc_1$ and $acc_2$ indicate the average classification accuracy (%) obtained using $K$-fold cross validation and a classification algorithm on the training sets of task 1 and task

2, respectively. $size$ denotes the tree size. $N_{correct}$ denotes the number of correctly classified images, and $N_{total}$ denotes the total number of images in the training set. It is noted that the tree structure employed in KSMTGP is different from that in standard GP and the tree size is typically smaller than 100, when the maximal tree size is 8.

**Algorithm 2:** Fitness Evaluation for Common Population

**Input :** $\mathcal{D}^1_{train}$: the training set of task 1; $\mathcal{D}^2_{train}$: the training set of task 2; $P^{com}$: the population to be evaluated.
**Output :** $P^{com}$: the evaluated population.

```
for each tree p in P^com do
    for each task t in {1,2} do
        for each image i in D^t_train do
            f_i = {f_{i,0}, ..., f_{i,m_p}} ← features extracted from image i using the tree of p;
        end
        D^t_tr ← the transformed training set with the features;
        D^t_nrom ← perform the min-max normalisation;
        acc_t ← use SVM with K-fold cross-validation on the normalised training set D^t_nrom to obtain the average accuracy;
    end
    size ← len(p);
    f_com(p) ← (1/2) Σ_{t=1}^{2} acc_t − size.
end
```

*2) Fitness Evaluation for Each Task:* With the individual representation in KSMTGP, each task is solved by using two trees, one from the task-specific population and the other is the best common tree. In the fitness evaluation, the performances of the two trees are evaluated using a fitness function, which is defined in Equations (5) and (6). It calculates the classification accuracy according to the training set of each task.

$$f_t = acc_t \tag{6}$$

The overall fitness evaluation process is described in Algorithm 3. This process starts by using the best common tree ($ct_{best}$) to transform the whole training set into features. When evaluating each individual, the features transformed by $ct_{best}$ will be directly used rather than performing transformation again, which saves computational cost.

**Algorithm 3:** Fitness Evaluation for Each Task

**Input :** $\mathcal{D}_{train}$: the training set; $(P,\ ct_{best})$: the population to be evaluated.
**Output :** $(P,\ ct_{best})$: the evaluated population.

```
for each image i in D_train do
    f_i = {f_{i,0}, ..., f_{i,m}} ← features extracted from image i using tree ct_best;
end
for each tree p in P do
    for each image i in D_train do
        v_i = {v_{i,0}, ..., v_{i,n_p}} ← features extracted from image i using tree p;
        {v_i, f_i} ← concatenating the features extracted by trees p and ct_best to form the feature set for describing image i;
    end
    acc ← normalise the transformed training set and use SVM with K-fold cross-validation to obtain the average accuracy;
    f(p) ← acc.
end
```

In the fitness evaluation for common population and the population for each task, a linear SVM is employed to obtain the classification accuracy based on the features learned by GP. The linear SVM is employed because it is commonly used in GP-based methods for image classification [15, 21]. The classification accuracy is obtained by performing $K$-fold cross validation on the training set. The average "test" accuracy of the $K$ folds are used in the fitness evaluation.

### *D. Tree Structure*

The algorithm framework of KSMTGP can cooperate with different GP tree structures that are able to transform images into features to achieve multitask feature learning for image classification. To investigate the performance of KSMTGP, this study uses a recently proposed tree structure of a baseline method, i.e., FGP in [21]. This tree structure is very flexible to allow GP to learn flexible numbers and types of features for different image classification tasks. KSMTGP uses the same tree structure to evolve the common trees and the task-specific trees. This section will briefly describe this tree structure, the function set and the terminal set.

The tree structure and an example tree is shown in Fig. 4. The tree structure has a number of different layers, including an input layer, filtering layers, pooling layers, a feature extraction layer, a feature concatenation layer, and an output layer. Each layer has different functions for different purposes. This tree structure allows GP to evolve variable-length trees that generate different types and numbers of features from images.

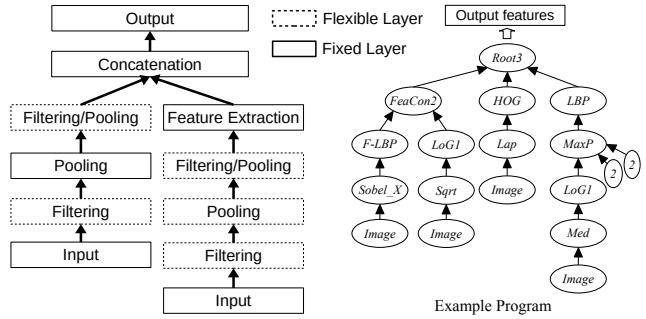



Fig. 4. The tree structure and an example tree (adapted from [21]).

*1) Function Set:* The function set has filtering functions, a pooling function, feature extraction functions, and concatenation functions, which are listed in Table I. The filtering functions take images and parameters if required as inputs and output a new image being processed. The filtering functions include Gaussian filters ($Gau$), Gaussian derivatives ($GauD$), Gabor filters ($Gabor$), Laplacian filter ($Lap$), Laplacian of Gaussian filters with two different parameter values ($LoG1$ and $LoG2$), Sobel edge detectors ($Sobel$, $sobelX$, $SobelY$), median filter ($Med$), mean filter ($Mean$), min filter ($Min$), max filter ($Max$), LBP as a filter ($LBP$-$F$), HOG as a filter ($HOG$-$F$), weighted sum ($W$-$Add$), weighted subtraction ($W$-$Sub$), rectified linear unit ($ReLU$), and Sqrt ($Sqrt$). The pooling function is max-pooling with two kernel size as two parameters ($MaxP$). The feature extraction functions are commonly used methods, i.e., $LBP$, $HOG$ and $SIFT$. The concatenation functions concatenate features or images into a feature vector. They are $FeaCon2$, $FeaCon3$, $Root2$, $Root3$,

and $Root4$. More details of these different functions can be found in [21].

TABLE I
FUNCTION SET

| Layer | Function |
|---|---|
| Filtering | $Gau$, $GauD$, $Gabor$, $Lap$, $LoG1$, $LoG2$, $Sobel$, $SobelX$, $SobelY$, $Med$, $Mean$, $Min$, $Max$, $LBP\text{-}F$, $HOG\text{-}F$, $W\text{-}Add$, $W\text{-}Sub$, $ReLU$, $Sqrt$ |
| Pooling | $MaxP$ |
| Feature Extraction | $SIFT, HOG, LBP$ |
| Concatenation | $FeaCon2$, $FeaCon3$, $Root2$, $Root3$, $Root4$ |

*2) Terminal Set:* The terminal set has $Image$ indicating the input image, and the parameters (i.e., $\sigma$, $o_1$, $o_2$, $\theta$, $f$, $n_1$, $n_2$, $k_1$, and $k_2$) for particular functions. The $\sigma$ terminal indicates the standard deviation of a Gaussian function in the $Gau$ and $GauD$ filters. It is an integer in the range $[1, 3]$. The $o_1$ and $o_2$ terminals denote the order of the Gaussian derivatives and are integers in the range $[0, 2]$. The $\theta$ terminal denotes the orientation of the $Gabor$ filter and is in the range $[0,\ 7\pi/8]$ with a step of $\pi/8$ [34]. The $f$ terminal denotes the frequency of the $Gabor$ filter and equals to $\frac{\pi}{2}/\sqrt{2}^{v}$, where $v$ is an integer in the range of $[0, 4]$ [34]. The $n_1$ and $n_2$ terminals are parameters for the $W\text{-}Add$ and $W\text{-}Sub$ functions and are in the range $[0,\ 1)$. The $k_1$ and $k_2$ terminals indicate the kernel size of the $MaxP$ function and are in the range $\{2, 4\}$.

## IV. EXPERIMENT DESIGN

This section designs the experiments, including comparison methods, benchmark problems of image classification with limited training data, and parameter settings.

### A. Comparison Methods

The performance of KSMTGP is compared with three GP-based methods, five classification algorithms, five traditional methods using manually designed features, and four methods based on CNNs, which are described as follows.

