Genetic Programming for Feature Learning in Image Classification

by

Ying Bi

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2020

Abstract

Image classification is an important and fundamental task in computer vision and machine learning. The task is to classify images into one of some pre-defined groups based on the content in the images. However, image classification is a challenging task due to high variations across images, such as illumination, viewpoint, scale variations, deformation, and occlusion. To effectively solve image classification, it is necessary to extract or learn a set of meaningful features from raw pixels or images. The effectiveness of these features significantly affects classification performance. *Feature learning* aims to automatically learn effective features from images for classification. However, feature learning is difficult due to the high variations of images and the large search space.

Genetic Programming (GP) as an Evolutionary Computation (EC) technique is known for its powerful global search ability and high interpretability of the evolved solutions. Compared with other EC methods, GP has a flexible representation of variable length and can search the solution space without any assumptions on the solution structure. The potential of GP in feature learning for image classification has not been comprehensively investigated due to the use of simple representations, e.g., functions and program structures.

The overall goal of this thesis is to further investigate and explore the potential of GP for image classification by developing a new GP-based approach with a new representation to automatically learning effective features for different types of image classification tasks.

Firstly, this thesis proposes a new GP-based approach with image descriptors to learning global and/or local features for image classification

by developing a new program structure, a new function set, a new terminal set, and a new fitness function. These new designs allow GP to detect small regions from the relatively large input image, extract features using image descriptors from the detected regions or the input image, and combine the extracted features for classification. The results show that the new approach significantly outperforms five GP-based methods, eight traditional methods, and three convolutional neural network methods in almost all the comparisons on eight different datasets.

Secondly, this thesis proposes a new GP-based approach with a flexible program structure and image-related operators for feature learning in image classification. The new approach learns effective features transformed by multiple layers, i.e., a filtering layer, a pooling layer, a feature extraction layer, and a feature concatenation layer, in a flexible way. The results show that the new approach achieves better performance than a large number of effective methods on 12 benchmark datasets. The solutions and features learned by the new approach provide high interpretability.

Thirdly, this thesis proposes the first GP-based approach to automatically and simultaneously learning features and evolving ensembles for image classification. The new approach can learn high-level features through multiple transformations, select effective classification algorithms and optimise the parameters for these classification algorithms to build effective ensembles. The new approach outperforms a large number of benchmark methods on 12 different image classification datasets.

Finally, this thesis proposes a multi-population GP-based approach with knowledge transfer and ensembles to improving both the generalisation performance and computational efficiency of GP-based feature learning algorithms for image classification. The new approach can achieve better generalisation performance and computational efficiency than baseline GP-based feature learning method. The new approach can achieve better performance on 11 datasets than a large number of benchmark methods, including many neural network-based methods.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my dear supervisors, **Prof. Mengjie Zhang** and **A/Prof. Bing Xue**. They have spent dedicated time and efforts to train my research skills and provided constructive and challenging feedbacks to improve my research work. Without their encouragement, motivation, inspiration, and guidance, this thesis would not have been achievable. Prof. Mengjie Zhang has supported me in many ways to develop not only my research skills but also presentation, teaching and other important skills. His wealth of knowledge in many aspects of artificial intelligence and expertise in genetic programming has helped me to shape this thesis, inspired me with many new ideas and developed my critical thinking. My special appreciations go in particular to A/Prof. Bing Xue for her valuable and insightful suggestions for my research work and always nice discussions with persistent encouragement. I am also grateful to her for dedicated time and detailed comments that helped me improve my research writing skills.

I would like to thank my dear friends for their friendship and support, which relax me from the stressful PhD life overseas. I am also grateful for friends and colleagues in the Evolutionary Computation Research Group (ECRG) for valuable suggestions and comments to improve my research and creating such an active research environment. I would like to thank all the members in the subgroup, Feature Analysis, Selection, and Learning (FASLIP), for the fruitful research discussions and informative knowledge sharing, which enriched my knowledge and inspired me a lot to complete

this thesis. I am also grateful to the staff and colleagues who have helped me over the past three years at Victoria University of Wellington (VUW).

I wish to thank my family for their understanding, consistent support and encouragement throughout my study. Thank you to my beloved parents and grandparents for their unconditional love and care. I am deeply grateful to my beloved husband Qie for his love, patience, support, and accompany. You have always been the source of love and happiness that encouraged me to complete this long journey.

I want to express my gratitude to the joint scholarship of Victoria University of Wellington and China Scholarship Council for the financial support for my PhD study. I would like to thank the School of Engineering and Computer Science (ECS) and Victoria University of Wellington (VUW) for the resources required to complete my thesis. Additionally, I would like to thank New Zealand eScience Infrastructure (NeSI), who provides sufficient computing resources to conduct computationally expensive experiments required by the development of this thesis.

Finally, I wish to thank all the reviewers and examiners who have spent the time to read my thesis and provided useful comments and suggestions. Thank you to those who have not been mentioned above but helped me or enriched my life.

List of Publications

1. **Ying Bi**, Bing Xue, and Mengjie Zhang. "Genetic Programming with Image-Related Operators and A Flexible Program Structure for Feature Learning in Image Classification". *IEEE Transactions on Evolutionary Computation*. 15pp, 2020. DOI: 10.1109/TEVC.2020.3002229. (15 June 2020 published online).

2. **Ying Bi**, Bing Xue, and Mengjie Zhang. "Genetic Programming with A New Representation to Automatically Learn Features and Evolve Ensembles for Image Classification". *IEEE Transactions on Cybernetics*. 15pp, 2020. DOI: 10.1109/TCYB.2020.2964566. (30 Jan 2020 published online).

3. **Ying Bi**, Bing Xue, and Mengjie Zhang. "An Effective Feature Learning Approach Using Genetic Programming with Image Descriptors for Image Classification". *IEEE Computational Intelligence Magazine*. vol. 15, no. 2, 2020. pp. 65-77.

4. Bo Peng, Shuting Wan, **Ying Bi**, Bing Xue, and Mengjie Zhang. "Automatic Feature Extraction and Construction Using Genetic Programming for Rotating Machinery Fault Diagnosis". *IEEE Transactions on Cybernetics*. 15 pp, 2020. DOI: 10.1109/TCYB.2020.3032945 (25 Nov 2020 published online).

5. **Ying Bi**, Bing Xue, and Mengjie Zhang. "Genetic Programming-Based Discriminative Feature Learning for Low-Quality Image Clas-

sification". Submitted to *IEEE Transactions on Cybernetics*.

6. **Ying Bi**, Bing Xue, and Mengjie Zhang. "A Fast Genetic Programming-Based Feature Learning Approach with Knowledge Transfer and Ensembles for Image Classification". Submitted to *IEEE Transactions on Evolutionary Computation*.

7. **Ying Bi**, Bing Xue, and Mengjie Zhang. "Instance Selection Based Surrogate-Assisted Genetic Programming for Feature Learning in Image Classification". Submitted to *IEEE Transactions on Cybernetics*.

8. **Ying Bi**, Bing Xue, Mengjie Zhang. "Multi-Objective Genetic Programming for Feature Learning in Face Recognition". Submitted to *Applied Soft Computing*.

9. **Ying Bi**, Bing Xue, Mengjie Zhang. "Using a Small Number of Training Instances in Genetic Programming for Face Image Classification". Submitted to *Information Sciences*.

10. Harith Al-Sahaf, **Ying Bi**, Qi Chen, Andrew Lensen, Yi Mei, Yanan Sun, Binh Tran, Bing Xue, and Mengjie Zhang. "A Survey on Evolutionary Machine Learning". *Journal of the Royal Society of New Zealand*. vol. 49, no. 2. 2019. pp. 205-228.

11. **Ying Bi**, Bing Xue, and Mengjie Zhang. "A Survey on Genetic Programming to Image Analysis". *Journal of Zhengzhou University (Engineering Science)*. vol. 39, no. 06. 2018. pp. 3-13. (In Chinese).

12. **Ying Bi**, Bing Xue, and Mengjie Zhang. "An Automated Ensemble Learning Framework Using Genetic Programming for Image Classification". In *Proceedings of 2019 Genetic and Evolutionary Computation Conference (GECCO 2019)*. ACM Press. Prague, Czech Republic. 13-17 July 2019. pp. 365-373.

13. **Ying Bi**, Bing Xue, Mengjie Zhang. "Evolving Deep Forest with Automatic Feature Extraction for Image Classification Using Genetic Programming". In *Proceedings of The Sixteenth International Conference on Parallel Problem Solving from Nature (PPSN 2020)*, Lecture Notes in Computer Science. Vol. 12269. Springer. Leiden, The Netherlands, 5-9 September 2020. pp. 3-18.

14. **Ying Bi**, Bing Xue, and Mengjie Zhang. "Automatically Extracting Features for Face Classification Using Multi-Objective Genetic Programming". In *Proceedings of 2020 Genetic and Evolutionary Computation Conference (GECCO) Companion*. ACM Press. Cancun, Mexico. 8-12 July 2020. pp 117–118.

15. **Ying Bi**, Bing Xue, and Mengjie Zhang. "Genetic Programming-Based Feature Learning for Facial Expression Classification". In *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*. Glasgow, UK. 19-24 July 2020. pp 1-8.

16. **Ying Bi**, Bing Xue, and Mengjie Zhang. "An Evolutionary Deep Learning Approach Using Genetic Programming with Convolution Operators for Image Classification". In *Proceedings of 2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE Press. Wellington, New Zealand. 10-13 June 2019. pp. 3197-3204.

17. **Ying Bi**, Mengjie Zhang, and Bing Xue. "Genetic Programming for Automatic Global and Local Feature Extraction to Image Classification". In *Proceedings of 2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE Press. Rio de Janeiro, Brazil. 8-13 July 2018. pp. 1-8. (**Nominated for Best Student Paper Award**)

18. **Ying Bi**, Bing Xue, Mengjie Zhang. "A Gaussian Filter-Based Feature Learning Approach Using Genetic Programming to Image Classification". In *Proceedings of the 31st Australasian Joint Conference on Artificial Intelligence (AI2018)*, Lecture Notes in Computer Science. vol.

11320. Springer. Wellington, New Zealand. 11-14 December 2018. pp. 251-257.

19. **Ying Bi**, Bing Xue, Mengjie Zhang. "An Automatic Feature Extraction Approach to Image Classification Using Genetic Programming". In *Proceeding of the 21st European Conference on Applications of Evolutionary Computation (EvoApplications 2018)*. Lecture Notes in Computer Science. Parma, Italy. 4-6 April 2018. pp. 421- 438.

20. **Ying Bi**, Mengjie Zhang and Bing Xue. "An Automatic Region Detection and Processing Approach in Genetic Programming for Binary Image Classification". In *Proceedings of 2017 the 32ed International Conference on Image and Vision Computing New Zealand (IVCNZ 2017)*. IEEE Press. Christchurch, New Zealand. 4-6 December 2017. pp. 1-6.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter presents the introduction of the thesis, including the problem statement, the motivations, the research goals, the major contributions, and the overall organisation of the thesis.

1.1 Problem Statement

In computer vision and machine learning, image classification is an important and fundamental task of assigning images to one of pre-defined groups based on the content in the images. Image classification has a wide range of applications in various fields, such as biological images [166], facial images [104], remote sensing images [148], and medical images [23]. Image classification is also a key component of other visual tasks, such as object detection, object recognition, video annotation, and image retrieval [27, 173]. However, image classification is a challenging task due to high variations across images, such as background, illumination, viewpoint, scale, deformation, and occlusion variations.

Feature extraction is an important step of an image classification system. Feature extraction is the process of extracting important and representative information from an image that describes the image data and reduces the dimensionality of the image data. The effectiveness of the

image features significantly affects the performance of the classification system [99]. However, due to the high image variations, it is difficult to obtain a set of effective features for classification. Many image descriptors have been developed to describe images or keypoints of images as features, e.g., Grey-Level Co-occurrence Matrix (GLCM) [96, 97], Histogram of Oriented Gradients (HOG) [60], Local Binary Patterns (LBP) [169], and Scale-Invariant Feature Transform (SIFT) [147]. However, domain knowledge are often required to design such an effective method for feature extraction when solving a new task.

Feature learning, also known as representation learning, aims to learn effective representations that capture useful and underlying information of the data to build classifiers or other predictors for solving a task [33]. The representation of the data often refers to raw data or a number of features that describe the domain. Feature learning is a general term that emphasises the change/optimisation of the representation of the data using a learning algorithm to solve a task. During the feature learning process, a number of operations related to features, e.g., feature extraction, feature construction and feature selection, may be involved to find the optimal representation of the data for a task. In image classification, feature learning techniques can identify and extract effective features from raw images via a learning process without domain knowledge. Figure 1.1 shows the overall process of feature learning for image classification. The feature space search may include the feature extraction, feature selection or feature construction process that is able to find the best representation (feature set). A measure, e.g., the classification performance, is used to evaluate the effectiveness (goodness) of the learned features and guide the search in the feature learning process. After finding the best features (the feature learning process is terminated), a classifier can be trained and used to classify images in the test set.

Feature learning is a difficult task due to the high variations across images and the large search space. Many algorithms have been developed

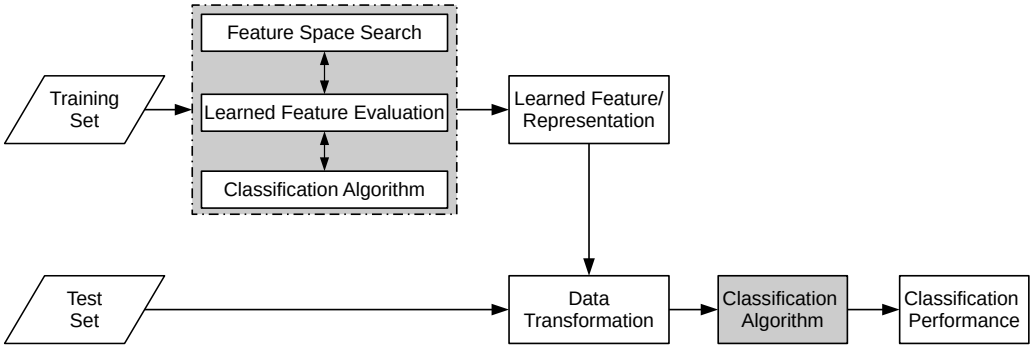


Figure 1.1: Overall process of feature learning for image classification.

for image classification with simultaneous and automatic feature learning. Representative methods are Neural Network (NN)-based methods, e.g., Convolutional Neural Networks (CNNs), [100, 180], Auto-Encoders (AEs) [183], and Restricted Boltzmann Machine (RBM) [69]. However, most of these methods have limitations, i.e., they require a large number of training instances, have fixed model complexity or need rich expertise to design the architectures. In addition, the learned solutions/models/features are often difficult to understand and interpret.

Genetic Programming (GP) is an Evolutionary Computation (EC) technique that is able to automatically evolve computer programs for solving a problem without using extensive domain knowledge [121]. Compared with other EC techniques, e.g., Particle Swarm Optimisation (PSO), Genetic Algorithms (GAs) and Differential Evolution (DE), GP has a more flexible representation of variable length and can search the solution space without any assumptions on the solution structure. In addition, the tree-based solutions of GP often have high interpretability and understandability, providing insights into why it can achieve good performance. Existing GP-based algorithms have achieved promising results in image classification [11, 16, 138, 197] and many other image-related tasks, such as image segmentation [141, 142], edge detection [90] and salient object detection [3]. However, the potential of GP has not been systematically investi-

gated in feature learning for image classification. The majority of existing GP methods use very simple functions and program structures, thus their performance is limited [8, 164, 190, 200, 241, 242]. These limitations require further investigation and development of individual representations, program structures and functions in GP to explore its potential in feature learning for various types of image classification tasks.

1.2 Motivations

1.2.1 Challenges of Image Classification

Image classification is a fundamental task in computer vision. However, it is very difficult due to the following challenges:

1. **High-dimensional data.** Image data often have very high dimensionality, which causes the classification based on the pixel values to be very challenging. For instance, a 100×100 grey-scale image has 10,000 pixel values with integers between $[0, 255]$, which is high-dimensional data. This raises an important and challenging problem of how important information is learned/extracted to represent the image data and reduce its dimensionality. Another potential challenge is that dealing with high-dimensional data may lead to high computational cost, which needs to be addressed.

2. **Image variations.** In the real world, images are often sampled in different environments under various conditions, which contain a large number of variations, including viewpoint, scale, illumination variations, as well as deformations and occlusions. The large variations require the image classification system to have high generalisation ability and robustness. In addition, image variations vary with the datasets and the types of image classification tasks. For solving different image classification tasks, it is often necessary to develop a

method that can deal with various types of variations, which is challenging.

3. **Domain knowledge requirement.** To effectively solve image classification, a number of processes/operations may be needed, e.g., image processing, keypoint detection, feature extraction, and feature selection. Many of these operations require domain knowledge in order to obtain a set of effective features for classification. The classification performance relies highly on the quality of the extracted/selected features. However, domain experts are not always at hand. It takes time to find the experts and they are costly to employ [15].

4. **Flexibility and adaptability.** Most existing methods are only effective for particular problems (domains), leading to image classification system that lack flexibility and adaptability. For example, a face classification system may not be effective for other object image classification tasks. With poor flexibility and adaptability, it is often necessary to develop a new method for solving a new task. To address this, it is necessary to develop a method with high flexibility and adaptability that can achieve good performance in different types of image classification tasks. However, different types of image classification tasks often involve a wide range of image variations and are more difficult to solve than a single type of image classification task. This challenge requires a feature learning method that can adaptively and dynamically extract different types of domain-specific and effective features for image classification.

1.2.2 Why Genetic Programming (GP)

GP is an EC technique that automatically evolves computer programs to solve problems without extensive domain knowledge. GP has been successfully applied to image analysis, including edge detection [90], feature-ground segmentation [142], texture analysis [11], and image feature ex-

traction/description [15]. The following aspects demonstrate why GP is good for image classification.

1. **Flexible representation.** Compared with other EC techniques, such as PSO, GAs and DE, where the lengths of the representation/genotype are usually fixed, the tree-based GP method has a more flexible representation. The flexible representation allows GP to evolve solutions with variable lengths/depths for solving a problem. Compared with other well-known image classification algorithms, such as CNN-based methods [126, 239, 199, 100], where the model architecture/complexity is pre-defined, GP can evolve solutions with variable sizes/depths.

2. **Constraints and different data types.** As a commonly used and enhanced version of GP, Strongly Typed GP (STGP) [161] or Grammatically-based GP [225] is able to evolve programs with functions or terminals that operate on appropriate data types. These GP methods are easy to implement and can manipulate multiple data types, which is very suitable for solving a relatively complex problem by employing different types of functions/operators, such as arithmetic functions, logic functions, filtering operators, and edge detectors, in their function sets.

3. **Integration of multiple tasks.** GP is able to simultaneously deal with multiple tasks using a single tree representation. For instance, GP is able to simultaneously deal with region detection, feature extraction, feature construction, and image classification [13, 14, 27, 138]. Instead of dealing with each task independently and separately, GP has the potential to find solutions for each task and automatically assemble these solutions into a solution to the original problem [122].

4. **Learning ability.** Compared with traditional image feature extraction methods, GP has the capability to learn features from raw images and automatically adjust the learned features to the classifiers

to achieve better classification performance [11, 16]. GP is able to effectively find the best solution from a large search space to a problem/task.

5. **Interpretability (understandability).** The solutions evolved by GP have potentially high interpretability. Unlike other image classification techniques, such as deep NNs and CNNs, where the mapping from inputs to outputs looks like a "black-box", the programs/solutions of GP are usually expressed as syntax trees with functions and variables and can often be translated into a set of symbolic rules. A GP tree can be interpreted and analysed to provide valuable insights into how it solves the problem and why it achieves good performance.

6. **Building blocks/knowledge discovery.** GP is able to automatically form building blocks during the evolutionary process. The building blocks represent the knowledge discovered from the problem by learning. The interrelationships among the relevant variables and operators are often included in the building blocks. Such building blocks can be extracted and transferred as useful knowledge to improve the learning performance of solving other related/similar problems.

1.2.3 Limitations of Existing Work

In recent decades, many image classification tasks have been successfully solved by deep learning methods. Among all the deep learning methods, CNNs have become the leading method for image classification with impressive performance [100, 126, 199, 239]. Deep CNNs have hundreds or thousands of layers of non-linear transformation processors to learn an effective representation of the input data to perform image classification [134, 180]. However, deep CNN methods have the following limitations. First, deep models with a huge number of trainable parameters require a large number of labelled training instances/images to train. In many

domains, such as medicine, biology and remote sensing, the number of labelled instances is often limited due to the high cost of collecting data. Second, deep models often have poor interpretability due to the "black-box" mapping from inputs to outputs. It is difficult to explain and understand what features are extracted/learned and why they are effective for classification. Third, rich domain expertise is often required to design an effective architecture for CNN [248]. The architecture of CNN is a key factor to achieve promising results as it can be found from [100, 126, 199, 239] that CNNs with different architectures perform differently. Fourth, the model complexity of CNNs is fixed, indicating that flexibility and adaptability is limited. Due to these limitations, it is still necessary to explore and develop new approaches to image classification.

GP has been successfully applied to learn features from images for classification [13, 14, 16, 27, 138, 197]. Based on the number of learned features, existing GP-based feature learning methods can be broadly classified into two groups, i.e., learning one feature [13, 14, 27, 138] and learning multiple features [10, 16, 197]. It is also noted that GP has been applied to evolve CNNs for image classification, such as in [64, 206, 250], but these methods use CNNs for feature learning rather than GP. Therefore, these methods are not included in any of these two groups. The methods in the first group can integrate multiple tasks, i.e., region detection, feature extraction and feature construction, into a single tree to produce one high-level feature [13, 14, 27, 138]. The learned single feature is often used to build a simple classifier to perform a binary classification task. The limitation is that these methods cannot be directly employed to solve a multi-class classification problem, which many image classification tasks belong to. The methods in the second group often learn a number of features from images and use traditional classification algorithms, e.g., k-Nearest Neighbour (KNN) and Support Vector Machines (SVMs), to perform classification based on the learned features [10, 16, 197]. However, most of these methods have simple functions and program structures and have only been examined on a

limited number of image classification tasks. Due to these limitations, the potential of GP in image feature learning has not been extensively and systematically investigated. In addition, the GP-based feature learning methods have seldom been examined on large benchmark image classification datasets and compared with state-of-the-art algorithms. These limitations motivate this thesis to develop and further explore a new GP-based approach to feature learning for image classification.

Existing image-related operators, such as filtering operators (e.g., Gaussian filter, derivatives of Gaussian, Laplacian filter, Laplacian of Gaussian filter, Gabor filter, and Sobel edge detector), pooling operators (e.g., max-pooling) and image descriptors (e.g., GLCM, LBP, HOG, and SIFT), are able to transform image pixels and/or detect/describe informative features from images. These operators have been widely used in image preprocessing and feature extraction. Employing image-related operators in GP is effective for learning domain-specific features to achieve promising classification performance [138, 197]. GP has a flexible representation, enabling it to have many possible ways to combine/employ these operators to achieve effective feature learning [27, 138, 197]. Existing works have not comprehensively and systematically investigated the use of different image-operators in GP to achieve effective image feature learning. To address this, it is necessary to investigate a new individual representation, a new function set, a new terminal set, and a new fitness function for GP.

Ensemble methods have been widely used for classification and can achieve better generalisation performance than using a single classifier [91, 237]. Traditional ensemble methods for image classification often include the processes of feature extraction, base learners/classifier selection, training and combination [65]. The key process is to extract a set of informative features from the images, but domain knowledge is often required. In addition, traditional methods often separately perform these processes, which may limit the performance of the constructed ensemble. Automating the processes of feature learning and ensemble learning is able to ad-

dresses these limitations. GP has been applied to construct ensembles for classification [75] or automatically learn features for image classification [197]. However, no existing work combines both in GP to achieve automatically and simultaneously learn features and evolve ensembles for image classification.

When the number of training instances is large, the computational cost of GP-based feature learning algorithms is often very high. However, many well-known image classification tasks have a large number of training instances, e.g., MNIST [136] has 60,000 training images. To allow GP to achieve feature learning on such large datasets, improving its efficiency is very important. However, very few works have successfully addressed this issue. Using a small number of instances intuitively reduces the computational cost. However, unless the used small dataset can cover a wide range of image variations, this may lead to poor generalisation performance on unseen data. To improve both computational efficiency and generalisation performance, it is necessary to develop a new GP-based feature learning approach for image classification by addressing this limitation.

1.3 Goals

The overall goal of this thesis is to investigate the potential capability of GP for feature learning in image classification by developing a new GP-based approach to automatically learning effective features for different types of image classification tasks. To achieve this overall goal, the following four objectives have been established to guide the research.

1. Develop a new feature learning approach using GP to automatically select and combine existing *image descriptors* to extract rich and discriminative *global and/or local* features for different image classification tasks. This approach is expected to evolve solutions with high interpretability that can extract effective features to achieve better

classification performance than other GP-based methods, traditional methods and CNN-based methods.

Existing image descriptors, such as HOG [60], SIFT [147] and LBP [167, 168], are well-developed and quite effective for dealing with certain image variations. With a flexible tree-based representation, it is possible to employ them in GP to be automatically selected and combined/integrated to learn high-level features. However, this use of these effective descriptors in GP to achieve feature learning has not been fully investigated. In general, there are two types of features, i.e., global features extracted from the whole image and local features extracted from the regions of interest. Most of the existing methods extract one type of features, i.e., global features or local features, which are not flexible and effective for solving different types of image classification tasks.

2. Develop a new GP-based approach with *a flexible program structure* and *image-related operators* for feature learning in image classification. This approach is expected to learn various types and numbers of image features that can achieve better classification performance than a large number of competitive methods on different benchmark datasets, including large benchmark datasets.

   Existing image-related operators have been employed as GP functions to evolve solutions that extract high-level features from images, including filtering operators [197]. However, the filtering operators and the image descriptors have not been simultaneously investigated in GP to effectively use them to achieve feature learning. To effectively use these operators in GP, a program structure is typically required to integrate different functions and terminals into a single tree. Existing GP program structures, e.g., multi-tier [27], two-tier [14] and multi-layer [197], are often restricted in a certain way and cannot be used when new image-related operators are introduced in

GP.

3. Develop a GP-based approach with a new representation to automatically and simultaneously *learn features and evolve ensembles* for image classification. This approach is the first approach that uses GP to automatically construct ensembles from raw images for classification. This approach is expected to extract informative features, select effective classification algorithms and optimise key parameters of the classification algorithms to construct an effective ensemble with high diversity that is able to achieve better image classification performance than existing methods and the GP method proposed in the second objective.

   Ensemble methods have been widely used for solving classification problems [91, 237]. An ensemble of classifiers often achieves better generalisation performance than a single classifier in classification problems [247]. Using ensemble methods to solve image classification, a key process is to extract a set of informative features to build a set of classifiers that are individually good and diverse overall. GP has been used to evolve ensembles for classification [75] or automatically learn features for image classification [11, 197]. However, no existing work combines both in GP to automatically and simultaneously learn features and evolve ensembles for image classification. With a flexible tree-based representation, it is possible to integrate the processes of feature learning and ensemble learning into a single GP tree to automatically evolve effective solutions for image classification from raw images.

4. Develop a *multi-population* GP-based approach with *knowledge transfer and ensembles* to achieve *fast* feature learning for effective image classification. This approach is expected to improve both the effectiveness and efficiency of the GP-based feature learning algorithm proposed in the second objective on different types of image classifi-

cation tasks. This approach is also expected to achieve better classification performance than a large number of existing algorithms and the GP methods proposed in the second and third objectives.

Many well-known image classification tasks have a large number of training instances, e.g., MNIST [136] has 60,000 training images. Learning features from such a large dataset using GP-based feature learning algorithms is computationally expensive. Therefore, improving the efficiency of the GP-based feature learning algorithms is important and necessary. However, very few works have successfully addressed this issue. A multi-population algorithm framework is expected to use multiple small populations to perform feature learning from small subsets of the original training set, respectively, which is able to reduce the computational cost. Knowledge transfer can be used to extract and reuse useful knowledge across populations to improve the learning performance of the algorithm. However, very few works of knowledge transfer in GP-based feature learning algorithms have been proposed. The generalisation performance can be further improved by creating an ensemble for image classification.

1.4 Major Contributions

This thesis makes the following contributions.

1. This thesis shows how existing effective image descriptors are integrated into GP to achieve automatic and simultaneous global and/or local feature learning for image classification. This thesis develops a new GP-based approach with a new program structure, a new function set, including five representative image descriptors in the global and local scenarios, and a new terminal set. The new approach can achieve region detection, feature extraction and feature

combination, simultaneously and automatically. A new fitness evaluation process is developed to improve the generalisation ability of the learned features. The results demonstrate that the proposed approach achieves significantly better performance than the other GP-based methods, traditional methods using effective features and the CNN-based methods on different types of image classification tasks.

Part of this contribution has been published in:

- [40] Ying Bi, Bing Xue, and Mengjie Zhang. "An Effective Feature Learning Approach Using Genetic Programming with Image Descriptors for Image Classification". *IEEE Computational Intelligence Magazine*, vol. 15 , no. 2, 2020. pp. 65-77.
- [44] Ying Bi, Mengjie Zhang, and Bing Xue. "Genetic Programming for Automatic Global and Local Feature Extraction to Image Classification". In *Proceedings of IEEE Congress on Evolutionary Computation*. IEEE Press. Rio de Janeiro, Brazil. 8-13 July 2018. pp. 1-8. (**Nominated for the Best Student Paper Award**).

2. This thesis demonstrates how different types of image-related operators, i.e., filtering, pooling, and image descriptors, can be integrated into GP to achieve feature learning for image classification. This thesis develops a new GP-based approach with a new flexible program structure, a new function set and a new terminal set. The flexible program structure is constructed by an input layer, filtering layers, pooling layers, a feature extraction layer, a feature concatenation layer, and an output layer. The new function set has a large number of image-related operators. With these designs, the new approach can learn three types of features, i.e., features from filtering and pooling, features from feature extraction (by image descriptors), and the combination of features from filtering/pooling and feature extraction. The proposed approach can be easily applied to solve different image classification tasks by learning effective features. The

experimental results on 12 benchmark datasets of varying difficulty show that the proposed approach achieves better performance than a number of effective algorithms.

Part of this contribution has been published in:

- [43] Ying Bi, Bing Xue, and Mengjie Zhang. "Genetic Programming with Image-Related Operators and a Flexible Program Structure for Feature Learning in Image Classification". *IEEE Transactions on Evolutionary Computation*. 15pp, 2020. DOI: 10.1109/TEVC.2020.3002229 (15 June 2020 published online).
- [35] Ying Bi, Bing Xue, Mengjie Zhang. "An Automatic Feature Extraction Approach to Image Classification Using Genetic Programming". In *Proceeding of the 21st European Conference on Applications of Evolutionary Computation (EvoApplications 2018)*. Lecture Notes in Computer Science. Parma, Italy. 4-6 April 2018. pp. 421- 438.
- [36] Ying Bi, Bing Xue, Mengjie Zhang. "A Gaussian Filter-Based Feature Learning Approach Using Genetic Programming to Image Classification". In *Proceedings of the 31st Australasian Joint Conference on Artificial Intelligence (AI2018)*, Lecture Notes in Computer Science. vol. 11320. Springer. Wellington, New Zealand. 11-14 December 2018. pp. 251-257.
- [39] Ying Bi, Bing Xue, and Mengjie Zhang. "An Evolutionary Deep Learning Approach Using Genetic Programming with Convolution Operators for Image Classification". In *Proceedings of 2019 IEEE Congress on Evolutionary Computation (CEC 2019)*. IEEE Press. Wellington, New Zealand. 10-13 June 2019. pp. 3197-3204.

3. This thesis develops a GP-based approach with a new representation to automatically and simultaneously learn features and evolve

ensembles for image classification. A new individual representation, a new function set, including image-related operators, classification functions and combination functions, and a new terminal set, including key parameters for the classification functions, are developed. The new approach can automatically learn high-level features through multiple transformations, and select various classification algorithms and their corresponding parameters to construct an effective ensemble. The diversity issue when building the ensembles can be automatically addressed by the proposed approach using the strategies of input feature manipulation and learning parameter manipulation. The proposed approach significantly outperforms a large number of benchmark methods, including deep learning methods, on many image classification datasets of varying difficulty. This is the first work using GP to construct ensembles for classification from raw images.

Part of this contribution has been published in:

- [42] Ying Bi, Bing Xue, and Mengjie Zhang. "Genetic Programming with A New Representation to Automatically Learn Features and Evolve Ensembles for Image Classification". *IEEE Transactions on Cybernetics*. 15pp, 2020, DOI: 10.1109/ TCYB.2020.2964566. (30 Jan 2020 published online).
- [38] Ying Bi, Bing Xue, and Mengjie Zhang. "An Automated Ensemble Learning Framework Using Genetic Programming for Image Classification". In *Proceedings of 2019 Genetic and Evolutionary Computation Conference (GECCO 2019)*. ACM Press. Prague, Czech Republic. 13-17 July 2019. pp. 365-373.
- [41] Ying Bi, Bing Xue, Mengjie Zhang. "Evolving Deep Forest with Automatic Feature Extraction for Image Classification Using Genetic Programming". In *Proceedings of The Sixteenth International Conference on Parallel Problem Solving from Nature (PPSN*

2020), Lecture Notes in Computer Science. Vol. 12269. Springer. Leiden, The Netherlands, 5-9 September 2020. pp. 3-18.

4. This thesis develops a new GP-based feature learning algorithm with multiple populations, knowledge transfer and ensembles to improve both the effectiveness and efficiency for image classification. A new multi-population algorithm framework is developed to use multiple small populations to learn features from small subsets of the training set, respectively. With this design, this computational cost is reduced theoretically and empirically. A new knowledge transfer method is developed to improve the learning performance of multiple small populations. A new fitness function is proposed to evaluate the individuals by providing more accurate information on how the classifier performs using the learned features. A new combination strategy is introduced to combine the best solutions found by the proposed approach to create an effective ensemble for classification. Experimental results show that the proposed approach achieves better classification results and reduces the computational cost than the baseline GP-based feature learning algorithm. The comparisons with state-of-the-art algorithms show that the proposed approach achieves better or comparable performance in almost all the comparisons.

   Part of this contribution has been published in:

   - Ying Bi, Bing Xue, and Mengjie Zhang. A Fast Genetic Programming-Based Feature Learning Approach with Knowledge Transfer and Ensembles for Image Classification. Submitted to *IEEE Transactions on Evolutionary Computation*. June 2020.

1.5 Organisation of Thesis

The structure of the rest of this thesis is illustrated in Figure 1.2. Chapter 2 introduces the background and related work. The main contributions

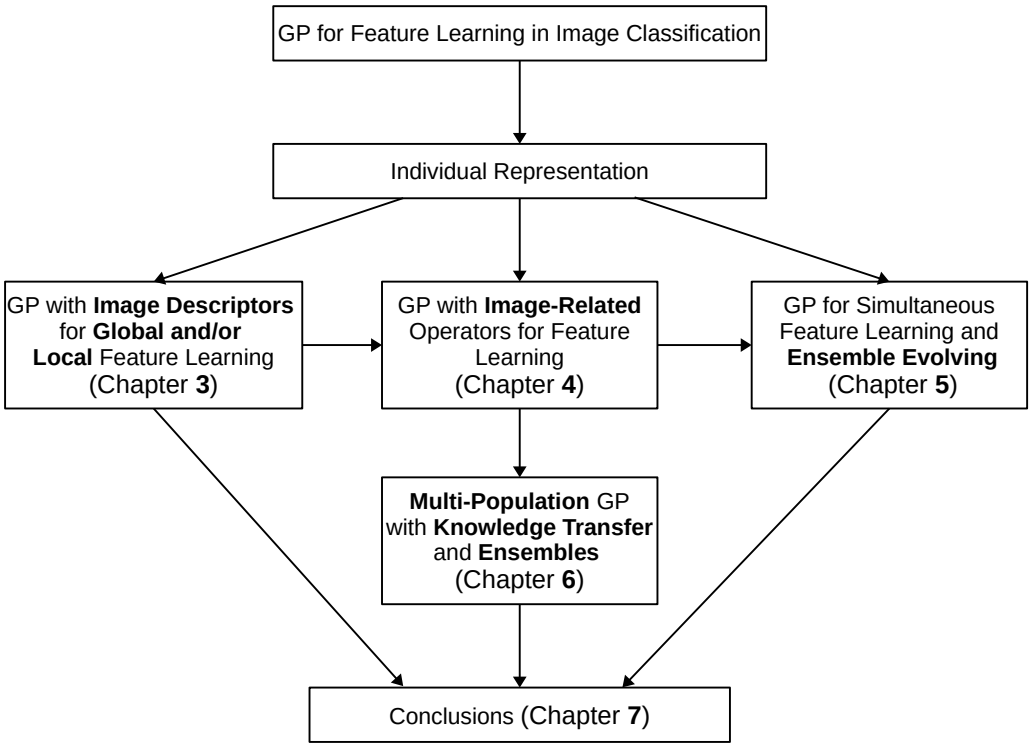


Figure 1.2: The overall structure of the contributions.

of this thesis are presented in Chapters 3-6. Each of the four chapters addresses one of the objectives and produces one major contribution. Chapter 7 concludes this thesis and points out future research directions.

Chapter 2 presents essential background concepts of this thesis, including computer vision, machine learning, classification, ensemble learning, transfer learning, feature learning, evolutionary computation, and genetic programming. It also reviews recent literature related to this thesis and summarises the limitations that this thesis aims to address.

Chapter 3 proposes a new GP-based approach to automatically select and combine existing image descriptors to extract rich and discriminative global and/or local features for different image classification tasks. A novel program structure (individual representation), a new function set with image descriptors and a new terminal set are proposed. To effec-

tively learn discriminative features, a new feature learning process and a new fitness evaluation process are developed. The new approach is examined on eight different image classification datasets, including multi-class classification datasets. Further analysis of computational cost and example solutions is conducted to provide deep insights into the new approach.

Chapter 4 proposes a new GP-based approach with image-related operators and a flexible program structure to feature learning for different image classification tasks. A flexible program structure, a new function set with many image-related operators, and a new terminal set are developed. The new approach is evaluated on 12 benchmark datasets of varying difficulty and compared with a large number of existing methods. Further analysis is conducted to provide an in-depth understanding of the new approach.

Chapter 5 develops a new GP-based approach to automatically learning effective features and evolving ensembles for image classification. A new representation with an input layer, filtering & pooling layers, a feature extraction layer, a concatenation layer, a classification layer, a combination layer, and an output layer is developed. Each functional layer, except for the input and output layers, has a number of different functions. The new approach is evaluated on 12 datasets of varying difficulty, including large datasets with noisy images. The performance of the new approach is compared with a large number of benchmark methods. Further analysis is conducted to analyse the effectiveness and diversity of the evolved ensembles.

Chapter 6 proposes a new GP-based feature learning approach with high generalisation performance and low computational cost for image classification. A new feature learning approach, including a new knowledge transfer method, a new fitness function, and a new combination strategy for constructing ensembles, is proposed. The new approach is examined on 11 different image classification datasets of varying difficulty and compared with the baseline GP-based feature learning method and a large

number of other existing methods. The results are presented and analysed in detail. Further analysis is conducted to show the effectiveness of the ensembles and the knowledge transfer method.

Chapter 7 summarises the work and concludes this thesis. The main contributions and key findings of this thesis are highlighted. A number of future research directions are pointed out.

Chapter 2

Literature Survey

2.1 Computer Vision

2.1.1 Overview

As a human, it is easy to describe the objects that we see from the world as our human vision system can perceive a 3D structure with enough information, such as the shape, appearance and colour of the objects. The perceptual psychologists have investigated the human vision system for decades to figure out how it works [213]. However, using a computer to simulate the human vision system is very complicated. For example, humans can easily track an object in a moving environment with a complex background, but it is difficult to achieve this using a computer. The task involves an inverse problem of vision [213], where human vision system has the ability to recover the unknowns when the information is insufficient, but computers cannot do this easily.

Researchers in computer vision try to use computers to reconstruct or describe the objects in images or videos as the human vision system can do [30, 113, 213]. A definition of computer vision from The British Machine Vision Association and Society for Pattern Recognition (BMVA) [1] quoted as follows:

> "It is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images".

Computer vision is a broad research field with different disciplines involved, such as geometry, mathematics, engineering, and physics. The research on computer vision not only focuses on theoretical development but also attempts to solve real-world problems about vision using computers, such as face recognition, motion capture and remote sensing.

A large number of computer vision applications have been developed in different fields, including autonomous navigation, medical imaging, surveillance, fingerprint recognition, and biometrics [113, 213]. However, the majority of applications consist of several typical computer vision tasks, such as object classification, object detection and object recognition.