The three GP-based methods are a single-task GP method with a single tree representation (FGP proposed in [21]), a single-task GP method with a multi-tree representation (MTFGP), and a multitask method (i.e., multifactorial GP, MFFGP). These three methods use the same tree structure as KSMTGP. MTFGP has a multi-tree representation to evolve two trees to solve a task and uses the genetic operators employed in [35]. MFFGP is based on the multifactorial framework to search for the best solutions for the two tasks, simultaneously [1]. Since no multitask GP methods proposed for image feature learning, the multifactorial framework is used for comparison to investigate whether the proposed multitask framework is better for image classification.

The five classification algorithms are SVM, random forest (RF) [36], k-nearest neighbour (KNN) [23], sparse representation-based classification (SRC) [37], and linear discriminant analysis (LDA) [38]. The aim is to investigate whether KSMTGP can learn features that are more effective than raw pixels for classification using limited training data.

The five traditional methods using manually designed features are domain-independent features (DIF) [39], histogram features (Histogram) [40], histogram of oriented gradients (HOG) [41], local binary patterns (LBP) [42], and scale-invariant feature transform (SIFT) [43]. These methods use the corresponding features as inputs of a linear SVM to perform classification. They represent traditional image classification methods and the aim of comparisons is to show whether KSMTGP can learn better features with limited training data.

Two CNNs and two deep CNNs with transfer learning are used for comparisons. The two CNNs are LeNet [44] and a five-layer CNN (CNN-5) [15]. We do not directly train deep CNNs (including state-of-the-art CNNs) for comparisons due to a small number of training instances are used. Instead, the features extracted from pre-trained state-of-the-art deep CNNs, i.e., InceptionV3 [45] and InceptionResNetV2 [46], are used for comparisons. These two CNNs were trained on ImageNet [47] and the extracted features are fed into SVMs for classification. From our preliminary experiments, we found the features extracted from InceptionV3 and InceptionResNetV2 achieved better results than features from other famous pretrained deep CNN models, e.g., VGGNet, ResNet and DenseNet. Therefore, we only choose these two best methods for comparisons. Note that these methods use single-task settings. The aim of comparisons is to investigate whether the proposed method can achieve better performance than these popular CNN methods for image classification.

### B. Benchmark Problems

Due to no multitask feature learning problems are available for examining the performance of the proposed method, we construct six problems based on the task type of 12 different image classification datasets. These datasets have different images sizes, numbers of classes and sizes of training and test sets. They are FEI_1 [48], FEI_2 [48], JAFFE [49], RAFD [50], ORL [51], EYALE [52], KTH [53], Outex [54], Office_D [55], Office_W [55], COIL1 [56], and COIL2 [56]. These datasets are chosen because they represent typical image classification tasks, i.e., facial expression classification (FEI_1, FEI_2, JAFFE, and RAFD), face classification (ORL and EYALE), texture classification (KTH and Outex), and object classification (Office_D, Office_W, COIL_1, and COIL_2), and contain a wide range of image variations, e.g., in rotation, pose, illumination, scale, and occlusion. These will comprehensively show the effectiveness of the proposed approach. Example images from these datasets as shown in Fig. 5.

According to the types of tasks, six problems of multitask image feature learning are established, including two facial expression classification problems, one face classification problem, one texture classification problem, and two object classification problems. The details of them are listed in Table II. As this study focuses on limited training data, a small number of images are randomly selected to form the training sets. The fourth column of Table II shows the number of training images in each class, ranging from 5 to 25 except for the RAFD dataset. RAFD is a large dataset with 1,000 images per class. Therefore, 10% images are used as the training set and the remaining images are used as the test set. Table II also shows that the numbers of classes and the image sizes

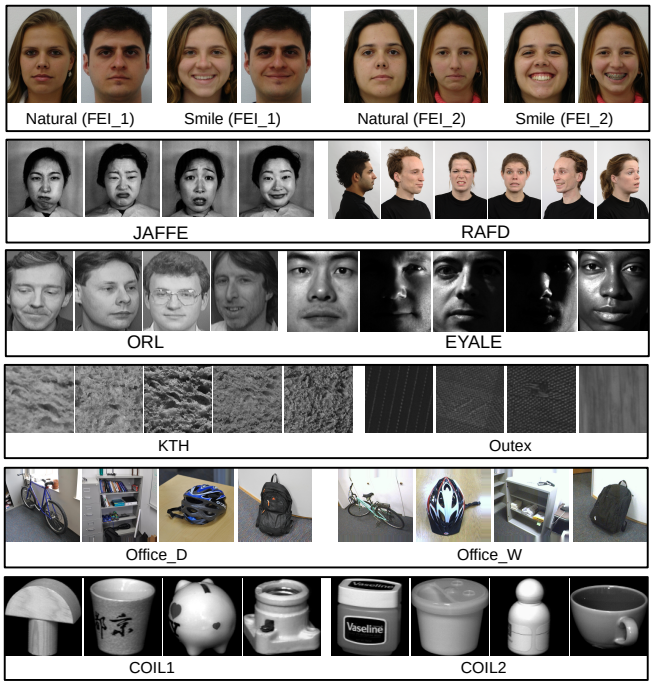



Fig. 5. Example images from the 12 image classification datasets.

TABLE II
SUMMARY OF THE DATASETS

| | Datasets | #Class | Image Size | Train Set (Per Class) | Test Set |
|---|---|---|---|---|---|
| **Problem 1** | FEI_1 | 2 | 40×60 | 50 (25 images) | 150 |
| | FEI_2 | 2 | 40×60 | 50 (25 images) | 150 |
| **Problem 2** | JAFFE | 7 | 64×64 | 70 (10 images) | 143 |
| | RAFD | 8 | 40×60 | 800 (100 images) | 7,240 |
| **Problem 3** | ORL | 40 | 60×50 | 200 (5 images) | 200 |
| | EYALE | 38 | 45×50 | 380 (10 images) | 2,044 |
| **Problem 4** | KTH | 10 | 50×50 | 200 (20 images) | 610 |
| | Outex | 24 | 64×64 | 480 (20 images) | 3,840 |
| **Problem 5** | Office_D | 31 | 50×50 | 155 (5 images) | 343 |
| | Office_W | 31 | 50×50 | 155 (5 images) | 640 |
| **Problem 6** | COIL_1 | 10 | 32×32 | 100 (10 images) | 620 |
| | COIL_2 | 10 | 32×32 | 100 (10 images) | 620 |

of the two tasks are different in most of the six multitask feature learning problems. To reduce the computational cost, some large images are resized to smaller images and the colour images are converted to gray-scale images.

### C. Parameter Settings

The parameters for KSMTGP are based on the commonly used settings of the GP community [14, 57]. In KSMTGP, the population size is 100, i.e, each population has a population size of 100. The maximal number of generation is 50. The crossover, mutation and elitism rates are 0.8, 0.19 and 0.01, respectively. The maximal tree depth is 8 and the minimum tree depth is 2. The tournament selection with a size of five is used for selecting individuals for crossover and mutation. For fair comparisons, the three GP-based methods (FGP, MTFGP and MFFGP) use the same parameter settings as KSMTGP. FGP and MTFGP deal with each task solely and use a population size of 100. MFFGP is a multitask method so that it uses a population size of 200 for the two tasks. These GP-based methods use SVM with $K$-fold cross validation for fitness evaluation and the fitness function is the classification accuracy of the training set. We set $K = 3$ for $K$-fold cross validation since the number of training instances is small.

The parameter settings for the non-GP-based methods follow the commonly used settings [23, 36, 58, 59]. The key parameters of these methods are listed as follows. The number of neighbours in KNN is one [23] and the penalty parameter in SVM is one [58]. In RF, the number of trees is 500 and the maximal tree depth is 100 [36]. In LeNet and CNN-5, the activation function is ReLU and the classification method is softmax. The number of epochs is 100 and the batch size is 20 due to a small number of training instances.

The GP-based methods are implemented using the DEAP [60] package. The classification algorithms are based on *scikit-learn* [61]. The two CNN methods, the InceptionV3 method and the InceptionResNetV2 method are based on *Keras* [59]. Each algorithm has been executed 30 independent times on each dataset/task using different random seeds.