Object classification is the task of assigning a class label to an object in the image [22, 63]. It is a challenging task due to the wide range of variations of images/objects in colour, shape, scale, illumination, orientation, and occlusions [219]. Object detection means to detect objects in an image against the background without considering the classes of the objects [22, 63]. The main purpose of object detection is to find the locations of the objects in an image [10]. Object recognition means to recognise the class labels and the locations of the objects from an image [22, 63]. Object recognition actually performs both object detection and object classification, which is more difficult.

Figure 2.1 shows a commonly used scheme of an object recognition system [22, 63]. Typically, an object recognition system has four key modules. The input of the system is an image and the output is the recognised object(s). The first important module is feature extraction, where image features, such as edge features, are extracted from the input image. A grouping method is then used to group all the extracted features into meaningful collections, such as the collection of edge features, which are called indexing primitives [63]. A matching algorithm is employed

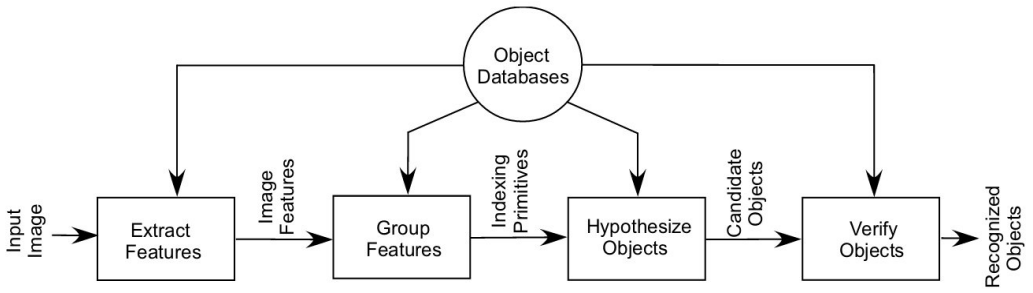


Figure 2.1: General object recognition system [22, 63].

to determine the candidate objects by using the grouped collections and the objects from the databases. Finally, a score is given to each candidate object and the best candidates with the highest scores are labelled as the recognised objects.

Besides the three typical tasks, i.e., object classification, object detection and object recognition, computer vision also contains other well-known tasks, including image segmentation, object tracking, edge detection, and content-based image retrieval. Among these tasks, the most general components are image classification, image preprocessing and feature extraction, which will be described in the following sections.

2.1.2 Image Classification

Image classification is the task of assigning images with one of the predefined class labels based on the content in the images. For example, an image classification algorithm/system can classify the images with a cat into the cat category and the images with a dog into the dog category, as shown in Figure 2.2 (the images in this figure are from [74]). Image classification includes a wide range of tasks according to the type of images, such as medical image classification, remote sensing image classification, face classification, texture classification, and biological image classification. Due to the high variations of background, illumination, viewpoint, scale, deformation, and occlusion across images, image classification is a

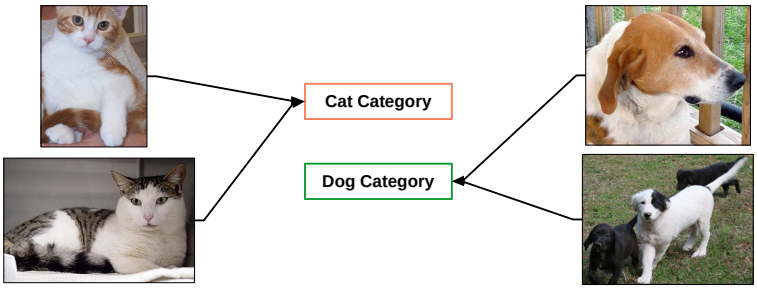


Figure 2.2: An example of cat and dog image classification. The example images are from [74].

challenging task.

2.1.3 Image Preprocessing Operators

Image operators are usually used in the image preprocessing process to obtain images with high equality or to satisfy certain requirements. Generally, images are collected from different conditions. The raw images may contain noise or have poor contrast, which is not suitable for direct analysis. Image preprocessing aims to deal with these images in order to obtain better quality or performance. Commonly used image preprocessing operators are briefly described as follows.

1. **Histogram equalisation:** Histogram equalisation aims to equalise the distribution of the intensities of an image [2]. It spreads the large values or small values of intensities out over a larger range, which increases the contrast of the overall image.

2. **Gaussian filter:** The Gaussian filter is known as a blurring/smoothing filter, which can be used to reduce the noise of an image [2]. In the filtering process, each pixel is transformed according to its neighbour pixels and the filter kernel. In Gaussian filter, the kernel is generated based on a 2D Gaussian function, where the standard deviation and the kernel size need to be set.

3. **Edge filters/detectors:** There are many edge filters in the literature to detect edges in an image [2]. A simple example is the $F = [-1 \ 0 \ 1]$ operator. In this filtering process, if the left neighbour pixel value is bigger than the right neighbour pixel value, it will return a negative value to replace the current pixel value. Otherwise, it will return a positive value. Edge filters are designed to find discontinuities along with different directions, e.g., horizontal and vertical. Two representative edge filters are Prewitt and Sobel edge detectors [2].

4. **Laplacian filter:** The Laplacian filter is designed for detecting the flat area or the area where has significant edges in an image [2]. The filter is generated by discretising and approximating the Laplacian operator. Compared with other edge filters, e.g., the Sobel edge detector, the Laplacian filters cannot detect edges from different directions.

5. **Laplacian of Gaussian (LoG) filter :** The Laplacian filter might produce noisy results [2]. A LoG filter is used to reduce the noise to obtain better results than the Laplacian filter. The results produced by the LoG filter is similar to that of performing a Laplacian filter on the results produced by a Gaussian filter.

6. **Difference of Gaussian (DoG) filter:** The DoG filter is designed for finding regions in an image where they are changes in different scales [2]. The DoG filter is created using two Gaussian filters with different but nearby scales.

7. **Gabor filter:** The Gabor filter is used to find elements in an image at a certain frequency and orientation with a certain phase [2]. A Gabor filter is created by using the product of a 2D Gaussian function with a 2D sinusoid function. To select a Gabor filter, it is necessary to determine the parameters, including phase, orientation and wavelength.

2.1.4 Image Descriptors and Feature Extraction

Image descriptors are developed by domain experts to extract/describe effective image features for particular tasks. An image descriptor may consist of a set of different operations or a number of mathematical equations/rules that are able to detect keypoints and describe informative image features. The process of transforming images into features using image descriptors and/or other operators is known as feature extraction. Image feature extraction aims to extract informative and discriminative information from an image that is helpful for the task as well as reducing the dimensionality of the image data. The image features can be broadly classified into two categories, global (holistic) features and local features [99]. Global features are the features extracted from the whole image based on all the pixel values, which can be considered as properties of an image, such as texture, colour and shape. Local features are the features extracted from keypoints or regions of interest, which are often detected from an image. In the process of global or local feature extraction, different image descriptors may be employed. Representative image descriptors include Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP), Scale Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF) and so on [99].

This section overviews several well-known image descriptors that can be used to extract global and/or local features from the images.

1. **Histogram:** The histogram method is a very simple and fast feature extraction method [2], which quantises the distribution of all the pixel values in an image as a feature vector. This method is commonly used for generating histogram vector according to the results produced by filters, descriptors or other operators.

2. **Grey-Level Co-occurrence Matrix (GLCM):** The GLCM method is designed by Haralick [96, 97] to characterise the texture of images by calculating the occurrences of the adjacent grey levels according to

the pre-defined distance and orientation. After obtaining the GLCM images, fourteen texture statistics are calculated as features for classification. The fourteen features are Angular Second Moment, Contrast, Correlation, Sum of Squares, Inverse Difference Moment, Sum Average, Sum Variance, Sum Entropy, Entropy, Difference Variance, Difference Entropy, two Information Measures of Correlation, and Maximal Correlation Coefficient [97].

3. **Local Binary Patterns (LBP):** The LBP descriptor [169] is a simple but effective texture description method. LBP compares each central pixel with its neighbour pixels in a sliding window to generate binary codes. Then the value of the central pixel is replaced by the sum of all the products of the binary code and pre-defined weights. The histograms of the generated LBP image are used as features for image analysis. Different versions of LBP can be found in [111, 169], where the rotation-invariant LBP (i.e., uniform LBP) is the widely used version.

4. **Histogram of Oriented Gradients (HOG):** The HOG method is a well-known shape and appearance feature extraction/description approach to human detection [60]. It contains a number of steps, including gamma and colour normalisation, gradient computation, weighted voting into spatial and orientation cells, and contrast normalisation over overlapping spatial blocks [60]. To simplify, the main idea of HOG is to extract locally normalised histogram features of gradient orientations (cell) in a densely overlapping grid (block). By using this approach, the local object appearance and shape are well-characterised and finally generate a feature vector.

5. **Scale Invariant Feature Transform (SIFT):** The SIFT method [147] is the most popular keypoint detection and description method for image matching or object detection. In this approach, local extreme points are detected by using the DoG filters with different scales

firstly. Then the Taylor function is used to optimise and eliminate low-contrast keypoints and the Hessian matrix is used to eliminate edge responses. For each keypoint, SIFT produces 128 histogram features of gradient magnitudes and orientations. Without detecting keypoints, a dense SIFT method is developed to extract features from images with less computational complexity [220]. This method is rotation, illumination and scale-invariant.

6. **Speeded Up Robust Features (SURF):** The SURF method [32] is an effective scale and rotation-invariant keypoint detector that is similar to SIFT but faster and more robust. Instead of using DoG and Gaussian derivatives in SIFT, the Hessian blob detector is used in SURF. The SURF descriptor generates a 64D feature vector for each detected keypoint rather than 128D by SIFT.

7. **Others:** Other image descriptors include Binary Robust Invariant Scalable Keypoints (BRISK) [139], Gradient Location-Orientation Histogram (GLOH) [157], and Fast Retina Keypoint (FREAK) [18].

2.2 Machine Learning and Classification

2.2.1 Overview of Machine Learning

Machine learning as a research field under the big umbrella of Artificial Intelligence (AI) [154] has received remarkable attention over the world in recent years. Machine learning aims to design computer programs that are able to automatically learn from data/experiences to solve problems [20, 159]. One famous definition of machine learning by Mitchell [159] quoted as follows:

> "A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P**, if

its performance at tasks in **T**, as measured by **P**, improves with experience **E**."

According to this definition, there are three main components in a machine learning system, i.e. experience **E** (data), task **T** and performance measure **P**. The performance measure is related to the task and the computer program. Thus machine learning is also defined as "Machine learning is programming computers to optimise a performance criterion using example data or past experience" by Alpaydin [20].

Machine learning is a big research area, involving a large number of algorithms and applications in various fields, such as engineering, finance, medical diagnosis, biometrics, automatics, and game playing. Depending on different types of feedback provided for learning, machine learning algorithms can be categorised into three main groups: supervised learning, unsupervised learning and reinforcement learning [189].

- In **supervised learning**, the input and the desired output are provided for the learning system, where the task is to learn models/functions that map the input to the target output [20]. Two well-known supervised learning tasks are regression and classification. In regression, the desired continuous output of each instance is known in advance, where one feedback of this learning may be the error such as mean squared error. In classification, the class labels of the input data are provided. The goal of this task is to learn models that can predict class labels given unseen data. Thus the feedback or measure may be classification accuracy or error rates. Typical supervised learning algorithms are Decision Trees, Naïve Bayes, Artificial Neural Networks, and Support Vector Machines.

- In **unsupervised learning**, only the input data is provided for the learning system, whose task is to find regularities/structures in the input data [20]. One representative example task is clustering, which aims to find clusters of inputs or group inputs into different groups.

The *k*-means clustering algorithm [98] is a widely used method for clustering, where a distance measure is often employed to guide the learning.

- In **reinforcement learning**, a learning agent is to explore the environment by taking actions. Each action has different impacts on the environment. The feedback in a form of rewards or punishments of the action is used to guide the learning. The output of a reinforcement learning system is a sequence of actions. Two well-recognised reinforcement learning algorithms are Temporal Difference learning and Q-learning [210].

Other well-known machine learning types include semi-supervised learning, transductive inference, transfer learning, on-line learning, and active learning [160]. More details can be seen from [160].

2.2.2 Classification

As classification (i.e., image classification) is the main task to be solved in this thesis, this section introduces the basic concepts of classification. Classification is a typical supervised learning task in machine learning. Classification is the process of classifying a set of instances/examples to pre-defined classes. An instance is typically an observation of the problem domain, consisting of variables/features and class labels. The overall process of classification consists of two phases. The first phase that learns a classifier is called **Training** and the latter phase that predicts class labels using the learned classifier to unseen data is called **Testing**.

During the training process, a classifier is trained or learned from instances with class labels. This collection of instances used in this phase is called the *training set*. The training process discovers important knowledge and rules in the training set via building models and/or adjusting parameters. The overall goal of training is to obtain a classifier (model)

and/or parameters of a classifier which can achieve the best performance (e.g., classification accuracy) on the training set. During the testing process, the learned classifier is used to predict the class labels for the data (instances), which are unseen to the training phase. This collection of instances is called the *test set*, which is from the same problem domain as the training set and often used to measure the performance of the learned classifier.

A common issue in classification is *overfitting*. It indicates that the learned models can perfectly fit the training data but poorly fit the unseen data. One possible reason is the training data contains noise and the learned models also learn the knowledge/rules from the noise [20]. To address this problem, another dataset called the *validation set* is employed in the training phase to determine when to stop training or to select the best model in the training phase. The validation set is independent of the training set and the test set. In classification, if the performance (accuracy) of the learned model on the validation set starts/keeps decreasing, the training process (cycles) will stop no matter whether the accuracy on the training set is increasing or not. Alternatively, the best model that achieves the highest performance on the validation set among the learned models at each training cycle is selected after completing all the training process.

The performance of learning algorithms is often examined on benchmark problems, where the performance/effectiveness of different algorithms can be compared. The problems are often formed by selecting datasets from public resources to researchers, such as the UCI Machine Learning Repository [1] [26] and Kaggle [2]. The learning algorithms can be directly applied to a dataset that has the publicly separated training, validation and test sets. However, many benchmark datasets only have a set of instances without being split or have a small number of instances. In these cases, different methods that split the data into different (i.e., train-

[1] `http://mlr.cs.umass.edu/ml/datasets.html`
[2] `https://www.kaggle.com`

ing, validation, test) sets can be employed to evaluate the performance of learning algorithms. These methods include *hold-out* and *cross-validation* [24, 119].

In the hold-out method, a dataset is split into two non-overlapping groups according to a specific percentage to form the training set and the test set. The test set keeps unseen to the training phase and is used to test the performance of the model/classifier learned by a learning algorithm. There are different cross-validation approaches, e.g., *k*-fold cross-validation, leave-one-out cross-validation (LOOCV) and leave-*p*-out cross-validation (LOPCV). The cross-validation methods average several hold-out estimators to different data splits as the final evaluation results of the model [24]. In the *k*-fold cross-validation method, the dataset is randomly partitioned into *k* folds with an approximately equal number of instances. Each time, *k-1* folds are used as training samples and the left one fold is used as the test set. The evaluation process repeats *k* times, with each fold used exactly once as the test set. LOOCV is an extreme case of *k*-fold cross-validation, where the value of *k* is equal to the number of instances. LOPCV is similar to LOOCV but it keeps *p* instances in the test set rather than one. Both LOOCV and LOPCV are exhaustive methods that are computationally expensive [24]. *k*-fold cross-validation is usually employed when the dataset is small.

Classification Algorithms

A number of classification algorithms (classifiers) have been developed to tackle classification in machine learning, including *k*-nearest Neighbours, Bayesian classifiers, Decision Tree classifiers, Support Vector Machines, and Logistic Regression. This section will give a brief introduction to these classification algorithms and other classification algorithms.

1. ***k*-Nearest Neighbours classifier (KNN):** KNN is the simplest non-parametric machine learning method [59]. When using KNN to solve

classification problems, a training set is given in advance. KNN calculates the distances of an instance in the test set to each instance in the training set. Then all the distances are compared and the *k* nearest neighbours are found. KNN classifies the instance in the test set to the class that is most represented by its k neighbours. One simple version is KNN with $k = 1$ where the class label of the instance in the test set is assigned as that of the nearest neighbour in the training set. Different distance measurements and different settings of *k* have impacts on the classification results. The most commonly used distance measurements are Euclidean distance, Minkowski distance and Manhattan distance.

KNN does not require assumptions about the feature space or distribution of the data, which leads it to work well in practice. KNN is very simple and easy to implement. However, since for each instance in the test set, a number of calculations of distance are required, it is expensive when the training set is large [80].

2. **Bayesian classifiers:** Bayesian classifiers are probabilistic methods for classification [159]. These classifiers are based on the well-known Bayes' rules to estimate the probability of each class given the attributes [205]. One assumption is the behaviour of the problem domain can be captured by using probability distributions given the training data. According to the Bayes theorem, the posterior probability used to predict the class label can be transformed to multiply the prior probability and a likelihood [205], which can be calculated and stored for the given training data.

   The Naïve Bayes classifier (NB) is the simplest and a commonly used classifier among the Bayesian classifiers. It is based on the assumption that all the features are conditionally independent, which makes the computation easier and requires less memory. In this method, only the prior probability and a number of conditional probabilities

of each feature given the class are required for calculating the posterior probability [205]. The advantages of NB include requiring few parameters, saving memory, and low computational cost. However, the drawback is that it is based on the above assumption, which may be invalid in some domains [205].

3. **Decision Tree classifiers (DT):** DT is a widely used non-parametric machine learning method [20, 159]. In classification, DT uses a tree to represent the learned discrete-valued function/relationships among the attributes and the class labels. A learned tree (which is also a classifier) consists of a root node, internal nodes and leaf nodes, where root node and internal nodes are attributes, and the leaf nodes are class labels. The learning process starts from choosing an attribute from the attribute set of the training data as the root node. A condition (the "if-then" rule) based on the chosen attribute is set to split the training data into subsets. This process of choosing attributes and building "if-then" rules to split data into smaller groups repeats. When the termination criterion is satisfied, a class label is selected as the leaf node. When all the branches of the tree have leaf nodes, the decision tree is learned. Using the learned tree to classify an unseen instance is very simple, as it simply makes decisions from the root node to the internal nodes until reaching the leaf node.

   During the learning process, the main problem is which attribute should be selected to form the internal node or the root node [159]. In the well-known Iterative Dichotomiser 3 (ID3) method, an *information gain* measure is employed to evaluate how well the candidate attribute splits the instances [159]. ID3 performs a greedy search to select the best attribute at each step. The growing of a decision tree stops when a homogeneous set of instances is obtained, which leads to a significant issue, i.e., overfitting. Two commonly used methods for avoiding overfitting are setting a good stopping criterion and per-

forming post-pruning on the learned decision tree.

A learned decision tree has very good interpretability and understandability, which provides insights in which features/attributes are important for classification. However, the decision tree might not perform well in separating non-rectangular regions [178].

4. **Support Vector Machines (SVMs):** SVMs are a set of popular machine learning methods [46]. SVMs employ a kernel function to transform the training data into a higher dimension. In the higher dimension, the main goal of SVMs learning is to find the optimal separating hyperplane or a set of optimal separating hyperplanes that can linearly and easily separate the instances of different classes. The optimal hyperplane is found by maximising the distances between the hyperplane and the nearest training instances (called support vectors) of different classes. The optimal hyperplane can be used as a decision boundary for a binary classification problem in the higher dimension.

   Kernel functions are important in SVMs. The most popular kernel functions are linear function, radial-basis function, sigmoid function, and polynomials. If the data are not linearly separable, the soft margin hyperplane that incurs the least error is expected to be found for classification [20].

   SVMs were primarily developed for binary classification. To deal with multiple classes classification, the commonly used methods for SVMs are to build *one-vs-all* (also called *one-vs-rest*) classifiers or to build *one-vs-one* classifiers [193, 224, 233]. These classifiers are then combined together for solving multi-class classification problems.

   By maximising the margin of the classification decision boundary, SVMs often obtain good generalisation performance. SVM is one of the most effective methods for classification and has been widely applied in computer vision tasks, including image classification [197].

5. **Logistic Regression (LR):** LR, known as softmax regression, is a popular classification algorithm and is widely used in the final layer of neural networks for classification. For a binary classification problem, LR assumes that there is a linear relationship between the variables and the log-odds of the probabilities of the positive ($p = P(Y = 1)$) and negative ($p = P(Y = 0)$) classes. The relationship can be formulated as $\ln \frac{p}{1-p} = w^T x + b$, where w represents a weight vector and b is the bias of the linear model. Given a set of training data, the optimal weights can be estimated using maximum likelihood estimation [124]. With the optimised weights and bias, the probability of an instance to belong to class 1 is $P(y = 1|x) = \frac{e^{w^T x+\beta}}{1+e^{w^T x+\beta}}$ and the probability of an instance to belong to class 0 is $P(y = 0|x) = \frac{1}{1+e^{w^T x+\beta}}$.

   For multi-class classification problems, multinomial LRs that are able to build multiple linear models have been developed [103].

   LR is a special case of the generalised linear model for classification. LR has high interpretability and is often easy to train. Different from SVMs, KNN, and the other algorithms, LR builds soft classifier that predicts the probabilities of instances to belong to each class. However, LR makes the assumption that the dependent variables and independent variables have a linear relationship, which may not hold in many real-world problems.

6. **Others:** Other well-known classifiers are Random Forests (RFs), Adaptive Boosting (AdaBoost), and Multilayer Perceptron (MLP). RF and AdaBoost are ensemble learning methods, which will be introduced in the following section. MLP is a type of feedforward Artificial Neural Network (ANNs), which consists of at least three layers and a number of neurons [92].

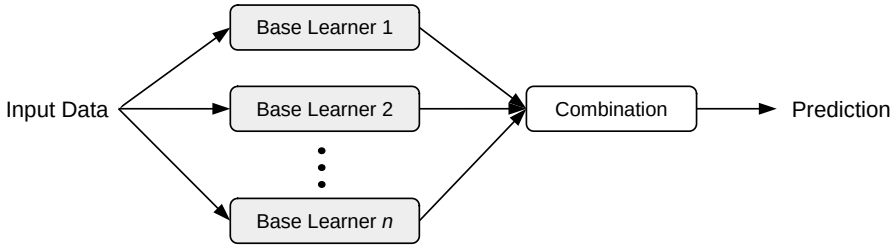


Figure 2.3: A general ensemble method/architecture [247].

2.2.3 Ensemble Learning

Ensemble learning is an active research field in recent years. Ensemble learning aims to build multiple base learners to solve problems [247]. Each base learner can be trained using a machine learning algorithm. Ensemble learning is also known as committee-based learning and learning multiple classifier systems [247].

A common architecture of an ensemble [247] is shown in Figure 2.3. In an ensemble, multiple base learners are employed. If the base learners are trained using the same machine learning algorithm, the constructed ensemble is called *homogeneous ensemble*. If the base learners in the ensemble are trained using different machine learning algorithms, the constructed ensemble is called *heterogeneous ensemble*. With multiple base learners, a combination method such as voting and averaging is often used to combine the outputs of classifiers to make a good prediction.

Typically, there are two well-known types of ensemble methods, i.e., boosting and bagging. A brief introduction of these two methods is described as follows.

1. **Boosting:** Boosting is the method that builds multiple base learners sequentially using training instances with different weights and combines the trained learners for solving a task. A general boosting procedure starts with training one base learner using the training instances with initial weights. The error or the misclassified instances by this trained learner (i.e., classifier for a classification task) can be

found. Based on this, the weights of the training instances can be updated by putting high weights to the misclassified instances and the data distributions can be changed. In the second training cycle, a new base learner is trained using the training instances with new weights. With a different data distribution, this base learner focuses more on correctly classifying the instances that have been misclassified by the previous base learner. This process repeats until all the training cycles are completed. Finally, a weighted ensemble of multiple trained learners can be constructed to solve the problem.

A commonly used boosting method for classification is the AdaBoost algorithm [87]. The AdaBoost algorithm defines how the training set s changed for building the next base learner and how multiple trained base learners are combined at the final step to construct an effective ensemble with high generalisation performance. More details of AdaBoost can be found in [247].

2. **Bagging:** Bagging is the method that builds multiple base learners in parallel using different training instances and combines them for prediction. Different from boosting, bagging can build or train multiple base learners simultaneously and each base learner is independent of the other base learners. Bagging uses bootstrap sampling [72] to create multiple data subsets with different distributions from the original training set. Bootstrap sampling generates samples with replacement, which indicates that an instance may appear more than once in a data subset. Then each sampled data subset can be used to build a base learner. Then an ensemble of multiple trained learners can be created using a combination/aggregation method, i.e., voting for classification and averaging for regression [247].

   A representative example of the bagging methods is Random Forest (RF) [48]. RF uses the idea of bagging to build an ensemble of decision trees. Besides, it uses randomised feature selection to introduce

high diversity to the ensemble. In RF, a subset of features are randomly selected and the corresponding data subset with the selected features is used to build the decision tree. It is noted that the randomised feature selection is different from that in building the decision tree. In other words, RF uses different datasets with different feature subsets to build the decision trees, which increases diversity. Therefore, RF often obtains a better generalisation performance than the other ensemble methods [247].

Combination methods can combine multiple trained learners for prediction, which are also important in ensemble learning. Two typical combination methods are **averaging** and **voting**. Averaging is used for combining numeric outputs, e.g., probabilities or predicted values for a regression task. The averaging methods include simple averaging and weighted averaging, which considers the weights of each base learner. In contrast, voting aims to combine trained learners that produce nominal outputs, e.g., class labels for a classification problem. The voting methods include majority voting, plurality voting, weighted voting, and soft voting.

An ensemble often achieves a better generalisation performance than a single base learner on unseen data [249]. The performance of an ensemble can be defined as $E = \overline{E} - \overline{A}$, where $\overline{E}$ denotes the average of the generalisation errors of the base learner and $\overline{A}$ denotes the average ensemble ambiguity (diversity) [127, 247]. As indicated by this equation, to obtain a good ensemble, the base learners should be accurate and diverse. In the ensemble building process, the diversity of classifiers is often considered as a key factor affecting the performance of ensembles [129]. The diversity indicates that the errors achieved by the base learner in the ensemble are uncorrelated, which is not straightforward to measure. The diversity issue is still an open issue in ensemble learning. Typical methods to enhance the diversity of ensemble are data sample manipulation, input feature manipulation, learning parameter manipulation, and output representation manipulation [247].

2.2.4 Transfer Learning

Many traditional machine learning methods assume that the training data and the test data have the same feature space and distributions [170]. When the distribution or feature space of the data is changed, a new model is required to be built on these new data. In many cases, the training data with labels in one task/domain may not be sufficient or available, but the training data in a similar/related domain can be easily obtained. Motivated by these, transfer learning or knowledge transfer has been proposed and investigated [223].

A simple transfer learning task may include four main components, i.e., source domain (D_s), source task (T_s), target domain (D_t) , and target task (T_t). A domain, $D(X, P(X))$, can be expressed by a feature space X and a marginal distribution $P(X)$. A learning task aims to learn a prediction function $f(\cdot)$ that maps from the feature space X to the label space Y. For example, $f(\cdot)$ for a classification task can be expressed as a conditional distribution $Q(Y|X)$. The source domain (D_s) and the target domain (D_t) are different if they have different feature spaces and/or marginal distribution. The source task (T_s) and the target task (T_t) are different if they have different label spaces and/or conditional distributions. Given a source task T_s and a source domain D_s, the purpose of transfer learning is to extract knowledge learned from the source domain and to use the extracted knowledge to improve the learning performance of the target task T_t on a target domain D_t. It is noted that multiple source domains can be used in transfer learning [170]

According to Pan *et al.* [170], there are three main research issues in this field: 1) what to transfer, 2) how to transfer, and 3) when to transfer. The first question asks what knowledge learned from the source domain can be transferred to the target domain or task. For different tasks or domains, not all the knowledge is useful and has positive effects on solving target tasks. It is necessary to discover some common or general knowledge crossing different domains that can be transferred. The second issue

is about the learning algorithms/methods that can be developed to perform transfer learning based on the extracted knowledge. The final issue determines in which situations the knowledge transfer should or should not be done. A bad situation occurs when the source domain and the task domain are not related or less related, which might lead to a significant issue called negative transfer in transfer learning. Negative transfer means knowledge transfer, which decreases the performance of learning in the target domain [170, 216].

Typically, transfer learning can be divided into the following three categories [170].

1. **Inductive transfer learning:** In an inductive learning task, the target task and the source task are different, but the target domain and the source domain can be the same or different. According to whether the labelled data in the source domain is available or not, the inductive transfer learning can be classified into two cases. The first case, where the labelled data of the source domain and the target domain is available, is similar to multi-task learning [52]. In the second case, the labelled data of the source domain are unavailable, while the labelled data of the target domain is available. This case is similar to self-taught learning [179].

2. **Transductive transfer learning:** In a transductive transfer learning task, the target task and the source task are the same, while the source domain and the target domain are different but related. In this case, the labelled data is available in the source domain but not available in the target domain. The transductive transfer learning can be divided into two cases according to whether the feature space across different domains is the same or not. The first case is the feature spaces of the source domain and the target domain are different. The second case is the feature spaces of the source and target domains are the same. This case is similar to domain adaptation [61].

3. **Unsupervised transfer learning:** In an unsupervised transfer learning task, the source task/domain and the target task/domain are different but related to each other, while no labelled data is available in both domains. Unsupervised transfer learning aims to solve problems in an unsupervised way without labelled data. This task includes clustering, dimensionality reduction and so on.

Many approaches have been proposed to address different transfer learning tasks [149, 170, 216, 223]. These most common approaches are instance-transfer, feature-representation-transfer, parameter-transfer, and relational-knowledge-transfer [170]. Instance-transfer aims to reuse parts of the data (instances) in the source domain for learning in the target domain. Two approaches, i.e., instance reweighting under inductive transfer learning scenarios and importance sampling under transductive scenarios can be used for instance-transfer. Feature-representation-transfer means discovering a good feature representation that could have positive impacts on learning in the target domain. Learning a good feature representation from the source domain can be in a supervised way or an unsupervised way. Parameter-transfer aims to extract some common parameters of the model learned from the source tasks and reuse them when learning in the target domain. This approach is based on the assumption that there are shared or common parameters cross source domains and target domains. The final approach is relational-knowledge-transfer, which is to transfer the relationship among the data from the source domain to the target domain.

2.3 Feature Learning

Feature learning, also known as representation learning, aims to learn effective representations that capture useful and underlying information of the data to build classifiers or other predictors for solving a task [33]. The

representation of the data often refers to raw data or a number of features that describe the domain. Feature learning is a general term that emphasises the change/optimisation of the representation of the data using a learning algorithm to solve a task. During the feature learning process, a number of operations related to features, e.g., feature extraction, feature construction and feature selection, may be involved to find the optimal representation of data for a task.

1. **Feature Selection:** Feature selection is also known as variable selection or attribute selection. The task of feature selection is to select a subset of relevant features from the original large set of features [165]. A large set of features are often employed to represent the data/domain. But these features may include irrelevant and redundant features, which are not useful for representing the data or solving a task, e.g., classification. Feature selection can address this issue by selecting a subset of relevant features. Ideally, the smallest subset that is necessary and sufficient to represent the data should be selected [120]. Feature selection has the advantage of reducing the dimensionality of the data, speeding up the learning process, simplifying the learned model, and/or increasing the performance [234].

2. **Feature Extraction:** Feature extraction is the task of extracting new informative features to represent the raw data via some functional mapping [145]. In some domains, such as image, video, and text, the original/raw data are not informative and cannot well represent the domain. When solving a task in these domains, e.g., image classification, a set of features are often extracted to represent an image and the task can be performed using the extracted features. The extraction of features may use or discover some equations or rules to transform the raw data into a set of features, such as calculating the histogram of an image and use the histogram as features. Feature extraction can change the representation of the data by introducing a

new feature space, while feature selection aims to narrow the current feature space. By extracting a small set of features, feature extraction can also reduce the dimensionality of the data.

3. **Feature Construction:** Feature construction aims to construct new high-level features from the original features to represent the data [145]. Feature construction typically creates new features by finding relationships among the current features using mathematical expressions. In general, the constructed features can be used to replace the original features or added to the original features to represent the data. In the former case, feature construction can also be a dimensionality reduction technique. In the latter case, feature construction expands the feature space, which is different from feature extraction. It is also noted that feature extraction and feature construction are often used interchangeably in some literature and application scenarios, such as in [163].

2.3.1 A Typical Feature Learning System for Classification

The aim of learning a set of features is to effectively solve a task. Thus, feature learning is highly related to the task. Fig. 2.4 shows a general system of feature learning for solving a classification problem. In this system, the features are learned from a training set. How to learn features from the data or how to search the feature space is often related to the algorithm(s) employed. There are many algorithms that can learn features in various ways. A typical example is the neural network-based method, which has different types of architectures to learn features/representation for solving a task. More details about these methods can be found in [123].

During the feature learning process, the performance of the learned features is evaluated using a measure. For a classification problem, the measure can be the performance of a classification algorithm using these learned features. The measure is important to guide the feature learning

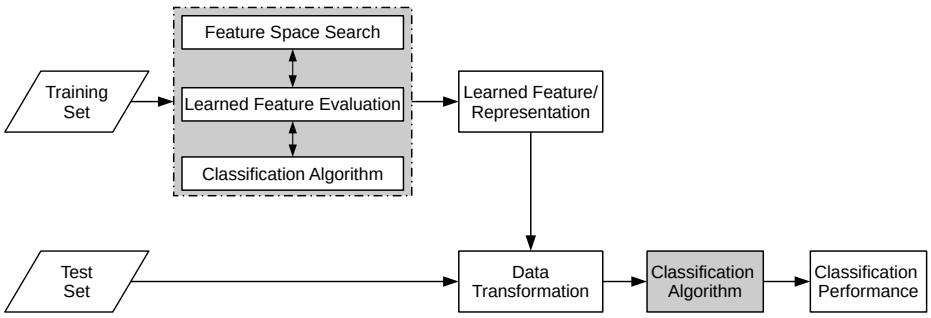


Figure 2.4: Overall process of feature learning for classification.

algorithm to find/search for the optimal feature set. The feature learning process is terminated when a pre-defined stopping criterion is satisfied.

After the feature learning process, the learned features are used to solve the task. In this process, the original data of the test set are transformed to obtain a new set of features. Then a learning algorithm is used to train a classifier using the transformed training set and the trained classifier is applied to classify each instance in the test set.

2.4 Evolutionary Computation

Evolutionary Computation (EC) is an area under the umbrella of Artificial Intelligence (AI) and machine learning (to a lesser extent). EC is the family of algorithms/techniques that are driven from the principles of biological evolution and social intelligence. The origins of EC can date back to the late 1950s, when the earliest EC paradigms were proposed [29, 28]. With the development of several decades, EC became a hot research area with significant attention received over the world. The EC techniques have been successfully applied to various problems, including planning, design, control, and classification [29].

The key idea of EC techniques is to search for the optimal solution from a population of individuals during the evolutionary process. A fitness function is set to guide the search in EC techniques. During the evolution-

ary process, genetic material or individuals are recombined or updated according to different rules. However, an individual with a better fitness value has a higher chance to be selected for the next generation, which simulates the principle of the "survival of the fittest".

Many EC techniques have been proposed and studied. Typically, these techniques can be classified into three groups, i.e., Evolutionary Algorithms (EAs), Swarm Intelligence (SI), and other techniques.

1. **Evolutionary Algorithms:** There are four well-known EAs in this field: Genetic Algorithms (GAs) [106, 107], Genetic Programming (GP) [121], Evolutionary Programming (EP) [82, 83], and Evolutionary Strategies (ES) [34, 182]. These algorithms simulate the biological evolution process, where a selection method and genetic operators, i.e., reproduction, crossover and mutation, are employed to generate new genetic material/populations. In GAs, each individual is typically encoded by a fixed-length (binary) string, which is called a chromosome. The selection method, and the reproduction, crossover and mutation operators are used to improve the population. GP has the same operations for updating the population, but the individual representation is different from that of GAs. More details of GP will be described in the following section. In EP, the population update relies on the mutation operation, which increases the diversity of the population. In ES, the recombination (crossover) and mutation operators are used to generate new populations. However, the selection operation for the recombination process is random-based [182], which is in contrast to that in GAs and GP. More details about EAs can be found in [81, 222].

2. **Swarm Intelligence:** SI is a collection of algorithms inspired by social behaviours of animals. Typical algorithms in this field include Particle Swarm Optimisation (PSO) [71, 118] and Ant Colony Optimisation (ACO) [66, 68]. PSO as one of the most popular algorithms

in SI was proposed by Kennedy and Eberhart in 1995 [71, 118]. PSO simulates the social behaviour of bird flocking or fish schooling. In PSO, the population is called a swarm, which consists of a number of particles (candidate solutions). PSO searches for the optimal solution by updating the position and the velocity of each particle in the search space through a number of iterations. It has been widely applied to job shop scheduling [195], electric power systems [21], feature selection [233], and parameter optimisation [110, 238]. ACO is inspired by foraging behaviours of ants, and the typical research works on ACO have been proposed by Dorigo and his co-workers [66, 68]. In ACO, there are three algorithmic components, i.e., constructing ant solutions, daemon actions, and updating pheromones, where new solutions are generated and the pheromone values of solutions are updated. Typically, good solutions with high pheromone values attract more ants to select these paths in the solution construction process. ACO has been widely applied to find optimal solutions for combinatorial optimisation problems [67]. Other techniques of SI include Bacterial Foraging Optimisation (BFO) [171], which inspired by the foraging behaviours of bacteria, and Artificial Bee Colony (ABC) [117], which simulates the foraging behaviours of a bee swarm.

3. **Other Techniques:** Other techniques have been proposed to find optimal solutions for problems according to different rules. In this field, the typical examples are Differential Evolution (DE) [203, 204], Memetic Algorithms (MAs) [56, 162], Evolutionary Multi-Objective (EMO) [58], Learning Classifier System (LCS) [108], and Estimation of Distribution Algorithm (EDA) [133]. In DE, each individual is encoded by a vector with real numbers. The population is updated using selection, mutation and crossover operators but in a different way in contrast to EAs. A MA is typically a combination of an EC technique used as a global search method and a local search method.

The global search method can explore the search space and the local search method can exploit the local region [214]. In MAs, the population is updated based on the global search method, such as GAs, DE and PSO, and individuals are often improved by a local search method, such as Tabu search and Simulated Annealing.

2.5 Genetic Programming

GP is an EC technique, aiming to automatically evolve computer programs to solve problems without the structure of the solution. During the past decades, GP has attracted much attention in various fields [174]. As GP is the main EC technique to be used in this thesis, this section introduces the basic concepts of GP.

The same as the other EAs, GP is driven from the biological evolution principles. GP is a population-based algorithm, where each individual in the population represents a solution. Unlike GAs, where a fixed-length string is used to encode a chromosome, the individual in GP is a computer program. The evolutionary process of GP is similar to that of GAs, where the selection method, the reproduction, crossover, and mutation operators are often employed to update the population during the evolutionary process. Simulating the principle of the "survival of the fittest", individuals with better fitness have higher chances to be selected to generate new individuals using genetic operators. The overall goal of the GP method is to evolve/find the best solution/program to the problem through a number of generations. A fitness function is used to evaluate individuals and guide the search of GP. The overall process of GP is shown in Algorithm 1.

2.5.1 Representation

There are several well-known GP variants with typically different representations, e.g., Linear GP [31, 47], Tree-based GP (GP) [121], Grammatically-

Algorithm 1: GP

Input : Pc: crossover rate, Pm: mutation rate, Pr: reproduction rate, $\mathcal{F}$: function set, $\mathcal{T}$: terminal set, P: population size, G: maximum number of generations.

Output: $Best\_Fitness$: the best fitness value; $Best\_Tree$: the best program tree.

```
1  begin
2      Select a tree generation method (i.e. full, grow, or
         ramped half-and-half);
3      Randomly initialise P individuals/trees according to F and T;
4      while the termination criterion is not satisfied do
5          Evaluate the fitness of each invididual in the population;
6          Update Best_Fitness and Best_Tree;
7          begin Population updating
               // Selection
8              Select P individuals from the current population;
               // Reproduction
9              Copy P * Pr individuals to the new population;
               // Crossover
10             Generate P * Pc offspring by swapping the branches of
                 parents according to the randomly selected nodes;
               // Mutation
11             Generate P * Pm new individuals by replacing the
                 selected mutation points with new randomly generated
                 branches;
12         end
13     end
14     Return Best_Fitness and Best_Tree.
15 end
```

based GP (or Grammar guided GP, GGGP) [155, 225], and Cartesian GP (CGP) [158]. Among these versions, the most commonly used one is perhaps the tree-based GP [174]. In this thesis, the tree-based GP is used. For simplification, the term GP is employed to indicate the tree-based GP.