## V. Results and Discussions

This section analyses and discusses the classification (generalisation) performance of KSMTGP by comparing it with the GP-based and non-GP-based comparison methods. Additional to the classification performance, this section also deeply analyses important aspects of KSMTGP, i.e., training performance, convergence behaviour, computational time, and the number of learned features, to provide more insights on its effectiveness.

### A. Classification Performance on the Test Sets

This section compares the classification performance of KSMTGP with the comparison methods on the six multitask learning problems of image classification. The classification accuracy (%) is used as the performance measure because it is the most commonly used. The results are listed in Tables III and IV. To show the significance of performance improvement, Wilcoxon rank-sum test with a significance level of 0.05 is used to compare KSMTGP with each of the comparison methods. In these tables, "+" and "–" denote KSMTGP achieves significantly better and worse performance than the compared method, respectively. "=" means no significant difference.

*1) Overall Performance:* Table III shows that KSMTGP outperforms the single-task GP (FGP), multi-tree GP (MTGP) and multi-factorial GP (MFFGP) methods by achieving similar or significantly better classification performance on six multitask learning problems of 12 image classification datasets, i.e., significantly better in 20 comparisons and similar 16 in out of the total 36 comparisons. Table IV shows that KSMTGP outperforms 14 non-GP-based methods in almost all the comparisons on the 12 image classification datasets as it achieves significantly better performance in 160 comparisons out of the total 168 comparisons. The results suggest that KSMTGP is an effective approach to automatically learning informative image features from two related or similar tasks for image classification. More detailed discussions and comparisons will be described in the following subsections.

TABLE III
CLASSIFICATION ACCURACY (%) OF FGP, MTFGP, MFFGP, AND KSMTGP ON THE TWELVE IMAGE CLASSIFICATION DATASETS. "+" MEANS KSMTGP IS SIGNIFICANTLY BETTER, "–" MEANS KSMTGP IS SIGNIFICANTLY WORSE, AND "=" MEANS NO SIGNIFICANT DIFFERENCE

| Problem | Dataset | FGP | | MTFGP | | MFFGP | | KSMTGP | |
|---|---|---|---|---|---|---|---|---|---|
| | | Max | Mean ± St.dev | Max | Mean ± St.dev | Max | Mean ± St.dev | Max | Mean ± St.dev |
| **Problem 1** | FEI_1 | 89.33 | 84.56 ± 3.50 + | **94.00** | 86.05 ± 4.27 = | 91.33 | 86.22 ± 2.83 = | **94.00** | **87.91 ± 3.39** |
| | FEI_2 | 92.00 | 87.16 ± 3.03 = | 92.67 | 86.55 ± 2.80 = | 92.67 | 87.98 ± 3.46 = | **94.67** | **88.24 ± 3.46** |
| **Problem 2** | JAFFE | 66.43 | 61.65 ± 3.35 + | 67.83 | 58.83 ± 4.46 + | 67.13 | 60.58 ± 3.23 + | **68.53** | **64.13 ± 2.60** |
| | RAFD | 49.83 | 44.82 ± 2.36 + | 49.64 | **46.95 ± 2.32** = | **50.26** | 43.14 ± 3.82 + | 49.82 | 46.45 ± 1.66 |
| **Problem 3** | ORL | 100.0 | **99.33 ± 0.45** = | 100.0 | 99.27 ± 0.57 = | 100.0 | 99.17 ± 0.62 = | 100.0 | 99.27 ± 0.53 |
| | EYALE | 99.71 | 98.52 ± 1.48 = | **99.80** | 99.04 ± 0.50 + | 99.71 | 98.76 ± 1.25 = | **99.80** | **99.35 ± 0.30** |
| **Problem 4** | KTH | 94.59 | 92.04 ± 1.32 + | 95.08 | 93.58 ± 0.96 = | **96.23** | 93.05 ± 1.87 + | 96.07 | **94.14 ± 1.11** |
| | Outex | 98.96 | 97.73 ± 0.74 + | 99.40 | 98.70 ± 0.54 = | 99.24 | 98.05 ± 0.72 + | **99.45** | **98.76 ± 0.30** |
| **Problem 5** | Office_D | 60.35 | 57.26 ± 1.87 + | 60.64 | 57.85 ± 2.29 = | 60.35 | 57.47 ± 2.40 + | **63.27** | **59.16 ± 2.10** |
| | Office_W | 61.41 | 56.10 ± 2.12 + | 61.09 | 56.80 ± 2.00 = | 60.31 | 56.09 ± 1.88 + | **61.88** | **57.22 ± 1.74** |
| **Problem 6** | COIL_1 | 93.71 | 91.52 ± 1.33 + | **94.68** | 92.40 ± 1.33 = | **94.68** | 92.10 ± 1.08 + | 94.52 | **92.89 ± 0.81** |
| | COIL_2 | 100.0 | 98.84 ± 0.92 + | 100.0 | 98.80 ± 1.25 + | 100.0 | 98.43 ± 1.69 + | 100.0 | **99.75 ± 0.41** |
| **Overall** | | | 9+, 3= | | 3+, 9= | | 8+, 4= | | |

*2) Comparisons with Single-Tasking GP (FGP):* The results in Table III show that KSMTGP achieves significantly better performance than FGP on nine datasets and similar performance to it on three datasets. On these three datasets, the mean test accuracy (%) obtained by KSMTGP is higher than that by FGP, although their performances are not significantly different. Comparing to solving each image classification task individually, jointly solving two similar or related tasks using the proposed KSMTGP approach is more effective. The multi-tree representation and the knowledge sharing mechanism in KSMTGP are the main reasons for the improvement of generalisation performance, which will be further analysed and discussed in the following subsections.

*3) Comparisons with Single-Tasking GP with a Multi-Tree Representation (MTFGP):* Compared with MTFGP, the proposed KSMTGP approach achieves significantly better performance on three datasets and similar performance on six datasets. Specifically, KSMTGP achieves better mean accuracy than MTFGP on ten datasets, although their classification results of the 30 runs are not significantly different. Comparing with MTFGP and FGP, it can be found that a multi-tree representation is more effective than a single-tree representation in GP for feature learning to image classification in most cases. A multi-tree representation allows a GP individual to represent more information about the images or the class distributions. However, a multi-tree representation does not necessarily improve the classification performance of GP due to the issue of overfitting when the number of training instances is small. For example, MTFGP achieves worse results than FGP on the JAFFE, ORL and COIL_2 datasets. The same as MTFGP, KSMTGP also uses an individual of two trees to represent a solution. Differently, KSMTGP learns a common tree from two similar or related tasks as a part of the multi-tree representation to represent more general knowledge/features, which can improve its generalisation performance.

*4) Comparisons with Multifactorial GP (MFFGP):* Table III shows that KSMTGP achieves significantly better performance than MFFGP on eight datasets and similar performance on four datasets. On these 12 datasets, KSMTGP achieves better mean accuracy than MFFGP. KSMTGP also achieves better maximal accuracy than MFFGP on nine datasets. Comparing the results obtained by MFFGP with FGP, it can be found that the classification performance is slightly improved by simultaneously solving two tasks. However, the improvement is not significant. Compared with MFFGP, the KSMTGP approach is more effective for multitask feature learning to image classification. Unlike an optimisation problem, which only needs to optimise the objective/fitness function, the feature learning and image classification problem need to improve the generalisation performance, e.g., the classification performance on the test set, which is vital and different from the objective/fitness function. Therefore, it is necessary to consider how to improve the generalisation performance when dealing with two learning tasks. The KSMTGP approach is able to learn both common and task-specific knowledge to describe features for image classification. The features described by the common tree learned from two tasks is more general, which potentially improves the generalisation performance.

*5) Comparisons with non-GP-based Methods:* Since KSMTGP is proposed for image classification, it is necessary to compare it with existing image classification methods, including different classification methods (i.e. SVM, RF, KNN, SRC, and LDA), the methods using different manually extracted features (i.e. DIF, Histogram, HOG, LBP, and SIFT), CNNs (i.e. LeNet and CNN-5), and deep CNNs with transfer learning (i.e. InceptionV3 and InceptionResNetV2). This section further analyses the performance of KSMTGP by comparing it with these different comparison methods.