In the tree-based GP, each individual/computer program is represented as a $syntax\ tree$. This tree is constructed by a number of primitives, i.e.,

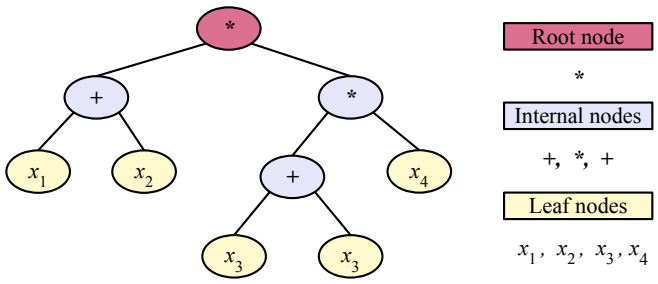


Figure 2.5: An example of GP tree.

functions and terminals. The functions can be used as the root node and the internal nodes, and the terminals can be used as the leaf nodes of a GP tree. Figure 2.5 shows an example of a GP tree, where x_1, x_2, x_3, and x_4 represent four features/variables/terminals used as the leaf nodes, $*$ is the root node, and $+$, $*$ are arithmetic functions used as the internal nodes. In Lisp S-expression, this tree is $(*(+\ x_1\ x_2)(*(+\ x_3\ x_3)\ x_4))$, which can be formulated as $((x_1 + x_2) * ((x_3 + x_3) * x_4))$.

2.5.2 Functions and Terminals

To represent a GP tree, a function set and a terminal set must be pre-defined. A terminal set typically consists of attributes/features, which are related to the target problem/task, and Ephemeral Random Constants (ERCs). ERCs are a number of randomly generated constants that form the leaf nodes of a GP tree. These terminals represent the inputs of a GP system. For example, for a feature selection task, the terminal set includes all the features and ERCs. GP is able to select a subset of these features and ERCs to form the leaf nodes of the evolved trees.

A function set consists of a number of functions or operators. The functions can be categorised into general functions and domain-specific functions. General functions include arithmetic functions, i.e., $+$, $-$, $*$, and protected division $\%$, logical functions, e.g., IF, AND, and OR, and other functions, e.g., cos, sin, log, and exp. These functions are commonly

used to solve relatively simple problems, e.g., feature selection, feature construction, classification, and symbolic regression [12]. For complex problems, such as image analysis, domain-specific functions are often required/employed to facilitate the solution representation.

Sufficiency and closure are two main criteria to construct/choose the function set and the terminal set for GP [121, 174]. Sufficiency means that all the pre-defined functions and terminals are sufficient to express a solution to the problem at hand. Closure includes type consistency and evaluation safety [174]. Type consistency means that the input and output types of functions are consistent. Evaluation safety avoids failures or errors generated by functions at run time.

2.5.3 Population Initialisation

Similar to other EC techniques, the initial population of GP is randomly generated. There are three main methods to generate a GP tree/population, i.e., $full$, $grow$, and $ramped\ half\text{-}and\text{-}half$ [121]. The maximum program size or the tree depth must be defined before population initialisation. The $full$ method generates a tree by randomly selecting functions from the function set as internal nodes or root node until it reaches the maximum tree depth. Then terminals are randomly selected to form the leaf nodes. All the leaf nodes in the tree generated by the $full$ method are at the same depth. In the $grow$ method, functions or terminals are randomly selected as nodes to form a tree. The tree stops growing when all the branches have terminal nodes or the maximum depth has been reached. Thus all the leaf nodes in the tree generated by the $grow$ method are at various tree depths. In the $ramped\ half\text{-}and\text{-}half$ method, a half population is generated by the $full$ method and the remaining half is generated by the $grow$ method. This method ensures a wider range of sizes and shapes of the generated trees. The $ramped\ half\text{-}and\text{-}half$ method is the most commonly used method for population initialisation in GP.

2.5.4 Fitness Evaluation

Fitness evaluation is an important component of the evolutionary process as it guides the search of GP to find better individuals. A fitness function is used to evaluate the fitness of each individual in the population. However, in most cases, choosing a fitness function is not easy. A good fitness function will lead the search smoothly towards the best solution, while a bad fitness function will mislead the search. Typically, the fitness function is related to the problem being tackled. For example, for a classification problem, the commonly used fitness function is classification accuracy [75].

2.5.5 Selection

In the evolutionary process, a selection method is employed to select individuals from the current population for genetic operations. Typically, the better the individual is, the larger chance it will have to be selected. There are a variety of selection methods proposed and used, such as rank selection, fitness-proportionate selection and tournament selection [93, 94, 121]. The tournament selection method is the most widely used selection method in GP [174]. This method selects the best individual from a number of randomly selected individuals by comparing their fitness values. The number of randomly selected individuals each time is determined by a parameter called tournament size in this selection method.

2.5.6 Genetic Operators

There are three main genetic operators used in GP, i.e., reproduction, crossover and mutation.

1. **Reproduction:** The reproduction operation is based on the selection method. It copies the selected individuals to the new generation. Since the majority of the selection methods are related to fitness, the

reproduction operation allows the population to keep good individuals in the new generation. *Elitism* is another strategy commonly used in GP for copying individuals to the new generation. In elitism, a small proportion of the best individuals in the population is copied to the new generation.

2. **Crossover:** The crossover operation is based on two selected individuals (parents) to generate two new individuals (offspring). In GP, the most commonly used form of crossover is *subtree crossover* [174]. Figure 2.6 shows an example of the subtree crossover operation, where the two trees on the left are parents and the two trees on the right are offspring. The crossover operation starts by randomly selecting two points in the parent trees and then swapping the subtrees according to the selected points to generate two new individuals. The crossover operation attempts to generate good individuals by recombination of the current genetic materials (trees).

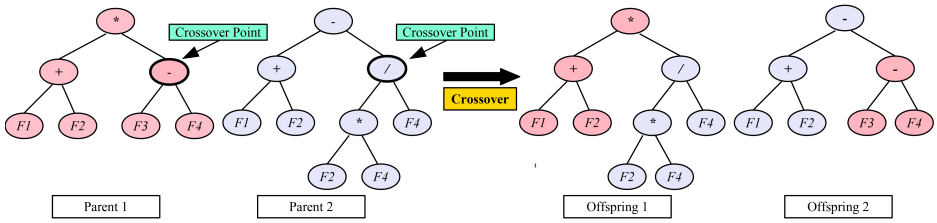


Figure 2.6: An example to show the **subtree crossover** operation.

3. **Mutation:** The mutation operation generates a new individual based on one selected individual. In GP, the *subtree mutation* method is used to randomly select a point of a tree and then replace this point with a randomly generated subtree to generate a new tree. Figure 2.7 shows an example of the subtree mutation operation, where the tree on the right is the newly generated tree. In GP, the mutation operation is very important to maintain the diversity of the population by introducing new genetic material into the population.

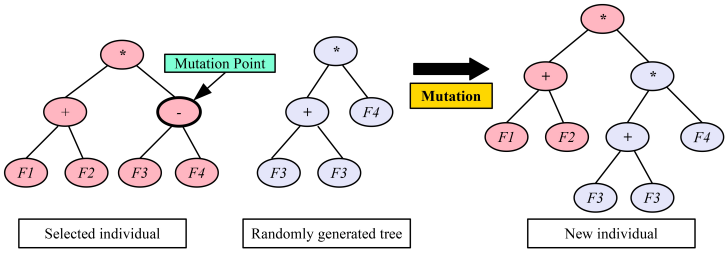


Figure 2.7: An example to show the **subtree mutation** operation.

2.5.7 Strongly Typed Genetic Programming

According to the closure criteria of functions, the traditional tree-based GP method only deals with one type of data, which limits the applicability of GP on complex problems that require multiple data types. To address this issue, Strongly Typed GP (STGP) was proposed as an enhanced version of GP by Montana in 1995 [161] to handle data type constraints. In STGP, each terminal, variable or constant parameter, specifies a data type. For each non-terminal (function), the input data type(s) and output data type are specified. These data types might be $Integer$, $Float$, and $Array$, which are often defined according to the requirements of the functions or problems. By these definitions, an internal node only takes particular non-terminals or terminals as its children nodes. The data type of the root node is the same as the final output data type of the STGP system.

In STGP, it is possible to define several functions for special purposes according to the problem [11]. In general, specific program structures are often developed to integrate functions or terminals with different data types into a single GP tree. STGP is also able to integrate different types of domain-specific functions, such as image preprocessing operators or filters, and general functions such as $+$, $-$, $*$, and $\%$, into a single tree [138]. STGP is commonly used in image analysis, including feature extraction [11, 16], edge detection [89] and image classification [14, 27, 138].

2.6 GP for Image Classification

This section reviews the closely related work on GP for image classification and the GP-based feature learning algorithms for other tasks.

Existing GP-based algorithms for image classification can be broadly classified into three groups: 1) learning features from pre-extracted image features for classification, 2) learning features from raw pixels/images for classification, 3) evolving neural networks for image classification. The first two groups are pure GP-based methods for feature learning in image classification, while the third group are neural network-based methods for image classification. This section will review the typical work on GP for image classification based on these three groups.

2.6.1 Learning from Pre-extracted Image Features

Nandi *et al.* [164] introduced GP for classifying breast mass into the benign and malignant categories based on the selected features. In the classification system, a number of Regions Of Interest (ROIs) were manually identified. Twenty-two features, i.e., four edge-sharpness measures, four shape factors and fourteen GLCM features were extracted from each ROI. Traditional feature selection methods were used to select a subset of features. Based on these selected features, GP was used to construct high-level features for classification. This method has achieved promising classification results. However, this method required domain experts to perform ROI identification and estimate shape factors in feature extraction.

Ryan *et al.* [190] developed a system to detect the first stage of cancer, where GP was employed for mammographic image classification. A set of preprocessing operations, i.e., background suppression, image segmentation, feature detection, and feature selection, were conducted in the classification system. Only two GLCM features, i.e., contrast and difference entropy of each segment, were manually selected to form the inputs of the GP system. This system has gained 100% accuracy on one dataset.

The limitation of this approach is clear, i.e., requiring human intervention and domain knowledge to extract or select effective features. The performance of this method relies on the performance of the extracted and selected features.

Ain *et al.* [7] developed a multi-tree GP algorithm for skin cancer image classification. First, four different sets of features were extracted from the skin cancer images, i.e., LBP features from the grey channel, LBP features from three colour channels, colour variation features of the lesion regions, and shape features of the lesion regions. Each individual of the multi-tree GP method had four trees and each tree could construct one high-level feature from a different set of features using a classic GP tree. Then the four features were fed into different classification algorithms, i.e., KNN, DT, MLP, RF, NB, and SVM, to perform classification, respectively. The results showed that the wrapper GP methods achieved better classification performance than different GP methods, i.e., single-tree GP wrapper and embedded methods, and multi-tree GP embedded method, and classical classification algorithms on binary classification datasets. The effectiveness of these wrapper GP methods has also been verified on two multi-class skin cancer image classification datasets.

Ain *et al.* [9] further extended the multi-tree GP method to construct multiple features for skin cancer image classification. This method used five GP trees to independently construct features from five different types of pre-extracted features and then used a classification algorithm for classification. This method has achieved better performance than the GP methods constructing one high-level feature and the other classification methods using different types of features on two skin cancer image classification datasets. Although these two methods in [7, 9] have achieved promising results and showed high interpretability, they have a limitation that requires domain knowledge and human intervention to extract effective features from images.

In summary, the aforementioned methods typically use a simple repre-

sentation of GP and different features/inputs, i.e., different terminal sets. These GP-based methods are easy to implement. However, the classification performance of these GP-based methods on image classification highly relies on the performance of the pre-extracted features. Generally, it is important to extract a number of task-specific features to improve the classification performance, such as the texture features for mammographic or dermoscopic images. Domain experts or human intervention is typically needed in these processes, e.g., to perform image preprocessing and feature extraction.

2.6.2 Learning from Raw Pixels/Images

Learning A Single Feature

Atkins *et al.* [27] developed a multi-tier GP method (simplified as 3TGP in [13]) to perform automatic feature extraction and image classification. Based on STGP, this method has three tiers, i.e., an image filtering tier, an aggregation tier and a classification tier, to process the input image, extract and construct high-level features for classification. Simultaneously, the evolved GP tree can be a classifier for binary classification. In the aggregation tier, small regions were detected and important pixel statistics were extracted. The classification tier was able to construct a high-level feature for classification. This method has achieved better results on binary image classification datasets over the methods using manually extracted features.

Al-Sahaf *et al.* [13] proposed a two-tier GP (2TGP) method to feature extraction and image classification from raw images. Compared with 3TGP, the representation of 2TGP was simplified by only having the aggregation tier and the classification tier. Therefore, 2TGP could be faster than 3TGP. The results showed that 2TGP could achieve better or comparable results in different image classification datasets. Two variants of 2TGP were proposed in [14] to automatically detect regions with more flexible

shapes and sizes. However, the performance of the 3TGP and 2TGP methods was only evaluated on binary image classification datasets.

A GP-HoG method was proposed by Lenson *et al.* [138] to employ the HOG descriptor as a function in 2TGP to achieve automatic region detection, feature construction, feature extraction, and image classification using raw images. From the automatically detected regions, this method extracted the histogram features or distance features of HOG. The extracted features were further constructed for classification. The GP-HoG method showed how the advanced HOG descriptor was integrated into GP to achieve high-level feature extraction for effective image classification. Compared to the 2TGP method in [13], the GP-HoG method has obtained better results on all the three datasets.

Evans *et al.* [76] proposed a GP method (ConvGP) with a convolution layer, an aggregation layer and a classification layer for image classification. The convolution layer has convolution and pooling operators. The parameters of the filters can be automatically selected by ConvGP. A single tree of ConvGP can simultaneously perform convolution, pooling, region detection, feature extraction, feature construction, and classification. The results showed ConvGP achieved better classification performance than the other classification algorithms but worse classification performance than Convolutional Neural Networks (CNNs) on four binary image classification datasets.

Evans *et al.* [78] extended the ConvGP method [76] by developing a memetic algorithm based on GP and gradient descent for image classification. This method had the same representation as ConvGP, but it used gradient descent to search for the best filters in the convolutional layer. With this local search operator, this method achieved better classification performance than ConvGP but used much longer training time on four image classification datasets.

In summary, GP has been developed to learn/construct one high-level feature for classification from raw images. These GP methods typically

have a specific representation to perform multiple tasks using a single tree. The GP tree can also be employed as a classifier for binary image classification. The methods benefited from the flexible representation of GP and showed a promise in image classification. However, these methods may not be effective for difficult image classification tasks by only learning one feature. In addition, these methods have only been examined for binary image classification tasks. However, many image classification tasks are multi-class classification tasks.

Learning Multiple Features

Shao *et al.* [197] proposed a multi-objective GP (MOGP) method to automatically learn features from images for classification. MOGP has four layers, i.e., a data input layer, a filtering layer, a max-pooling layer, and a concatenate layer. Each layer employs different types of functions for different purposes. MOGP simultaneously optimised two objectives, i.e., classification accuracy and tree size. In MOGP, Principal Component Analysis (PCA) was employed to reduce the dimension of the learned features and a linear SVM was used to perform classification based on the selected features. The results showed that MOGP obtained better performance than several traditional feature extraction methods and a five-layer CNNs on four different image classification datasets. However, MOGP could produce a large number of features from an input image. Additionally, this method has a very complex feature learning process, which is computationally expensive.

Liu *et al.* [146] proposed a GP method to learn spatio-temporal motion features for action recognition. The idea behind this method was similar to that in [197], while this method used 3D sequence as inputs and employed a number of functions/operators for video processing in its function set. In addition, this method only employed max pooling function in the pooling layer. The results showed that this method obtained better performance than CNNs and Deep Belief Networks (DBNs) on several

well-known datasets. This method also showed good interpretability.

Agapitos *et al.* [5] developed a greedy layer-wise GP method for handwritten digits recognition. This method has a filter bank layer, a transformation layer, an average pooling layer, and a classification layer. Specifically, the filter bank layer has a collection of filters to convolve the input image. Similar to this method, Suganuma *et al.* [207] developed a GP method by using more layers with filters for feature learning. This method has two stages of feature learning, where the first stage uses a combination of image processing filters and the second stage uses a combination of evolved filters. The structures of these two methods are very similar to CNNs, but their performance has not been compared with CNNs.

Al-Sahaf *et al.* [11] proposed a GP method to automatically evolve rotation-invariant texture descriptors for texture classification with a small number of training instances. This method used a procedure similar to LBP to generate a feature vector from an image using a sliding window. The evolved GP descriptor is a classical GP tree with a specific root node. The GP tree consists of the arithmetic functions as internal nodes and the pixel statistics of the sliding window as leaf nodes. The employed pixel statistics are the mean, max, standard deviation, and min values of a sliding window, which allows the learned features to be rotation-invariant. With a distance-based fitness function, this method could learn effective features from a small number of training instances. The results showed that this method achieved better performance than the methods using existing texture descriptors (e.g., LBP, GLCM and other LBP variants) for texture image classification. However, this method learned a fixed number of features from the images, which is not efficient and flexible.

Al-Sahaf *et al.* [16] developed a dynamic GP method to generate a flexible number of texture features for classification, which addressed the limitation of the method in [11]. Several root nodes with a flexible number of children nodes were developed in this approach to allow it to produce variable lengths of features. The results showed that this method achieved

better performance than the method in [11] by learning a flexible length of features. However, these two GP descriptors inspired by the LBP descriptor were originally proposed for describing texture features, which might not be effective for the other types of image classification tasks.

Price and Anderson [175] designed a GP method for image descriptor learning and applied multiple kernel-learning SVM for classification. This method employed a set of image-related operators, such as Canny, Hough Circle and Harris Corner Detector, as functions to learn image descriptors. The results demonstrated that this method achieved promising performance in image classification on one dataset. However, this method should be examined on more image classification datasets to show its effectiveness. In addition, many other image-related operators can be employed in GP to learn effective features, which has not been comprehensively investigated.

Price *et al.* [176] further improved the method in [175] by developing a new GP method that could have a better search ability to learn effective features for image classification. This method had a set of image-related operators as functions and grey, red, blue, and green images as terminals. New crossover and mutation operators were introduced in this method to control the bloat of GP and improve the diversity of the population. The results showed that this method achieved better classification performance on a difficult scene classification dataset than a CNN-based method.

Iqbal *et al.* [114] developed a GP method with transfer learning for complex image classification tasks. The building blocks of GP trees evolved from a source task were extracted and reused in GP to solve another similar or different task. This method investigated the use of the extracted knowledge in the process of individual initialisation and mutation operation. The extensive results showed that transfer learning can be effectively employed to improve the performance of GP crossing different image classification tasks, i.e., texture classification and object classification.

Almeida and Torres [19] proposed a GP method to evolve effective similarity functions from a set of existing functions in remote sensing image classification. The evolved GP tree transformed time series raw pixels into a set of features to form the dissimilarity matrix, which was fed into KNN for classification. Compared with the commonly used methods that used existing similarity functions, the GP method achieved better performance in remote sensing image classification.

To summarise, several GP-based methods have been proposed to automatically learn multiple features for image classification from raw images. These methods have specific GP representations (i.e., program structures, function sets and terminal sets) that define how features are learned from the images. These methods could automatically extract informative and discriminative features from raw images and achieve promising performance in image classification. GP has a flexible representation, indicating that there are many possible ways to learn effective features. However, this has not been systematically investigated. In addition, most of these methods have only addressed very limited image classification tasks. The potential of GP in feature learning for different image classification tasks has not been comprehensively investigated. Therefore, it is necessary to further explore the potential of GP in feature learning for image classification.

2.6.3 Evolving Neural Networks for Image Classification

Suganuma *et al.* [206] proposed a GP method to automatically evolve architecture of CNNs (CGP-CNNs) for image classification. This method was based on Cartesian GP (CGP). A set of commonly used functional models of CNNs, e.g., convblock, resblock, max pooling, average pooling, summation, and concatenation, were designed as GP functions. In CGP-CNNs, each GP individual represents a CNN with a specific architecture, which can be trained using backpropagation for image classification. The

results showed that CGP-CNNs achieved better classification performance and found models with relatively fewer parameters compared with the state-of-the-art CNNs. However, this method is computationally expensive.

Zhu *et al.* [250] proposed a tree-based GP method for evolving architectures of CNNs for image classification. The leaf nodes of a GP tree were the resblocks and the internal or root nodes were normal functions for block assembling. This approach benefits from the flexible representation of GP to evolve solutions with a variable length/depth. A dynamic crossover operator was developed in this method. The results showed that this method achieved results competitive with the state-of-the-art automatic or semi-automatic neural architecture search algorithms on the well-known CIFAR10 dataset. Compared with CGP-CNNs, this method has achieved better classification but found models with more parameters.

Diniz *et al.* [64] designed grammar-based GP to find the architectures of CNNs for image classification. This method used simple grammar to encode an architecture of CNN, i.e., the convolutional layer, the pooling layer and the fully-connected layer. This method has been examined on the CIFAR10 dataset and compared with another automatic neural architecture search method. However, this method only searched very simple connections of the different layers and cannot search for important parameters of the layers, e.g., the number of filters and the kernel sizes.

To sum up, these existing works showed the success of using GP to evolving CNNs for image classification. However, there are very limited works in this field, which may be due to the high computational cost. In addition, these methods focused mainly on the object classification benchmark datasets, i.e., CIFAR10 and CIFAR100 [125]. However, there are many other image classification tasks in the real world. It is necessary to develop new GP-based approaches for different types of image classification tasks.

2.6.4 GP for Feature Learning in Other Tasks

Rodriguez-Coayahuitl *et al.* [186] developed a GP method to learn effective and deep representations of data in a way that is similar to neural networks. This method employed GP forests of standard trees to learn a set of features from raw images. This method has multiple structured layers to learn deep representation. Importantly, this GP method can have an encoder forest and a decoder forest to learn a representation in a way that is similar to auto-encoder. The GP auto-encoder was further investigated in [188], where a GP auto-encoder with an online learning method using different population dynamics and genetic operators was proposed. The reconstruction performance of this method has been extensively investigated on image datasets. However, these methods could be employed to solve many other tasks, such as image classification and object detection, to further demonstrate its effectiveness on representation learning.

Rodriguez-Coayahuitl *et al.* [187] proposed a convolutional GP to automatically evolve filters for image denoising, where each GP tree can be employed as a filter to convolve the image. This method can have one layer to construct one filter or have multiple layers to generate multiple filters to convolve the image sequentially. The idea of this method is similar to CNNs, but the filters generated by this method are different from those in CNNs. However, the performance of this method has only been examined on image denoising tasks, while CNNs can achieve promising results in many other image-related tasks.

Liang *et al.* [142] applied GP to construct high-level features for figure-ground segmentation from seven different types of features, respectively. The GP tree could be used as a classifier to classify the pixels of an image into the object and non-object classes. The effectiveness of the GP methods with different terminal sets was investigated and compared. The results showed the GP methods could achieve better results than four conventional segmentation techniques on different datasets.

Tran *et al.* [217] proposed GP methods to automatically construct multi-

ple class-dependent and class-independent features from high-dimensional data. The main difference of GP for class-dependent feature construction and class-independent feature construction is the fitness function, which uses the class labels or does not. Multi-tree GP was employed to construct multiple features for classification. The results showed the effectiveness of GP-based feature construction for high-dimensional classification. However, these methods only learned a fixed number of features, which may not be effective for classification.

La Cava *et al.* [131] developed a multidimensional GP method with Lexicase selection and age-fitness Pareto survival to automatically learn multiple features using a stack-based representation. In the stack-based GP, each individual produced multiple features. Based on the learned features, a distance-based metric was employed for classification. The results showed that this method achieved better performance than many classification algorithms. However, this method could learn a small number of features from the data.

Wu *et al.* [228] applied GP to automatically evolve image descriptors that can extract features from various images for image registration. This GP method employed a set of simple arithmetic operators as functions and the statistics of the region as terminals to evolve descriptors, which can reduce noise interference of images. The results showed that this method achieved better performance than the other four descriptors and one GP method on five datasets.

La Cava and Moore [130] proposed a multi-objective multidimensional GP method to learn effective features for symbolic regression. In the multidimensional GP, each individual produced a set of features. Then the learned features were weighted to form a linear model for symbolic regression. In this method, new genetic operators were developed to improve learning performance. The results showed the effectiveness of this method over the other baseline methods on a large number of regression problems.

Fu *et al.* [88] developed several GP methods to automatically extract edge features from the images. Based on raw pixels, GP was able to construct low-level, Gaussian-based and Bayesian-based edge detectors, respectively. The results showed the effectiveness and high interpretability of the GP methods on edge detection.

Besides, GP has also been successfully applied to many other tasks, some of which can be found in [4, 6, 12, 37]

2.7 Other Algorithms for Image Classification

2.7.1 Traditional Methods

Traditional image classification methods often use existing feature extraction methods, e.g., LBP [169], SIFT [147] and HOG [60], to manually extract features from the images. Then the traditional classification algorithms, e.g., SVM, KNN, NB, DT, and RF, can be used to perform classification using the extracted features.

Chapelle *et al.* [55] extracted colour histogram features from images and investigated the use of SVM with a new kernel function for image classification. The results verified the effectiveness of SVM with a heavy-tailed RBF kernel over SVM with traditional polynomial or Gaussian radial basis function for image classification. However, this method needs much effort to tune the parameters for the RBF kernel.

Bosch *et al.* [45] developed an image classification method that was able to extract shape and appearance features from the detected regions of interests. Based on the extracted features, RF is employed to perform classification. Compared with the SVM classifier, RF achieved better classification results on a large object image classification dataset.

Lu and Weng [148] discussed existing image classification methods from different perspectives and the methods for improving classification performance. However, the majority of the existing traditional methods

require image preprocessing and feature extraction, which are time-consuming and need domain expertise. In addition, it is often difficult to extract informative features for effective image classification due to the high variations across the images.

Recently, Sparse Representation-based Classification (SRC) has been popular for image classification, specifically face classification. Wright *et al.* [227] proposed SRC for face classification. This method used a set of pre-extracted face features and aimed to find a representation of each test instance using the training set. Based on the residuals, the classification decision can be made by assigning an instance to the class with the smallest residual. One clear advantage of this method is its robustness to occlusion and uniform corruption. The SRC method has been extended to many other image classification tasks, e.g., hyperspectral image classification [240] and ship classification [231].

To sum up, image classification is a big research field with many applications, e.g., face classification, object classification, and remote sensing image classification. More recent reviews on image classification can be found in [151, 198, 232, 244]. Although many traditional methods have been developed for image classification, most of these methods require domain knowledge to determine what features are extracted and how features are extracted from the images to effectively solve this task. The manually extracted features may not be the optimal set for solving the task, which limits the image classification performance.

2.7.2 Neural Network-based Methods

In recent decades, Neural Network (NN)-based methods have achieved impressive results in image classification, including Auto-Encoder (AE), CNNs, DBNs, and Deep Boltzmann Machines (DBMs). The NN-based methods directly learn distinctive and compact representations from raw pixels through layer-by-layer non-linear transformations for image classi-

fication. Among these methods, CNN is the most commonly used method for feature learning and image classification. A typical CNN is constructed by a number of layers, including an input layer, convolutional layers, pooling (sub-sampling) layers, fully-connected layers, and an output layer. CNNs are able to handle a number of image variations, such as scale, shift and distortion, and have been widely applied to image classification in recent years.

Krizhevsky *et al.* [126] proposed a deep CNN (AlexNet) to deal with the 1,000-class classification task of the well-known ImageNet Large Scale Visual Recognition Challenge (ILSVRC). AlexNet consists of five convolutional layers, three max-pooling layers and three fully connected layers. This method achieved a top-1 error rate of 37.5% and a top-5 error rate of 15.3% on the ILSVRC2010 test set, which was better than many methods. In ILSVRC2012, AlexNet achieved a top-5 error rate of 15.3%, which was significantly better than that achieved by the second-best entry with an error rate of 26.2%. However, AlexNet has 60 million parameters and 650,000 neurons, which require huge computing resources.

Szegedy *et al.* [212] designed a deep CNN called GoogLeNet for image classification and detection in ILSVRC2014. In GoogLeNet, an inception module was developed, which can concatenate the feature maps from several convolutional layers and max-pooling layers. By using the inception module, the parameters of each layer dramatically decreased. In particular layers of this method, averaging pooling was used rather than max-pooling. The GoogLeNet ranked the first in ILSVRC2014 on image classification and detection tasks. The advanced version of GoogLeNet can be found in [211], which is called Inception-v4.

Another famous CNN is the VGGNet, which was the runner-up in ILSVRC2014, proposed by Simonyan and Zisserman [199]. The effect of CNNs' depth on the classification accuracy was investigated. Based on the analysis, very deep CNNs architecture (up to 19 weight layers) was developed for image classification. This work demonstrated that increas-

ing depth of CNNs was good for classification accuracy improvement. In VGGNet, only 3×3 convolutional filter and 2×2 pooling were used. However, this model requires large memory and computing resources due to a large number of parameters (138 million parameters).

He *et al.* [100] developed a residual CNN (ResNet) for image classification and detection. A deep residual learning framework with residual representation and shortcut connection was introduced to address the degradation problem with increasing depth. The ResNet was easy to be optimised due to a smaller number of parameters compared to VGGNet, and was able to increase its performance when the depth was increased. The performance of ResNet has been verified on two image classification datasets i.e. CIFAR-10 and ILSVRC2015, and other tasks. ResNet has won first place in the ILSVRC2015 classification task with 3.57% top-5 error rate. An extremely deep CNN achieved by the ResNet method can be found in [101] for image classification, which reaches over 1,000 layers.

Shao *et al.* [196] proposed a new method based on existing CNN models for object localisation, object classification, object detection, and scene classification. In the object classification task, different kinds of deep models were trained, including Inception-V3, Inception-V4, Residual Network, Inception-ResNet-v2, and Wide Residual Network. The ten most difficult categories were chosen from the 1,000 classes to compare the performance of these models. This designed method achieved a 2.99% top-5 error rate in the ILSVRC2016 classification task.

Hu *et al.* [109] designed a Sequeeze-and-Excitation (SE) block in CNNs (SENet) for image classification. The SE block aims to enhance the informative features and suppress less useful features by using useful global information. Different variants of SENets, such as SE-ResNet, SE-Inception, SE-ResNeXt, and SE-Inception-ResNet, were designed and compared. A small ensemble of SENets was employed to deal with object classification in ILSVRC2017 and obtained 2.252% top-5 error rate on the test set. This method won first place in ILSVRC2017 on classification. Other datasets,

such as Place365-Challenge dataset [246] was also used to evaluate the performance of this method.

Huang *et al.* [112] develop a densely connected CNN (DenseNet) for image classification. DenseNet has a number of dense blocks, where all the layers are directly connected with each other. The feature maps of the previous layers are concatenated and passed to the next layer. This design allows $L(L+1)/2$ connections in a L-layer network. The advantage of DenseNet over ResNet or other CNNs is their improved flow of information and gradients in the network, which makes them easy to train. The performance of DenseNet has been examined on four benchmark image classification datasets, i.e., CIFAR-10, CIFAR-100, SVHN, and ImageNet. The results showed that DenseNet achieved better performance than the other CNN methods.

Many other types of CNNs have been developed, where general domain knowledge was utilised to learn invariant features for image classification. Bruna and Mallat [49] proposed an invariant Scattering Convolution Network (ScatNet) for image classification. ScatNet has multiple layers of wavelet transform convolutions with non-linear modulus and averaging operators to learn invariant information from images to achieve good classification performance. This method has achieved better performance than a Gaussian kernel SVM and a generative PCA classifier on digital recognition and texture classification.

Based the concept of PCA, Chan *et al.* [53] developed the famous PCANet for image classification. PCANet can have multiple stages/layers of convolutions to extract/learn high-level features. The filters for convolutions were generated by PCA. The final layers of PCANet were binary hashing and blockwise to obtain the final output/features for classification. Two variants of PCANet, RandNet and LDANet, have been developed by using different filters for convolutions, i.e., randomly generated or learned from Linear Discriminant Analysis (LDA). PCANet has achieved promising results in many well-known image classification benchmark datasets,

including face recognition, digits recognition and texture classification.

Qian and Zhang [177] developed a Feedforward Convolutional Conceptor Neural Network (FCCNN) for image classification. This method built a simple but fast representation learning model by integrating components of CNNs, PCA, binary thresholding (BT), and non-temporal conceptor classifiers. The results showed the effectiveness of FCCNN over the other CNN variants on MNIST variant datasets.

Li and Gong [140] proposed a self-paced CNN (SPCN) by assigning weights to the training instances during the learning process to enhance the learning robustness of CNN. The method has gained better performance than a number of effective algorithms on six benchmark datasets.

AE is well-known as an unsupervised feature learning algorithm. The learned features of AE can be fed into a commonly used classification algorithm such as SVM for classification. Rifai *et al.* [183] developed Contrastive AE (CAE) by using a new cost function with a well-chosen penalty term. This method has been examined on seven image classification datasets and compared with other types of AEs, including 3-layers Stack AE (SAE-3) and 3-layers denoising AE with binary masking noise (DAE-b-3). As variants of NNs, DBN and DBM have also been applied by Larochelle *et al.* [132] to digit recognition.

In summary, NN-based methods are popular and effective for image classification. More related works can be found in [95, 180, 245]. CNNs have gained remarkable success on large-scale image classification in recent years. However, these deep methods have several limitations. First, the deep models with a huge number of trainable parameters require a large number of labelled training instances/images to train. For example, AlexNet has 138 million parameters, which can only be trained well when sufficient data are provided. In many real-world applications, such as medical image classification, biological image classification and remote sensing image classification, the number of labelled instances is often limited due to the high cost to collect data. In these domains, it could be dif-

ficult to obtain a deep model to achieve good classification performance. Second, the deep models often have poor interpretability due to the "black-box" mapping from inputs to outputs. It is difficult to explain and understand what features are extracted/learned and why they are effective for classification. Third, rich domain expertise is often required to design an effective architecture for CNN [248]. CNNs can integrate many different modules/functions, such as inception, dropout, activation functions, regularisation, residual, and batch normalisation, which is difficult to manually find an effective architecture of CNNs with good performance from the scratch. The architecture of CNN is a key factor to achieve promising results, as it can be found from [100, 112, 126, 199, 239] that CNNs with different architectures perform differently. Fourth, the model complexity of CNNs is fixed, indicating that the flexibility and adaptability of the models are limited. Due to these limitations, it is still necessary to explore and develop new approaches for image classification.

2.7.3 Evolving Neural Networks

To address the limitation that designing deep CNNs requires rich domain knowledge, many methods have been proposed to automatically evolve the architectures for CNNs for image classification. These methods are also known as Neural Architecture Search (NAS) algorithms, which include reinforcement learning methods, EC methods, and others [73, 226]. Since most of these works focused on solving image classification, this section reviews recent works on EC for evolving CNNs (or NAS).

Real *et al.* [181] developed an evolutionary algorithm to evolve architecture for deep CNNs (known as Large-scale Evolution) for image classification. In this method, each individual represents a trainable CNN. A specific mutation operation was developed to change or increase the components of CNNs in the individual. The results showed that this method achieved better or competitive performance on CIFAR10 and CIFAR100

than the manually designed CNNs and the other NAS algorithms. However, the computational cost of this method is very high.

Xie and Yuille [230] employed GA to automatically search the architectures of CNNs (Genetic CNN). In this method, the network structure is encoded using a fixed binary string. At each generation, selection, crossover and mutation operators were used to generate a new population of solutions. The effectiveness of this method has been examined on three datasets, i.e., MNIST, CIFAR10 and CIFAR100.

Irwin-Harris *et al.* [115] developed an evolutionary algorithm with a graph encoding to evolve architectures of CNNs. The graph encoding is similar to a tree-based encoding, which is more flexible and has fewer constraints to the search space. Accordingly, the initialisation method and the individual generation method were proposed for creating a population of individuals. The graph encoding can be easily decoded to an architecture of CNN. The results showed that this method achieved similar performance to the manually designed CNNs and found CNN models with significantly fewer parameters.

Sun *et al.* [208] developed a GA method to automatically evolve architecture and connection weight initialisation values for CNNs (this method is knowns as EvoCNN) on image classification. In EvoCNN, GA was employed to search the CNN structures, consisting of convolution layers, pooling layers, and full connected layers, and to find the optimal mean and standard deviation values of weights. Besides, EvoCNN simultaneously searched for the numbers of filters in the convolutional layers and the types of pooling. The results showed that EvoCNN achieved better performance on nine datasets than state-of-the-art algorithms. However, EvoCNN has a high computational cost.

Sun *et al.* [209] developed a GA method to automatically design CNNs (CNN-GA) for image classification. In this method, a variable-length representation and a corresponding crossover operation were developed to better search for the CNN structures. Skip connections were employed

in this method for dealing with complex data. Besides, this method addressed the drawbacks of high computational cost by developing a new fitness evaluation process. The results showed that this method achieved better performance on two benchmark datasets than eight manually-designed CNNs, seven automatic+manual tuning and five automatic CNN architecture design algorithms.

In summary, these methods addressed the limitations requiring rich expertise to design the architectures of CNNs for image classification. However, one major limitation of these methods is the high computational cost. For example, the Large-scale Evolution [181] used 2,750 Graphics Processing Unit (GPU) days, Genetic CNN [230] consumed 27 GPU days, and CNN-GA [209] used 35-40 GPU days to obtain the best structures of CNNs on the CIFAR10 and CIFAR100 datasets. In addition, the NN-based methods still suffer from the poor interpretability of the learnt models.

2.7.4 Ensemble Methods

To employ an ensemble method for image classification often requires a different design since before the general classifier training process the feature extraction process is performed to extract meaningful features from raw images. Dittimi and Suen [65] used CNNs to extract image features and used PCA to reduce the dimension (number) of the features. Then different base learners, i.e., RF, DT and SVM, were trained on the features. Finally, ensemble methods such as stacking or bagging were used to obtain the combined predictions.

Forcén *et al.* [85] developed a weighted ensemble method for image classification. A set of features were extracted and employed to train many binary classifiers. The binary classifiers were combined using neural networks and the weights of the classifiers were calculated to build a weighted ensemble. The experimental results showed that a weighted ensemble could improve the classification accuracy on two datasets.

Sergyan [194] proposed a method that extracts statistics of the histogram features from images and employed a distance-based similarity function in a content-based image retrieval system. This method was fast due to the small number of features employed. However, this method may not be effective for difficult image classification tasks due to the lack of sufficient information.

Kumar *et al.* [128] proposed an ensemble of different CNNs for medical image classification. Based on the AlexNet and GoogLeNet models with weights trained from ImageNet, this method fine-tuned these two CNNs to obtain the outputs (treated as features) of the final layer. The ensemble was built using five classifiers, i.e., the softmax classifier in the fine-tuned AlexNet, the softmax classifier in the fine-tuned GoogLeNet, the SVM classifier trained using the features from AlexNet, the SVM classifier trained using the features from GoogLeNet, and the SVM classifier trained using the concatenated features from GoogLeNet and AlexNet. The results showed that the built ensemble achieved better classification performance than the other CNN baselines on two datasets. However, this method is computationally expensive and requires a large number of training instances to fine-tune deep CNNs.

Xia *et al.* [229] investigated five different strategies, i.e., bagging, boosting, random subspace, rotation-based, and boosted rotation-based, to construct RF ensembles for hyperspectral image classification. In these methods, the spatial features, i.e., extended multiattribute extinction profiles, were used as features/inputs for classification. The results showed the effectiveness of the five ensemble methods on two commonly used benchmark datasets. The results also showed that the rotation-based and boosted rotation-based methods were the most effective methods among all these methods. However, these methods were only developed for hyperspectral image classification and have not been tested on other types of image classification tasks.

Overall, ensemble methods have shown to be effective approaches for

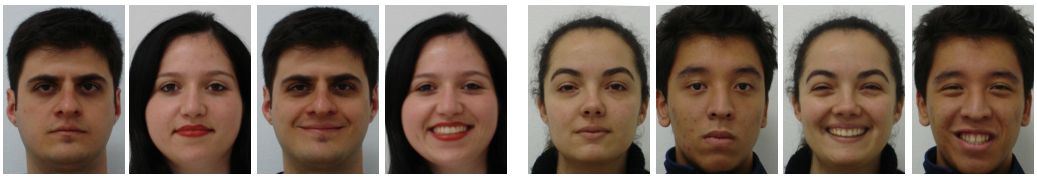


Figure 2.8: Example images from the **FEI_1** and **FEI_2** datasets, respectively.

image classification. However, most methods require rich domain knowledge or human intervention to build an effective ensemble of diverse and accurate classifiers for image classification. In addition, most of these methods used manually extracted features to build ensembles, which might not be effective. Thus, the potential of ensemble methods for image classification has not been fully investigated.

2.8 Benchmark Datasets

Throughout this thesis, 15 different image classification datasets of varying difficulty, image sizes, numbers of images and classes, are employed to examine the effectiveness of the proposed GP-based approaches. The details of these datasets, i.e., image size, number of classes, number of instances, and image type, are summarised in Table 2.1.

The **FEI_1** and **FEI_2** datasets [215] are the tasks of facial expression classification based on images. These two datasets contain frontal facial images with a neutral or smiling expression. Each of these two datasets include 200 260×360 colour images sampled from 200 Brazilian with a different appearance, hairstyle and adorn. Example images from these two datasets are shown in Figure 2.8.

The **VGDB** dataset is to identify Vincent Van Gogh's paintings [84]. This dataset has 330 colour images in two classes, i.e., Vincent Van Gogh's (VG) paintings or not Vincent Van Gogh's paintings (Non_VG). The im-

Table 2.1: Benchmark Datasets

Dataset	Image size	Number of Classes	Number of Images	Image Type
FEI_1	360×260	2	200	Colour
FEI_2	360×260	2	200	Colour
VGDB	(600-14,000)× (600-14,000)	2	330	Colour
ORL	92×112	40	400	Grey
JAFFE	256×256	7	213	Grey
KTH	200×200	10	810	Grey
EYALE	192×168	38	2,424	Grey
FS	About 250×300	13	3,859	Grey
MB	28×28	10	62,000	Grey
MRD	28×28	10	62,000	Grey
MBR	28×28	10	62,000	Grey
MBI	28×28	10	62,000	Grey
Rectangle	28×28	2	51,200	Grey
RI	28×28	2	62,000	Grey
Convex	28×28	2	58,000	Grey

Figure 2.9: Example images from the **VGDB** dataset.

ages in this dataset have very high quality with a large range of resolution in pixels (width and height) from 600 to 14,000. This task is very difficult as all the images are abstract without any particular objects inside and the painting style is hard to capture. Example images from this dataset can be found in Figure 2.9.

Figure 2.10: Example images of some classes from the **ORL** dataset.

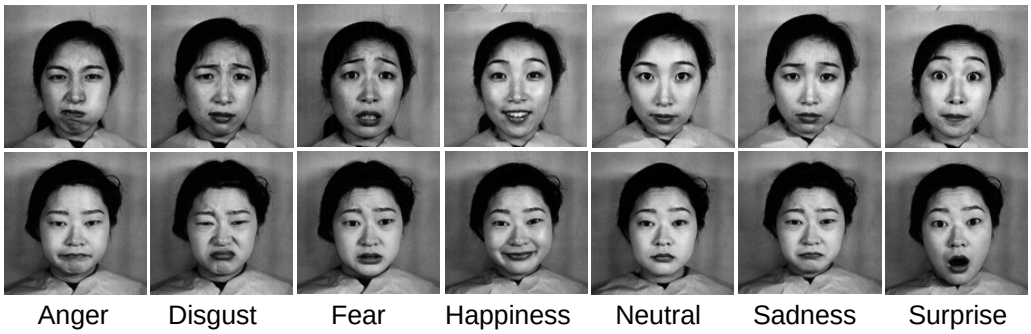


Figure 2.11: Example images from the **JAFFE** dataset.

The **ORL** dataset [191] is a small dataset for face recognition/classification. This dataset includes 400 facial images of 40 different people i.e., 10 images per class. The faces in the images include variations of gender, expressions, rotations, occlusion by glass, illumination, and facial details.

The **Japanese Female Facial Expression (JAFFE)** dataset [150] is a well-known facial expression classification dataset. It has seven common facial expressions, i.e., anger, disgust, fear, happiness, neutral, sadness, and surprise. This dataset has 213 images of 7 different expressions sampled from 10 Japanese females. The size of the images is 256×256. Example images from each class of JAFFE are shown in Figure 2.11.

The **kth-tips2 (KTH)** dataset [153] is a texture classification dataset, containing 810 images equally distributed in 11 classes, i.e., aluminium foil, styrofoam, brown bread, linen, corduroy, cotton, orange peel, cracker, sponge, sandpaper. The images are sampled on nine different scales with three poses under four illumination conditions, which indicates the difficulty of classification. Example images from the KTH dataset are shown

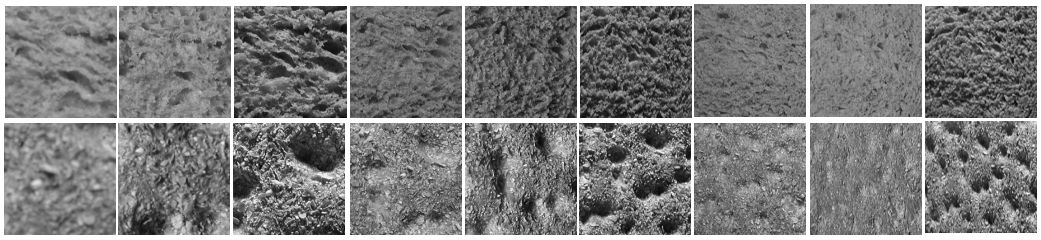


Figure 2.12: Example images from the **KTH** dataset.

Figure 2.13: Example images from the **EYALE** dataset.

in Figure 2.12, where each row represents images from one class. It is clear that the images have variations of scale, illumination and pose.

The **Extended Yale B (EYALE)** dataset [137] is a face classification task, having 2,424 192×168 facial images from 38 different people, i.e., about 64 images per class. The facial images in this dataset are sampled under different poses and illumination conditions. The images of EYALE are resized, cropped and manually-aligned. Example images of EYALE are showed in Figure 2.13.

The **13 categories of natural scenes (FS)** dataset [79] is a challenging task of scene analysis and classification. This dataset contains 3,859 natural images in 13 groups, i.e., bedroom, suburb, kitchen, living room, office, coast, forest, highway, inside city, mountain, open country, street, and tall building. The average size of the images in this dataset is approximately 250×300. The natural images are acquired under different conditions and have high variations, which makes the task difficult. Example images of each class from the FS dataset are shown in Figure 2.14.

The remaining seven datasets are the **mnist-basic (MB)** [132], **mnist-rot (MRD)** [132], **mnist-back-rand (MBR)** [132], **mnist-back-image (MBI)** [132], **Rectangle** [132], **rectangle-image (RI)** [132], and **convex sets (Con-**

Figure 2.14: Examples image of each class from the **FS** dataset.

vex) [132] datasets. These datasets have separated training and test sets [3]. The MB dataset [132] is a large dataset of digit images. The MB dataset is a subset of the famous MNIST benchmark dataset. This dataset has 62,000 images of 10 classes, i.e., 50,000 images in the training set and 12,000 images in the test set. The MRD, MBR and MBI datasets are three variants of the MB dataset obtained by adding factors of variations, including rotation, random background and image background. The MRD dataset includes digit images with rotation by an angle generated uniformly between 0 and 2π. The MBR dataset has images with random background and the MBI dataset has images with random images added as their background. The MRD, MBR and MBI datasets are more difficult than the MB dataset due to these additional variations. The Rectangle, RI and Convex datasets are the tasks of object classification with two classes. The Rectangle and RI datasets have images with rectangle objects and the task is to recognise whether each rectangle in an image has a larger width or length. The RI dataset is more difficult than the Rectangle dataset due to the addition of the random background. The Convex dataset has images with convex or non-convex objects, i.e., two classes. Example images from these seven datasets are shown in Figure 2.15.

These datasets include a broad range of image classification tasks, i.e.,

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

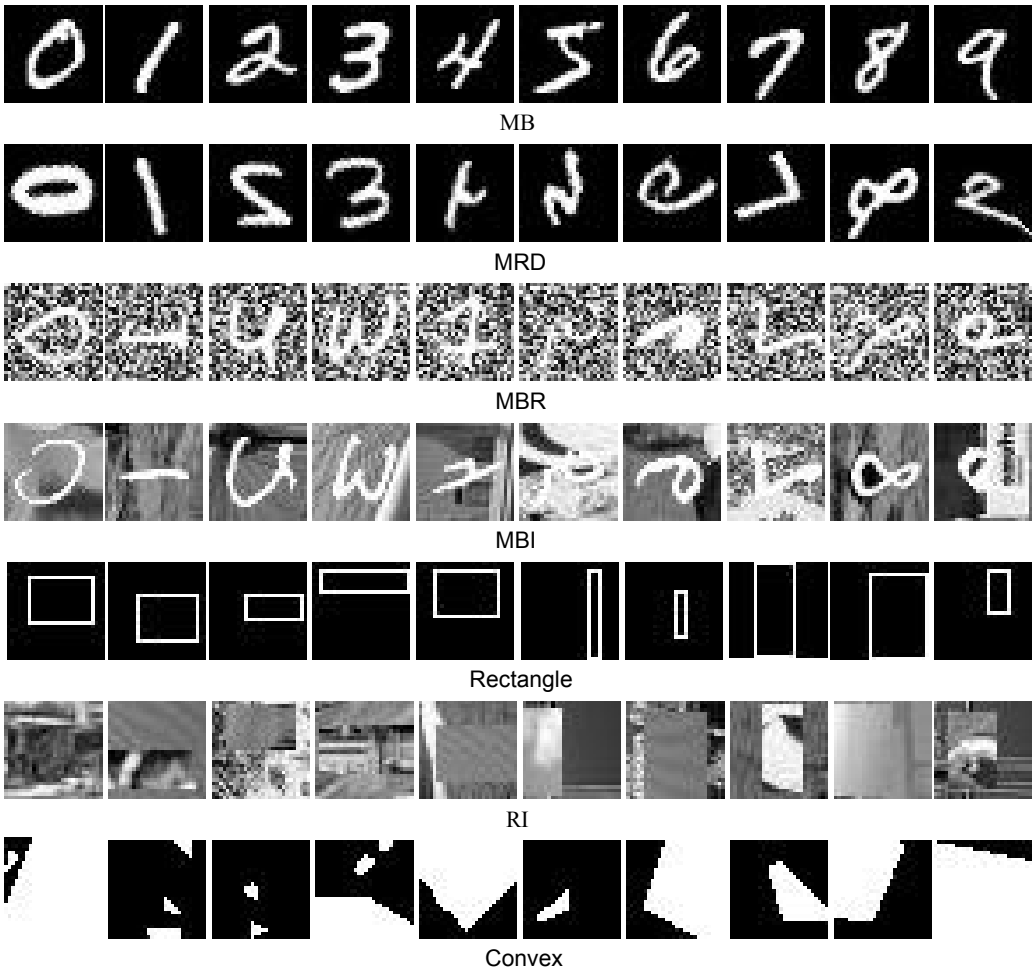


Figure 2.15: Example images from the **MB, MRD, MBR, MBI, Rectangle, RI**, and **Convex** datasets.

facial expression classification: FEI_1, FEI_2 and JAFFE; face classification/recognition: ORL and EYALE; texture classification: KTH; scene classification: FS; object classification: MB, MRD, MBR, MBI, Rectangle, RI, and Convex, and painting classification: VGDB. In these image classification tasks, different image variations are included, e.g., rotation, scale, illumination, pose, facial expression, and occlusion. In the object classification tasks, the variations also include the random background in MBR and

RI, rotations in MRD, and additional image background in MBI. The variety of image classification tasks and the image variations are two main considerations for selecting these datasets as benchmark datasets in this thesis. These datasets will be employed to comprehensively investigate the performance of the proposed approaches on different types of image classification tasks. This is also a limitation of existing GP methods, which have only been examined on a very limited number and type of image classification tasks. It is noticeable that the MB, MRD, MBR, MBI, Rectangle, RI, and Convex datasets are large datasets with over 50,000 images for training and testing. To the best of our knowledge, no GP-based methods have been examined on these datasets.

2.9 Chapter Summary

This chapter reviewed the main concepts of computer vision, machine learning and evolutionary computation, particularly image preprocessing operators, image descriptors, image feature extraction, classification, ensemble learning, transfer learning, feature learning, and GP. This chapter also reviewed and discussed the related works of GP, traditional methods, CNNs, and ensemble methods for image classification.

This chapter summarised the limitations of the existing works that form the motivations of this research. Image classification is a challenging task and a set of informative features are important for effectively solving it. Traditional image classification methods often require domain knowledge to manually extract a set of features from images. The manually extracted features may not be the optimal features for solving this task so that the classification performance is limited. In recent years, NN-based methods, particularly CNNs, have been widely applied to image classification and showed promising results. However, most of these NN-based methods have a number of limitations, such as requiring a large number of training instances, being computationally expensive, and having poor

interpretability. Compared with these NN-based methods, GP has a flexible tree-based representation, enabling it to evolve solutions with variable depths/lengths and high interpretability. Existing works show the potential ability of GP with different representations to learn one or multiple features for image classification. However, existing GP methods also have other limitations, which are discussed as follows.

- Existing works show that employing image-related operators, such as filters, in GP can potentially improve the image classification performance [175, 197]. Existing image descriptors, such as HOG, SIFT and LBP, can extract effective features that are invariant to certain variations. However, the use of these effective descriptors in GP to achieve feature learning has not been fully investigated. In addition, global and local features are two types of features that can be employed for image classification. These two types of features are effective for different image domains. However, most existing methods can only learn one type of image features, i.e., global features or local features, which are not flexible and effective for solving different types of image classification tasks.

- The image-related operators, including image filters and image descriptors, have not been simultaneously investigated in GP to effectively use them to achieve effective feature learning. Existing GP program structures, e.g., multi-tier [27], two-tier [14] and multi-layer [197], are often restricted in a certain way and cannot be used when new image-related operators introduced in GP. Therefore, it is necessary to develop a new GP program structure to effectively use these operators for feature learning.

- Ensemble methods have shown to be an effective approach to image classification [128, 229]. However, very few GP-based ensemble methods have been developed for image classification. GP has a flexible tree-based representation and can integrate the processes of

feature learning and ensemble learning into a single GP tree. However, GP has never been applied to automatically and simultaneously learn features and evolve ensembles from raw images for image classification.

- It is often computationally expensive to use GP to learn features from a large number of training instances, e.g., the famous MNIST dataset [136] has 60,000 training images. However, very few works have successfully addressed this issue in the GP-based feature learning algorithms. Therefore, it is necessary to develop a new GP approach that can improve both the computational efficiency and the classification performance on image classification.

The following four chapters of this thesis will show how GP is employed to tackle these limitations.

Chapter 3

GP with Image Descriptors for Global and/or Local Feature Learning

3.1 Introduction

Many existing image descriptors, such as Histogram of Oriented Gradients (HOG) [60], Scale-Invariant Feature Transform (SIFT) [147] and Local Binary Patterns (LBP) [169], have been developed for handling image variations and widely used for feature extraction. With a flexible tree-based representation, it is possible to use existing well-developed image descriptors as GP functions to automatically learn high-level features for image classification. However, existing GP methods cannot directly employ these descriptors due to the limitations of the program structures (representations). Therefore, it is necessary to develop a new GP-based approach with a new representation to effectively employ these descriptors for feature learning. In addition, many existing methods only learn or extract one type of features, i.e., global features [11, 197] or local features [14, 138]. Typically, the application scenarios of global features and local features are often different. By capturing the overall characteristics (e.g.,

brightness level, texture and histogram) of an image, global features are often employed to classify images that do not have specific objects, such as texture images. In contrast, by extracting features (e.g., edge, shape and appearance) from the regions of interests, local features are often employed to classify object images, e.g., face images. However, from some image classification tasks, both global and local features may be needed. But it is often unknown how many and what types of global and/or local features are needed. Therefore, it is necessary to develop a new feature learning approach to automatically learning global and/or local features in a flexible way.

3.1.1 Chapter Goals

The overall goal of this chapter is to develop a new feature learning approach using GP (FLGP) to automatically select and combine existing image descriptors to extract rich and discriminative global and/or local features for different image classification tasks. To achieve this goal, a novel program structure (individual representation), a new function set and a new terminal set are developed in FLGP. To effectively learn discriminative features, a new feature learning process and a new fitness evaluation process are developed. Specifically, this chapter will investigate:

- how image descriptors are employed in FLGP to achieve global and/or local feature learning by developing a new program structure, a new function set and a new terminal set;
- how the generalisation performance of the learned features is improved by developing a new fitness evaluation process;
- whether FLGP can obtain better classification performance than existing GP-based methods, the methods using different manually extracted features, and three Convolutional Neural Network (CNN)-based methods; and

- whether and how the solutions/programs evolved by FLGP provide insights into the tackled image classification tasks.

3.1.2 Chapter Organisation

The remainder of this chapter is organised as follows. Chapter 3.2 describes the proposed FLGP approach. Chapter 3.3 presents the experiment design, i.e., benchmark datasets, baseline methods and parameter settings. The experimental results are discussed and analysed in Chapter 3.4. Further analysis of the convergence behaviour and the solutions/programs evolved by FLGP is presented in Chapter 3.5. Chapter 3.6 concludes this chapter.

3.2 The Proposed Approach

This section describes the proposed FLGP approach in detail. The overall algorithm, the new feature learning process, and the new fitness evaluation process are presented. Then it describes the main components of FLGP, i.e., the program structure, the function set, and the terminal set.

The FLGP approach is proposed to address the limitations of a GP method for automatic global and local feature extraction (GP-GLF), which is a method we previously proposed in [44]. The GP-GLF method is a feature learning algorithm that uses seven image descriptors as GP functions to learn global and local features for image classification. In GP-GLF, six different root nodes have been developed to combine the global features and local features extracted by descriptors, which allow GP-GLF to produce a combination of global and local features. Although GP-GLF has achieved better performance than a large number of methods using manually-extracted features, it has several limitations. First, GP-GLF learns a combination of global and local features, which is not effective and flexible. Second, GP-GLF uses a fixed train-test split of the dataset in the

fitness evaluation process and uses k-Nearest Neighbour (KNN) for classification. The learning performance can be further improved by exploring a new fitness evaluation process with a new classification algorithm. Third, the features generated by GP-GLF are from different descriptors and they may have different scales. For example, features described by the uniform LBP (uLBP) descriptor may be in the range of $[0,\ 100]$ and the features described by the SIFT descriptor are in the range of $[0,\ 1]$. In the GP-GLF method, these features are fed into a SVM for classification without normalisation, which may lead to bias towards features with large values. Therefore, feature normalisation is needed to rescale the range of the learned features. Fourth, the performance of GP-GLF has not been examined on multi-class image classification datasets.

Therefore, this chapter develops a new GP-based feature learning approach, i.e., FLGP, by addressing the above limitations of GP-GLF. The FLGP approach aims to automatically learn global and/or local features in a flexible way.

3.2.1 Overall Algorithm

The proposed FLGP approach can automatically evolve solutions/trees that are able to extract discriminative global and/or local features using existing image descriptors from the input image. The overall feature learning process of FLGP is shown in Figure 3.1, where the left part shows the general evolutionary learning process of GP and the right part shows the new fitness evaluation process. The new components and the basic configuration of the FLGP system are shown in Figure 3.2.

As shown in Figure 3.1, FLGP starts with population initialisation, where a number of programs/trees/individuals are randomly generated according to the new program structure, the new function and terminal sets. Then each individual is evaluated using the new fitness evaluation process. After fitness evaluation, a selection method and three genetic op-

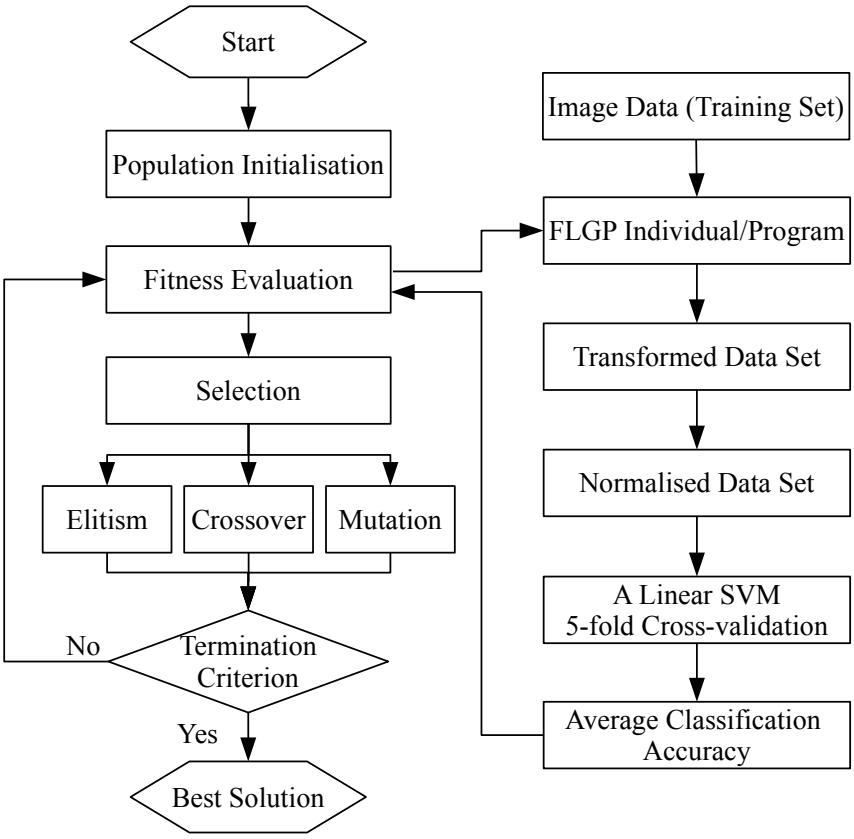


Figure 3.1: The overall feature learning process of FLGP.

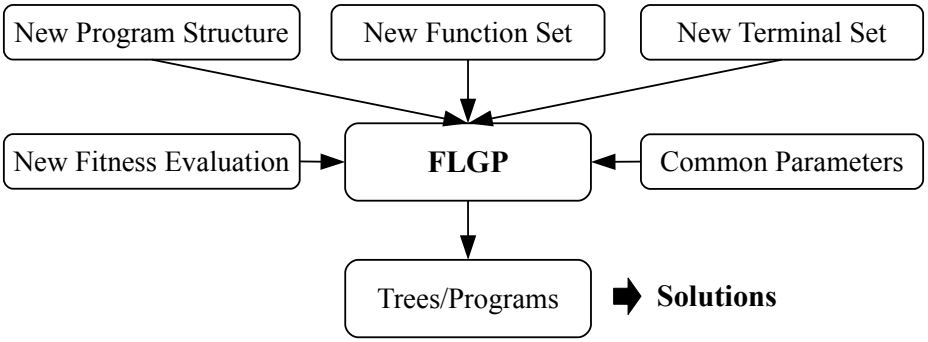


Figure 3.2: The new components and the basic configuration of the FLGP system.

erators, i.e., *elitism, crossover* and *mutation*, are used to obtain a new population for the next generation. The selection method selects individuals with better fitness values for crossover and mutation. The crossover and mutation operations change the nodes or branches of the FLGP trees to search for better trees/solutions. The evolutionary learning process is terminated when a pre-defined termination criterion is satisfied. If the termination criterion is not satisfied, the process of fitness evaluation and population generation repeat again. Otherwise, the evolutionary process ends and the best individual is returned.

During the evolutionary learning process, a fitness function is used to guide the search for the best individual. The right part of Figure 3.1 shows the new fitness evaluation process in FLGP. In this process, a training set, containing N $m \times l$ images $\{I_i\}_{i=1}^{N}$ and N labels $\{Y_i\}_{i=1}^{N}$, is used. Each FLGP individual/program, as a solution to feature extraction, transforms each image I_i to a feature vector F_i with the size of S. S is the number of the extracted features by the FLGP program. Then all the features $\{F_i\}_{i=1}^{N}$ are normalised and fed into a linear SVM together with the class labels $\{Y_i\}_{i=1}^{N}$ to perform classification. A linear SVM is employed because it is popular for image classification [197] and has fewer parameters compared with SVMs with other kernel functions. On image classification tasks, the classification accuracy is the most commonly used fitness function [138, 197]. To increase the generalization ability, stratified *k*-fold cross-validation is used for evaluating each individual and the mean accuracy of the *k* folds is set as the fitness value. $k = 5$ and $k = 10$ are commonly used for k-fold cross-validation [185]. To reduce the computational cost, we set k to 5 in FLGP.

Normalization of Extracted Features

The previous GP-GLF [44] method does not perform feature normalisation. However, the features produced by GP-GLF are the combination of features with different scales. For example, the uLBP features may be in

the range $[0,\ 100]$ and the SIFT features are in the range $[0,\ 1]$. This may lead to bias towards particular types of features such as uLBP features when using the combination of these features for classification. Therefore, in FLGP, the min-max normalisation method is used to rescale the output features F_i to $\overline{F}_i$, as shown in Equation (3.1).

$$\overline{F}_i = \frac{F_i - min(\{F_i\}_{i=1}^{N})}{max(\{F_i\}_{i=1}^{N}) - min(\{F_i\}_{i=1}^{N})}. \tag{3.1}$$

In addition to the above differences, the FLGP approach has a new representation (program structure), a new function set, and a terminal set, which will be described below.

3.2.2 New Program Structure

The proposed FLGP approach is based on STGP so that a new program structure is needed. The new program structure is extended from that of GP-GLF in [44] by improving its flexibility. The program structure of GP-GLF has a representation with a fixed tree depth, which limits the type of output features, i.e., the combination of global and local features. The new program structure of FLGP addresses these limitations by using a flexible structure to represent more possible ways of combining global and local features. The new program structure allows each solution to have a flexible tree depth and to produce various numbers of global and/or local features F_S.

The new program structure is shown in Figure 3.3 (a). It contains the tiers of input, region detection, feature extraction, feature concatenation, and output, where different functions are used at different functional tiers. The region detection tier aims to detect small regions of interest from a large input image. The region detection tier may exist or be absent in FLGP trees, which indicates that the solutions/trees of FLGP can be constructed without any region detection. This allows FLGP to also produce only global features. The feature extraction tier extracts global features

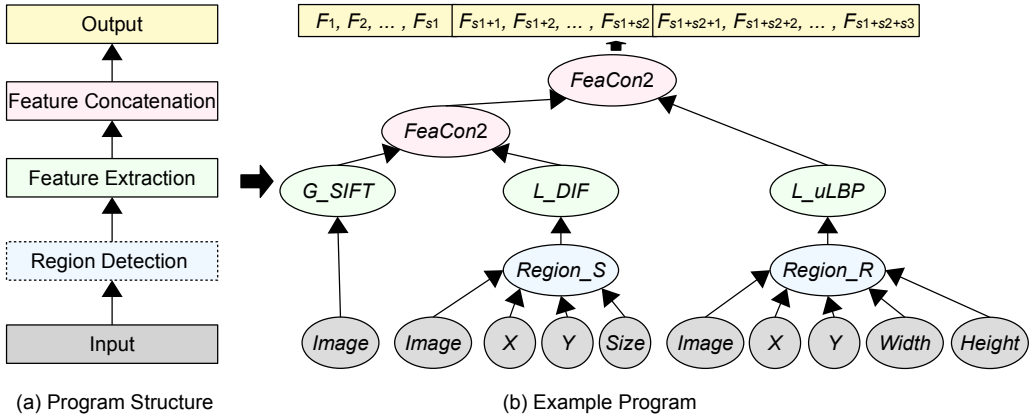


Figure 3.3: The new program structure of FLGP and an example program that describes a combination of global and local features.

from an input image or local features from the detected regions. Therefore, the feature extraction functions are developed in both the global and local scenarios. The feature concatenation tier concatenates the features from its child nodes to a feature vector. To further demonstrate the program structure, a typical example program/solution of FLGP is shown in Figure 3.3 (b), where different colours indicate inputs, outputs and different functions. In this program, there are region detection functions, i.e., $Region\_S$ and $Region\_R$, feature extraction functions, i.e., G_SIFT, L_DIF and L_uLBP, and feature concatenation function, i.e., $FeaCon2$.

The new program structure allows FLGP to produce three types of features. The first type is a combination of global and local features. As shown in the example program in Figure 3.3 (b), the output features are a combination of global SIFT features, local domain-independent features (DIF) and local uLBP features. The total number of the output features S equals $s_1 + s_2 + s_3$, where s_1 is the number of the SIFT features, s_2 is the number of the DIF features, and s_3 is the numbers of the uLBP features. These features are extracted by the G_SIFT, L_DIF and L_uLBP functions, respectively. The second type is a combination of local features. This can be achieved by building an FLGP tree where each input image

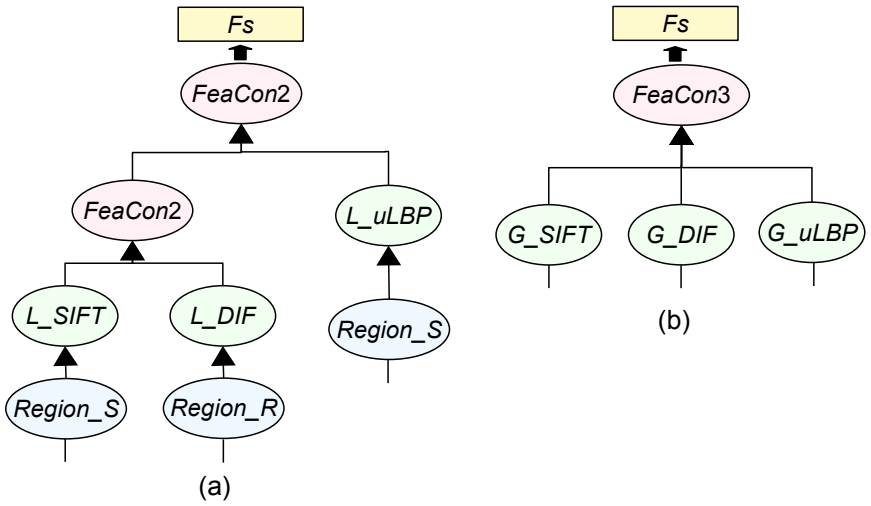


Figure 3.4: Two example program structures to describe (a) a combination of local features, and (b) a combination of global features.

(the $Image$ terminal) must connect with the region detection functions, as shown in Figure 3.4 (a). The third type is a combination of global features, which can be achieved by building an FLGP tree without using region detection functions, as shown in Figure 3.4 (b).

3.2.3 New Function Set

According to the new program structure, the function set of FLGP has three different types of functions: region detection functions, feature extraction functions and feature concatenation functions.

Region Detection Functions

The region detection functions are $Region\_S$ and $Region\_R$, which detect small square and rectangle regions from a large input image, respectively. The $Region\_S$ function takes an image $Image$ (I_i, the size is $m \times l$), X, Y, and $Size$ as inputs and returns a square region. The coordination of the top-left point of the region in $Image$ is (X, Y) and the size of the re-

gion is $Size$. Thus the detected region by $Region\_S$ is $Image[X : min((X + Size),\ m),\ Y : min((Y + Size),\ l)]$. Similar to $Region\_S$, the $Region\_R$ function detects the $Image[X : min((X + Width),\ m),\ Y : min((Y + Height),\ l)]$ region by taking the $Image$, X, Y, $Width$, and $Height$ as inputs. In the two functions, the $Image$, X, Y, $Size$, $Width$, and $Height$ are terminals, which will be described in the next subsection. The values of these terminals are randomly generated from pre-defined ranges and can be automatically optimised during the learning process. This indicates that the two functions can detect a region at an appropriate position with a suitable size.

Feature Extraction Functions

GP-GLF [44] uses seven descriptors, including the GLCM and Gabor descriptors. Since the GLCM, uniform LBP (uLBP) or Gabor descriptor is able to describe texture features, FLGP only uses uLBP instead of the three texture descriptors. Therefore, five representative descriptors, i.e., DIF [242], histogram (Hist), SIFT, HOG, and uniform LBP (uLBP), are used in FLGP as feature extraction functions to avoid a big search space. The five descriptors are representative methods to describe distribution, texture, shape, and appearance information of images. In FLGP, the five methods are developed in global and local scenarios. The methods in the global scenario are G_DIF, G_Hist, G_SIFT, G_HOG, and G_uLBP, which extract features from a whole image. The methods in the local scenario are L_DIF, L_Hist, L_SIFT, L_HOG, and L_uLBP, which extract local features from a detected region. The functions in the global scenario directly use the $Image$ terminal as their children node, while the functions in the local scenario employ the region detection functions as their children nodes. Each feature extraction function transforms an image or a region into a set of features F_s, where the number of features is s. The details of these functions are listed in Table 3.1. It is obvious that each function extracts a different number (s) of features from an image/region, as shown in the fourth column of Table 3.1.

Table 3.1: Feature Extraction Functions

Methods	Input	Output	#Features	Description
G_DIF/ L_DIF	1 Image/ Region	1 Vector	20	Domain independent features [242].
G_Hist/ L_Hist	1 Image/ Region	1 Vector	32	Histogram features of the image/region [99]. The number of bins is set to 32.
G_SIFT/ L_SIFT	1 Image/ Region	1 Vector	128	SIFT features. The image or detected region is considered as a keypoint [220].
G_HOG/ L_HOG	1 Image/ Region	1 Vector	Flexible	HOG features [60]. G_HOG/L_HOG extracts the mean value of each 20×20 / 10×10 grid with a step of 10 from a HOG image.
G_uLBP/ L_uLBP	1 Image/ Region	1 Vector	59	Uniform LBP histogram features [169]. In G_uLBP and L_uLBP, the radius is 1.5 and the number of neighbors is 8.

Feature Concatenation Functions

The feature concatenation functions are $FeaCon2$ and $FeaCon3$, which concatenate two feature vectors (F_{s_1} and F_{s_2}) and three feature vectors (F_{s_1}, F_{s_2} and F_{s_3}) to produce a feature vector F_S, respectively. The children nodes of the two functions can be the feature extraction functions and/or the feature concatenation functions. This allows FLGP to evolve trees with variable lengths to produce different numbers of features.

3.2.4 New Terminal Set

The new terminal set has six different terminals: $Image$, X, Y, $Size$, $Width$, and $Height$. The $Image$ terminal indicates the input grey-scale image, which is a two-dimension array ($m \times l$) with values in the range $[0,\ 1]$. The input image is normalised by dividing pixel values by 255, which is a commonly used method of normalising image data. The other terminals

of FLGP are ephemeral random constants. The X and Y terminals indicate the coordinates of the top-left point of a detected region in the image and are the parameters of the $Region\_S$ and $Region\_R$ functions. They are integers in the ranges $[0,\ m-20]$ and $[0,\ l-20]$, respectively. The $Size$, $Width$ and $Height$ terminals are the size or width and height of a detected region. Their values are in the range $[20,\ 50]$, indicating that the size of the detected region ranges from 20×20 to 50×50. It is noted that the value ranges of $Size$, $Width$ and $Height$ are smaller than that in [44] to narrow the search space.

3.3 Experiment Design

A number of experiments have been conducted to evaluate the performance of FLGP on feature learning. The experiments aim to investigate whether FLGP can achieve better performance than existing GP-based methods, the methods using manually extracted features, and simple CNN-based methods. This section describes the design of the experiments.

3.3.1 Benchmark Datasets

Eight different datasets of varying difficulty are used in the experiments to examine the effectiveness of FLGP. These datasets are FEI_1 [215], FEI_2 [215], VGDB [84], ORL [191], JAFFE [150], KTH [153], EYALE [137], and FS [79]. These datasets contain five types of image classification tasks, i.e., facial expression classification (FEI_, FEI_2 and JAFFE), object classification (EYALE and ORL), scene classification (FS), texture classification (KTH), and painting classification (VGDB).

Table 3.2 describes the details of these eight datasets, e.g., the image size, the number of classes, the number of images in the training and test sets. The original images in some datasets are resized and the colour images are converted to grey-scale images in order to reduce computational

Table 3.2: Dataset Properties

Dataset	Image size	# Classes	Training set	Test set
FEI_1	180×130	2	150	50
FEI_2	180×130	2	150	50
VGDB	200×200	2	247	83
ORL	92×112	40	280	120
JAFFE	128×128	7	140	73
KTH	100×100	10	600	210
EYALE	100×100	38	1,209	1,215
FS	100×100	13	1,928	1,931

time. These datasets are split into the training and test sets according to commonly used proportions. For the FEI_1, FEI_2, VGDB, and KTH datasets, 3/4 images are used to form the training sets and 1/4 images are used to form the test sets [116]. For the ORL dataset, which only has ten images per class, seven images per class are used for training and the remaining images are used for testing. For the JAFFE dataset, 20 images per class are used for training and the others are for testing. Since the EYALE and FS datasets are large, they are split into half and half to form the training set and the test set, respectively, according to [17].

3.3.2 Benchmark Methods

The benchmark methods are five GP-based methods, eight traditional methods using different types of manually extracted features, and three CNNs.

GP-based Methods

The five GP-based benchmark methods are GP-GLF [44], 2TGP [14], DIF+GP [242], Hist+GP, and uLBP+GP [8]. Since the new FLGP approach is an extension of GP-GLF, it is necessary to compare FLGP with GP-GLF. The 2TGP method automatically generates a high-level feature for classifica-

tion from the input image with simultaneous region detection, feature extraction and feature construction. The DIF+GP, Hist+GP and uLBP+GP methods construct high-level features for classification from pre-extracted features, i.e., 20 DIF features, 64 Hist features and 59 uLBP features, respectively. Since the 2TGP, DIF+GP, Hist+GP, and uLBP+GP methods are originally designed for binary image classification, they are only used for comparisons on binary image classification tasks, i.e., the FEI_1, FEI_2 and VGDB datasets. Because of the high computational cost of GP-GLF, it is too expensive to run it on the difficult multi-class classification tasks (probably needs several months to obtain all the results). Therefore, the comparisons of FLGP and GP-based methods, including GP-GLF, are only on the binary image classification tasks.

Traditional Methods

Eight traditional methods that use different well-known features are employed for comparisons. The features are DIF [242], Hist [99], GLCM [96], Gabor [144], SIFT [147], HOG [60], LBP [169], and uLBP [169] features. Most of these feature extraction methods have been introduced in Chapter 2. These features are fed into a linear SVM for classification. The DIF, SIFT, HOG, and ulBP features are extracted using the same functions as those employed in the function set of FLGP in the global scenario. Table 3.3 lists the detailed information of the eight feature extraction methods.

CNN-based Methods

Three CNN-based methods with different architectures are employed for comparisons. They are the LeNet [135], a five-layer CNN (CNN-5) [197] and an eight-layer CNN (CNN-8) [57]. LeNet is the very early version of CNN and its main parameters are the same as those in [135]. In CNN-5, there are two convolutional layers with 32 and 64 filters with a kernel size of 3×3, respectively, one max-pooling layer and two fully connected lay-

Table 3.3: Traditional Feature Extraction Methods for Comparisons

Methods	Description
DIF	Domain independent features the same as those in [242]
Hist	256 histogram features based on the pixel values of the image
GLCM	GLCM features [97]. Four different orientations are used and the contrast, dissimilarity, homogeneity, energy, correlation, and ASM are extracted from each GLCM
Gabor	Gabor bank features. 40 Gabor filters with eight different orientations at five scales are used [143]. The mean value of each 32×32 grid is extracted to form the features
SIFT	128 SIFT features. The whole image is used as a keypoint to generate a set of features [220]
HOG	A HOG image is generated by using the HOG descriptor with the same parameter settings in [60]. The mean value of each 20×10 grid is extracted from the HOG image
LBP	256 LBP histogram features [169]. In LBP, the number of neighbours is set to 8 and the radius is set to 1.5
uLBP	59 uniform LBP histogram features [169]

ers. The first fully connected layer has 128 neurons/nodes and the number of neurons of the final layer is the same as the number of classes. CNN-8 has four convolutional layers with 32, 32, 64, and 64 filters with a kernel size of 3×3, respectively, two max-pooling layers and two fully connected layers. These three methods use the popular rectified linear unit (ReLU) as the activation function and softmax for classification. Dropout is added after the pooling layer and the first fully connected layer with 0.25 and 0.5 probabilities, respectively, to avoid overfitting [201]. In these three CNN-based methods, the loss function is cross-entropy and the adaptive sub-gradient method is used to train the models [70]. The number of epochs

is set to 500, which allows the three methods to be fully trained on these datasets. It is noted that there are many deep CNN-based methods for image classification, which have been discussed in Chapter 2. The deep CNN-based methods require a large number of training instances. However, these benchmark datasets are not big and may not be sufficient to train the deep CNN-based methods with a large number of parameters. Instead of using deep CNNs, these three CNN-based methods with a small number of layers and parameters, which are able to be well trained on these benchmark datasets, are employed for comparisons.