The classification results of KSMTGP and the 14 non-GP-based methods are listed in Table IV. KSMTGP achieves significantly better or similar performance in 164 comparisons out of the total 168 ($14 \times 12$) comparisons. On several datasets, such as EYALE, KTH, Outex, Office_D, Office_W, the classification performances of these comparison methods are very low. The KSMTGP approach achieves much higher classification performance than these comparison methods, e.g., 8.69% higher on EYALE, nearly 20% higher on KTH, 11.23% on Outex, 10.18% on Office_D, and 7.76% on Office_W. On a few datasets, some comparison methods such as LeNet and SVM achieve better performance than KSMTGP. However, the performances of these methods vary with the datasets. For example, although LeNet achieves better performance than

TABLE IV
CLASSIFICATION ACCURACY (%) OF KSMTGP AND NON-GP-BASED COMPARISON METHODS ON THE TWELVE IMAGE CLASSIFICATION DATASETS. "+" MEANS KSMTGP IS SIGNIFICANTLY BETTER, "–" MEANS KSMTGP IS SIGNIFICANTLY WORSE, AND "=" MEANS NO SIGNIFICANT DIFFERENCE

| | Mean ± St.dev | Mean ± St.dev | Mean ± St.dev | Mean ± St.dev | Mean ± St.dev | Mean ± St.dev |
|---|---|---|---|---|---|---|
| Methods | **FEI_1** | **FEI_2** | **JAFFE** | **RAFD** | **ORL** | **EYALE** |
| SVM | 78.02 ± 0.12 + | 85.35 ± 0.12 + | **64.94 ± 0.30** – | 32.91 ± 0.63+ | 97.00 ± 0.00 + | 81.66 ± 0.09 + |
| RF | 86.56 ± 1.33 + | 82.47 ± 1.04 + | 50.12 ± 1.12 + | 24.61 ± 0.27 + | 96.28 ± 0.63 + | 74.35 ± 0.36 + |
| KNN | 48.67 ± 0.00 + | 48.00 ± 0.00 + | 14.69 ± 0.00 + | 13.36 ± 0.00 + | 82.50 ± 0.00 + | 27.74 ± 0.00 + |
| SRC | 82.00 ± 0.00 + | 84.00 ± 0.00 + | 56.64 ± 0.00 + | 29.82 ± 0.00 + | 92.50 ± 0.00 + | 90.66 ± 0.00 + |
| LDA | 86.67 ± 0.00 + | 85.33 ± 0.00 + | 51.05 ± 0.00 + | 31.05 ± 0.00 + | 97.00 ± 0.00 + | 81.56 ± 0.00 + |
| DIF | 48.67 ± 0.00 + | 54.00 ± 0.00 + | 19.58 ± 0.00 + | 18.44 ± 0.00 + | 83.50 ± 0.00 + | 19.23 ± 0.00 + |
| Histogram | 53.33 ± 0.00 + | 48.00 ± 0.00 + | 18.88 ± 0.00 + | 13.14 ± 0.00 + | 91.50 ± 0.00 + | 7.53 ± 0.00 + |
| HOG | 65.33 ± 0.00 + | 69.33 ± 0.00 + | 34.97 ± 0.00 + | 15.07 ± 0.00 + | 86.00 ± 0.00 + | 19.67 ± 0.00 + |
| LBP | 64.00 ± 0.00 + | 54.00 ± 0.00 + | 25.87 ± 0.00 + | 15.28 ± 0.00 + | 90.00 ± 0.00 + | 39.33 ± 0.00 + |
| SIFT | 86.67 ± 0.00 + | 86.67 ± 0.00 + | 55.94 ± 0.00 + | 23.49 ± 0.00 + | 98.50 ± 0.00 + | 67.95 ± 0.00 + |
| LeNet | **91.78 ± 1.71** – | **88.42 ± 1.49** = | 62.26 ± 3.67 = | 40.85 ± 4.20 + | 82.22 ± 2.78 + | 55.37 ± 3.41 + |
| CNN-5 | 77.64 ± 9.59 + | 79.49 ± 11.56 + | 50.07 ± 4.27 + | 32.09 ± 2.47 + | 92.43 ± 1.68 + | 65.80 ± 1.83 + |
| InceptionV3 | 61.38 ± 4.83 + | 68.20 ± 6.67 + | 51.23 ± 7.88 + | 20.69 ± 4.59 + | 94.88 ± 2.09 + | 66.33 ± 8.78 + |
| InceptionResNetV2 | 50.00 ± 0.00 + | 50.00 ± 0.00 + | 19.14 ± 6.61 + | 15.31 ± 2.44 + | 12.28 ± 20.88 + | 9.86 ± 13.01 + |
| **KSMTGP** | **87.91 ± 3.39** | **88.24 ± 3.46** | **64.13 ± 2.60** | **46.45 ± 1.66** | **99.27 ± 0.53** | **99.35 ± 0.30** |
| Overall | 13+, 1– | 13+, 1= | 12+, 1=, 1– | 14+ | 14+ | 14+ |
| | **KTH** | **Outex** | **Office_D** | **Office_W** | **COIL_1** | **COIL_2** |
| SVM | 33.27 ± 2.53 + | 23.78 ± 0.53 + | 26.10 ± 0.52 + | 34.10 ± 0.30 + | 91.45 ± 0.00 + | 96.15 ± 0.06 + |
| RF | 53.93 ± 0.91 + | 52.61 ± 0.36 + | 42.83 ± 1.16 + | 49.46 ± 0.67 + | 93.97 ± 0.47 – | 98.02 ± 0.36 + |
| KNN | 28.69 ± 0.00 + | 26.43 ± 0.00 + | 18.08 ± 0.00 + | 23.59 ± 0.00 + | 83.55 ± 0.00 + | 84.52 ± 0.00 + |
| SRC | 27.05 ± 0.00 + | 8.41 ± 0.00 + | 21.57 ± 0.00 + | 22.34 ± 0.00 + | 90.00 ± 0.00 + | 96.13 ± 0.00 + |
| LDA | 30.00 ± 0.00 + | 26.67 ± 0.00 + | 23.62 ± 0.00 + | 27.03 ± 0.00 + | 87.90 ± 0.00 + | 96.13 ± 0.00 + |
| DIF | 50.98 ± 0.00 + | 48.62 ± 0.00 + | 29.74 ± 0.00 + | 30.00 ± 0.00 + | 84.52 ± 0.00 + | 96.45 ± 0.00 + |
| Histogram | 37.54 ± 0.00 + | 71.04 ± 0.00 + | 23.03 ± 0.00 + | 23.12 ± 0.00 + | 68.55 ± 0.00 + | 83.39 ± 0.00 + |
| HOG | 42.79 ± 0.00 + | 21.82 ± 0.00 + | 30.61 ± 0.00 + | 25.16 ± 0.00 + | 62.92 ± 0.05 + | 71.29 ± 0.00 + |
| LBP | 74.26 ± 0.00 + | 87.53 ± 0.00 + | 33.53 ± 0.00 + | 37.97 ± 0.00 + | 85.16 ± 0.00 + | 97.58 ± 0.00 + |
| SIFT | 74.75 ± 0.00 + | 38.70 ± 0.00 + | 48.98 ± 0.00 + | 49.06 ± 0.00 + | 92.74 ± 0.00 = | 96.13 ± 0.00 + |
| LeNet | 55.61 ± 4.75 + | 72.86 ± 5.01 + | 33.81 ± 2.53 + | 36.15 ± 1.95 + | **96.02 ± 0.90** – | 95.38 ± 0.95 + |
| CNN-5 | 41.28 ± 6.61 + | 66.08 ± 7.99 + | 40.85 ± 3.10 + | 43.47 ± 1.99 + | 92.77 ± 1.80 = | 95.14 ± 1.09 + |
| InceptionV3 | 55.50 ± 11.60 + | 49.70 ± 16.33 + | 48.04 ± 4.71 + | 48.56 ± 2.54 + | 90.82 ± 1.04 + | 98.16 ± 0.66 + |
| InceptionResNetV2 | 25.65 ± 8.86 + | 13.55 ± 10.83 + | 6.47 ± 6.94 + | 9.83 ± 9.62 + | 36.95 ± 21.87 + | 46.67 ± 22.83 + |
| **KSMTGP** | **94.14 ± 1.11** | **98.76 ± 0.30** | **59.16 ± 2.10** | **57.22 ± 1.74** | **92.89 ± 0.81** | **99.75 ± 0.41** |
| Overall | 14+ | 14+ | 14+ | 14+ | 10+, 2=, 2– | 14+ |

KSMTGP on three datasets (FEI_1, FEI_2, and COIL_2), its performance on the other datasets (e.g., ORL, EYALE, KTH, Outex, Office_D, Office_W) is significantly lower than that of KSMTGP. Compared with these 14 methods, KSMTGP has better adaptability since it is able to achieve better performance on these different types of image classification tasks. The comparisons and analysis indicate that KSMTGP is more effective and adaptive than these representative image classification methods on different types of image classification tasks.