3.3.3 Parameter Settings

Parameter settings for the proposed FLGP approach are the most commonly used settings in the community of GP [114], which are described in Table 3.4. Note that we aim to develop a general method that with common settings can achieve a good performance on a variety of image classification tasks. Therefore, we do not conduct parameter tuning for FLGP, although it could improve the performance. The other four GP-based benchmark methods use the same parameter settings as FLGP except for the population size. The population size for the four GP-based methods is 500, while FLGP and GP-GLF use a smaller size of 100 in order to reduce the computational cost. The crossover, mutation, and elitism rates are 0.8, 0.19, and 0.01, respectively. The selection method is Tournament selection with size 7. The tree generation method is ramped-half-and-half. The tree depth is between 2 and 6. The termination criterion for all the GP methods is reaching the maximum number of generations.

3.3.4 Test Process and Experiment Settings

The overall test process of FLGP and the eight traditional methods are the same. The FLGP solution is used as a feature extraction/description method to transform the training set and the test set. Before feeding the

Table 3.4: GP Run Time Parameters

Parameter	Value	Parameter	Value
Generations	50	Crossover rate	0.8
Population size	100	Mutation rate	0.19
Population generation	Ramped half-and-half	Elitism rate	0.01
Selection type	Tournament (size=7)	Tree depth	2–6

datasets to SVM, the min-max normalisation method is used to rescale the feature range of the training and test sets. Note that the normalisation for the test set is based on the min and max values of features in the training set. A linear SVM is employed to train a classifier using the transformed and normalised training set and the classifier is tested on the transformed and normalised test set. The linear SVM is employed because it is popular for image classification [197] and it has fewer parameters compared with SVMs with other kernel functions. The classification accuracy of the test set is reported.

The implementations of all the GP-based methods are based on the *DEAP* (*Distributed Evolutionary Algorithm in Python*) [86] package. The implementations of the CNN-based methods are based on Keras [57] package. The implementation of the linear SVM method is based on the *scikit-learn* [172] package with default parameter settings for simplification. In SVM, the penalty parameter (C) is 1. The experiments of these methods run independent 30 times with different random seeds and the classification results are reported.

3.4 Results and Discussions

This section discusses and compares the classification results of the proposed FLGP approach, the five GP-based methods, the eight traditional methods, and the three CNNs on the eight datasets with relatively large

image sizes. The classification results are listed in Tables 3.5, 3.6 and 3.7. The results include maximum accuracy (Max), mean accuracy and standard deviation (Mean $\pm$ St.dev). The Wilcoxon rank-sum test with a 5% significance level is used to compare FLGP with a benchmark method to show the significance of performance improvement. The symbols "+" or "–" in these tables indicate that FLGP is significantly better or significantly worse than the compared method. The symbol "=" indicates the performance of FLGP is similar to the compared method. In Tables 3.5, 3.6 and 3.7, each small block lists the results on one dataset, and the maximum classification accuracy is highlighted in bold. The final row of each block summarizes the overall results of the significance test.

3.4.1 Overall Classification Performance

As mentioned in Section 3.3.2, the GP-GLF method and the other four GP-based benchmark methods are only used for comparisons on binary image classification tasks. Thus, there are 16 benchmark methods on the FEI_1, FEI_2 and VGDB datasets and 11 benchmark methods on the remaining five datasets. From the final rows of Tables 3.5, 3.6 and 3.7, it is obvious that the FLGP approach achieves significantly better or similar performance in almost all the comparisons. Specifically, the FLGP approach obtains 97 "+", 5 "=" and 1 "–" in the total 103 comparisons. The proposed FLGP approach significantly outperforms all the benchmark methods on one binary classification dataset, i.e., FEI_2, and on five multi-class classification datasets, i.e., ORL, JAFFE, KTH, EYALE, and FS. FLGP performs significantly better than or similar to any of the 16 benchmark methods on the FEI_1 dataset.

The proposed FLGP approach gains the maximum accuracy and the maximum mean accuracy among all the methods on seven datasets except for the VGDB dataset. Specifically, FLGP improves the maximum accuracy by 11% on the JAFFE dataset, 8.1% on the KTH dataset, 7.8% on

the FS dataset, 6% on the FEI_2 dataset, 1.7% on the ORL dataset, and 0.5% on the EYALE dataset. The proposed FLGP approach improves the mean accuracy by 9% on the FS dataset, 6.9% on the KTH dataset, 2.5% on the FEI_2 dataset, 2.2% on the JAFFE dataset, and 1.3% on the ORL dataset. On the VGDB dataset, FLGP performs worse than the LBP method. The FLGP approach does not use the LBP descriptor (FLGP only uses uniform LBP to extract 59 features) so that it cannot achieve classification performance as good as the LBP features on this dataset. Overall, it is clear that FLGP is more effective than any of the benchmark methods on different types of image classification tasks.

The experimental results demonstrate the effectiveness of FLGP on feature learning. The main reasons to explain why FLGP is effective are the developments of the feature learning process, the new program structure, the function set and the terminal set in FLGP. With the utilization of five representative image descriptors in global and local scenarios, respectively, the proposed FLGP approach can extract high-level invariant features with the potential capability of increasing classification performance. The program structure enables FLGP to effectively search for optimal functions and terminals to form solutions that flexibly produce various numbers of global and/or local features. The overall feature learning process enables FLGP to find optimal solutions with high generalization ability.

3.4.2 Comparisons with Five GP-based Methods

Comparisons Between FLGP and GP-GLF

Table 3.5 shows that the proposed FLGP approach achieves significantly better performance than the GP-GLF method on the three binary classification datasets. Compared with GP-GLF, FLGP improves the mean accuracy by 6.7% on the FEI_1 dataset, 10.8% on the FEI_2 dataset and 9.8% on the VGDB dataset. The FLGP approach improves the maximum accuracy by

Table 3.5: Classification Accuracy (%) of the FEI_1, FEI_2 and VGDB Datasets

Methods	Max	Mean±St.dev	Max	Mean±St.dev	Max	Mean±St.dev
	FEI_1		**FEI_2**		**VGDB**	
2TGP	96.00	88.13±6.22+	94.00	85.47±5.98+	63.86	61.61±1.45+
DIF+GP	80.00	56.67±6.88+	72.00	60.33±8.38+	68.67	61.41±3.51+
Hist+GP	70.00	48.93±7.22+	60.00	48.80±6.14+	84.34	75.98±2.58=
uLBP+GP	66.00	50.87±7.48+	72.00	48.73±7.87+	79.52	69.40±4.36+
GP-GLF	96.00	89.07±3.92+	92.00	82.47±5.77+	74.63	65.12±4.73+
DIF	74.00	61.13±4.89+	72.00	62.80±6.10+	66.27	55.62±10.25+
Hist	54.00	48.13±3.38+	54.00	50.13±2.53+	62.65	62.21±0.79+
GLCM	50.00	49.67±0.75+	54.00	50.13±0.72+	62.65	53.33±9.80+
Gabor	82.00	71.60±7.87+	74.00	65.67±5.14+	63.86	56.02±8.28+
SIFT	82.00	82.00±0.00+	78.00	78.00±0.00+	60.24	60.24±0.00+
HOG	94.00	94.00±0.00+	88.00	88.00±0.00+	57.83	57.23±0.68+
LBP	68.00	62.47±3.49+	66.00	57.60±3.56+	**84.34**	**80.56±3.23**–
uLBP	64.00	56.87±5.18+	56.00	51.93±2.34+	81.93	71.48±8.14=
LeNet	**98.00**	94.40±1.96=	94.00	90.80±1.83+	65.06	58.07±4.84+
CNN-5	**98.00**	95.60±1.50=	90.00	85.00±3.00+	65.06	61.45±2.09+
CNN-8	**98.00**	94.20±2.09=	94.00	90.02±2.27+	61.45	56.87±4.57+
FLGP	**98.00**	**95.80±3.24**	**100.0**	**93.27±3.78**	81.93	74.94±3.72
Overall		**13+, 3=**		**16+**		**13+, 2=, 1–**

2% on the FEI_1 dataset, 8% on the FEI_2 dataset and 7.3% on the VGDB dataset. From the results, it is clear that FLGP is more effective than GP-GLF for feature learning to image classification. As mentioned in Section 3.2, FLGP has been developed to address the limitations of the previous GP-GLF method. The experimental results show that this goal was successfully achieved. FLGP is more effective than GP-GLF by producing three types of features, i.e., a combination of global and local features, a combination of global features, a combination of local features. This allows FLGP to automatically find suitable types and numbers of features to improve the classification accuracy for a given task.

Comparisons with The Other Four GP-based Methods

From Table 3.5, it is noticeable that the proposed FLGP approach achieves significantly better results in 11 comparisons and similar results in 1 comparison out of the total 12 comparisons. Importantly, compared with the four GP-based methods (i.e., 2TGP, DIF+GP, Hist+GP, and uLBP+GP), the proposed FLGP approach improves the mean accuracy by over 7% on the FEI_1 and FEI_2 datasets. Moreover, FLGP achieves the maximum accuracy on the FEI_1 and FEI_2 datasets. On the VGDB dataset, FLGP obtains a similar performance to Hist+GP. The Hist+GP method uses 256 histogram features as inputs while the G_Hist/L_Hist function in FLGP only extracts 32 histogram features, which may be the reason why FLGP cannot achieve classification performance as good as the Hist+GP method.

By automatically extracting a set of global and/or local features from raw pixels, FLGP achieves significantly better performance than DIF+GP, Hist+GP and uLBP+GP in most comparisons. The results show the potential of GP in feature learning from raw pixels. The 2TGP method can learn features from raw images. However, FLGP is more effective than 2TGP on these three datasets. The 2TGP method learns only one high-level feature from an image to perform classification. In contrast, FLGP learns a set of high-level features from an image, which is more effective for classification.

3.4.3 Comparisons with Eight Traditional Methods

Tables 3.5, 3.6 and 3.7 show that FLGP achieves significantly better results in 62 comparisons out of the total 64 comparisons. Specifically, FLGP performs significantly better than any of the eight benchmark methods on seven datasets except for VGDB. Compared with the eight traditional methods (i.e., DIF, Hist, GLCM, Gabor, SIFT, HOG, LBP, and uLBP), FLGP improves the mean accuracy by 10.8% on EYALE, 9.1% on FS, 7.1% on JAFFE, and 6.9% on KTH. Moreover, FLGP obtains the maximum accuracy

Table 3.6: Classification Accuracy (%) of the ORL, JAFFE and KTH Datasets

Methods	Max	Mean±St.dev	Max	Mean±St.dev	Max	Mean±St.dev
	ORL		**JAFFE**		**KTH**	
DIF	85.00	85.00±0.00+	35.62	35.62±0.00+	56.19	56.19±0.00+
Hist	97.50	97.50±0.00+	19.18	19.18±0.00+	51.43	51.43±0.00+
GLCM	2.50	2.50±0.00+	15.07	15.07±0.00+	23.33	23.33±0.00+
Gabor	59.17	57.06±0.88+	46.58	43.15±1.57+	44.29	42.87±0.71+
SIFT	98.33	98.33±0.00+	73.97	73.97±0.00+	81.43	81.43±0.00+
HOG	96.67	96.67±0.00+	72.60	72.60±0.00+	51.43	51.37±0.16+
LBP	87.50	87.50±0.00+	21.92	21.92±0.00+	87.62	87.62±0.00+
uLBP	94.17	94.17±0.00+	23.29	23.29±0.00+	80.95	80.95±0.00+
LeNet	93.33	89.92±1.88+	79.45	68.93±6.95+	78.57	72.00±6.43+
CNN-5	97.50	96.33±0.77+	80.82	78.90±1.26+	84.29	81.48±1.84+
CNN-8	96.67	94.17±1.79+	61.64	52.47±6.15+	82.38	80.71±1.54+
FLGP	**100.0**	**99.58±0.70**	**91.78**	**81.14±4.69**	**95.71**	**94.52±0.90**
Overall		**11+**		**11+**		**11+**

on seven datasets except for VGDB. FLGP has an increase by over 10% in terms of the maximum accuracy on the FEI_2, JAFFE, EYALE, and FS datasets. The VGDB dataset is the only one which FLGP performs worse than one of the eight traditional methods, i.e., LBP. The FLGP approach does not use the LBP descriptor so that it cannot achieve classification performance as good as the LBP features on VGDB.

The experimental results show that the features learned by FLGP are more effective than the well-known hand-crafted features for image classification. Using traditional methods often requires domain expertise to extract a set of effective features for classification. The new FLGP approach can automatically learn features from the images. The design of FLGP enables it to automatically find the best combination of global and/or local features, which are effective for image classification. Compared with the features used in the eight methods, the features learned by FLGP are more

Table 3.7: Classification Accuracy (%) of the EYALE and FS Datasets

Methods	Max	Mean±St.dev	Max	Mean±St.dev
	EYALE		**FS**	
DIF	26.42	26.42±0.00+	33.51	33.51±0.00+
Hist	11.03	11.03±0.00+	21.23	21.23±0.00+
GLCM	5.10	4.97±0.12+	13.88	13.88±0.00+
Gabor	36.71	36.27±0.21+	22.68	22.28±0.19+
SIFT	88.40	88.40±0.00+	63.13	63.13±0.00+
HOG	74.32	74.32±0.00+	30.81	30.81±0.00+
LBP	46.42	46.42±0.00+	62.45	62.45±0.00+
uLBP	56.13	56.13±0.00+	66.13	66.13±0.00+
LeNet	92.35	89.38±1.58+	54.17	51.05±2.35+
CNN-5	99.26	98.57±0.52+	58.88	55.35±1.39+
CNN-8	90.86	88.19±1.03+	69.24	66.21±1.98+
FLGP	**99.75**	**99.21±0.38**	**77.01**	**75.22±0.65**
Overall		**11+**		**11+**

flexible in type and number. FLGP can learn various numbers of features of three types, i.e., a combination of global and local features, a combination of global features and a combination of local features.

3.4.4 Comparisons with Three CNN-based Methods

Compared with LeNet, CNN-5 and CNN-8, FLGP achieves significantly better performance on seven datasets and similar performance on the remaining one, the FEI_1 dataset. Importantly, FLGP improves the mean accuracy by over 13% on the VGDB and KTH datasets, and by over 9% on the FS dataset compared with the three CNNs. Surprisingly, CNN-8 performs worse than CNN-5 on six datasets except for the FEI_2 and FS datasets, which indicates that an increase in the depth of CNNs cannot guarantee an increase in classification accuracy. A more complex model may require more training instances/samples in order to obtain satisfactory results, but the numbers of instances for these datasets are not large. Compared with

the three methods with pre-defined model complexity, the flexible representation allows FLGP to evolve solutions with various depths, which is more flexible for solving different image classification tasks.

The results indicate that the features learned by FLGP are more effective than those by the three CNNs with different architectures for image classification. Compared with the three CNNs, FLGP uses a simpler program structure and a set of functions and terminals but achieves better performance on different types of datasets. FLGP can learn various numbers and types of features, which is more flexible than the three CNNs. FLGP can learn not only global features but also local features from automatically detected regions. However, it may be difficult for the three CNNs to achieve this.

3.5 Further Analysis

This section further analyses the convergence behaviours of the FLGP method. It also analyses the best solutions/programs/trees evolved by FLGP to fully understand why it achieves good performance and to demonstrate good interpretability.

3.5.1 Convergence Behaviour of FLGP

The convergence behaviours of the proposed FLGP approach on the FEI_1, ORL, KTH, and EYALE datasets are shown in Figure 3.5. The X-axis indicates the number of generations and the Y-axis indicates the fitness value, i.e., the classification accuracy (%). This figure shows that the proposed FLGP approach can converge within 50 generations on these four datasets. On the ORL dataset, the proposed FLGP approach converges faster and reaches the maximum accuracy of 100% at very early generation, specifically, within 10 generations. On the other datasets, such as KTH, the proposed FLGP approach converges slowly and shows its search ability even

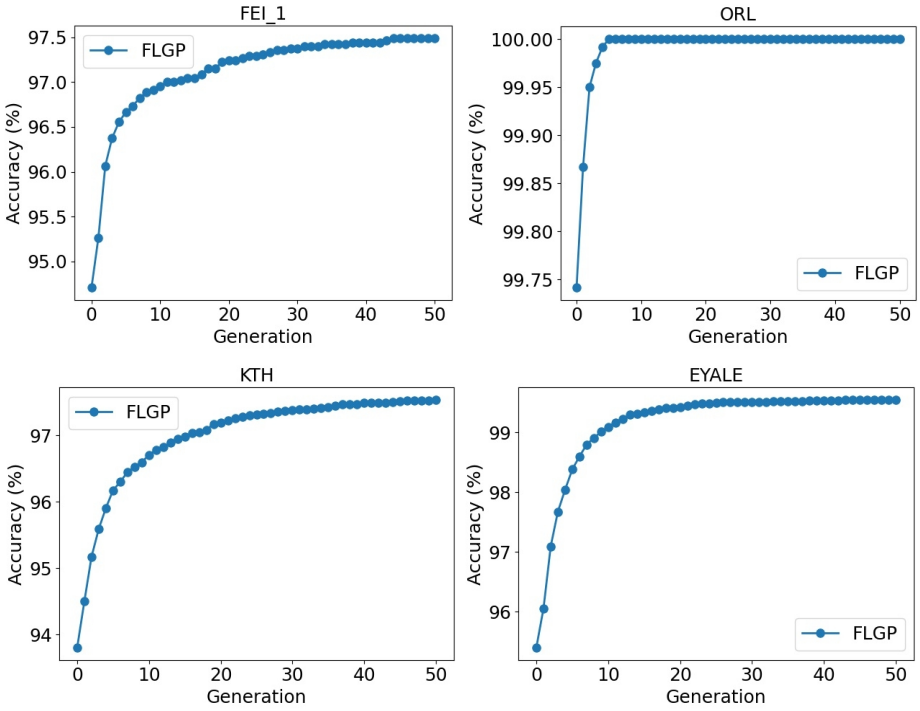


Figure 3.5: Convergence curve of the proposed FLGP approach on the FEI_1, ORL, KTH, EYALE datasets.

at the final several generations. But the convergence of FLGP becomes stable after 30 generations by achieving a high fitness value.

To sum up, this analysis shows that the proposed FLGP approach is able to converge to a high fitness value within the given number (50) of generations. Additionally, it is clear that the convergence behaviours of FLGP vary with the datasets.

3.5.2 Evolved Programs

An Example Program on the FEI_2 Dataset

An example program/tree of FLGP on the FEI_2 dataset is visualised in Figure 3.6. This example program achieves 100% classification accuracy on both the training and test sets. Two example images from the two

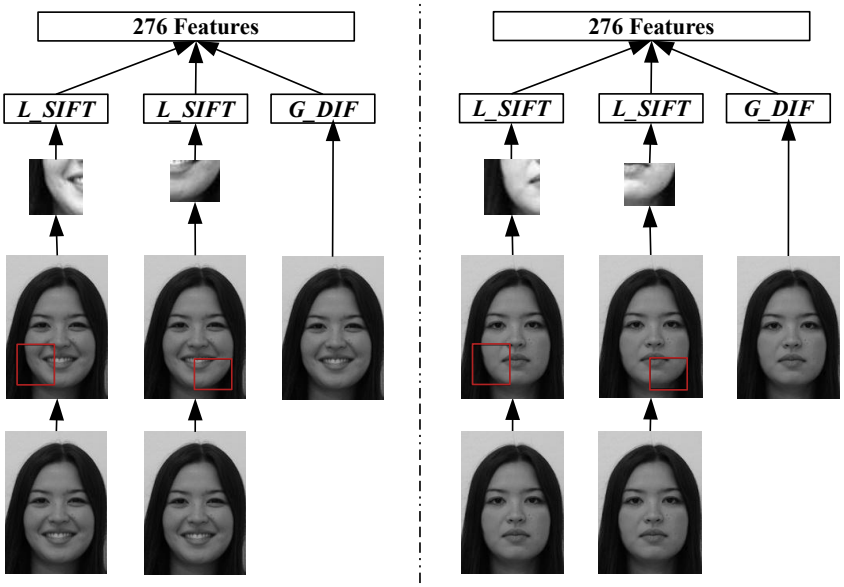


Figure 3.6: An example program evolved by FLGP on the **FEI_2** dataset.

classes (smile and natural) are used for visualisation to show what and how features are extracted. This solution detects a 50×50 region using $Region\_S$ and a 38×46 rectangle region using $Region\_R$ from the input image. From each detected region, the example program extracts 128 SIFT features using L_SIFT. Together with the extracted 20 DIF features from the whole image by G_DIF, the example program is able to produce 276 features from an input image.

From Figure 3.6, it can be seen that the $Region\_S$ function detects the chin and mouth area of the left face and the $Region\_S$ function detects a similar area of the right face. It is obvious that the two regions capture the most discriminative information between the two different classes. For example, the detected areas contain the teeth in the happy face image but not in the natural face image and capture the difference in mouth shapes between the two expressions.

This example program finds a combination of local SIFT features and global DIF features for classification. The combined features are more effective for classification than the individual global DIF and SIFT features. The traditional method using DIF features only achieves a maximum accuracy of 72% and the method using SIFT features only achieves a maximum accuracy of 78% on the FEI_2 dataset. The example program improves the classification performance by detecting the regions of interest and extracting meaningful local features from the detected regions. The analysis shows that FLGP detects informative regions and extract discriminative global and/or local features for classification.

Example Programs on the ORL Dataset

Three different example programs/trees of FLGP on the ORL dataset are shown in Fig 3.7. These example programs achieve 100% accuracy on both the training set and the test set. These three example programs detect one, two and two regions from the input image, respectively. The example images with detected regions are shown in Figure 3.8.

Using the three example programs, different types of global and local features are extracted to form the final output features. The first program extracts global Hist, uLBP and SIFT features from an input image and local uLBP features from a detected 30×46 region. The detected region includes the mouth area of the face, which is discriminative under two different expressions. The second program is able to produce a combination of global SIFT features, local Hist features from a 24×29 region and local uLBP features from a 44×46 region (the size of the detected region is 46×46, but part of the region is outside the input image). From Figure 3.8, it can be found that the chin and mouth areas of the face are detected, where discriminative local features can be extracted from. The third program is able to extract global DIF, HOG and uLBP features, local uLBP features from a 27×26 region and from a 27×30 region (the size of the detected region is 30×30, but part of the region is outside the input image), as shown in

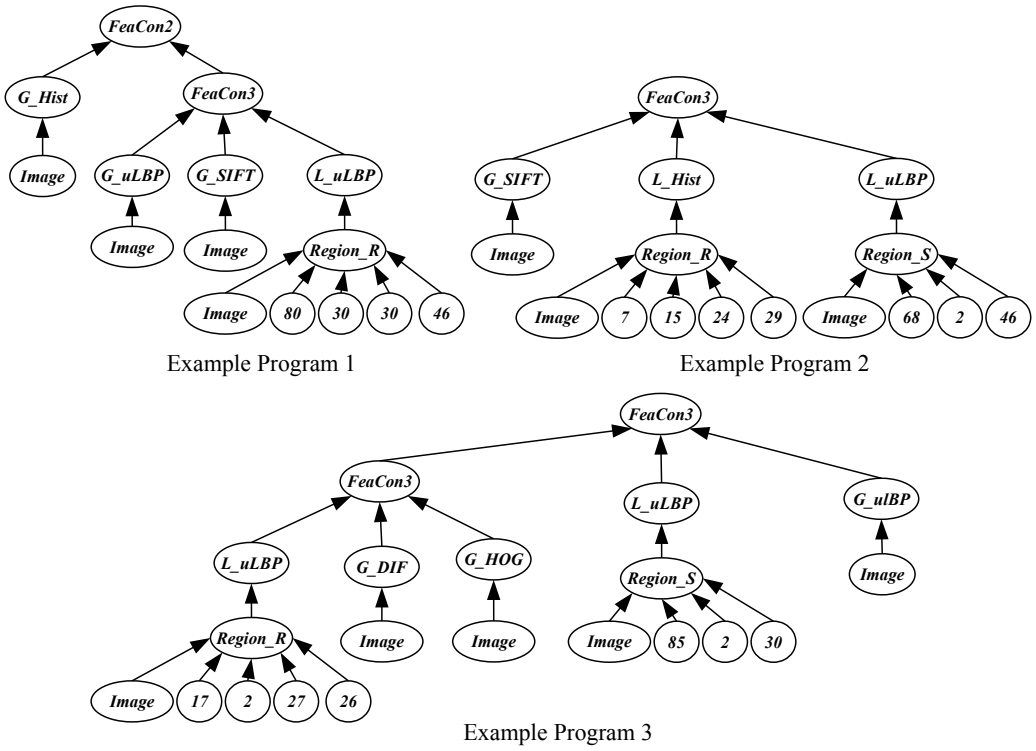


Figure 3.7: Three example programs evolved by FLGP on the **ORL** dataset.

Figure 3.8.

By analysing the three different programs, some patterns can be found. Both the first and second programs extract the Hist, uLBP and SIFT features rather than the HOG and DIF features. The third program extracts the uLBP, DIF and HOG features rather than the SIFT and Hist features. This pattern shows that different combinations of global and local features can achieve the same classification performance. Generally, without domain expertise, it is very difficult to find such a combination of features for image classification. FLGP is able to automatically find the most effective combinations. Moreover, the flexible representation and the population-based search enable FLGP to have good global search ability to find multiple optimal solutions (combinations of features) for a given task. From the three example programs, it can be found that the uLBP features are one of

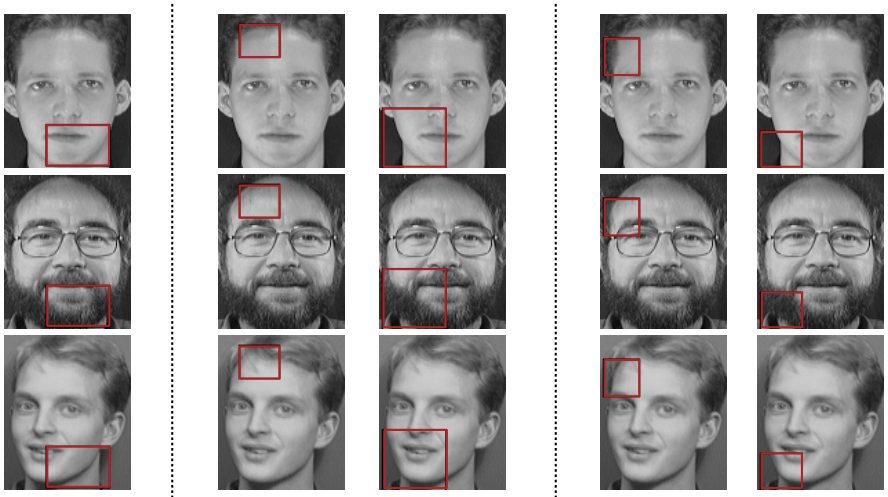


Figure 3.8: The detected regions by the three example programs showed in Figure 3.7 on the **ORL** dataset.

the most important and meaningful features in the face images of ORL. By visualising the programs evolved by FLGP, it is very clear what types of features are extracted and why they are effective.

3.5.3 Analysis on the Feature Extraction Functions

To further analyse FLGP, we recorded the ten best FLGP programs/trees of each run (totally 300 programs) on each dataset. The frequency of occurrences of the ten feature extraction functions in the global (G_DIF, G_Hist, G_SIFT, G_HOG and G_uLBP) and local (L_DIF, L_Hist, L_SIFT, L_HOG, and L_uLBP) scenarios in these programs are ranked. The results of the ranking are listed in Table 3.8, where 1 indicates the most frequently used function and 10 indicates the least frequently used function in these programs.

From Table 3.8, it is clear that the frequency-rank of the feature extraction functions varies with the datasets. Specifically, the L_SIFT function is the most frequently used on the FEI_1, FEI_2, JAFFE, and EYALE datasets,

Table 3.8: Ranking of All the Feature Extraction Functions on Each Dataset

Function	FEI_1	FEI_2	VGDB	ORL	JAFFE	KTH	EYALE	FS
G_DIF	5	9	7	7	7	5	8	4
G_Hist	10	10	3	5	9	6	5	6
G_SIFT	8	5	8	1	2	2	3	2
G_HOG	3	4	10	4	5	7	9	9
G_uLBP	9	7	1	2	10	1	4	1
L_DIF	6	6	5	9	3	4	7	7
L_Hist	4	8	2	6	8	3	6	8
L_SIFT	1	1	9	10	1	10	1	10
L_HOG	7	3	4	8	4	9	10	5
L_uLBP	2	2	6	3	6	8	2	3

the G_uLBP function is the most frequently used on the VGDB, KTH and FS datasets, and the G_SIFT function is the most frequently used on the ORL dataset. Moreover, the frequently used functions on one dataset may be less frequently used on the other datasets. For example, L_SIFT is the most frequently used on four datasets but it is the least frequently used on the ORL, KTH and FS datasets. The G_Hist function is the least frequently used on the FEI_1, FEI_2 and JAFFE datasets, but it is frequently used on the VGDB dataset. This confirms the difficulty of feature extraction. In contrast, FLGP automatically finds the best feature extraction methods or combinations of them to extract features.

It can be seen from Table 3.8 that FLGP learns more local features than global features on most face datasets, i.e., FEI_1, FEI_2, JAFFE, and EYALE, which confirms that local features are more effective for object classification. In contrast, FLGP learns more global features than local features on the non-object datasets, i.e., VGDB, KTH and FS, as global features are more effective. The analysis shows that FLGP learns the best feature extraction functions or combinations of them to extract effective global and/or local features for image classification.

3.6 Chapter Summary

In this chapter, a GP-based feature learning approach was developed to automatically learn effective global and/or local features for image classification. A novel program structure, a new function set with five existing image descriptors in the global and local scenarios, and a new terminal set were proposed. To effectively learn discriminative features, a new feature learning process and a new fitness evaluation process were developed in FLGP. These designs allowed FLGP to automatically learn various numbers of global and/or local features from different types of images. The performance of FLGP was examined on eight different image classification datasets of varying difficulty and compared with a number of benchmark methods to show its effectiveness.

This chapter showed that the use of image descriptors in GP could help to learn effective features for image classification. The results showed that FLGP achieved significantly better performance in almost all the comparisons on eight different image classification datasets. The results confirmed the effectiveness of FLGP in learning effective features for image classification. FLGP has a flexible program structure, a new function set and a new feature learning process to learn various types of features. The comparisons of FLGP and GP-GLF showed that the classification performance was improved in FLGP with these new designs. Compared with the other GP-based methods that constructed one high-level feature, FLGP was more effective for image classification by automatically learning a set of features from images. The results showed that the features learned by FLGP were more effective than many manually extracted features, i.e., achieving significantly better classification results in almost all the comparisons. The comparisons with the three CNN-based methods demonstrated that FLGP was more effective by evolving small solutions of variable lengths. Further analysis of the example solutions of FLGP confirmed the good interpretability of the solutions and revealed that FLGP learned

discriminative features using a simple solution. The analysis showed that FLGP detected informative regions from a large input image and found the most effective feature extraction functions to extract features from the regions/images.

This chapter showed the potential of GP with existing image descriptors in feature learning and image classification. The FLGP approach learned global and/or local features from relatively large images. Besides the image descriptors employed in FLGP, many other image-related operators, such as image filters, can be employed in GP to achieve effective feature learning. However, no GP-based methods have been developed to simultaneously employ image descriptors and filters to feature learning for image classification. Therefore, the next chapter will investigate the use of image filters and descriptors in GP to automatically learn features for image classification.

Chapter 4

GP with Image-Related Operators for Feature Learning

4.1 Introduction

Image-related operators include image descriptors, e.g., Histogram of Oriented Gradients (HOG) [60], Scale-Invariant Feature Transform (SIFT) [147] and Local Binary Patterns (LBP) [169], and image filters, e.g., Gaussian filter, mean filter, min filter, Laplacian filter, and Sobel filter. The previous chapter investigated the use of image descriptors in GP for feature learning and achieved promising results in image classification. Existing GP-based algorithms have employed image filters as functions for feature learning [197]. However, no GP-based methods have been developed to simultaneously employ image descriptors and filters to learn features for image classification. To effectively use these image-related operators in GP, a program structure typically needs to integrate different functions and terminals into a single tree. Several GP methods with different program structures, e.g., multi-tier [27], two-tier [14] and multi-layer [197], have been developed, but these program structures are often restricted somehow and cannot be used when new image-related operators are employed/needed. For example, the program structure in [27] has filtering,

aggregation, and classification layers, which indicates that the output features should be transformed via these three layers. This may not be effective and flexible in learning features that need various types of transformations. In addition, very few GP-based feature learning methods have been examined on large/big benchmark datasets and compared with state-of-the-art algorithms for image classification.

4.1.1 Chapter Goals

The overall goal of this chapter is to develop a new GP-based approach with image-related operators and a flexible program structure to learn features for different image classification tasks. The new approach is called FGP in short. To achieve this, a flexible program structure, a new function set with image-related operators, and a new terminal set are developed in FGP. The new approach is evaluated on 12 benchmark datasets of varying difficulty and compared with a large number of state-of-the-art methods. Further analysis is conducted to provide an in-depth understanding of the new approach. Specifically, this chapter will investigate

- how different image-related operators are employed in FGP to learn various numbers and types of features by developing a new program structure, a new function set and a new terminal set;
- whether FGP can achieve better classification performance than the methods using raw pixels, the methods using well-known features, and two CNN-based methods;
- whether FGP can achieve better classification performance than a large number of state-of-the-art algorithms on datasets with a large number of training and test instances;
- whether FGP can learn various types and numbers of features from different images; and

- whether the solutions evolved by FGP and the learned features can be easily explained to provide insights on why they can achieve good results.

4.1.2 Chapter Organisation

The remainder of this chapter is organised as follows. Chapter 4.2 describes the details of the proposed FGP approach. Chapter 4.3 designs the experiments. The results are discussed and compared in Chapter 4.4. Chapter 4.5 further analyses the proposed FGP method in terms of the convergence curve, the learned features and the evolved trees/programs. Chapter 4.6 concludes this chapter.

4.2 The Proposed Approach

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

4.2.1 Overall Algorithm

The framework of the proposed FGP approach is outlined in Algorithm 2. The FGP approach starts with population initialisation, where N (population size) individuals/trees are randomly generated using a commonly used tree generation method: *ramped half-and-half*. Each FGP individual is built by selecting functions from the new function set to construct internal/root nodes and selecting terminals from the new terminal set to construct the leaf nodes. Each individual is then evaluated through the fitness evaluation process to have a fitness value. After fitness evaluation, the best individual and a hash table, $Cache\_Table$, are updated. The $Cache\_Table$ is used to avoid evaluating the individuals that have been already evaluated in past generations. At each generation, the selection method and

three genetic operators, i.e., *subtree crossover, subtree mutation* and *elitism*, are employed to generate a new population to replace the current one. The evolutionary process is terminated when the maximum number of generations is reached. Finally, the best individual is returned.

Algorithm 2: Framework of FGP

Input : X_train: the training images; Y_train: the labels of the training images.
Output : $Best\_Individual$: the best individual.

$Cache\_Table \leftarrow \emptyset$;
$P_0 \leftarrow$ Initialise the population using the *ramped half-and-half* method according to the new program structure, the new function set and the terminal set;
Evaluate P_0 using Algorithm 3;
Update $Best\_Individual$ and $Cache\_Table$;
$g \leftarrow 0$;
while $g < G$ **do**
    $I \leftarrow$ The best individuals of P_g using elitism operator;
    $S \leftarrow$ Individuals selected from P_g using tournament selection;
    $O_{g+1} \leftarrow$ Offspring generated from S using subtree crossover and subtree mutation operators;
    Evaluate the fitness of each individual p in O_{g+1} using Algorithm 3;
    $P_{g+1} \leftarrow O_{g+1} \cup I$;
    Update $Best\_Individual$ and $Cache\_Table$;
    $g \leftarrow g + 1$
end
Return $Best\_Individual$.

A new fitness evaluation process is developed in FGP to evaluate each individual, as described in Algorithm 3. On image data, GP is often known as a computationally expensive method, especially when the number of instances is large. To avoid evaluating the same subtrees, subtree caching strategy has been developed in GP on image data [184]. Inspired by this, a hash table $Cache\_Table$ is employed in FGP to store individuals and their

Algorithm 3: Fitness Evaluation

Input : $Cache\_Table$: the hash table to store evaluated individuals and their fitness values; X_train: the training images; Y_train: the labels of the training images; p: the individual to be evaluated.

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

```
if p in Cache_Table then
    f(p) ← the fitness value of p in Cache_Table;
else
    Use p to transform X_train into features X_features;
    Normalise X_features using the min-max normalisation
     method;
    Feed normalised X_features and Y_train into a linear SVM
     using stratified k-fold cross-validation;
    f(p) ← average test accuracy of k folds
end
Return f(p).
```

fitness values so that repeated individual can be directly assigned a fitness value without evaluating it on the training data. To balance the search time in $Cache\_Table$ and the evaluation time of an individual, only the N_c best unique individuals and the previous population (P_g) are stored in $Cache\_Table$. With the $Cache\_Table$, the new fitness evaluation process is described in Algorithm 3. It starts by checking whether the individual is in the $Cache\_Table$. If the individual is in the $Cache\_Table$, the fitness value is directly assigned to the individual. Otherwise, the individual is evaluated on a training set using a linear SVM. The linear SVM is chosen because it is commonly used for image classification [197]. In this process, the FGP individual transforms each image in X_train to a number of features to form $X\_features$. Then $X\_features$ is normalised using the min-max normalization method and fed into a linear SVM using the stratified *k*-fold cross-validation method. The stratified *k*-fold cross-validation method splits the dataset ($X\_features$ and class labels) into *k* folds by preserving the class

ratio. Each time $k-1$ folds are used to build a SVM classifier and the classifier is tested on the remaining one fold. The average test accuracy of the *k* folds is set as the fitness value for the individual. In FGP, *k* is set to be 5 instead of 10 used in [197] to reduce the computational cost of the fitness evaluation.

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

4.2.2 Flexible Program Structure

A flexible program structure is developed in FGP to integrate functions and terminals into a single tree. The development of the new program structure is based on three motivations. First, in state-of-the-art GP methods on feature learning in [146, 197], the program structure has two main components, the filtering layer and the pooling layer, connected in a bottom-up manner. This program structure is fixed, which may not be effective for learning invariant features as learned by CNN using multiple layers' transformations. To this end, we relax this constraint to design a more flexible program structure, which allows FGP to evolve programs with multiple filtering and pooling layers. Second, integrating feature extraction functions into GP for feature learning has achieved promising results, as shown in Chapter 3. However, this method has a limitation of fixed tree depth, and the learned features may not be invariant to noise without filtering/denoising process. To address this, a flexible filtering/pooling layer is added between the input layer and the feature extraction layer. Third, hybrid features/representations, i.e., combined features extracted by filtering/pooling and features extracted by traditional feature extraction methods, have been seen in CNN-based methods with promising per-

formance [50]. But no existing GP methods can achieve this. Therefore, the flexible program structure of FGP can combine the two different types of features to produce hybrid features.

The program structure of FGP is based on STGP [161], which has types constraint on functions (input types and output types) and terminals (output types). In STGP, each function can only use particular functions or terminals as child nodes, where its input types must be the same as the output types of its child nodes. Based on STGP, a program structure is developed in FGP to integrate functions and terminals of different types into trees. The new program structure and three typical example programs are shown in Figure 4.1. The new program structure has several different layers, i.e., an input layer, filtering layers, pooling layers, a feature extraction layer, a concatenation layer, and an output layer. The input layer feeds the image and ephemeral random constants into the FGP system. The filtering layer performs filtering operations or other operations on the image. The pooling layer conducts max-pooling to the image with size reduction, which is in contrast to that in [146, 197]. The feature extraction layer extracts features from the image using several well-known feature extraction methods. The concatenation layer concatenates/combines features from different processes, i.e., filtering/pooling and feature extraction, into a feature vector to form the output of the FGP system.