### *B. Training Results and Convergence Behaviours*

The average training results (i.e. fitness values according to Eq. (6)) and the convergence behaviours of the GP-based methods, i.e., FGP, MTFGP, MFFGP, and KSMTGP, of the 30 runs are shown in Fig. 6. The single-task FGP method achieves the best fitness values than the other three methods on five datasets, i.e., JAFFE, ORL, Office_D, Office_W, and COIL_1. The single-task FGP with a multi-tree representation achieves the best training results than the other three methods on four datasets, i.e., FEI_2, RAFD, KTH and Outex. On the COIL_2 dataset, MFFGP achieves the best training results. On the remaining two datasets, i.e., FEI_1 and EYALE, the training results of these methods are very close. Compared with the three GP methods, KSMTGP achieves worse training results on almost all the datasets. However, KSMTGP achieves better testing (generalisation) performance than these three GP-based methods, as shown in Table III. The results show that KSMTGP may be less overfitted to the training set by having a common feature set from the two tasks to perform image classification compared with these three methods.

Figure 6 shows that the four GP-based methods have similar convergence behaviours, although they have different starting points due to the representation or searching mechanism differences. On most of these datasets, the four GP-based methods can converge to high fitness values, i.e., over 90% accuracy. On some datasets, such as JAFFE, RAFD, Office_D, and Office_W, the single-task FGP method performs better than the other three GP methods. Different from these three methods, the best fitness values of the KSMTGP approach are not always increased over the generations. The best fitness values may decrease due to the change of the common tree in the individual representation, which leads to some fluctuations in the convergence curve.

### *C. Computational Time*

The average training (i.e. evolutionary learning) time and the testing time (i.e. classification time) of the four GP-based methods are shown in Figures 7 and 8. Note that the training time is the sum of the training time for the two datasets because MFFGP and KSMTGP solve two tasks

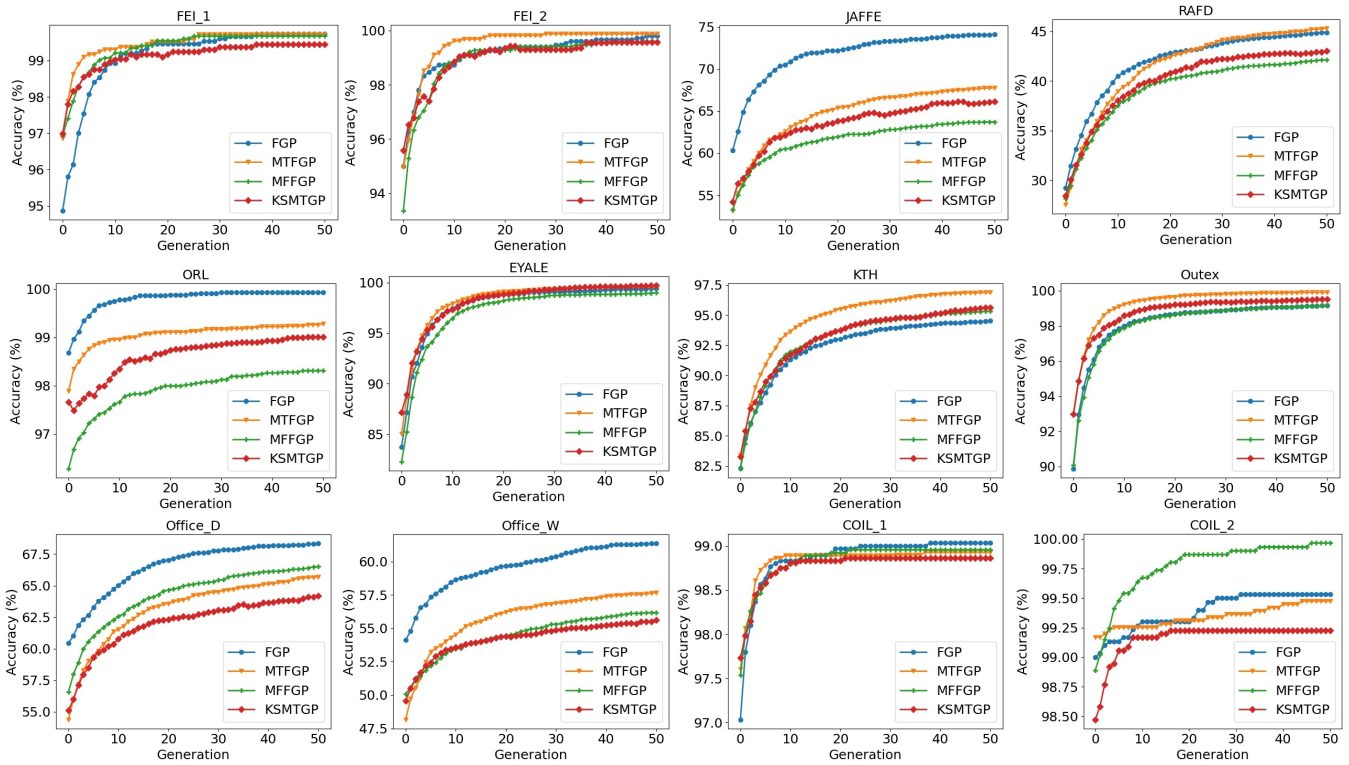



Fig. 6. Convergence behaviours of the FGP, MTFGP, MFFGP, and KSMTGP methods over 50 generations on the 12 image classification datasets.

simultaneously. Compared with FGP and MFFGP, which use one tree, MTFGP and KSMTGP that use two trees as a solution to a task need longer training time on all the datasets. It is reasonable because adding one more tree will increase the time of fitness evaluation. Compared with MTFGP, KSMTGP uses longer training time on four problems (eight datasets) and less training time on two problems (four datasets). The reason may be that KSMTGP evolves some complex trees to achieve good performance. As it can be seen from the testing time in Fig. 8 that KSMTGP uses slightly longer time on classifying the EYALE and Outex datasets. However, since the training of these methods can be offline and the testing process is very fast, the training time is not so important. But the analysis still provides insights on the computational cost of these methods.

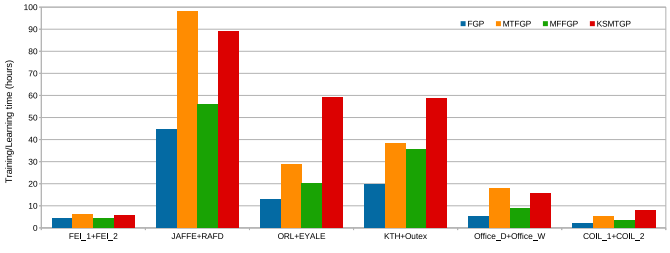



Fig. 7. The average training/learning time (hours) of FGP, MTFGP, MFFGP, and KSMTGP on the six problems of 12 image classification datasets.

The testing/classification time is more important in real-world applications. Although KSMTGP uses longer training time, its overall testing time is short. Except for the RAFD dataset, KSMTGP uses less than four minutes on the 11

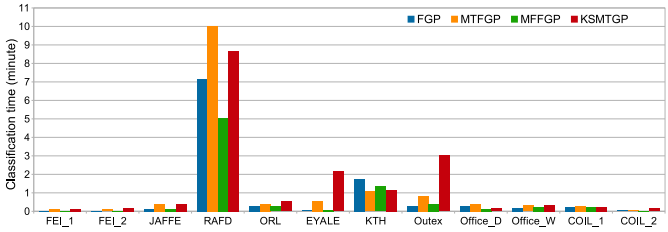



Fig. 8. The average testing/classification time (minutes) of FGP, MTFGP, MFFGP, and KSMTGP on the 12 image classification datasets.

datasets. It also can be found that the testing time of KSMTGP is close to that of the other three GP-based methods. RAFD is a large dataset with 7,240 testing images so that these GP-based methods use longer time. The comparisons show that KSMTGP is fast on most of the datasets in testing even with the use of two evolved trees to extract features for image classification.