More importantly, as shown in Figure 4.1, the layers drawn with a dashed line are flexible, indicating that they may be absent in an FGP program. These flexible layers allow the FGP program to have multiple filtering and pooling layers to extract features, which are similar to those in CNNs. The layers that are drawn with a solid line are fixed layers to make sure that there are feature transformations from the input to the output. Using this flexible program structure, three typically different types of features can be produced by FGP. The first type combines features from the feature extraction process as the *Example Program 1* shown in Figure 4.1. The second type combines features from the filtering and/or pool-

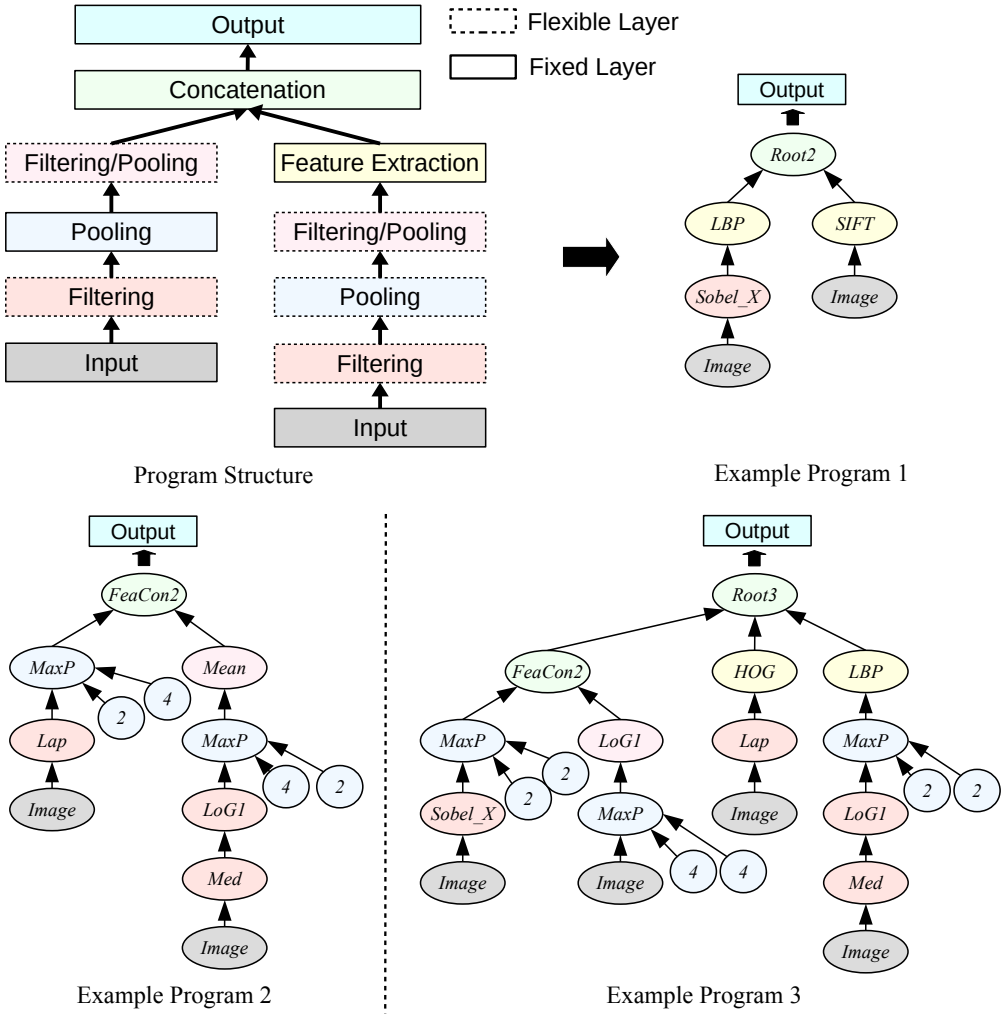


Figure 4.1: The program structure of the FGP approach and three typical example programs.

ing processes as the *Example Program 2* shown in Figure 4.1. The third type combines features from the feature extraction process and the filtering and pooling processes, as shown in the *Example Program 3* in Figure 4.1.

This program structure allows FGP to evolve shallow/small trees that contain a few functions or to evolve deep/large trees with multiple pooling and/or filtering layers. With this program structure, FGP can produce various types and numbers of features, which are flexible for solving dif-

ferent image classification tasks. Associated with this program structure, a number of functions and several terminals are employed in FGP, which will be described in the following subsections.

4.2.3 New Function Set

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

Filtering Functions

There are 19 functions employed in the filtering layer of FGP, as listed in Table 4.1. The Gau function takes an image and standard deviation σ as inputs and returns an image convolved by a Gaussian kernel. $GauD$ has three parameters, i.e., standard deviation σ, o_1 and o_2. The o_1 and o_2 represent orders of the derivative along the X and Y axis, respectively. The $Gabor$ filter is generated by a Gabor wavelet function. It has θ and f as parameters, which indicate the orientation of the kernel and the wavelength ($\lambda = 1/f$) of the sinusoid function in the Gabor wavelet function. The Lap function is generated by discretising and approximating the Laplacian operator, and it can detect flat areas or edges. The $LoG1$ and $LoG2$ functions convolve the Laplacian filter by the Gaussian function, which reduces noise in the image. The standard deviation of the Gaussian function in $LoG1$ and $LoG2$ is set as 1 and 2, respectively. Among these filters, the Gau, Med and $Mean$ filters are often employed for image denoising and smoothing. The filters, including $GauD$, Lap, $LoG1$, $LoG2$, $Sobel$, $SobelX$, and $SobelY$, can detect edges or flat areas in the image.

The kernel sizes for the $Mean$, Med, Min, and Max functions are 3×3,

Table 4.1: Filtering Functions

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

which is a commonly used kernel size. The kernel sizes for the other filters are based on their parameters or default settings. For example, the kernel sizes of the Gau and $GauD$ functions are related to their parameter σ; and the kernel size of the $Sobel$, $SobelX$ and $SobelY$ functions is 3×3.

Besides the above filters, the $LBP\text{-}F$, $HOG\text{-}F$, $W\text{-}Add$, $W\text{-}Sub$, $ReLU$, and $Sqrt$ functions listed in Table 4.1 are also employed. The $LBP\text{-}F$ and $HOG\text{-}F$ functions return a LBP image and a HOG image for an input image, respectively. The two functions produce high-level feature maps that may be informative. The $W\text{-}Add$ and $W\text{-}Sub$ functions are used to add or subtract two weighted images with different or the same sizes, where the weights are n_1 and n_2. In the case where the sizes of the two images are not the same, $W\text{-}Add$ and $W\text{-}Sub$ overlap the image pixels at coordinates

(0, 0), cut the exceeding part of the larger images, and then perform the add or subtract operation. The $ReLU$ is the rectified linear unit, which is commonly used in CNNs. The $Sqrt$ function calculates the square root of each pixel value in an image and is protected by returning 1 if the pixel value is negative. The $ReLU$ and $Sqrt$ functions rescale the input image by transforming the pixel values from the negative to non-negative.

Pooling Functions

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

Feature Extraction Functions

Many image descriptors can be employed as GP functions to extract informative features. To reduce the search space of FGP, the three most commonly used methods, i.e., HOG, LBP, and SIFT, are employed for feature extraction. Table 4.2 lists the details of these functions.

Concatenation Functions

To concatenate features produced by different functions, five functions ($Root2$, $Root3$, $Root4$, $FeaCon2$, and $FeaCon3$) are employed and devel-

Table 4.2: Feature Extraction Functions

Function	Input	Output	Description
$SIFT$	1 Image	1 Vector	SIFT descriptor. 128 features are extracted from the image [220]
LBP	1 Image	1 Vector	LBP descriptor. It extracts 59 uniform LBP histogram features. In the LBP method, the radius is 1.5 and the number of neighbours is 8 [169]
HOG	1 Image	1 Vector	HOG descriptor. In HOG, the orientation is 9, the cell size is 8×8 and the block size is 3×3 [60]. The mean value of each 4×4 grid is extracted from an HOG image

Table 4.3: Concatenation Functions

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

oped. The descriptions of these functions are listed in Table 4.3. Each concatenation function can be used as the root node of a program tree or a child node of another concatenation function. This means that the tree depth of the concatenation layer is flexible. With these functions, FGP trees can output various numbers of features from an input image.

4.2.4 New Terminal Set

The terminal set of FGP contains the input image ($Image$) and the parameters for the functions, i.e., σ, o_1, o_2, θ, f, n_1, n_2, k_1, and k_2. More details of them are listed in Table 4.4. The $Image$ terminal represents the input

Table 4.4: Terminal Set

Terminal	Type	Description
$Image$	Image	The input grey-scale image (2D array containing image pixel values in the range of $[0,\ 1]$)
σ	Integer	The standard deviation of the Gaussian filter. It randomly initialized from the range of $\{1, 2, 3\}$
o_1, o_2	Integer	The order of the Gaussian derivatives. They are randomly initialised from the range of $\{0, 1, 2\}$
θ	Float	The orientation of the $Gabor$ filter. It is in the range of $[0,\ 7\pi/8]$ with a step of $\pi/8$ [144]
f	Float	The frequency of the $Gabor$ filter. It equals to $\frac{\frac{\pi}{2}}{\sqrt{2}^v}$, where v is an integer in the range of $\{0, 1, 2, 3, 4\}$ [144]
n_1, n_2	Float	The parameters for the W-Add and W-Sub functions. They are randomly generated from the range of $[0,\ 1)$
k_1, k_2	Integer	The kernel size of the $MaxP$ function. They are in the range of $\{2, 4\}$

image, which is a 2D array, and the values in the array are normalised into $[0, 1]$ dividing them by 255. The other terminals are ephemeral random constants of FGP and only appear in the trees where the corresponding functions are used. The values of these terminals are randomly selected from a pre-defined range at the initialisation step.

4.3 Experiment Design

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

4.3.1 Benchmark Datasets

Twelve widely used image classification datasets are employed to examine the performance of FGP. They are FEI_1 [215], FEI_2 [215], ORL [191], KTH [153], FS [79], MB [132], MRD [132], MBR [132], MBI [132], Rectangle [132], RI [132], and Convex [132] benchmark datasets. These datasets represent different types of image classification tasks, i.e., facial expression classification (FEI_1 and FEI_2), face recognition (ORL), texture classification (KTH), scene classification (FS), digit recognition (MB, MRD, MBR, and MBI), and object classification (Rectangle, RI and Convex). The images in these datasets are grey-scale or converted to grey-scale images to reduce the computational cost. The details of these datasets are listed in Table 4.5.

Table 4.5: Summary of the 12 Benchmark Datasets

No.	Dataset	Image Size	Training Set Size	Test Set Size	#Class
1	FEI_1	60×40	150 (75)	50	2
2	FEI_2	60×40	150 (75)	50	2
3	ORL	50×55	240 (6)	160	40
4	KTH	50×50	480 (48)	330	10
5	FS	55×55	1,300 (100)	2,559	13
6	MB	28×28	12,000	50,000	10
7	MRD	28×28	12,000	50,000	10
8	MBR	28×28	12,000	50,000	10
9	MBI	28×28	12,000	50,000	10
10	Rectangle	28×28	1,200	50,000	2
11	RI	28×28	12,000	50,000	2
12	Convex	28×28	8,000	50,000	2

For the FEI_1 and FEI_2 datasets [215], 75 images randomly selected from each class form the training set and the remaining 25 images form the test set. For the ORL dataset [191], which has a small number of images per class, six images randomly selected from the 10 images construct each class of the training set and the remaining images construct the test set.

For the KTH dataset [153], 48 images randomly selected from each class are used for training and the renaming 33 images are used for testing. The training set of the FS dataset [79] has 100 randomly selected images per class and the test set has the remaining images, since the dataset is large.

The MB, MRD, MBR, MBI, Rectangle, RI, and Convex datasets [132] have separated training and test sets [1], which can be directly used in experiments. The MB, MRD, MBR, and MBI datasets are subsets of the famous MNIST benchmark dataset. The training sets of the MB, MRD, MBR, and MBI datasets have 12,000 images, and the test sets have 50,000 images, respectively. The Rectangle dataset has 1,200 images for training and 50,000 images for testing. The RI dataset has 12,000 images for training and has 50,000 images for testing. The Convex dataset has 80,000 images for training and 50,000 images for testing.

4.3.2 Benchmark Methods

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

On datasets 1-5, 12 different methods are used as benchmark methods. They are six commonly used classification algorithms using raw pixels, four SVM methods using different pre-extracted features, and two CNN-based methods with different architectures. The six commonly used classification algorithms are linear SVMs, KNN, logistic regression (LR), RF, adaptive boosting (AdaBoost), and extremely randomised trees (ERF).

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

These methods use the normalised raw pixel values of images as inputs to train the classifiers. The four SVM methods are uLBP+SVM LBP+SVM, HOG+SVM and SIFT+SVM, which use uniform LBP, LBP, HOG, or SIFT features as inputs of SVMs for classification. The uniform LBP, LBP, HOG, and SIFT features are extracted by the methods described in Table 4.2. The final two benchmark methods are a five-layer CNN (CNN-5) [197] and an eight-layer CNN (CNN-8) [57]. CNNs are well known for image classification so that it is necessary to compare FGP with CNNs.

On datasets 6-12, 18 existing methods are used as benchmark methods. These methods have been reported recently or are representative methods for image classification. The classification results of these 18 methods are collected from the corresponding papers. These methods are SVM+RBF [132], SVM+Poly [132], SAE-3 [183], DAE-b-3 [183], CAE-2 [183], SPAE [236], RBM-3 [183], ScatNet-2 [49, 53], RandNet-2 [53], PCANet-2 (softmax) [53], LDANet-2 [53], NNet [132], SAA-3 [132], DBN-3 [132], FCCNN [177], FCCNN (with BT) [177], SPCN [140], and EvoCNN [208]. Most of these methods are NN-based methods, which have been introduced in Chapter 2. Note that in several methods, including SVM+RBF, SVM+Poly, NNet, SAA-3, and DBN-3, model selection has been conducted to find the best parameters using a training set and a validation set. Then these methods with the best parameters were trained using the training set and tested on the test set. The EvoCNN method is a deep learning method, which uses an evolutionary algorithm to automatically search for the best architectures of CNNs and has achieved the best performance on some of these benchmark datasets [208].

4.3.3 Parameter Settings

The parameter settings for FGP are based on the commonly used settings in the community of GP [114]. In FGP, the maximum number of generations G is 50 and the population size N is 500. The crossover rate Pc is 0.8,

the mutation rate Pm is 0.19, and the elitism rate Pe is 0.01. The selection method is the tournament selection with size 7. The tree depth is between 2-6 at the initialisation step, and the maximum tree depth is 8. Note that in FGP, which is based on STGP, the type constraint is more important than the depth constraint. Therefore, a tree may have a depth of over eight. As a new parameter, Nc, the number of individuals stored in the $Cache\_Table$ is set to $6 * N$ (N for the previous population and $5 * N$ for the best individuals at the past generations) based on the assumption that $6 * N$ is efficient and effective. In general, the value of Nc can be any number, but too large one may lead to a long searching time in $Cache\_Table$, and if a too small one, it would only store very limited individuals, which would make the $Cache\_Table$ not very useful. Note that the parameter settings for FGP are kept the same on the 12 different datasets for generality, although performing parameter tuning for FGP could further improve its performance on these datasets.

The parameter settings for the six classification algorithms SVM, KNN, LR, RF, AdaBoost, and ERF refer to [235, 249]. In KNN, the number of nearest neighbours is set to 1 [16]. In SVM and LR, the penalty parameter C is set to 1 [235]. In RF, ERF and AdaBoost, the number of trees is set to 500, and the maximum tree depth is set to 100 [249]. In CNN-5 and CNN-8, the commonly used ReLU function is used as the activation function and softmax is used for classification [57]. To avoid overfitting, dropout is added after the pooling layer and the first fully connected layer with 0.25 and 0.5 probabilities, respectively [201]. The maximum number of epochs is set to 500, and the batch size is set to 128, which is a commonly used value.

The implementation of FGP is based on the DEAP (*Distributed Evolutionary Algorithm in Python*) [86] package. The implementations of the classification algorithms are based on the *scikit-learn* [172] package and the implementations of CNNs are based on *Keras* [57]. The experiments of FGP on each dataset conduct 30 independent runs to avoid the experimental

bias, which follows the conventions of the EC communities. Each of the benchmark methods has been run 30 times on datasets 1-5 to obtain the classification results.

4.3.4 Test Process

The test procedure of the best individual/tree found by FGP is shown in Figure 4.2. In the process, the training set and the test set are used (note that only the training set are employed during the evolutionary process). The best FGP individual/tree is used to transform the training and test images into features. Then the transformed training and test sets are normalised using the min-max normalisation method to scale the features [102]. Note that the normalisation of the test set is based on the minimum and maximum values of each feature in the training set. The normalised training set is used to train a linear SVM classifier. The trained classifier is tested on the normalised test set to obtain the classification error rate.

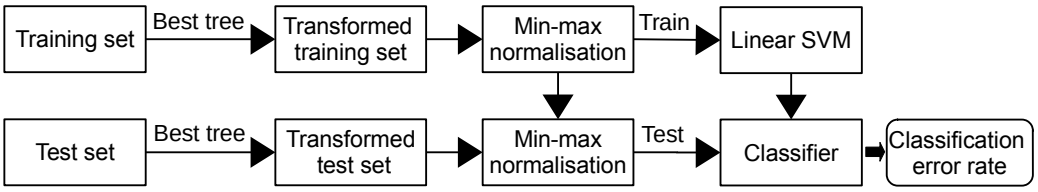


Figure 4.2: The test procedure of the best individual/tree found by FGP.

4.4 Results and Discussions

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

Table 4.6: Classification Error Rates (%) of Datasets 1-3

	FEI_1		FEI_2		ORL	
Methods	Min	Mean±St.dev	Min	Mean±St.dev	Min	Mean±St.dev
SVM	10.00	10.00±0.00+	12.00	12.00±0.00+	5.62	5.62±0.00+
KNN	68.00	68.00±0.00+	92.00	92.00±0.00+	5.62	5.62±0.00+
LR	8.00	8.00±0.00+	12.00	12.00±0.00+	6.25	6.25±0.00+
RF	2.00	**2.93±1.01**–	10.00	10.80±1.13+	6.88	7.67±0.63+
AdaBoost	20.00	21.33±1.32+	20.00	24.00±3.44+	40.62	47.73±4.00+
ERF	6.00	6.73±0.98+	8.00	9.40±0.93+	2.50	3.29±0.59+
uLBP+SVM	34.00	43.27±3.66+	32.00	37.47±3.52+	12.50	12.58±0.21+
LBP+SVM	32.00	35.40±1.83+	26.00	30.20±0.00+	11.88	12.48±0.20+
HOG+SVM	4.00	4.00±0.00–	18.00	18.00±0.00+	8.75	8.75±0.00+
SIFT+SVM	44.00	44.00±0.00+	38.00	38.00±0.00+	6.25	6.25±0.00+
CNN-5	2.00	4.60±1.30=	2.00	4.73±1.62–	3.12	4.71±1.06+
CNN-8	2.00	4.67±1.32=	4.00	9.07±1.87=	5.00	6.96±1.09+
FGP	2.00	5.53±2.67	4.00	8.67±3.36	**0.00**	**1.37±1.04**
Overall		8+, 2=, 2–		10+, 1=, 1–		12+

4.4.1 Classification Results on Datasets 1-5

The classification results on datasets 1-5, i.e., FEI_1, FEI_2, ORL, KTH, and FS, are listed in Tables 4.6 and 4.7. The results are the minimum classification error rate (Min), the average classification error rate of 30 runs and the standard deviation (Mean±St.dev). To show the significance of performance improvement, the Wilcoxon rank-sum test with a 95% significance interval is used to compare FGP with a benchmark method. In Tables 4.6 and 4.7, the symbols "+" and "–" indicate that FGP achieves significantly better and worse results than the compared method. The symbol "=" denotes that FGP achieves similar results to the compared method. In Tables 4.6 and 4.7, the best error rate and the average error rate on each dataset are highlighted in bold. The final row of each bock in the table summaries the overall results of the significance test.

From Tables 4.6 and 4.7, it can be found that FGP achieves significantly

Table 4.7: Classification Error Rates (%) of Datasets 4-5

	KTH		FS	
Methods	Min	Mean±St.dev	Min	Mean±St.dev
SVM	53.03	55.41±2.83+	79.37	79.71±0.15+
KNN	65.76	65.76±0.00+	75.65	75.65±0.00+
LR	51.21	51.21±0.00+	76.51	76.51±0.00+
RF	40.00	42.19±0.83+	62.64	63.47±0.49+
AdaBoost	62.12	66.56±1.37+	82.53	86.96±1.47+
ERF	38.48	40.17±0.86+	62.06	62.85±0.36+
uLBP+SVM	21.21	26.71±4.18+	50.21	66.73±8.90+
LBP+SVM	16.36	17.29±0.51+	46.50	49.55±1.80+
HOG+SVM	42.73	44.04±0.64+	87.89	92.09±2.47+
SIFT+SVM	34.24	34.24±0.00+	39.08	39.08±0.00+
CNN-5	14.24	17.44±1.87+	49.86	51.97±1.16+
CNN-8	23.64	28.37±3.18+	50.84	53.21±1.01+
FGP	**1.21**	**3.93±1.13**	**25.52**	**29.41±1.74**
Overall		12+		12+

better performance in 54 comparisons out of the 60 comparisons. More importantly, FGP significantly outperforms the 12 benchmark methods on the ORL, KTH and FS datasets, which are the face recognition, texture classification and scene classification tasks. On these three datasets, FGP not only obtains the minimum error rate but also achieves the best mean error rate among all the methods. It improves the error rate by 13.03% on KTH and 13.56 % on FS. In terms of the other two datasets, which are facial expression classification tasks, FGP is significantly better than seven methods on FEI_1 and than nine methods on FEI_2. The experimental results show that FGP is very effective for dealing with different types of image classification tasks. The main reasons are the development of the program structure and the function set within a number of image-related operators, which allow FGP to produce different types and numbers of effective features in more flexible ways.

Compared with SVM, KNN, LR, RF, AdaBoost, and ERF, which use

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

4.4.2 Classification Results on Datasets 6-12

On datasets 6-12, 18 baseline methods with published results are used for comparisons. Note that some of the 18 methods have not been examined on the Rectangle, RI, and Convex datasets so that there are 15 benchmark methods on Rectangle and RI and 11 benchmark methods on Convex. Table 4.8 lists the classification error rates (%) of FGP and 18 benchmark methods. Each column of Table 4.8 shows all the results on one dataset and the minimum error rate is highlighted in bold. The results of FGP, including the minimum error rate (best), the mean error rate (mean) and the standard deviation (std), are listed at the bottom of Table 4.8. Since the benchmark methods only have the best classification results, we compare

the FGP approach with them using the best error rate. The symbol "+" in the table denotes that FGP is better than the compared method in terms of the best error rate. The final row of Table 4.8 summarises the ranking results of FGP among all the methods on each dataset.

Table 4.8: Classification Error Rates (%) of Datasets 6-12

Methods	MB	MRD	MBR	MBI	Rectangle	RI	Convex
SVM+RBF [132]	3.03(+)	11.11(+)	14.58(+)	22.61(+)	2.15 (+)	24.04(+)	19.13(+)
SVM+Poly [132]	3.69(+)	15.42(+)	16.62(+)	24.01(+)	2.15(+)	24.05(+)	19.82(+)
SAE-3 [183]	3.46(+)	10.30(+)	11.28(+)	23.00(+)	2.14(+)	24.05(+)	–
DAE-b-3 [183]	2.84(+)	9.53(+)	10.30(+)	16.68(+)	1.99(+)	21.59(+)	–
CAE-2 [183]	2.48(+)	9.66(+)	10.90(+)	15.50(+)	1.21(+)	21.54(+)	–
SPAE [236]	3.32(+)	10.26(+)	9.01(+)	13.24(+)	–	–	–
RBM-3 [183]	3.11(+)	10.30(+)	6.73(+)	16.31(+)	2.60(+)	22.50(+)	–
ScatNet-2 [49, 53]	1.27(+)	7.48(+)	12.30(+)	18.40(+)	0.01(+)	8.02(+)	6.50(+)
RandNet-2 [53]	1.25(+)	8.47(+)	13.47(+)	11.65(+)	0.09(+)	17.00(+)	5.45(+)
PCANet-2 (softmax) [53]	1.40(+)	8.52(+)	6.85(+)	11.55(+)	0.49(+)	13.39(+)	4.19(+)
LDANet-2 [53]	**1.05**	7.52(+)	6.81(+)	12.42(+)	0.14(+)	16.20(+)	7.22(+)
NNet [132]	4.69(+)	18.11(+)	20.04(+)	27.41(+)	7.16(+)	33.20(+)	32.25(+)
SAA-3 [132]	3.46(+)	10.30(+)	11.28(+)	23.00(+)	2.41(+)	24.05(+)	18.41(+)
DBN-3 [132]	3.11(+)	10.30(+)	6.73(+)	16.31(+)	2.60(+)	22.50(+)	18.63(+)
FCCNN [177]	2.43(+)	8.91(+)	6.45	13.23(+)	–	–	–
FCCNN (with BT) [177]	2.68(+)	9.59(+)	6.97(+)	10.80(+)	–	–	–
SPCN [140]	1.82(+)	9.81(+)	**5.84**	9.55(+)	0.19(+)	10.60(+)	–
EvoCNN (best) [208]	1.18	**5.22**	**2.80**	**4.53**	0.01(+)	**5.0**3	4.82(+)
FGP (best)	1.18	**7.37**	6.54	**7.48**	**0.00**	**6.10**	**1.54**
FGP (mean)	1.30	8.44	7.34	10.35	0.12	7.34	1.84
FGP (std)	0.06	0.6	0.42	1.41	0.11	0.61	0.19
Rank	2/19	2/19	4/19	**2/19**	**1/16**	**2/16**	**1/11**

From Table 4.8, it can be found that FGP achieves a smaller error rate than any of the benchmark methods on two datasets, i.e., Rectangle and Convex, and ranks second on four datasets, i.e., MB, MRD, MBR, and RI. Importantly, FGP improves the error rate by 2.65% on Convex. Note that these datasets have been widely used by these effective methods so that even 1% improvement in error rate is very difficult to achieve. On the MB

dataset, the FGP approach achieves 1.18% error rate, which is better (or similar) than any of the 18 benchmark methods except for LDANet-2. The LDANet-2 method achieves 1.05% error rate on the MB dataset, which is slightly better than FGP of 1.18% error rate. Although FGP is worse than LDANet-2 on the MB dataset, it is better than LDANet-2 on the other six datasets. The MRD, MBR and MBI datasets are three variants of the MB dataset by adding additional factors to make it more difficult. On the MRD dataset, FGP achieves an error rate of 7.37%, which is better than 17 benchmark methods and worse than EvoCNN. On the MBR dataset, FGP ranks fourth among all the benchmark methods. On the MBI dataset, FGP achieves an error rate of 7.48%, which is better than 18 methods and only worse than EvoCNN. On the Rectangle and Convex datasets, FGP achieves better results than any of the benchmark methods. It is noticeable that FGP finds the perfect solution on Rectangle. The RI dataset is an extension of the Rectangle dataset and it is more difficult. Most methods perform worse on the RI dataset, e.g., LDANet-2 obtains 16.20% error rate, SPCN obtains 10.60% error rate, and NNet obtains 33.20% error rate. FGP obtains an error rate of 6.10%. On the Convex dataset, FGP achieves the best error rate of 1.54%, which is better than that of any benchmark method.

Compared with EvoCNN, which is a state-of-the-art deep learning algorithm, FGP achieves better or the same error rates on the MB, Rectangle and Convex datasets. FGP is a non-neural network-based algorithm, while EvoCNN is a CNN-based algorithm, where an evolutionary algorithm is used to search for the best architecture of CNNs. EvoCNN was designed to find a more complex CNN-based solution for image classification so that it can achieve better performance on the other four difficult datasets. Compared with EvoCNN, FGP can find simpler solutions with several image-related operators. In addition, the EvoCNN method requires to run on Graphics Processing Unit (GPU), while FGP runs on Central Processing Unit (CPU), which is less computationally expensive.

These advantages enable FGP to be an alternative to effective feature learning for image classification.

The comparisons demonstrate that the proposed FGP approach achieves better results than the 18 existing effective algorithms for object classification. Comparing the classification results on the MB dataset with those on the MRD, MBR and MBI dataset, it is obvious that FGP is less affected by rotation and background changes in images than the 18 effective methods. This indicates that FGP can learn invariant features from images with these additional factors. FGP has a function set with a number of image-related operators having different functionalities, such as denoising, detecting edges and extracting invariant features. These functions enable FGP to find optimal solutions with the capability of dealing with different image variations.

4.5 Further Analysis

This section further analyses the FGP approach to provide insights on why it achieves better results. First, the convergence behaviours of FGP on different datasets are analysed. Second, the evolved example programs/solutions of FGP are analysed to understand what features are learned. Third, the frequency of the image-related operators in best-of-the-run programs/trees is calculated and analysed. Third, the datasets with the learned features are visualised and compared with the original raw pixels and the commonly used features. This provides insights on how the hidden structures of the datasets are changed by the solutions of FGP.

4.5.1 Convergence Behaviour of FGP

The convergence behaviours of FGP on the FEI_2, KTH, MB, MBI, MBR, MRD, Convex, and RI datasets are shown in Figure 4.3. The X-axis indicates the number of generations and the Y-axis indicates the fitness value,

i.e., the classification accuracy (%). It can be found from Figure 4.3 that FGP has different convergence behaviours on different datasets. Specifically, FGP has fast convergence speed at the early stage of generation. When approaching the maximum number generation, the convergence speed of FGP becomes slow on most of the datasets. Figure 4.3 shows the FGP can converge to a high fitness value after 50 generations. However, on some difficult datasets such as MBI, the FGP approach has not been converging at the end of generations. These difficult datasets may need a larger number of generations or population size to allow FGP to find better solutions and coverage to a higher fitness value. This analysis shows that the proposed FGP approach is able to converge to a high fitness value within the given number (50) of generations on most of the datasets.

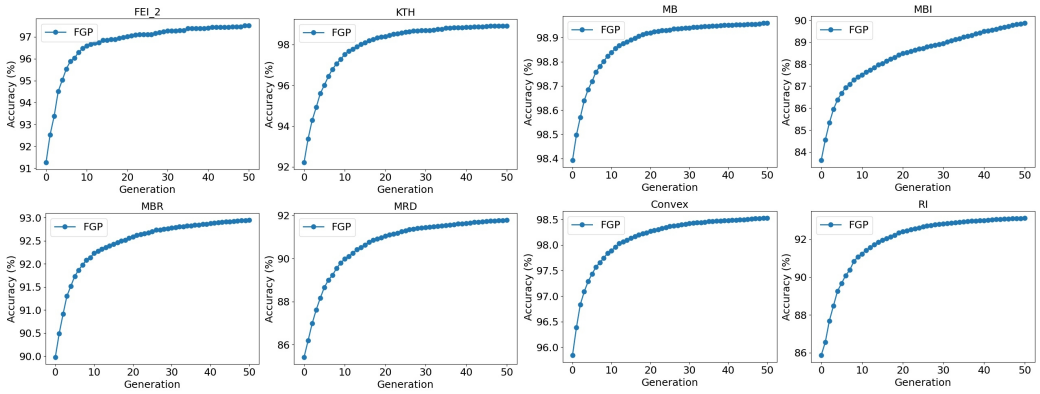


Figure 4.3: Convergence Behaviours of FGP on the FEI_2, KTH, MB, MBI, MBR, MRD, Convex, and RI datasets.

4.5.2 Number of Learned Features

The numbers of features produced by the best FGP programs/trees are shown in Figure 4.4. On the FEI_2, ORL, MBR, Rectangle, and RI datasets, the average number of features learned by FGP is about or less than 500. On the remaining datasets except for MRD, the average number of learned features is more than 500 but less than 1000. On the MRD dataset, FGP

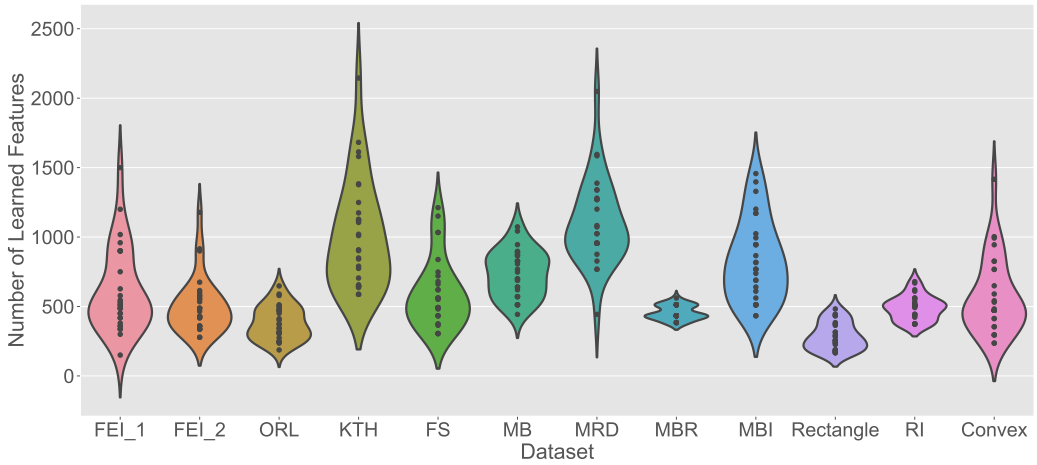


Figure 4.4: Distribution of the number of features learned by FGP on each dataset.

learns an average number of 1100 features. The smallest number of learned features on the 12 datasets is smaller than 500. This analysis confirms that FGP can learn various numbers of features from different datasets.

4.5.3 Evolved Programs/Solutions

An Example Program on FEI_1

An example program evolved by FGP on the FEI_1 dataset is visualised in Figure 4.5. It achieves 98% accuracy on both the training and test sets. To show how the program extracts features, two example images from the natural and smile classes are used for visualisation, as shown in Figure 4.5. The example program has filtering and pooling functions to describe the features from the input image. The edge filters, i.e., $SobelY$ and $GauD$, are used as nodes in the two branches of the example program. These operators can extract edges from the images before applying the pooling operators. The two example images are different in facial expressions. From Figure 4.5, it is clear that using different filters can obtain informative features that enlarge the difference between the two classes. Finally, the ex-

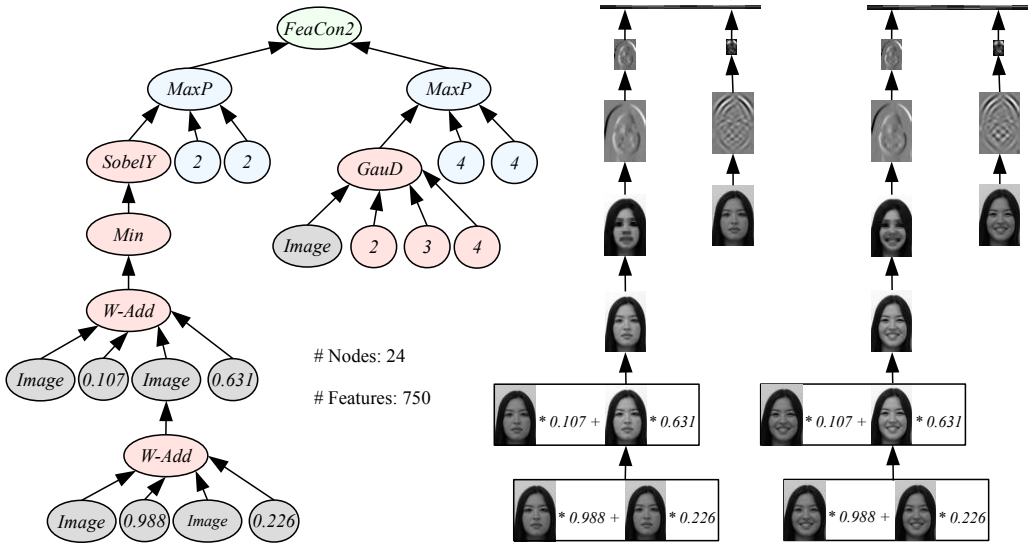


Figure 4.5: Example program evolved by FGP on the **FEI_1** dataset.

ample program with 24 nodes produces 750 features from an input 60×40 image, i.e., 600 features extracted by the left branch and 150 extracted by the right branch.

Example Programs on Rectangle

Three example programs of the proposed FGP approach on the Rectangle dataset are visualised in Figure 4.6. The three programs achieve 100% classification accuracy on both the training set and the test set. The three programs learn 433, 187 and 246 features from the input image, respectively. Meanwhile, their tree sizes (the number of nodes) are 23, 9 and 11, respectively. In contrast to the example program in Figure 4.5, which describes features using filtering and pooling functions, the three example programs generate features using feature extraction, filtering and pooling functions. The three example programs extract SIFT and LBP features from the input image or the images after the filtering functions, e.g., Med, $LoG1$, Gau, and $LoG2$. Since Rectangle has images with rotation and scale variations, the

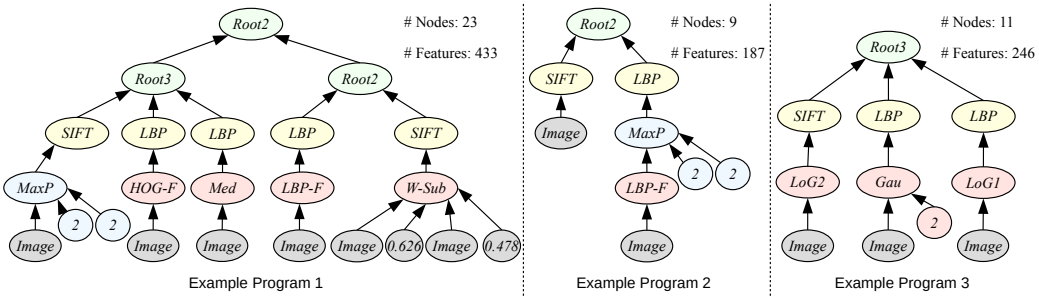


Figure 4.6: Example programs evolved by FGP on the **Rectangle** dataset.

features that are invariant to these variations are more discriminative than the other types of features. Therefore, the LBP and SIFT features are extracted by the example programs. It is noticeable that the LBP and $SIFT$ functions in the three example programs have different child nodes, which indicate that each LBP or $SIFT$ function extracts different features.