### *D. Number of Learned Features*

The average number of features learned by the four GP-based methods are compared in Fig. 9. The four GP-based methods can learn a flexible number of features from different image datasets. The feature number ranges from 200 to 1,200 on different datasets. Compared with FGP and MFFGP, MTFGP and KSMTGP learn a larger number of features for classification. It is reasonable because MTFGP and KSMTGP use two trees to extract features from an image, while FGP and MFFGP use one tree. By learning a larger number of

features, MTFGP and KSMTGP achieve better generalisation performance than FGP and MFFGP, as shown in Table III. However, a large number of features does not necessarily improve the generalisation performance. Compared with MTFGP, KSMTGP learns a smaller number of features on the FEI_2, JAFFE, Office_D, Office_W, COIL_1 datasets but achieves better generalisation performance. On the EYALE, KTH and COIL_2 datasets, KSMTGP learns a larger number of features and achieves better performance than MTFGP. The results show that KSMTGP is able to learn a reasonable number of features to achieve better generalisation performance than the other three GP-based methods.

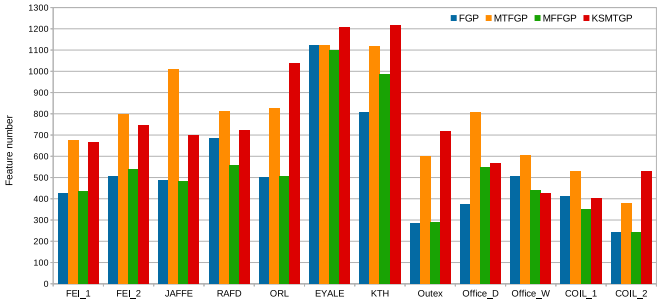



Fig. 9. The average number of features learned by FGP, MTFGP, MFFGP, and KSMTGP on the 12 image classification datasets.

## VI. Further Analysis

This section further analyses the trees, including the common and task-specific trees, evolved by the KSMTGP approach to show its effectiveness and generalisability.

### A. Example Individual Analysis

Two example individuals, i.e., three trees evolved by KSMTGP, on Problem 6, i.e. the COIL_1 and COIL_2 datasets, are used for further analysis. The three trees are a common tree ($\Phi_{com}$) and two task-specific trees ($\Phi_{t1}$ and $\Phi_{t2}$). The $\Phi_{com}$ and $\Phi_{t1}$ trees are for solving the task of classifying COIL_1 and achieve an accuracy of 93.06%. The $\Phi_{com}$ and $\Phi_{t2}$ trees are for classifying COIL_2 and achieve an accuracy of 100%. These trees are listed as follows. The visualisation of them is shown in Fig. 10.

$$\Phi_{t1} = Root2(SIFT(Gau(Lap(Image),4)), SIFT(Gau(ReLU(LoG2(Image)),4))) \quad (7)$$

$$\Phi_{com} = Root2(SIFT(Image), HOG(Image)) \quad (8)$$

$$\Phi_{t2} = Root3(LBP(Image), HOG(Gabor(Sqrt(W\text{-}Sub(Gau(Image,1),0.79,SobelY(Image),0.994)),2,3)), SIFT(LoG1(LBP\text{-}F(Med(Min(Image)))))) \quad (9)$$

The common tree $\Phi_{com}$ is simple and can extract a combination of SIFT and HOG features from an input image ($Image$). The number of extracted features are $128 + 64 = 192$. From the results in Table III, it can be found that the SIFT features perform better than the other features on the COIL_1 and COIL_2 datasets. In addition, the HOG features can capture the shape and appearance of the objects in COIl_1 and COIL_2 images. Therefore, the common tree has the $SIFT$ operators to extract a combination of SIFT and HOG features as the shared features across the two tasks.

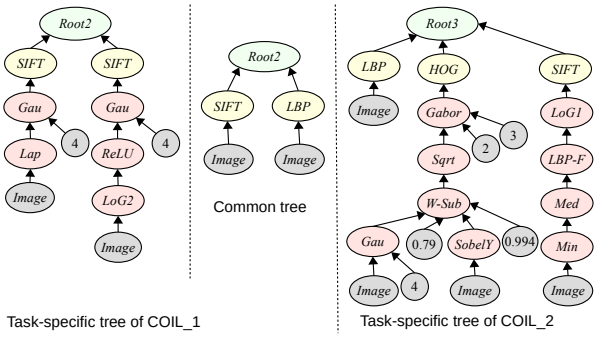



Fig. 10. Visualisation of the three example trees on Problem 6.

The $\Phi_{t1}$ tree is slightly more complex than the common tree and can extract a combination of SIFT features from an input image. Before feature extraction, a number of filtering operations, including Laplacian filtering ($Lap$), Gaussian filtering ($Gau$) and Laplacian of Gaussian filtering ($LoG2$), are used to process the input image. The $\Phi_{t1}$ tree is able to extract 256 ($128\times2$) features from an input image. In total, 448 features are extracted for classifying the COIL_1 dataset. The $\Phi_{t2}$ tree is for classifying the COIL_2 dataset. It can extract a combination of LBP, HOG and SIFT features, i.e., 251 ($59 + 64 + 128$) features. The LBP features are extracted from the original images, and the HOG and SIFT features are extracted from the images processed by different filtering operations, including Gaussian filtering, Gabor filtering ($Gabor$), median filtering, and Laplacian of Gaussian filtering. In total, 443 features are extracted for classifying the COIL_2 dataset.

The analysis shows that KSMTGP learns a simple but effective common tree across two different image classification tasks, which is expected and reasonable since the common tree is to extract features general to both tasks. KSMTGP evolves a complex task-specific tree together with the common tree to effectively solve an individual task. IKSMTGP is able to evolve common and task-specific trees that extract a flexible number of features from images.

### B. Analysis on Common and Task-specific Trees

In this subsection, the performances of the common tree and the task-specific trees on the same dataset are analysed. Furthermore, the common and task-specific trees are transferred to solve other image classification tasks, which are different from the ones they learned from, to further show their generality and transferability.

*1) Classification Performance on the Same Datasets:* The classification performances of the common tree and the task-specific tree are further analysed to show the effectiveness of the new individual representation in KSMTGP. Table V shows the classification results obtained by using these trees from the 30 runs on the datasets where they were learned from.

Specifically, the accuracy is obtained by using one of these two trees to extract features for classifying the same datasets.

Table V shows that the classification performance of the two trees is better than that of the common tree and the task-specific tree on the majority of the datasets. Using two trees can represent more information (i.e., features) of the datasets, which potentially increase the performance. The new search mechanism enables KSMTGP to evolve two trees that perform the best on the training set. Therefore, using the combination of the features extracted by the two trees can achieve better performance than using those by each single of the two trees.

TABLE V
CLASSIFICATION ACCURACY (%) OBTAINED BY TWO TREES, THE COMMON TREE, AND THE TASK-SPECIFIC TREE ON EACH DATASET. TWO TREES ARE A COMMON TREE AND A TASK-SPECIFIC TREE

| | **Two trees** Mean ± St.dev | **Common tree** Mean ± St.dev | **Task-specific tree** Mean ± St.dev |
|---|---|---|---|
| FEI_1 | **87.91 ± 3.39** | 84.36 ± 2.85 | 84.07 ± 6.83 |
| FEI_2 | **88.24 ± 3.46** | 86.78 ± 1.21 | 83.64 ± 7.22 |
| JAFFE | 64.13 ± 2.60 | **64.29 ± 1.84** | 54.41 ± 11.90 |
| RAFD | **46.45 ± 1.66** | 35.06 ± 1.42 | 42.44 ± 3.03 |
| ORL | **99.27 ± 0.53** | 97.82 ± 0.81 | 98.90 ± 1.21 |
| EYALE | **99.35 ± 0.30** | 95.26 ± 0.87 | 99.05 ± 0.60 |
| KTH | **94.14 ± 1.11** | 87.65 ± 2.70 | 91.02 ± 3.46 |
| Outex | **98.76 ± 0.30** | 93.46 ± 1.42 | 96.69 ± 2.48 |
| Office_D | **59.16 ± 2.10** | 56.84 ± 0.99 | 46.99 ± 10.52 |
| Office_W | **57.22 ± 1.74** | 55.26 ± 0.48 | 44.35 ± 8.31 |
| COIL_1 | **92.89 ± 0.80** | 92.58 ± 0.00 | 86.88 ± 10.92 |
| COIL_2 | **99.75 ± 0.41** | 97.42 ± 0.00 | 97.81 ± 2.00 |

Table V shows that the classification performance of the task-specific trees is better than that of the common trees on six datasets, i.e., JAFFE, RAFD, ORL, EYALE, KTH, Outex, and COIL_2. The performances of the common trees and the task-specific trees vary with datasets. An important finding is that the performance of the common trees is more stable than that of the task-specific trees on each dataset because the standard deviation values obtained by the common trees are smaller than those by the task-specific trees. During the evolutionary process, the performance of the common trees is evaluated on the two tasks (in Algorithm 2), while the performances of the task-specific trees are jointly evaluated with the common tree (in Algorithm 3). This leads to the common trees themselves are more effective and general than the task-specific trees.

*2) Classification Performance Across Different Datasets:* The common and task-specific trees learned from one dataset are applied to extract features for classifying another different dataset to further analyse their generality and transferability. Table VI lists the results obtained by these trees transferred from one dataset to another dataset. The results are from the trees of the 30 independent runs. The experiments are conducted across the same type of image classification tasks, i.e., facial expression classification and object classification. In the first column of Table VI, "X→Y" indicates that the trees learned from the X dataset are used to classifying the Y dataset. Note that this section aims to further analyse the performances and transferability of the trees learned by KSMTGP to provide more insights into the new approach.

Comparing the results in Table VI with those in Table IV, it can be found that these transferred GP trees can achieve

TABLE VI
CLASSIFICATION ACCURACY (%) OBTAINED BY APPLYING THE GP TREES LEARNED FROM ONE DATASET TO ANOTHER DIFFERENT DATASET

| Datasets | **Two trees** Mean ± St.dev | **Common tree** Mean ± St.dev | **Task-specific tree** Mean ± St.dev |
|---|---|---|---|
| FEI_1→ JAFFE | 55.66 ± 3.36 | 57.78 ± 1.63 | 50.44 ± 7.36 |
| JAFFE→ FEI_1 | 87.64 ± 3.67 | 87.05 ± 2.75 | 81.35 ± 8.42 |
| FEI_2→ RAFD | 27.50 ± 3.89 | 25.28 ± 1.62 | 23.89 ± 4.59 |
| RAFD→ FEI_2 | 85.69 ± 2.13 | 84.76 ± 2.64 | 84.22 ± 2.40 |
| Office_D→ COIL_1 | 91.67 ± 1.08 | 92.15 ± 0.32 | 89.02 ± 3.19 |
| COIL_1→ Office_D | 49.72 ± 3.28 | 48.69 ± 0.00 | 37.68 ± 11.10 |
| Office_W→ COIL_2 | 99.83 ± 0.35 | 99.86 ± 0.44 | 96.33 ± 9.59 |
| COIL_2→ Office_W | 52.88 ± 5.13 | 50.62 ± 0.00 | 39.62 ± 8.61 |

better performance than most of the comparison methods, such as SRC, RF, KNN, LDA, Histogram, HOG, LBP, SIFT, InceptionV3, and InceptionResNetV2, on these datasets. InceptionV3 and InceptionResNetV2 use the models pretrained on ImageNet to extract features from images for classification. Compared with InceptionV3 and InceptionResNetV2, the trees (i.e., feature extractors) learned by KSMTGP are more effective for solving other tasks by achieving better classification performance on the datasets different from the datasets being used to learn them. This indicates that the trees learned by KSMTGP have good transferability.

The results in Table VI show that the common trees achieve better classification performance than the task-specific trees on most datasets, including the FEI_1, FEI_2, JAFFE, EYALE, KTH, Office_D, and COIL_2 datasets. The common trees are more effective than the task-specific trees when applying them to solve other image classification tasks. This is because the common trees are jointly learned from the two tasks so that they can represent general features that are effective for multiple image classification tasks. The task-specific trees represent features that are specifically effective for a particular dataset. However, with image-related operators and descriptors as internal nodes, the task-specific trees still can represent effective features for image classification, although they are not as effective as the common trees.

To sum up, the analysis shows that using both the common and task-specific trees can achieve better performance than using one of them, individually. Two trees can represent richer information of the datasets to achieve better generalisation performance. The common trees are learned jointly from the two tasks and can represent general but effective features across two different tasks. Further analysis shows the high transferability of both the common and task-specific trees, and the common trees have better transferability than the task-specific trees due to the more general information learned from two image classification tasks.

## VII. CONCLUSIONS

The goal of this paper was to develop a multitask GP approach to feature learning for image classification with limited training data. This goal has been successfully achieved by developing the KSMTGP method with explicit knowledge sharing. A new individual representation, a new evolutionary search process, and corresponding fitness functions were de-

veloped to allow KSMTGP to automatically find the best common tree and task-specific trees that extract informative and effective features for image classification. The performance of KSMTGP was examined on six problems of 12 image classification datasets with limited training data and compared with a large number of competitive methods.

The proposed KSMTGP method achieved better or similar generalisation performance than the single-task GP method, the single-task GP with a multi-tree representation method, the multifactorial GP method, and 14 non-GP-based methods. The results showed that the new individual representation with a common tree and the knowledge sharing mechanism improved the generalisation performance of KSMTGP. Further analysis on example trees showed that KSMTGP learned a simple yet effective common tree for knowledge sharing across two tasks. Compared with the task-specific trees, the common trees achieved better and more stable classification performance. In addition, the common trees have better transferability than the task-specific trees evolved by KSMTGP.

As a starting point, this study shows the potential capability of multitask GP for feature learning in image classification. The proposed GP-based method can discover the shared knowledge in a form of a tree with a variable length and shape by making the best use of the flexible representation of GP. The high transferability of GP tree has also been demonstrated in this study. There is still a large space in this direction for future research. First, this paper focuses on simultaneously solving two tasks. It is worth investigating a new multitask GP method that can solve more than two tasks, simultaneously. Second, this paper treats a multitask learning problem as the same type of image classification tasks and allows GP to determine what to learn for sharing. It is also important to set a multitask learning problem according to the task relatedness. However, designing a task relatedness measure is very challenging because image data are represented by raw pixels and their feature space keeps changing during the feature learning process. In the future, we will investigate these potential directions.

## REFERENCES


[1] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: toward evolutionary multitasking," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 343–357, 2015.

[2] L. Feng, L. Zhou, J. Zhong, A. Gupta, Y.-S. Ong, K.-C. Tan, and A. Qin, "Evolutionary multitasking via explicit autoencoding," *IEEE Trans. Cybern.*, vol. 49, no. 9, pp. 3457–3470, 2018.

[3] J. Ding, C. Yang, Y. Jin, and T. Chai, "Generalized multitasking for evolutionary optimization of expensive problems," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 44–58, 2017.

[4] M. Gong, Z. Tang, H. Li, and J. Zhang, "Evolutionary multitasking with dynamic resource allocating strategy," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 858–869, 2019.

[5] L. Feng, L. Zhou, A. Gupta, J. Zhong, Z. Zhu, K.-C. Tan, and K. Qin, "Solving generalized vehicle routing problem with occasional drivers via evolutionary multitasking," *IEEE Trans. Cybern.*, 2019, DOI: 10.1109/TCYB.2019.2955599.

[6] A. Gupta, Y.-S. Ong, L. Feng, and K. C. Tan, "Multiobjective multifactorial optimization in evolutionary multitasking," *IEEE Trans. Cybern.*, vol. 47, no. 7, pp. 1652–1665, 2016.