Example Programs on the Other Datasets

The best-of-the-run programs and the number of learned features on the other ten datasets except for the FEI_1 and Rectangle datasets are listed in Table 4.9. On the FEI_2 dataset, the SIFT and HOG features are extracted by the example program after the corresponding filtering and pooling operations on the input image. On the ORL dataset, the example program extracts the LBP and SIFT features from raw images or the images after filtering or pooling. On the KTH dataset, the LBP, HOG and SIFT features are extracted by the example program. On the FS dataset, the LBP and SIFT features are extracted by the example program. On the object classification tasks, i.e., the MB, MRD, MBR, and MBI datasets, compared with the LBP and HOG features, more SIFT features are extracted by these example programs from the raw images or images after pooling/filtering. The reason may be that the SIFT features can better capture the salient information of objects and are invariant to scale and rotation variations. On the RI and Convex datasets, the LBP and SIFT features are extracted

Table 4.9: An Example of Best-of-the-Run Program on Each Dataset

Dataset	Evolved Program	# Features
FEI_2	$Root3(SIFT(MaxP(W\text{-}Sub(Image, 0.453, Image, 0.641), 4, 2)), HOG(W\text{-}Add(LoG2(Image), 0.031, W\text{-}Sub(ReLU(SobelY(Image)), 0.9, W\text{-}Add(LoG1(Image), 0.206, GauD(Image, 2, 0, 1), 0.776), 0.837), 0.837)), HOG(Lap(Image)))$	428
ORL	$Root4(SIFT(Image), SIFT(LoG1(W\text{-}Add(Image, 0.873, Image, 0.178))), LBP(MaxP(ReLU(Image), 4, 4)), LBP(Image))$	374
KTH	$Root2(Root4(Root3(HOG(Gabor2(Image, 8, 4)), LBP(W\text{-}Sub(Image, 0.093, Image, 0.259)), LBP(Min(Image))), SIFT(SobelX(Image)), SIFT(Max(Image)), SIFT(SobelY(Image))), Root4(HOG(Image), HOG(Gau2(Image, 4)), LBP(W\text{-}Sub(LoG1(W\text{-}Sub(Image, 0.093, Image, 0.093)), 0.093, LoG1(W\text{-}Sub(LoG1(W\text{-}Sub(Image, 0.093, Image, 0.168)), 0.765, LoG1(GauD(Image, 3, 3, 3)), 0.935)), 0.935)), SIFT(Min(Gau(Image, 4)))))$	1121
FS	$Root3(Root4(LBP(Max(W\text{-}Add(Image, 0.24, Image, 0.24))), LBP(SobelY(SobelX(W\text{-}Add(Image, 0.24, Image, 0.24)))), Root4(LBP(Max(Lap(Image))), LBP(LoG1(Image)), Root4(LBP(Max(Sobel(Image))), SIFT(SobelX(Image)), Root4(LBP(Max(LBP(Image))), LBP(Sobel(Image)), Root2(LBP(W\text{-}Add(Image, 0.24, Image, 0.24)), SIFT(LBP\text{-}F(Image))), LBP(SobelY(Image))), LBP(SobelY(Sobel(Image)))), LBP(SobelY(SobelY(Image)))), LBP(LBP\text{-}F(Image))), LBP(Image), SIFT(Image))$	1151
MB	$Root4(SIFT(Image), HOG(Image), SIFT(SobelY(Image)), Root2(Root2(SIFT(W\text{-}Add(\ \ SobelX(Max(SobelY(Image))), 0.074, Gau(Gau(Max(Image), 3), 3), 0.665)), SIFT(Gabor(Lap(Gabor(SobelY(Image), 8, 5)), 8, 2))), Root2(Root2(SIFT(SobelX(Image)), SIFT(SobelX(Max(Image)))), Root2(SIFT(Mean(LoG1(MaxP(Image, 2, 2)))), SIFT(SobelX(Sqrt(Image)))))))$	1073
MRD	$Root2(Root4(SIFT(Gau(Gau(W\text{-}Sub(Relu(GauD(Image, 2, 0, 2)), 0.713, LoG2(Image), 0.52), 4), 4)), SIFT(Mean(GauD(Image, 4, 1, 1))), SIFT(Gau(Gau(W\text{-}Sub(ReLU(GauD(Image, 2, 2, 0)), 0.713, LoG2(Image), 0.52), 4), 2)), LBP(Sqrt(Image))), Root4(SIFT(Sobel(MaxP(Gau(Gau(Sqrt(Image), 4), 4), 2, 2))), SIFT(Sobel(Gau(Image, 2))), SIFT(ReLU(SobelX(Sqrt(Image)))), SIFT(Sqrt(Image))))$	955
MBR	$Root4(SIFT(Max(MaxP(GauD(Gabor(Gabor(Gabor(Image, 3, 1), 3, 1), 3, 1), 1, 0, 0), 2, 2))), SIFT(Max(MaxP(GauD(Gabor(GauD(Gabor(Image, 7, 2), 1, 0, 0), 7, 2), 1, 0, 0), 2, 2))), SIFT(Max(MaxP(GauD(Gabor(Image, 3, 2), 1, 0, 0), 2, 2))), SIFT(Mean(Image)))$	512
MBI	$Root2(Root4(SIFT(SobelY(Image))SIFT(Lap(ReLU(W\text{-}Sub(W\text{-}Add(Image, 0.329, Image, 0.791), 0.317, Max(Max(Image)), 0.329)))), SIFT(ReLU(W\text{-}Sub(W\text{-}Add(Image, 0.329, Image, 0.791), 0.317, Max(Max(Max(Image))), 0.329))), SIFT(SobelX(ReLU(W\text{-}Sub(W\text{-}Add(Image, 0.329, Image, 0.791), 0.317, Max(Max(Image)), 0.329))))), Root4(HOG(Image), SIFT(MaxP(Max(W\text{-}Add(Max(Image), 0.419, Gabor(Image, 3, 1), 0.911)), 2, 2)), SIFT(Max(W\text{-}Add(Max(Image), 0.187, Image, 0.192))), SIFT(ReLU(SobelX(Image)))))$	945
RI	$Root2(Root4(LBP(Sobel(Sobel(GauD(W\text{-}Add(Image, 0.386, Image, 0.259), 2, 0, 0)))), SIFT(ReLU(Gabor(GauD(Max(Gabor(Image, 0, 4)), 2, 0, 0), 0, 4))), LBP(Sobel(Sobel(GauD(Max(Gabor(Image, 4, 4)), 2, 1, 0)))), SIFT(Gau(Max(Gabor(Lap(Image), 4, 4)), 1))), Root4(LBP(Sobel(Gabor(Lap(GauD(Image, 1, 3, 0)), 0, 2))), SIFT(Gau(Max(Gabor(Image, 6, 3)), 2)), LBP(Sobel(Image)), LBP(Sobel(GauD(Max(Gabor(Image, 0, 4)), 1, 0, 0)))))$	679
Convex	$Root2(Root2(Root3(Root3(LBP(W\text{-}Sub(Image, 0.961, Image, 0.379)), LBP(W\text{-}Sub(ReLU(Image), 0.372, Image, 0.961)), LBP(W\text{-}Sub(ReLU(Image), 0.609, Sqrt(Mean(Image)), 0.379))), LBP(Lap(Mean(Image))), LBP(Image)), LBP(W\text{-}Sub(Image, 0.379, Gau(Image, 1), 0.961))), Root2(Root3(Root4(LBP(MaxP(Gau(Image, 1), 2, 2)), LBP(W\text{-}Sub(ReLU(Image), 0.609, Sqrt(Mean(Image)), 0.961)), LBP(W\text{-}Sub(Image, 0.961, Lap(Mean(Image)), 0.961)), LBP(Sqrt(Mean(Image)))), LBP(Sqrt(Mean(Image))), LBP(W\text{-}Sub(ReLU(Image), 0.609, Sqrt(Mean(Image)), 0.379))), LBP(W\text{-}Sub(Image, 0.379, Image, 0.961))))$	767

using many feature concatenation functions, feature extraction functions, filtering functions, and pooling functions.

By analysing these example programs, it can be found that FGP evolves programs/solutions that describe features using feature extraction, filtering and pooling functions. With the flexible program structure, FGP can produce three different types of features, i.e., features produced by feature extraction functions, features produced by filtering and/or pooling functions, the combination of features produced by feature extraction and filtering or pooling. The example programs show that most of the output features are of the first type, and few output features are of the second type (as the example program evolved by FGP on the FEI_1 dataset showed in Figure 4.5). A possible reason is that the features extracted from the pooling and filtering operations are not invariant to particular variations, such as rotation.

4.5.4 Analysis on Image-Related Operators in the Best-of-the-Run Programs

Analysis of the frequencies of the image-related operators/functions in 30 best-of-the-run programs/trees on each dataset is conducted to further understand the relationship between the datasets and the functions that are used in the final program. The frequency of each function (%) is shown in Figure 4.7.

From Figure 4.7, it is clear that the frequencies of the functions vary with the dataset. On the FEI_1 and FEI_2 datasets, the most frequently used functions are HOG, LBP, $SobelX$, and Max. On the ORL dataset, the LBP, $SIFT$, $SobelX$, and $LBP\text{-}F$ functions appear most frequently in the best programs. On the KTH dataset, the HOG, LBP, and $SIFT$ functions are frequently used. In the best programs on the FS dataset, the most frequently appearing functions are LBP and $SIFT$. This indicates that FGP mostly learns LBP and SIFT features on the FS dataset. On the

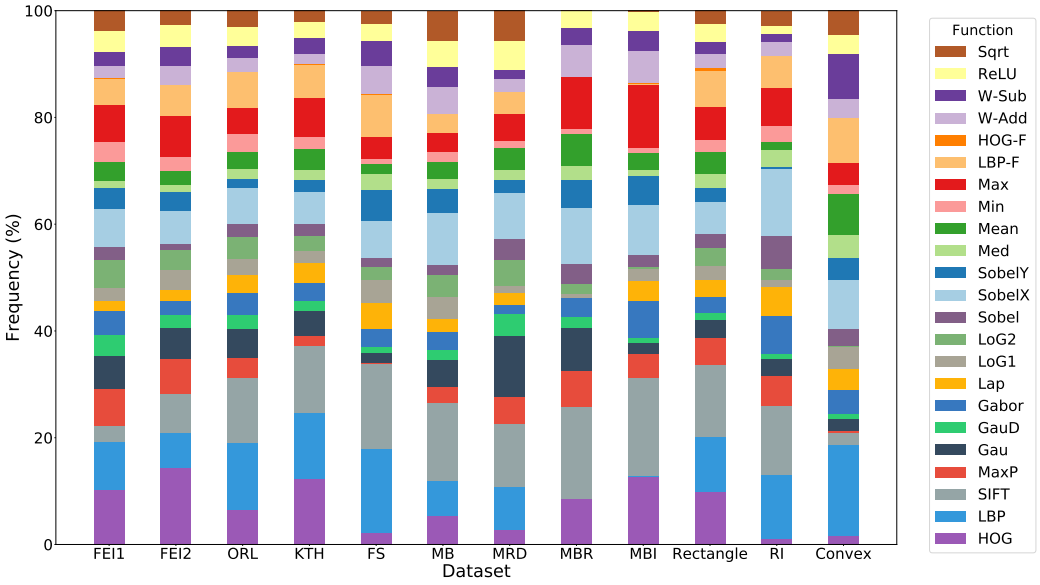


Figure 4.7: The frequency (%) of each function in the best program of 30 runs by FGP on each dataset.

MB dataset, the most frequently used functions are $SIFT$, $SobelX$, and LBP. On the MRD dataset, the best programs frequently use the Gau function, which is able to reduce noise. On the MBR and MBI datasets, no LBP functions are used to extract features. This may be because LBP is not invariant to noise. On the Rectangle dataset, the best programs tend to extract LBP, HOG, and SIFT features. On the RI dataset, the SIFT and LBP features are frequently extracted. On the Convex dataset, the most frequently used functions are LBP, $SobelX$, LBP-F, and W-Sub. This indicates that LBP features are frequently extracted by the programs of FGP on the Convex dataset. The analysis of the frequently used functions indicates which types of features are mostly learned by FGP. The results of this analysis are consistent with the results of the analysis in Section 4.5.3.

It can be found from Figure 4.7 that the frequently used functions on one dataset might not be (mostly) used on another dataset. For example, LBP is not used on the MBR and MBI datasets but frequently used on the Convex, ORL and FS datasets. The $Sqrt$ function is seldom used on

the MBR and MBI datasets. The $SobelY$ function is seldom used on the RI and Convex datasets but is commonly used on the FS and MBI datasets. The $MaxP$ function is mostly used on the MBI and RI datasets but is seldom used on the FS and Convex datasets. This confirms the difficulty of manually combining the different functions to achieve feature extraction/learning because different datasets require different functions to extract features. FGP can automatically combine these functions to build solutions that can extract/learn effective features for classification.

In summary, the analysis of the number of features learned by FGP shows that FGP can learn various numbers of features from different datasets. The analysis of the example programs shows that FGP mostly learns features from filtering and pooling; and features from feature extraction. The frequency of the image-related operators shows an overall picture on the use of image-related operators in the best programs of FGP. The analysis indicates that the frequencies of the image-related operators vary with the datasets. This confirms the difficulty of manually combining the different functions to achieve feature extraction/learning. The analysis gives insights into what image-related operators are used to build the solutions of FGP and what types of features are extracted/learned from different images.

4.5.5 Visualisation of the Learned Features

A popular visualisation method, t-distributed stochastic neighbour embedding (t-SNE) [152], is employed to visualise the features learned by FGP. The t-SNE method is a non-linear dimension reduction technique, which maps high-dimensional data into two- or three-dimensional data. The resulting low-dimensional data can be easily visualised in a scatter plot, which shows how well the similarities within each class is preserved. Compared with other visualisation methods, t-SNE produces better visualisation results [152]. Therefore, we use t-SNE for visualisation.

Three large datasets, i.e., MB, MBI and Rectangle, are used to run the experiments for visualisation because MB is representative and FGP achieves the best performance on the MBI and Rectangle datasets. The MB and MBI datasets have 12,000 training images and 50,000 test images. The Rectangle dataset has 1,200 training images and 50,000 test images. To reduce the computational cost of t-SNE, each experiment randomly selects 5,000 images from each dataset for visualisation. To show how the hidden structure of the data is changed by the FGP program, the original data, the LBP features, the HOG features, and the SIFT features are visualised for comparisons. The visualisation results are plotted in a scatter, and the class label of each point is used to give a specific colour to the point in the plot. The parameter settings for t-SNE are the same as those in [152].

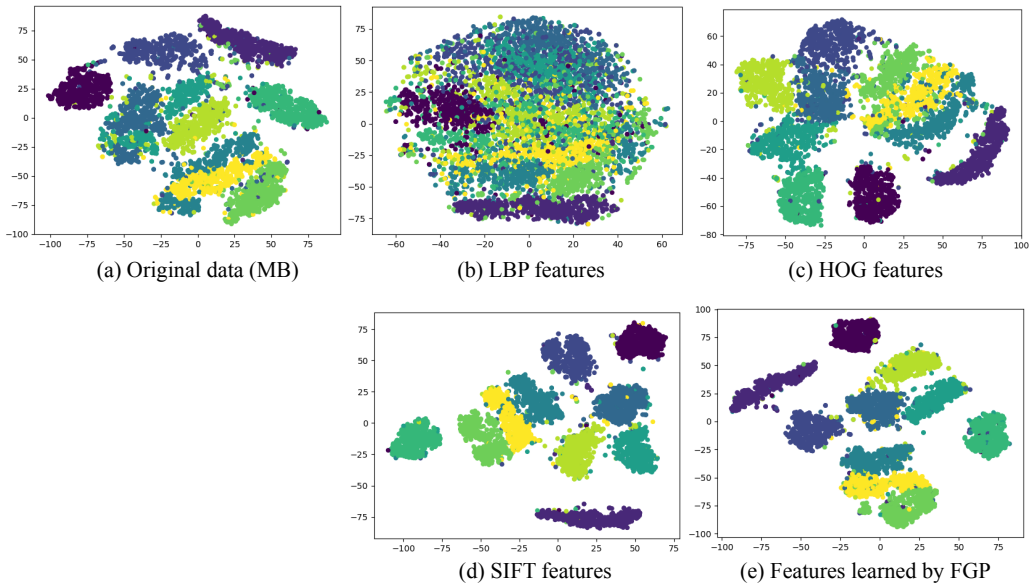
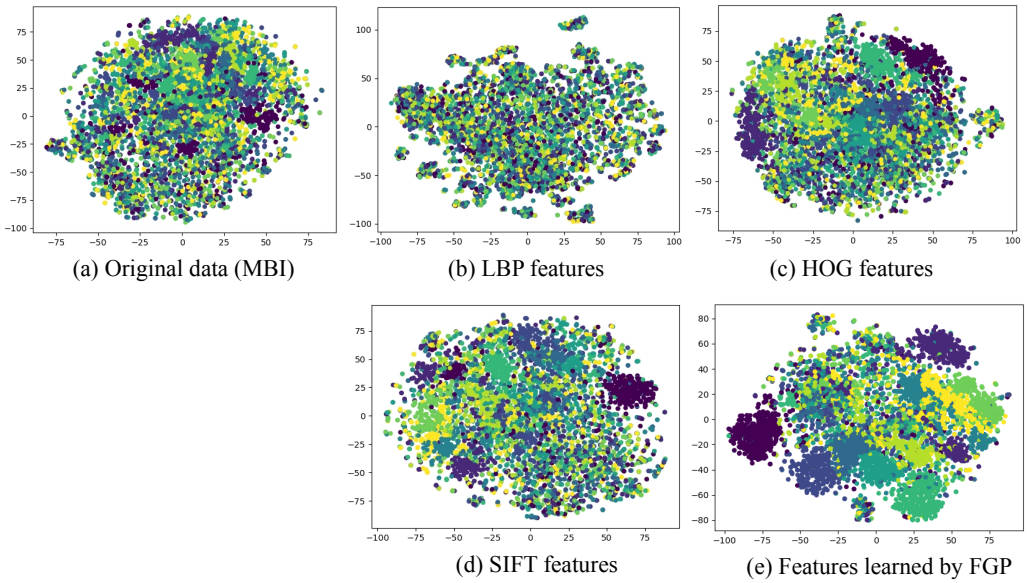


Figure 4.8: The visualisation results of the ten classes from the **MB** dataset (each colour represents one class). These figures are the visualisation results using (a) raw pixels, (b) LBP features, (c) HOG features, (d) SIFT features, and (e) the features learned by FGP, respectively.

The visualisation results of the MB dataset are shown in Figure 4.8. It is clear that the ten classes of the original MB data and the transformed MB



(a) Original data (MBI) (b) LBP features (c) HOG features

(d) SIFT features (e) Features learned by FGP

Figure 4.9: The visualisation results of the ten classes from the **MBI** dataset (each colour represents one class). These figures are the visualisation results using (a) raw pixels, (b) LBP features, (c) HOG features, (d) SIFT features, and (e) the features learned by FGP, respectively.

data with the HOG features, the SIFT features, and the learned features by FGP are well clustered after 1000 iterations using t-SNE. The visualisation results of the LBP features are poor as the plot shows a high mixture of different classes. In Figures 4.8 (a), (c) and (d), there are still some points being clustered into wrong classes as each cluster of points contains other points with different colours. In contrast, Figure 4.8 (e) shows clearer clusters of the data transformed by the example program of FGP. This figure shows that fewer points have been clustered into wrong classes, and each cluster is clearer than those in Figures 4.8 (a)-(d).

The visualisation results of the original MBI data (Figure 4.9) show a high mixture of different classes as it is very difficult to distinguish a cluster from the scatter plot. The reason is that the images of MBI have noise, which makes the visualisation of MBI more difficult than that of MB. The visualisations of the transformed MBI data using the features learned by

FGP (Figure 4.9 (e)) have a clearer plot than those of the original data (Figure 4.9 (a)) and the data with the LBP (Figure 4.9 (b)), HOG (Figure 4.9 (c)) and SIFT (Figure 4.9 (d)) features, respectively. It can be observed that several clusters of points exist in Figure 4.9 (e) even if some points are not well clustered. Comparing the visualisation results in Figure 4.8 with the results in Figure 4.9, it is obvious that the hidden structure of the MBI data is more complex than that of the MB data so that MBI is more difficult than MB. The results reveal that the programs/solutions evolved by FGP transform the original data into a space where the new data can be easily clustered by t-SNE, and the hidden structure can be well captured.

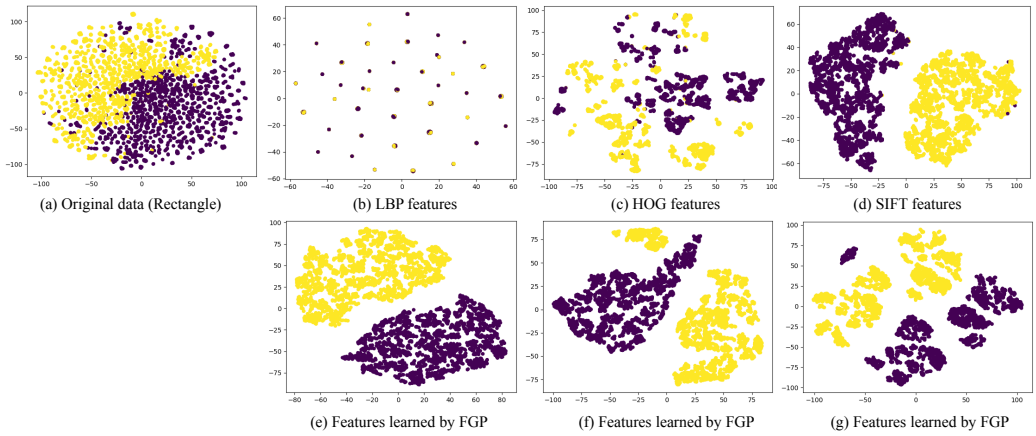


Figure 4.10: The visualisation results of the ten classes from the **Rectangle** dataset (each colour represents one class). These figures are the visualisation results using (a) raw pixels, (b) LBP features, (c) HOG features, (d) SIFT features, and (e-g) the features learned by the three example programs of FGP showed in Figure 4.6, respectively.

The visualisation of the Rectangle dataset is simpler than that of the MB and MBI datasets as Rectangle only has two classes. Figure 4.10 shows the visualisation results of the original data, the LBP features, the HOG features, the SIFT features, and the features learned by the proposed FGP approach using three different evolved programs, respectively. Figures 4.10 (a), (c) and (d) have clear scatter plots, where the two classes of Rectan-

gle are shown in two different colours. However, it is obvious that many points are clustered into the wrong classes in Figures 4.10 (a), (c) and (d). The visualisation results of the different classes are not clear when using LBP features, which indicates that the LBP features are not effective on the Rectangle dataset. The scatter plots are clearer in Figure 4.10 (e)-(g) than those in Figures 4.10 (a)-(d). It is noticeable that all the points are clustered into the correct classes by t-SNE using the data transformed by three different GP programs/solutions. The visualisation results confirm the search ability and superiority of FGP in finding the optimal solutions to transform the data into a new feature space where the new data can be easily classified into different classes.

4.6 Chapter Summary

In this chapter, the FGP approach with a flexible program structure and image-related operators was developed to learn effective features for different types of image classification tasks. With the flexible program structure, a new function set and a new terminal set, the FGP approach can evolve solutions with variable lengths to extract various numbers and types of features from raw images. The performance of FGP was examined on 12 datasets, including the datasets having a large number of training and testing instances (i.e., the variants of MNIST), and compared with a large number of existing methods.

The experimental results showed that FGP achieved significantly better performance than the 12 commonly used methods on five small image classification datasets of varying difficulty. The experimental results showed that FGP achieved better classification performance than the existing methods to which it has been compared on the other seven big datasets with a large number of instances. Compared with a state-of-the-art deep learning method, i.e., EvoCNN, FGP achieved better performance on two datasets and comparable performance on the remaining five datasets. The

results demonstrated that FGP is an effective and promising approach to feature learning for different types of image classification tasks.

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

In this chapter and the previous chapter (Chapter 3), two different GP-based approaches were developed to use image descriptors and/or image filters to learn effective features for classifying relatively large or small images. The two approaches use a single classifier to perform classification. The classification performance can be further improved by using ensemble methods. The next chapter will develop a new GP-based approach to construct ensembles to improve classification performance.

Chapter 5

GP for Simultaneous Feature Learning and Ensemble Evolving

5.1 Introduction

The previous two chapters proposed two GP-based approaches with image descriptors and other image-related operators for feature learning in image classification. These two approaches use a single classifier for classification. However, the classification performance can be further improved by using ensemble classifiers. This chapter proposes an ensemble method for image classification using GP.

Ensemble methods have been widely used for solving classification problems [91, 237]. An ensemble consists of multiple classifiers to solve a classification problem and often achieve a better generalisation performance [247]. Traditional ensemble methods for image classification often have the processes of feature extraction, base learner or classifier selection, training and combination [65]. In these processes, rich domain knowledge is required, especially for feature extraction. In addition, the diversity of the classifiers in the ensemble is a key factor to build an effective ensemble, but it is an open issue due to the difficulty of quantisation/measurement [129]. GP has been applied to evolve ensembles for

classification [75, 77, 218]. But most methods cannot be directly used for tackling image classification tasks due to the necessity and difficulty of feature extraction. On the other hand, GP has been applied to automatically learn features for image classification [11, 14, 27, 197], including the methods described in the previous two chapters. But these methods only employ a single classifier for classification. Their classification performance could be further improved by using ensemble classifiers. Therefore, this chapter proposes a new GP-based approach to automatically learning features and evolving ensembles for image classification.

5.1.1 Chapter Goals

The overall goal of this chapter is to develop a new effective GP-based approach to automatically learning features and evolving ensembles for image classification. The new approach is named Improved Ensemble GP method (IEGP). To achieve this goal, a new representation with an input layer, a filtering & pooling layer, a feature extraction layer, a concatenation layer, a classification layer, a combination layer, and an output layer is developed in IEGP. Except for the input and output layers, each of the remaining layers has a number of functions, which allow the new approach to automatically evolve combinations of them to form the solutions. Each solution produces the combined predictions of class labels for the input images. The IEGP approach is evaluated on 12 datasets of varying difficulty, including datasets with a large number of training and test images, and compared with state-of-the-art methods. Specifically, this chapter will investigate

- how the processes of feature learning and ensemble evolving are integrated into a single GP tree by developing a new program structure, a new function set and a new terminal set;
- whether IEGP can achieve better classification performance than the methods using raw pixels, the methods using well-known features,

two CNN-based methods, the FGP method described in the previous chapter, and other GP-based methods;

- whether IEGP can achieve better classification performance than a large number of state-of-the-art algorithms on the datasets with a large number of training and test instances;
- whether IEGP can select and optimise the parameters of the classification algorithms in the evolved ensemble, and address the diversity issue; and
- whether the solutions of IEGP can provide high interpretability of what features are learned, what classification algorithms are used to build the ensemble, and why they achieve good performance.

5.1.2 Chapter Organisation

The remainder of this chapter is organised as follows. Chapter 5.2 describes the proposed approach. Chapter 5.3 designs the experiments. The results are discussed and compared in Chapter 5.4. Further analysis is presented in Chapter 5.5. Chapter 5.6 concludes this chapter.

5.2 The Proposed Approach

The general process of traditional ensemble methods for image classification is shown in Figure 5.1. The overall process includes feature extraction, base classifiers selection, training and combination [65]. In these processes, domain knowledge or human intervention is often required to determine what and how effective features are extracted from raw images, what the base learners are used, how the parameters are set for the base learners, and how the trained base learners are combined to construct an ensemble.

With a flexible tree-based representation, it is possible to integrate these processes into a single GP tree to automatically evolve effective ensem-

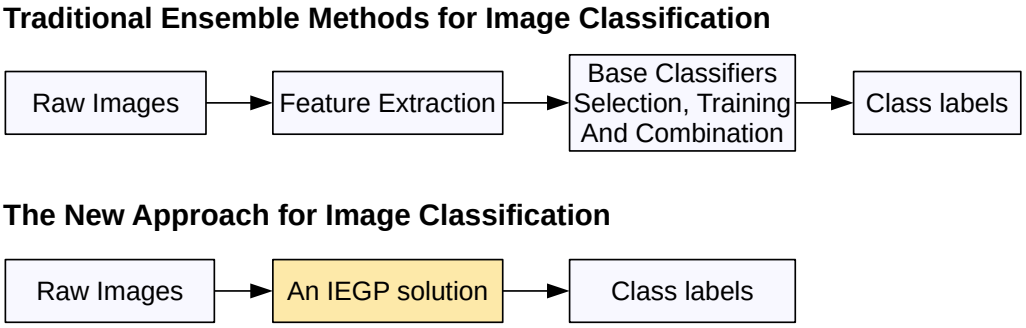


Figure 5.1: The outlines of the traditional ensemble methods [65] and the new approach (IEGP) for image classification.

bles with simultaneous feature learning for image classification from raw images. Our preliminary work [38] developed an automated ensemble framework using GP (EGP) for image classification and achieved promising results on different types of image classification tasks. The EGP method can learn features using filtering and pooling operators and use these features to build classifiers of an ensemble. The main advantage of EGP is the significant reduction of domain knowledge requirement in the whole process, where the selections of image-related operators, classification algorithms and combination methods are automatically conducted and optimised during the evolutionary learning process. However, the EGP method has shown an inferior performance on large image classification datasets, which might happen because the learned features are not effective. In addition, EGP cannot automatically select parameters for the classification algorithms, which is not flexible and efficient for different image classification tasks. Therefore, this chapter significantly improves the EGP method by developing the IEGP approach to automatically learning meaningful features and selecting suitable parameters for classification algorithms to build effective ensembles for image classification.

As shown in Figure 5.1, the inputs of an IEGP solution are raw images and the outputs are class labels. All the processes, such as feature extrac-

tion, base learner selection and combination, are coupled within a single IEGP solution/program. To achieve this, a novel individual representation, a new function set and a new terminal set are developed in IEGP. This section will describe these three important components of IEGP, and then outline the overall algorithm of IEGP for image classification.

5.2.1 Novel Individual Representation

GP has a tree-based representation, which is known for the variable lengths of the evolved solutions. The individual representation of the proposed IEGP approach is based on STGP [161], where each function has input and output types, and each terminal has an output type. To define the type constraints, a new program structure is developed, as shown in Figure 5.2. The new program structure comprises input, filtering & pooling, feature extraction, concatenation, classification, combination, and output layers. Each layer, except for the input and output layers, has different functions for different purposes. The input layer represents the input of the IEGP system, i.e., the terminals. The filtering & pooling layer has filtering and pooling functions, which operate on images. The feature extraction layer extracts informative features from the images using existing feature extraction methods. The concatenation layer concatenates features produced by its children nodes into a feature vector. The filtering & pooling, feature extraction and concatenation layers belong to the process of feature learning, where informative features are learned from raw images. The learned features can be directly fed into any classification algorithm for classification. Therefore, a classification layer is connected with the concatenation layer. The classification layer has several classification functions that can be used to train the classifiers using the learned features. The combination layer has several combination functions to combine the outputs of the classification functions. The classification and combination layers belong to the process of ensemble learning, where the classification functions are se-

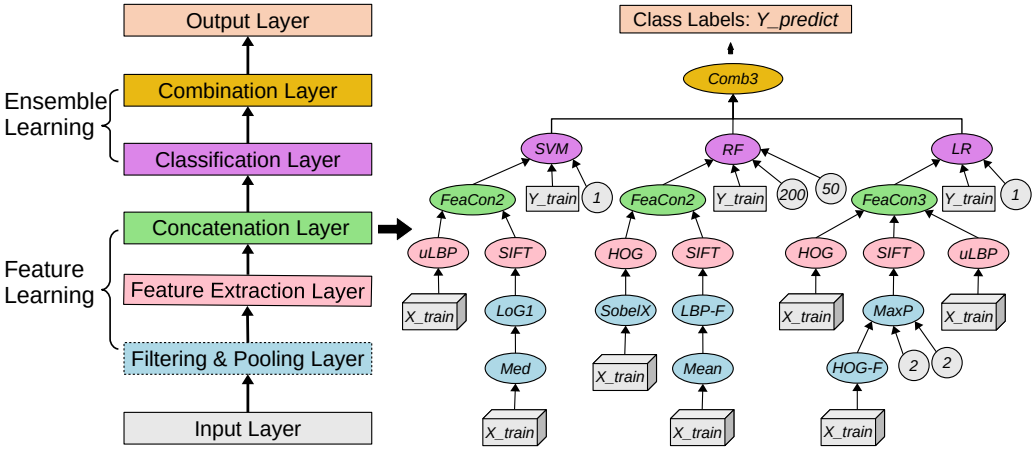


Figure 5.2: The new program structure of IEGP and an example solution/program that can be evolved by IEGP.

lected and trained, and the outputs of the classifiers are combined. Finally, the output layer performs the plurality voting on the outputs produced by the combination layer to obtain the combined predictions.

Compared with the program structure of the EGP method [38], the new program structure of the IEGP approach has one more layer, i.e., the feature extraction layer. The features learned by the previous EGP method are from the filtering and pooling operations, which might not be robust to variations such as scale and rotation. To improve the effectiveness of the learned features, a feature extraction layer is inserted between the filtering & pooling layer and the concatenation layer. The feature extraction layer utilises three well-developed image descriptors, i.e., LBP, SIFT, and HOG, to extract invariant and informative features from raw images. The three feature extraction functions will be introduced in Section 5.2.2.

An example solution of IEGP is shown in Figure 5.2 to further demonstrate the new representation. Figure 5.2 shows that different layers have different functions, which will be introduced in the following subsections. In the feature learning layers, i.e., the filtering & pooling layer and the feature extraction layer, a set of different image-related operators, such as

$LoG1$, HOG and $SIFT$, are employed. The classification layer has different classification functions, e.g., SVM, RF and LR and these functions have new additional leaf nodes in contrast to that in EGP. These leaf nodes are important parameters of the classification functions. In the combination layer, new combination functions, e.g., $Comb3$, are used to combine the predicted class labels in a flexible way. Therefore, the output of the example program in Figure 5.2 is the combined output of the SVM, RF and LR classifiers from the three branches.

Diversity of Ensembles

There are four commonly used strategies to enhance the diversity of an ensemble [249], (1) data sample manipulation, (2) input feature manipulation, (3) learning parameter manipulation, and (4) output representation manipulation. The ensembles evolved by IEGP can automatically address the diversity issue, adopting the strategies belonging to input feature manipulation and learning parameter manipulation. The new program structure allows that different classification functions have different tree branches in an evolved program, as the example program shown in Figure 5.2. The different tree branches with functions can produce various features to form the inputs of the classification functions, which is the input feature manipulation strategy. The parameters of the classification functions are developed as terminals of IEGP, which allows IEGP to automatically fine-tune the parameters during the evolutionary learning process (this will be introduced in subsection 5.2.3). This is the learning parameter manipulation. In addition, the new program structure allows IEGP to evolve ensembles of the same or different classifiers, which is more flexible than the other ensemble methods with fixed classifiers, such as in [249].

Flexibility of Representation

The new program structure enables IEGP to evolve programs with various tree sizes (nodes) and depths. In the new representation, the input layer and the output layer have a fixed tree depth of one. The feature extraction layer transforms images into features, which is an irreversible data transformation so that the depth of this layer is one. Similarly, the depth of the classification layer is one, where the inputs are transformed into class labels. The remaining layers, i.e., the filtering & pooling layer, the concatenation layer, and the combination layer, have a flexible/variable tree depth, which indicates that the tree depth can be automatically adjusted according to the problems. This maintains the flexibility of the evolved solutions by IEGP. Another point is that the filtering & pooling layer may not be necessary for a particular task. Therefore, the filtering & pooling layer is developed as a flexible layer, which indicates that it might be or might not be comprised in the IEGP programs. As shown in the example program in Figure 5.2, the inputs X_train can be directly fed into the feature extraction functions $uLBP$ and HOG without any filtering & pooling operations.

5.2.2 New Function Set

The new function set has a number of different functions for different purposes. The functions for each layer are summarised in Table 5.1. The introduction of these functions is in a direction from the bottom layer to the top output layer.

Filtering & Pooling Functions

The filtering & pooling functions operate on arrays. They take a number of images as inputs and conduct corresponding operations on the input image. The filtering functions keep the size of the output images consistent with the input size. The pooling function reduces the size of

Table 5.1: Function Set

Layer	**Function**
Filtering & Pooling	Gau, $GauD$, $Gabor$, Lap, $LoG1$, $LoG2$, $Sobel$, $SobelX$, $SobelY$, $LBP\text{-}F$, $HOG\text{-}F$, Med, $Mean$, Min, Max, $Sqrt$, $W\text{-}Add$, $W\text{-}Sub$, $ReLU$, $MaxP$
Feature Extraction	$SIFT$, HOG, $uLBP$
Concatenation	$FeaCon2$, $FeaCon3$, $FeaCon4$
Classification	LR, SVM, RF, ERF
Combination	$Comb3$, $Comb5$, $Comb7$

images by subsampling maximum values from a small window of the image. Gau is the Gaussian filter generated by a 2D Gaussian function $Gau(x,\ y) = \frac{1}{2\pi\sigma^2}\exp[-\frac{x^2+y^2}{2\sigma^2}]$, where the standard deviation σ is a terminal of IEGP. $GauD$ represents the derivative of the Gaussian function with the standard deviation σ in the orders $(o_1,\ o_2)$ along the X axis and the Y axis, respectively. The three parameters are terminals of IEGP. The kernel of the $Gabor$ function is generated according to $Gabor(x,\ y) = Gau(x, y) * sin[(\frac{2\pi(cos\theta x+sin\theta y)}{\lambda} + \psi)]$. The frequency f ($\frac{1}{\lambda}$) and the orientation θ are developed as terminals of IEGP. Lap is the Laplacian filter, which is commonly used to detect the flat area. In the case that the results produced by Laplacian are noisy, the Laplacian of Gaussian filter is used to convolve the image produced by Laplacian using the Gaussian filter. $LoG1$ and $LoG2$ are the Laplacian of Gaussian filters, where the σ for the Gaussian function is 1 and 2, respectively. $SobelX$, $SobelY$ and $Sobel$ conduct edge detection on the images. $HOG\text{-}F$ and $LBP\text{-}F$ produce the HOG and LBP images with informative features. Med, $Mean$, Min, and Max convolve images by returning the median, mean, minimum, and maximum values of each 3×3 sliding window, respectively. $Sqrt$ returns the sqrt root of each pixel value and is protected by returning 1 if the pixel value is negative. $W\text{-}Add$ and $W\text{-}Sub$ take two images and two weights as inputs and calculate the

weighted sum or subtraction of the images. If the input images have different sizes, the two functions will overlap the two images at the left top point and cut them to have the same sizes for sum or subtraction. $ReLU$ returns 0 if the pixel value is negative. Otherwise, it returns the pixel value.

Feature Extraction Functions

Three commonly used feature extraction methods, uniform LBP (uLBP) [169], HOG [60] and SIFT [147], are used as functions for IEGP. These functions transform the input image into a set of effective features, which are invariant to certain variations. The $uLBP$ function extracts 59 histogram features of the LBP image. In $uLBP$, the radius is set to 1.5 and the number of neighbours is set to 8 [169]. The HOG function extracts the mean value of each 4×4 grid from the HOG image to form the feature vector. The parameter settings for the HOG method refer to [60]. The $SIFT$ function produces 128 features for each image by taking the image as a keypoint [220]. It is noticeable that the three functions produce various numbers of features.

Concatenation Functions

The concatenation functions in IEGP are different from that in the EGP method [38]. The $FeaCon2$, $FeaCon3$ and $FeaCon4$ functions convert two, three and four vectors as a vector by concatenating them, respectively. The concatenation functions can take the feature extraction functions or the concatenation functions as their children nodes, which indicates that a combination of features is produced by each concatenation function.

Classification Functions

Any commonly used classification algorithm can be used as a function in the classification layer for IEGP. To narrow the search space, the previous EGP method [38] employs six classification algorithms, logistic regression

(LR), k-nearest neighbour (KNN), support vector machine (SVM), random forest (RF), extremely randomised trees (ERF), and adaptive boosting ($AdaBoost$), according to [235]. However, since KNN is an instance-based method and is computationally expensive when the training data is large, the KNN function is not employed in IEGP. The $AdaBoost$ method achieves an inferior performance on image classification [38] so that it is also removed from the function set. Therefore, only four classification functions are employed in IEGP. They are LR, SVM, RF, and ERF, including linear classification methods and tree-based (non-linear) classification methods. Note that RF and ERF are ensemble methods and the IEGP approach can build ensembles of ensembles.

The previous EGP method uses fixed parameters for the classification functions, which is not effective and efficient for solving different tasks. Therefore, the key parameters for LR, SVM, RF, and ERF are developed as terminals of IEGP to allow their values to be automatically generated or optimised during the evolutionary process. These key parameters and their ranges are introduced in Section 5.2.3.

Combination Functions

The previous EGP method [38] conducts voting multiple times, which is computationally expensive. To address this, three new functions are developed in IEGP to combine the classifiers effectively. The functions are $Comb3$, $Comb5$ and $Comb7$, which take 3, 5 and 7 class labels as inputs and concatenate these labels to form the output, respectively. These functions can be root or internal nodes of a GP tree, which represents different ways of combining the classifiers, as shown in Figure 5.3. In the left example (Ensemble 1) of Figure 5.3, the $Comb3$ function is used as the root node. The inputs for $Comb3$ are 1 from SVM, 1 from RF, and 0 from LR. The $Comb3$ function combines these inputs and returns 110. In the right example (Ensemble 2) of Figure 5.3, $Comb3$ is an internal node and $Comb5$ is the root node. The outputs of $Comb3$ are 012 and the outputs of $Comb5$ are

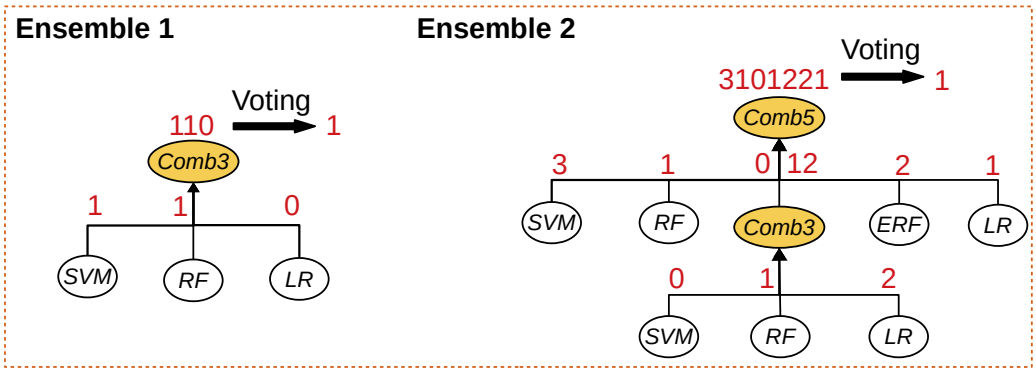


Figure 5.3: Example ensembles with the new $Comb3$ and $Comb5$ functions as root nodes and internal nodes.

3101221 from the nodes of SVM, RF, $Comb3$, ERF, and LR. After obtaining the outputs from the root node, the plurality voting is conducted to produce the predicted class label for an instance/image. As shown in the Figure 5.3, the final output (class label) of Ensemble 1 is 1 (from 110) and the final output (class label) of Ensemble 2 is 1 (from 3101221).

5.2.3 New Terminal Set

The terminal set represents the inputs of the IEGP system. Table 5.2 lists the terminals of IEGP and their ranges. The $X\_train = \{X_i\}_{i=1}^{N}$ represents N input training images and the $Y\_train = \{Y_i\}_{i=1}^{N}$ represents the class labels of the N images. In X_train, X_i represents an image of size $M \times L$, where the pixel values in the image are in the range of $[0,\ 1]$. In Y_train, Y_i is an integer representing the class label. The σ, o_1, o_2, θ, n_1, n_2, k_1, and k_2 terminals are the parameters for particular functions in the filtering & pooling layer and their values are in the commonly used ranges.