[7] J. Lin, H.-L. Liu, B. Xue, M. Zhang, and F. Gu, "Multi-objective multitasking optimization based on incremental learning," *IEEE Trans. Evol. Comput.*, vol. 24, no. 5, pp. 824–838, 2020.

[8] K. K. Bali, Y.-S. Ong, A. Gupta, and P. S. Tan, "Multifactorial evolutionary algorithm with online transfer parameter estimation: MFEA-II," *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 69–83, 2019.

[9] 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.

[10] H. Al-Sahaf, Y. Bi, Q. Chen *et al.*, "A survey on evolutionary machine learning," *J. Roy. Soc. New Zeal.*, vol. 49, no. 2, pp. 205–228, 2019.

[11] 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.

[12] R. Kemker, R. Luu, and C. Kanan, "Low-shot learning for the semantic segmentation of remote sensing imagery," *IEEE Geosci. Remote Sens. Lett.*, vol. 56, no. 10, pp. 6214–6223, 2018.

[13] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020, DOI: 10.1109/TPAMI.2020.2992393.

[14] 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.

[15] 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.

[16] Y. Bi, B. Xue, and M. Zhang, "Genetic programming-based discriminative feature learning for low-quality image classification," *IEEE Trans. Cybern.*, 2021, DOI: 10.1109/TCYB.2021.3049778.

[17] A. Argyriou, T. Evgeniou, and M. Pontil, "Multi-task feature learning," in *Proc. NeurIPS*, 2007, pp. 41–48.

[18] Z. Kang, K. Grauman, and F. Sha, "Learning with whom to share in multi-task feature learning," in *Proc. ICML*, vol. 2, no. 3, 2011, p. 4.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] L. Zhou, L. Feng, K. C. Tan, J. Zhong, Z. Zhu, K. Liu, and C. Chen, "Toward adaptive knowledge transfer in multifactorial evolutionary computation," *IEEE Trans. Cybern.*, 2020. DOI: 10.1109/TCYB.2020.2974100.

[26] X. Zheng, A. K. Qin, M. Gong, and D. Zhou, "Self-regulated evolutionary multitask optimization," *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 16–28, 2019.

[27] J. Zhong, L. Feng, W. Cai, and Y.-S. Ong, "Multifactorial genetic programming for symbolic regression problems," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 50, no. 11, pp. 4492–4505, 2020.

[28] A. Kattan, D. Faiyaz, Y.-S. Ong, and A. Agapitos, "Genetic programming multitasking," in *Proc. SSCI*, 2020, pp. 1004–1012.

[29] Y. Li, X. Tian, T. Liu, and D. Tao, "On better exploring and exploiting task relationships in multitask learning: Joint model and feature learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1975–1985, 2017.

[30] H.-C. Shin, H. R. Roth, M. Gao *et al.*, "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[31] A. Maurer, M. Pontil, and B. Romera-Paredes, "The benefit of multitask representation learning," *J. Mach. Learn. Technol.*, vol. 17, no. 1, pp. 2853–2884, 2016.

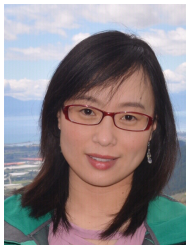
[32] L. M. Antonio and C. A. C. Coello, "Coevolutionary multiobjective evolutionary algorithms: Survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 22, no. 6, pp. 851–865, 2017.

[33] Q. Chen, M. Zhang, and B. Xue, "Feature selection to improve

generalization of genetic programming for high-dimensional symbolic regression," *IEEE Trans. Evol. Comput.*, vol. 21, no. 5, pp. 792–806, 2017.
[34] C. Liu and H. Wechsler, "Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition," *IEEE Trans. Image Process.*, vol. 11, no. 4, pp. 467–476, 2002.
[35] 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.
[36] Z.-H. Zhou and J. Feng, "Deep forest," *Natl. Sci. Rev.*, vol. 6, no. 1, pp. 74–86, 2018.
[37] 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.
[38] 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.
[39] 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.
[40] A. I. Awad and M. Hassaballah, "Image feature detectors and descriptors," *Stud. Comput. Intell.*, Springer, Cham. 2016.
[41] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE CVPR*, vol. 1, 2005, pp. 886–893.
[42] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, 2002.
[43] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
[44] 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.
[45] C. Szegedy, V. Vanhoucke *et al.*, "Rethinking the inception architecture for computer vision," in *Proc. IEEE CVPR*, 2016, pp. 2818–2826.
[46] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. AAAI*, 2017, pp. 4278–4284.
[47] J. Deng, W. Dong, R. Socher *et al.*, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE CVPR*, 2009, pp. 248–255.
[48] C. E. Thomaz, "Fei face database," *online: http://fei.edu.br/˜cet/facedatabase.html*, 2012.
[49] M. Lyons, S. Akamatsu, M. Kamachi *et al.*, "Coding facial expressions with Gabor wavelets," in *Proc. IEEE AFGR*, 1998, pp. 200–205.
[50] O. Langner *et al.*, "Presentation and validation of the radboud faces database," *Cogn. Emot.*, vol. 24, no. 8, pp. 1377–1388, 2010.
[51] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proc. IEEE Workshop Appl. Comput. Vis.*, 1994, pp. 138–142.
[52] 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.
[53] P. Mallikarjuna *et al.*, "The kth-tips2 database," *Computational Vision and Active Perception Laboratory, Stockholm, Sweden*, pp. 1–10, 2006.
[54] T. Ojala, T. Maenpaa, M. Pietikainen, J. Viertola, J. Kyllonen, and S. Huovinen, "Outex-new framework for empirical evaluation of texture analysis algorithms," in *Object recognition supported by user interaction for service robots*, vol. 1. IEEE, 2002, pp. 701–706.
[55] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proc. ECCV*. Springer, 2010, pp. 213–226.
[56] S. A. Nene *et al.*, "Columbia object image library (coil-100)," 1996.
[57] M. Iqbal, B. Xue, H. Al-Sahaf, and M. Zhang, "Cross-domain reuse of extracted knowledge in genetic programming for image classification," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 569–587, 2017.
[58] 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.
[59] F. Chollet *et al.*, "Keras," https://keras.io, 2015.
[60] F.-A. Fortin and et al., "DEAP: Evolutionary algorithms made easy," *J. Mach. Learn. Res.*, vol. 13, no. Jul, pp. 2171–2175, 2012.
[61] F. Pedregosa, G. Varoquaux, and et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

**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.

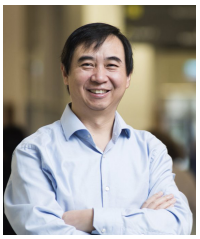
Dr Bi is a member of IEEE CIS and ACM SIGEVO. She has been serving as an organizing committee member of IEEE CEC 2019 and Australasian AI 2018, an organiser of a workshop in ICDM 2021, a special session in SSCI 2021 and a special session in IDEAL 2021, and a program committee member of over ten international conferences including IJCAI, GECCO, IEEE CEC, IEEE SSCI, and Australian AI. 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 neural networks, image analysis, transfer learning, and multi-objective machine learning.

Dr Xue is currently the Chair of IEEE Computational Intelligence Society (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 Vice-Chair IEEE CIS Task Force on Evolutionary Deep Learning and Applications. She has also served as associate editor of over ten 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 Re- search Center, Agricultural University of Hebei, Hebei, China, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 1989, 1992, and 2000, respectively.

He is currently Professor of Computer Science, Head of the Evolutionary Computation Research Group, and the Associate Dean (Research and Innovation) in the Faculty of Engineering at VUW. His current research interests include evolutionary computation, particularly genetic programming, particle swarm optimization, and learning classifier systems with application areas of image analysis, multi-objective optimization, feature selection and reduction, job shop scheduling, and transfer learning. In these areas, he has published over 700 research 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, and chair for the IEEE CIS Emergent Technologies Technical Committee and the Evolutionary Computation Technical Committee, and a member of the IEEE CIS Award Committee. He is a vice-chair of the IEEE CIS Task Force on Evolutionary Feature Selection and Construction, a vice-chair of the Task Force on Evolutionary Computer Vision and Image Processing, and the founding chair of the IEEE Computational Intelligence Chapter in New Zealand. He is also a committee member of the IEEE NZ Central Section.