The important parameters for the classification functions are designed as terminals of IEGP. The parameters are C, NT and MD. The C terminal is an integer and is related to the penalty term/parameter for the classification functions LR and SVM. The range for C is set to $[-2,\ 5]$ [25], which

Table 5.2: Terminal Set

Terminal	Type	Description
X_train	Array	N training images with a size of $M \times L$
Y_train	Array	The class labels of N training images
σ	Integer	The standard deviation of the Gaussian filter in the Gau and $GauD$ functions. Its range is $\{1,\ 2,\ 3\}$
o_1, o_2	Integer	The orders of the Gaussian derivatives. They are in range $\{0,\ 1,\ 2\}$
θ	Float	The orientation of the $Gabor$ function. It is in the range of $[0,\ 7\pi/8]$ with a step of $\pi/8$ [143]
f	Float	The frequency of the $Gabor$ function. It equals to $(\pi/2)/(\sqrt{2}^{v})$, where v is an integer in the range of $[0,\ 4]$ [143]
n_1, n_2	Float	The parameters for the *W-Add* and *W-Sub* functions. They are randomly generated from the range of $[0,\ 1)$
k_1, k_2	Integer	The kernel size for $MaxP$. They are in range $\{2,\ 4\}$
C	Integer	The penalty term/parameter in LR and SVM is 10^C, where C is in the range of $[-2,\ 5]$ according to [25]
NT	Integer	The number of trees in RF and ERF. It is in the range of $[50,\ 500]$ with a step of 10
MD	Integer	The maximum tree depth of the decision tree in RF and ERF. It is in the range of $[10,\ 100]$ with a step of 10

results in a penalty term/parameter (10^C) in the range of $\{10^{-2},\ 10^{-1},\ \ldots,\ 10^{4},\ 10^{5}\}$. The number of decision trees and the maximum tree depth are two important parameters for RF and ERF according to [249]. Therefore, they are developed as terminals, NT and MD. The values for NT is in the range of $[50,\ 500]$ with a step of 10 and the values for MD is in the range of $[10,\ 100]$ with a step of 10. The maximum values for NT and MD are set according to that in [249]. To reduce the computational cost, a smaller NT and MD are desired to be found for RF and ERF. In ad-

dition, a step of 10 is used to limit the dimension of the search space of IEGP.

5.2.4 Overall Algorithm

With the new program structure, the new function set and the new terminal set, the IEGP approach can automatically learn features and ensembles for image classification. The overall algorithm of IEGP is described in Algorithm 4. The flowchart of the overall algorithm (include the training and test processes) is shown in Figure 5.4.

The IEGP approach starts with randomly initialising the population P_0 using the *ramped half-and-half* method. Each individual in P_0 is evaluated by a fitness function. During the evolutionary learning process (at generation g), the elitism, subtree crossover and subtree mutation operators are used to create a new population P_g. The new P_g is then evaluated. When g equals the maximum number of generations, the evolutionary learning process stops and the best individual (ensemble) is returned.

During the evolutionary learning process, a subtree caching method is used to reduce the evaluation time since GP is known for being computationally expensive on image data [184]. A $Cache\_Table$ is used to store individuals and their fitness values. At generation 0, the $Cache\_Table$ stores the P_0 with fitness values. At generation g, the $Cache\_Table$ stores the best individuals of the past generations and all the individuals in generation $g-1$ [184]. To evaluate individual o at g $(g > 1)$, a search is conducted in $Cache\_Table$ to check whether o has been evaluated before. If o has been in the $Cache\_Table$, the fitness value will be directly assigned to o, otherwise, o is evaluated using the fitness function. Generally, any individual can be stored in $Cache\_Table$ but a tradeoff between the searching time and the evaluation time for an individual should be considered. Therefore, in IEGP, the size of $Cache\_Table$ is set to $6 \times N_p$ (N_p is the population size).

Algorithm 4: Framework of IEGP

Input : X_train: N training images; Y_train: the class labels of N training images.

Output : $Best\_Individual$: the best program tree.

```
P_0 ← Initialise a population based on the new representation, the new function set and the new terminal set;
g ← 0;
Cache_Table ← ∅;
for each individual p in P_0 do
    f_p ← Evaluate p using the fitness function
end
Cache_Table ← P_0;
while g < G do
    I ← Best individuals of P_g using elitism operator;
    S ← Selected individuals from P_g by tournamanet selection;
    O_{g+1} ← Offspring generated from S using subtree crossover and subtree mutation;
    for each individual o in O_{g+1} do
        if o in Cache_Table then
            f_o ← the fitness value of o in Cache_Table;
        else
            f_o ← Evaluate o using the fitness function;
        end
    end
    P_{g+1} ← O_{g+1} ∪ I;
    Update Best_Individual and Cache_Table;
    g ← g + 1;
end
Return Best_Individual.
```

Training Process and Fitness Function

In the training process of IEGP, the training data X_train and Y_train are fed into the IEGP system. Since the classification functions in each tree require training, we employ stratified *k*-fold cross-validation on the training set (X_train and Y_train) to build and evaluate the classifier in the training

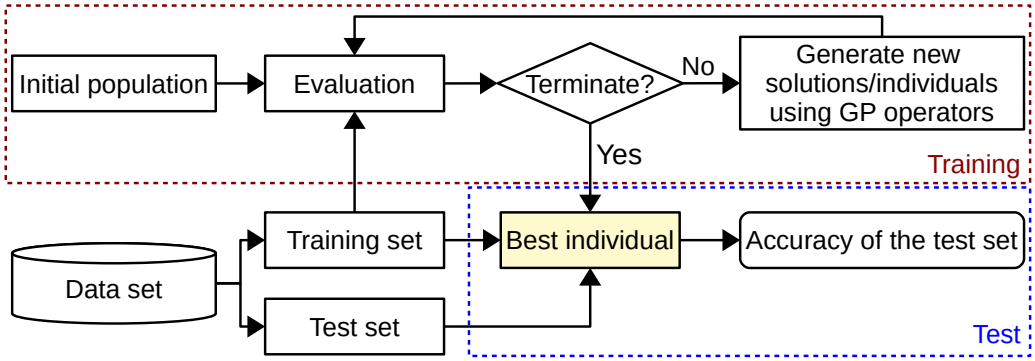


Figure 5.4: The flowchart of the overall IEGP algorithm.

process of IEGP. For each classification function in the evolved IEGP tree, $k-1$ folds are used to train the classifier and the remaining one fold is used as the test set. The trained classifier is employed to obtain the predicted class labels for the test set (the one fold). This process is repeated k times to obtain the predicted class label for each instance in X_train. The value of k is set to 3 to reduce the computational cost, which is also recommended in [249]. The predicted class labels from different nodes (classification functions) are then combined by the combination functions and voted by the plurality voting to form the output $Y\_predict$.

The fitness function for IEGP is the classification accuracy, which is the percentage of the number of correctly classified images out of the total number of images in the training set. In IEGP, the classification accuracy is calculated according to $Y\_predict$ and Y_train.

After the evolutionary process, the best IEGP tree (ensemble) is re-trained using X_train and Y_train without k-fold cross-validation (X_train. The ensemble of trained classifiers is used for classifying images in the test set X_test.

Test Process

The test process is to use the best tree/ensemble evolved by IEGP to classify the images in an unseen (test) set X_test. The ensemble of the trained

classifiers is used to classify images in the test set. It is noted that the images in the test set are transformed by the nodes of feature learning in the IEGP tree. Then the features are fed into the trained classifiers to obtain the predicted class labels. The class labels from multiple classifiers are combined and the final prediction can be obtained. Based on the class labels, the accuracy of the test set is calculated and reported.

5.3 Experiment Design

A number of experiments are conducted to show the effectiveness of the proposed approach. The detailed design of experiments is described in this section.

5.3.1 Benchmark Datasets

The performance of the proposed IEGP approach is examined on 12 benchmark datasets, which are the same as those in Chapter 4. These datasets are FEI_1 [215], FEI_2 [215], ORL [191], KTH [153], FS [79], MB [132], MRD [132], MBR [132], MBI [132], Rectangle [132], RI [132], and Convex [132]. These datasets include a broad variety of image classification tasks, i.e., facial expression classification: FEI_1 and FEI_2; face recognition: ORL; texture classification: KTH; scene classification: FS; and object classification: MB, MRD, MBR, MBI, Rectangle, RI, and Convex. The details of these datasets are listed in Table 4.5.

5.3.2 Benchmark Methods

For the datasets 1-5, we use 14 different benchmark methods for comparisons. The aim is to comprehensively investigate whether IEGP can learn informative features and evolve effective ensembles for image classification. These benchmark methods are six classification algorithms using

raw pixels, i.e., SVM, KNN, LR, RF, AdaBoost, and ERF, four SVM methods using different features, i.e., uLBP+SVM, LBP+SVM, HOG+SVM, and SIFT+SVM, two CNNs, i.e., CNN-5 and CNN-8, the EGP method [38], and the FGP method proposed in Chapter 4. Except for EGP and FGP, the other 12 benchmark methods are the same as those employed in Chapter 4.

The datasets 6-12 have public training and test sets, so the test accuracy reported in literature can be directly used for comparisons without reimplementing the methods. The results of these methods are collected from corresponding references on datasets 6-12. Therefore, there are 20 comparison methods on datasets 6-12, i.e., SVM+RBF [132], SVM+Poly [132], SAE-3 [183], DAE-b-3 [183], CAE-2 [183], SPAE [236], RBM-3 [183], ScatNet-2 [49, 53], RandNet-2 [53], PCANet-2 (softmax) [53], LDANet-2 [53], NNet [132], SAA-3 [132], DBN-3 [132], FCCNN [177], FCCNN (with BT) [177], SPCN [140], EGP [38], FGP proposed in Chapter 4, and EvoCNN [208]. Note that most of these methods are neural network-based methods and parameter tuning was used in some methods to optimise performance. Except for EGP and FGP, the other benchmark methods are the same as those employed in Chapter 4.

5.3.3 Parameter Settings

The parameter settings for IEGP are these commonly used settings within the GP community [114]. The population size is set to 100, which is smaller than in FGP due to the high computational cost of training multiple classifiers in a GP tree. The maximum number of generations is 50. The elitism rate is 0.01, the crossover rate is 0.8 and the mutation rate is 0.19. Tournament selection with a size of seven is used to select individuals for mutation and crossover during the evolutionary learning process. The tree depth is between two and eight. In IEGP, the type constraint has higher priority than the depth constraint so that the tree depth may be larger than 8 in some cases.

The implementation of IEGP is in Python using the DEAP (*Distributed Evolutionary Algorithm in Python*) [86] package. The implementations of the classification algorithms in IEGP and the benchmark methods are based on the *scikit-learn* package [172] and the *Keras* package [57]. Note that the other parameters (except for the optimised ones) of these classification algorithms are the default settings in *scikit-learn* for simplification. The experiment of IEGP on each dataset has been executed for 30 independent runs and the best tree of each run has been tested on the test set.

5.4 Results and Discussions

This section discusses and analyses the classification performance of the proposed IEGP approach and the benchmark methods, including the EGP method, on the 12 benchmark datasets.

5.4.1 Classification Results on Datasets 1-5

The classification results, i.e., the maximum accuracy (Max), the average accuracy and standard deviation (Mean$\pm$St.dev) of 30 runs by the IEGP approach and the benchmark methods on the six datasets are listed in Tables 5.3 and 5.4. The Wilcoxon rank-sum test with a 5% significance level is used to compare the IEGP approach with a benchmark method to show the significance of the differences. The symbols "+", "−" and "=" in Tables 5.3 and 5.4 indicate that the IEGP approach is significantly better, significantly worse or similar than/to the benchmark method. The final row of each table summaries the overall results of the significance test. The best maximum accuracy and mean accuracy of each dataset are highlighted in bold in Tables 5.3 and 5.4.

Table 5.3 shows that the IEGP approach performs significantly better than or similar to any of the benchmark methods on FEI_1 and FEI_2, which are facial expression classification tasks. The IEGP approach finds

Table 5.3: Classification Accuracy (%) of Datasets 1-3

	Max	Mean±St.dev	Max	Mean±St.dev	Max	Mean±St.dev
	FEI_1		**FEI_2**		**ORL**	
SVM	90.00	90.00±0.00+	88.00	88.00±0.00+	94.38	94.38±0.00+
KNN	32.00	32.00±0.00+	8.00	8.00±0.00+	94.38	94.38±0.00+
LR	92.00	92.00±0.00+	88.00	88.00±0.00+	93.75	93.75±0.00+
RF	98.00	**97.07±1.01**=	90.00	89.20±1.13+	93.12	92.33±0.63+
AdaBoost	80.00	78.67±1.32+	80.00	76.00±3.44+	59.38	52.27±4.00+
ERF	94.00	93.27±0.98+	92.00	90.60±0.93+	97.50	96.71±0.59+
uLBP+SVM	66.00	56.73±3.66+	68.00	62.53±3.52+	87.50	87.42±0.21+
LBP+SVM	68.00	64.60±1.83+	74.00	69.80±0.00+	88.12	87.52±0.20+
HOG+SVM	96.00	96.00±0.00=	82.00	82.00±0.00+	91.25	91.25±0.00+
SIFT+SVM	56.00	56.00±0.00+	62.00	62.00±0.00+	93.75	93.75±0.00+
CNN-5	98.00	95.40±1.30+	98.00	95.27±1.62+	96.88	95.29±1.06+
CNN-8	98.00	95.33±1.32+	96.00	90.93±1.87+	95.00	93.04±1.09+
EGP	**100.0**	96.20±2.06=	**100.0**	**98.07±1.70**=	99.38	97.44±1.26+
FGP	98.0	94.47±2.67+	96.00	91.33±3.36+	**100.0**	**98.63±1.04**=
IEGP	**100.0**	96.67±2.55	**100.0**	96.20±3.66	**100.0**	98.29±0.97
Overall		11+, 3=		13+, 1=		13+, 1=

the best accuracy of 100% on these two datasets. Although RF achieves the best mean accuracy on FEI_1 and EGP achieves the best mean accuracy on FEI_2, there is no significant difference between IEGP and EGP or RF in the performance on the two datasets.

The classification results in Table 5.3 show that IEGP achieves similar performance to FGP and significantly better performance than any of the other 13 benchmark methods on ORL. ORL is a face recognition task of 40 classes and has a very small number of training images, which is challenging for some methods needing a large number of training instances such as CNN-5 and CNN-8. IEGP found the best accuracy of 100% and the best mean accuracy of 98.29% on ORL, which shows the effectiveness of IEGP on datasets with a small number of training instances.

The classification results on the KTH and FS datasets in Table 5.4 indi-

Table 5.4: Classification Accuracy (%) of Datasets 4-5

	KTH		FS	
	Max	Mean±St.dev	Max	Mean±St.dev
SVM	46.97	44.59±2.83+	20.63	20.30±0.15+
KNN	34.24	34.24±0.00+	24.35	24.35±0.00+
LR	48.79	48.79±0.00+	23.49	23.49±0.00+
RF	60.00	57.81±0.83+	37.36	36.53±0.49+
AdaBoost	37.88	33.44±1.37+	17.47	13.04±1.47+
ERF	61.52	59.83±0.86+	37.94	37.15±0.36+
uLBP+SVM	78.79	73.29±4.18+	49.79	33.27±8.90+
LBP+SVM	83.64	82.71±0.51+	53.50	50.45±1.80+
HOG+SVM	57.27	55.96±0.64+	12.11	7.91±2.47+
SIFT+SVM	65.76	65.76±0.00+	60.92	60.92±0.00+
CNN-5	85.76	82.56±1.87+	50.14	48.03±1.16+
CNN-8	76.36	71.63±3.18+	49.16	46.79±1.01+
EGP	87.88	77.53±5.17+	67.17	61.07±2.91+
FGP	**98.79**	96.07±1.13=	74.48	71.59±1.74+
IEGP	98.48	**96.43±1.26**	**92.54**	**89.63±1.47**
Overall		13+, 1=		14+

cate that IEGP achieves significantly better results in almost all the comparisons. The KTH dataset has texture images and the FS dataset has natural scene images. The SVM, KNN, LR, RF, and AdaBoost methods achieve very low accuracy on these two datasets, which indicates that using raw pixels to classify these two datasets is not effective. However, simple feature extraction cannot improve accuracy as the uLBP+SVM, LBP+SVM, HOG+SVM, and SIFT+SVM methods achieve low accuracy as well. By automatically learning features and evolving ensembles for classification, IEGP achieves the best results on these two datasets, i.e., a maximum accuracy of 98.48% on KTH and a maximum accuracy of 92.45% on FS. Importantly, the IEGP approach improves the mean accuracy by 0.36% on KTH and by 18.04% on FS. The results indicate that IEGP is very effective for texture classification and scene classification.

Table 5.5 summarises the overall results of the significance tests in the

Table 5.5: Summary of Significance Test on Datasets 1-5

	A	B	C	D
Significantly better (+)	29	19	10	6
Similar (=)	1	1	0	4
Significantly worse (–)	0	0	0	0

A: compare IEGP with SVM, KNN, LR, RF, AdaBoost, and ERF
B: compare IEGP with uLBP+SVM, LBP+SVM, HOG+SVM, and SIFT+SVM
C. compare IEGP with CNN-5 and CNN-8
D. compare IEGP with EGP and FGP

comparisons of different benchmark methods on the six datasets. IEGP achieves significantly better results than or similar results to any of the six classification algorithms using raw pixels (in case A), which indicates that IEGP can learn informative features for effective image classification. Compared with the four methods (in case B), which use uLBP, LBP, SIFT, and HOG features as inputs for SVM, IEGP achieves significantly better results than or similar results to any of them. The results show that IEGP is more effective than the four methods by learning informative features and evolving ensembles for classification. Compared with CNN-5 and CNN-8 (in case C), IEGP achieves significantly better results on the five datasets, which shows that IEGP is more effective than simple CNN-based methods. The main advantage of IEGP over the CNN-based methods is the flexible length and depth of the evolved solutions. IEGP does not need to define the model/solution structure (complexity) as it is able to find a suitable model during the evolutionary learning process. In addition, CNNs often require a large number of training instances. In contrast, IEGP can evolve effective solutions from a relatively small number of training instances, as it does on the ORL dataset. Compared with FGP, the IEGP approach achieves significantly better performance on three datasets and similar performance on two datasets. The FGP approach only uses SVM for classification, while IEGP uses ensemble classifiers for classification.

The results indicate that the classification performance of these datasets can be further improved by using GP to evolve ensembles for classification. The results show that IEGP performs significantly better than or similar to EGP. Compared with EGP, the IEGP approach uses a new representation, a new function set and a new terminal set, which allow it to learn more effective features and to find more suitable classification algorithms with appropriate parameters to form effective ensembles for classification.

5.4.2 Classification Results on Datasets 6-12

The classification accuracy (%) of the datasets 6-12 are listed in Table 5.6. On these datasets, 20 methods are employed for comparisons, where the results are collected from the corresponding references. Note that some of these 20 methods may not have results on some datasets such as RI, Rectangle and Convex. In Table 5.6, each column shows the results of all these methods on one dataset. The results obtained by IEGP are listed at the bottom of the table. Because most of these benchmark methods have only reported the best results on these datasets, we compare IEGP with them using the best results. In Table 5.6, the symbol "+" denotes that IEGP achieves better accuracy than the corresponding benchmark method. The final two rows of Table 5.6 summarise the ranking results of IEGP among all these methods.

Table 5.6 shows that IEGP achieves better (or similar) classification accuracy than any of the benchmark methods with reported results on one dataset, i.e., Rectangle. On the MB, MRD, RI, and Convex datasets, the best accuracy of IEGP ranks second among all the methods, which indicates that only one method achieves better accuracy than IEGP on any of these four datasets. On the remaining two datasets, the best accuracy of IEGP ranks third among all the methods on MBR and ranks fourth on the MBI dataset, which indicates that only two and three methods achieve better accuracy than IEGP on these two datasets, respectively. IEGP achieves

Table 5.6: Classification Accuracy (%) of Datasets 6-12

Method	MB	MRD	MBR	MBI	Rectangle	RI	Convex
SVM+RBF [132]	96.97(+)	88.89(+)	85.42(+)	77.39(+)	97.85(+)	75.96(+)	80.87(+)
SVM+Poly [132]	96.31(+)	84.58(+)	83.38(+)	75.99(+)	97.85(+)	75.95(+)	80.18(+)
SAE-3 [183]	96.54(+)	89.70(+)	88.72(+)	77.00(+)	97.86(+)	75.95(+)	–
DAE-b-3 [183]	97.16(+)	90.47(+)	89.70(+)	83.32(+)	98.01(+)	78.41(+)	–
CAE-2 [183]	97.52(+)	90.34(+)	89.10(+)	84.50(+)	98.79(+)	78.46(+)	–
SPAE [236]	96.68(+)	89.74(+)	90.99(+)	86.76(+)	–	–	–
RBM-3 [183]	96.89(+)	89.70(+)	93.27(+)	83.69(+)	97.40(+)	77.50(+)	–
ScatNet-2 [49, 53]	98.73(+)	92.52(+)	87.70(+)	81.60(+)	99.99(+)	91.98(+)	93.50(+)
RandNet-2 [53]	98.75(+)	91.53(+)	86.53(+)	88.35(+)	99.91(+)	83.00(+)	94.55(+)
PCANet-2 (softmax) [53]	98.60(+)	91.48(+)	93.15(+)	88.45(+)	99.51(+)	86.61(+)	95.81(+)
LDANet-2 [53]	**98.95**	92.48(+)	93.19(+)	87.58(+)	99.86(+)	83.80(+)	92.78(+)
NNet [132]	95.31(+)	81.89(+)	79.96(+)	72.59(+)	92.84(+)	66.80(+)	67.75(+)
SAA-3 [132]	96.54(+)	89.70(+)	88.72(+)	77.00(+)	97.59(+)	75.95(+)	81.59(+)
DBN-3 [132]	96.89(+)	89.70(+)	93.27(+)	83.69(+)	97.40(+)	77.50(+)	81.37(+)
FCCNN [177]	97.57(+)	91.09(+)	93.55(+)	86.77(+)	–	–	–
FCCNN (with BT) [177]	97.32(+)	90.41(+)	93.03(+)	89.20(+)	–	–	–
SPCN [140]	98.18(+)	90.19(+)	94.16	90.45	99.81(+)	89.40(+)	–
EvoCNN (best) [208]	98.82	**94.78**	**97.20**	**95.47**	99.99(+)	**94.97**	95.18(+)
EGP (best) [38]	97.19(+)	–	–	–	99.91(+)	–	93.97(+)
FGP (best)	98.82	92.63(+)	93.46(+)	92.52	**100**	93.90(+)	**98.46**
IEGP (best)	98.82	94.28	93.59	89.41	**100**	94.88	98.26
IEGP (mean)	98.69	93.78	92.65	88.42	99.94	89.02	97.76
IEGP (std)	0.08	0.24	0.35	0.64	0.05	2.1	0.26
Rank	2/21	2/20	3/20	4/20	**1/18**	2/17	**2/13**

100% accuracy on the Rectangle dataset, although it is only 0.01% higher than the best results of the benchmark methods. Note that these benchmark methods have been explored extensively on these datasets, therefore, even 1% improvement in accuracy is very difficult to achieve.

On the MB dataset, the IEGP approach achieves better (or the same) results than any of the 19 benchmark methods except for LDANet-2. IEGP achieves a maximum accuracy of 98.82%, which is slightly lower than the accuracy of 98.95% achieved by LDANet-2. Although IEGP achieves

worse results than LDANet-2 on MB, it achieves better results on the other six datasets. The three variants of the MB dataset, i.e., the MRD, MBR, and MBI datasets, are more difficult than the MB dataset by adding variational factors including rotation and background change. On the MRD dataset, IEGP achieves better results than any of the 19 benchmark methods except for EvoCNN. On the MBR dataset, IEGP is better than any of the 18 benchmark methods except for SPCN and EvoCNN, which are CNN-based methods. On the MBI dataset, IEGP is better than any of the 17 benchmark methods except for SPCN, EvoCNN, and FGP. The SPCN method is more effective than IEGP on these two datasets but less effective on the other four datasets. Particularly, IEGP achieves 94.28% accuracy on MRD which is much higher than the accuracy (90.19%) achieved by SPCN. EvoCNN is a state-of-the-art CNN-based method by automatically evolving the architectures of CNNs. Compared with EvoCNN, IEGP achieves worse results on some difficult datasets, i.e., the MBR and MBI datasets, but IEGP achieves better (same) results on the MB, Rectangle and Convex datasets. This indicates that IEGP as a GP method is effective and promising for image classification. On the Rectangle dataset, IEGP achieves 100% accuracy. RI as a variant of Rectangle is more difficult. IEGP achieves the best accuracy among all the methods on the RI dataset except for EvoCNN. But the difference between the best results achieved by EvoCNN and IEGP on the RI dataset is very small, i.e., 94.97% (EvoCNN) vs. 94.88% (IEGP). On the Convex dataset, IEGP obtains a maximum accuracy of 98.26%, which is slightly lower than the best accuracy (98.46%) obtained by FGP among all the methods, but 3% higher than that achieved by EvoCNN. Compared with the FGP approach, the IEGP approach achieves better performance on three datasets, the same performance on two datasets and worse performance on two datasets. FGP only uses a single classifier for classification, while IEGP constructs an ensemble of accurate and diverse classifiers for classification. The results confirm the effectiveness of IEGP for constructing ensembles of various classifiers for image classification.

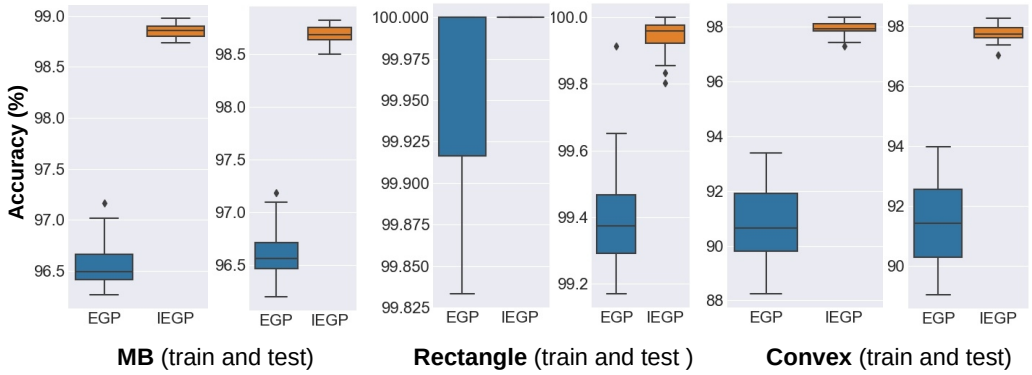


Figure 5.5: Comparisons of the distributions of the training and test results obtained by the EGP and IEGP approaches on the MB, Rectangle and Convex datasets.

Compared with EGP, the IEGP approach is more effective for classifying large-scale datasets. On three datasets, i.e., MB, Rectangle and Convex, IEGP achieves better results than EGP. Importantly, IEGP achieves a maximum accuracy of 98.26% on Convex, which is 4% higher than that achieved by EGP. To further compare EGP with IEGP, Figure 5.5 show the distributions of the training accuracy and test accuracy of the EGP and IEGP methods on the MB, Rectangle and Convex datasets. From this figure, it is clear that IEGP achieves much higher maximum classification accuracy and median accuracy than EGP on these three datasets. These results indicate that IEGP achieves better performance than EGP. In addition, the results obtained by IEGP are more clustered than those obtained by EGP, which means that IEGP is more stable than EGP. The results demonstrate that IEGP significantly improves the EGP method by having a new representation and a new function set for image classification.

5.5 Further Analysis

This section compares the convergence behaviours of EGP and IEGP, and analyses the trees/solutions evolved by IEGP to further understand what

features they extract and what classifiers in the evolved ensembles are built for image classification.

5.5.1 Convergence Behaviour of IEGP

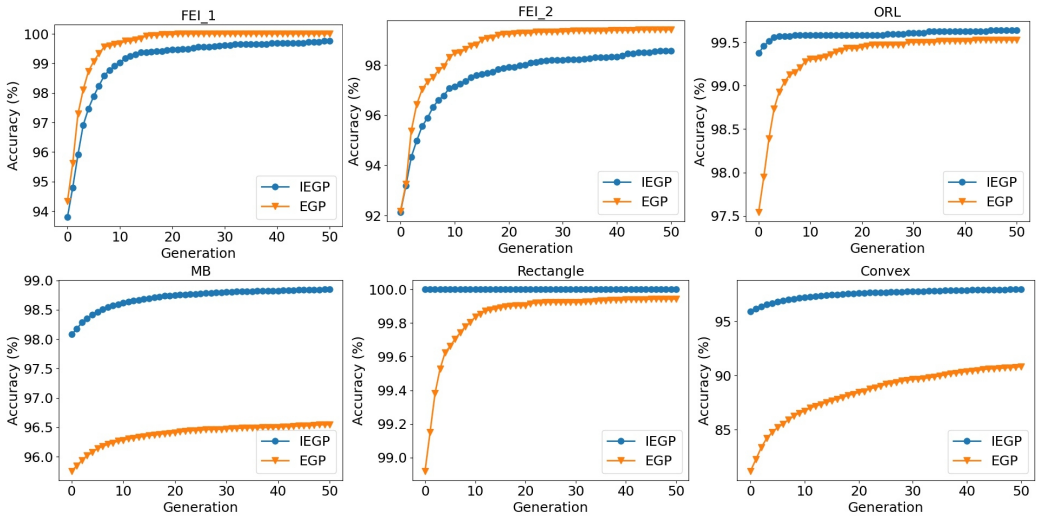


Figure 5.6: Convergence curve of the EGP method and the proposed IEGP method on the FEI_1, FEI_2, ORL, MB, Rectangle, Convex datasets.

Figure 5.6 shows the convergence behaviours of EGP and IEGP on the FEI_1, FEI_2, ORL, MB, Rectangle, Convex datasets. It can be found that the start points of IEGP are better than EGP on most of these datasets, indicating that the new designs of the program structure and the function set can improve the quality and effectiveness of the whole population. Compared with EGP, IEGP has faster convergence speed and can find better solutions with larger fitness value. Specifically, on the FEI_1 and FEI_2, the start points of EGP and IEGP are close and IEGP obtains better results than EGP at the final generation. This indicates the new individual representation of IEGP allows it to find better solutions with higher fitness values than the EGP method with the same parameter settings. In addition, it can be found from Figure 5.6 that IEGP can converge to a high point after 50 generations on these datasets.

5.5.2 Visualisation of Example Solutions

An Example Solution on MRD

Since IEGP achieves the best results on the MRD dataset, the best program/solution on this dataset is used for analysis and visualisation. The solution is visualised in Figure 5.7. This solution achieves 93.8% accuracy on the training set of MRD and 94.28% accuracy on the test set. Note that the example solution is used for testing so that the X_train node is replaced with the $Images$ node and the Y_train node is removed for simplification. It is clear from Figure 5.7 that the example solution is an ensemble of three ERF classifiers with different parameters. The left and right classifiers have 450 decision trees with a maximum tree depth of 60, while the middle classifier has 450 decision trees with a maximum tree depth of 30. It is obvious that the example solution is an ensemble of ensembles.

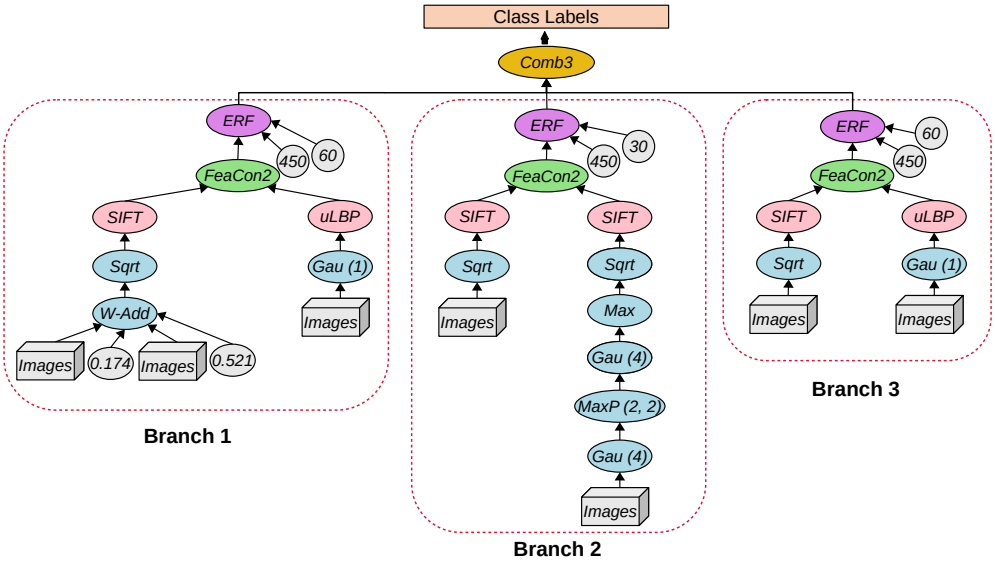


Figure 5.7: An example solution found by IEGP on the **MRD** dataset.

As shown in this figure, it is obvious that the three classifiers are trained using different features. The first branch uses the features that are the combination of 128 SIFT features and 59 uLBP features. Before extracting fea-

tures using the $SIFT$ and $uLBP$ functions, each image is processed by the $Sqrt$ function or the Gau function (with a standard deviation value of 1). The second branch uses 256 (128×2) SIFT features extracted from the images after the corresponding transformations, such as the $Sqrt$, Max, Gau, and $MaxP$ functions. The third branch uses 128 SIFT features and 59 uLBP features for classification. The $Sqrt$ function and the Gau function are employed to rescale and smooth the images before feature extraction. The $SIFT$ and $uLBP$ functions transform the processed images into features and the features are fed into the classification functions. By this analysis, it is clear that each branch extracts various numbers and types of features from images after corresponding filtering or pooling operations and uses these features to build different classifiers. This indicates that the inputs for each classifier in the example solution are different, which enhances the diversity of the classifiers in the ensemble.

An Example Solution on MBR

An example solution on the MBR dataset is visualised in Figure 5.8. This solution achieves 93.38% accuracy on the training set and 93.21% accuracy on the test set. Different from the solution in Figure 5.7, where the classifiers in the ensemble are trained from the same classification algorithm, this example solution is an ensemble of three classifiers trained from different classification algorithms. This shows that the IEGP approach can evolve ensembles of the same or different classifiers, which is very flexible for solving different image classification tasks.

In the ensemble in Figure 5.8, the first classifier is ERF, having 480 decision trees with a maximum tree depth of 90. The second classifier is SVM, where the value of the penalty parameter is 10^{-1}. The third classifier is LR, where the value of the penalty parameter is 10. From Figure 5.8, we can see that the three classifiers are trained using features extracted by different feature extraction functions in their children branches. The features extracted from this dataset are the combinations of the SIFT fea-

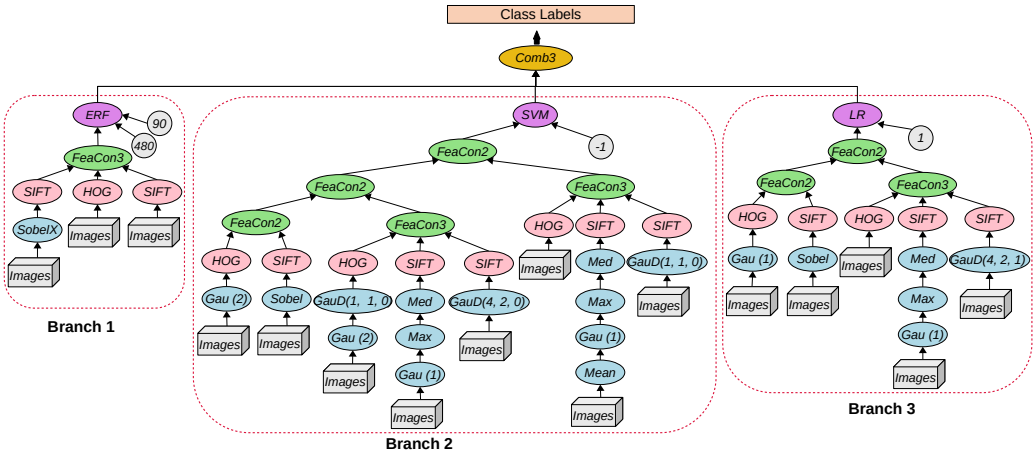


Figure 5.8: An example solution found by IEGP on the **MBR** dataset.

tures and the HOG features. Meanwhile, different filtering functions are employed to process the image before feature extraction. Accordingly, different inputs for the three classifiers further enhance the diversity of the constructed ensemble.

5.5.3 Effectiveness of the Evolved Ensembles

To further analyse the performance of the example ensembles in Figures 5.7 and 5.8, we calculate the accuracy of each classifier in the example ensembles on the test set of MRD and MBR, respectively. Table 5.7 lists the results obtained by the example ensembles (the second row) and the results obtained by each classifier in the branches circled in Figures 5.7 and 5.8 (the third to the fifth rows). In addition, we use the raw pixels as inputs to train three classification algorithms that are used to build the example ensembles in Figures 5.7 and 5.8, and build a new ensemble of the trained classifiers using the plurality voting. The results obtained by this ensemble are listed in the final two rows of Table 5.7.

The ensemble in Figure 5.7 achieves 94.28% accuracy on the test set of MRD and the ensemble in Figure 5.8 achieves 93.21% accuracy on the test

Table 5.7: Classification Accuracy (%) on the Test Sets of MRD and MBR

Method	**MRD**	**MBR**
Ensemble in Figure 5.7 or Figure 5.8	**94.28**	**93.21**
Classifier in branch 1 in Figure 5.7 or Figure 5.8	93.31	88.37
Classifier in branch 2 in Figure 5.7 or Figure 5.8	93.53	92.84
Classifier in branch 3 in Figure 5.7 or Figure 5.8	93.30	91.29
Ensemble of three ERFs using raw pixels	88.51	
Ensemble of ERF, SVM and LR using raw pixels		66.04

set of MBR, which are better than any of the three single classifiers or the ensemble using raw pixels, as listed in Table 5.7. Comparing the example ensembles with the single classifiers, we can find that the combination of these three classifiers using the voting function enhances the classification accuracy. The reason may be that IEGP automatically selects classification functions to build an ensemble during the evolutionary process, which results in a good combination of classifiers. This indicates that IEGP can find a good ensemble to achieve better generalisation performance than a single classifier. Compared the example ensembles in Figure 5.7 and Figure 5.8 with the new ensembles listed in the final two rows of Table 5.7, it is obvious that feature extraction is necessary and important for improving the classification performance. The ensembles of classifiers built using raw pixels only achieve 88.51% on MRD and 66.04% on MBR, which are much lower than those of the ensembles found by IEGP. This indicates that the features learned by IEGP are more discriminative than raw pixels and can further boost the performance of the ensembles. This shows that one of the objectives that uses the proposed IEGP approach to learn effective features has been successfully achieved.

To conclude, further analysis shows that the ensembles found by IEGP have high diversity by using different features to build classifiers to form ensembles. The analysis reveals that IEGP can find ensembles using the

classifiers trained from the same or different classification algorithms. The analysis also shows that IEGP finds good ensembles of classifiers to achieve higher generalisation performance than a single classifier. In addition, the features learned by IEGP are more discriminative than raw pixels for classifying images.

5.6 Chapter Summary

In this chapter, the IEGP approach was proposed to automatically learn effective features and evolve ensembles for image classification. A new individual representation, a new function set, and a new terminal set were developed to allow IEGP to learn informative features and evolve ensembles of diverse classification algorithms. The parameters of the classification algorithms in the evolved ensemble can be automatically set during the evolutionary process. The diversity issue of ensembles can be automatically addressed by IEGP with a tree-based representation. In addition, the evolved ensembles have variable lengths or sizes, which is suitable for dealing with different types of image classification tasks.

This chapter showed the effectiveness of the IEGP approach by testing it on 12 image classification datasets of varying difficulty and comparing it with a large number of effective methods. The comparisons showed that IEGP achieved better performance than the six methods using raw pixels, the four methods using well-known features, two CNN-based methods, and the two GP methods (EGP and FGP). On the datasets with a large number of training and test instances, i.e., datasets 6-12, the IEGP approach achieved better performance than all the benchmark methods on one dataset, ranked second on three datasets, third on one dataset, and fourth on the remaining one dataset. Compared with EvoCNN, the IEGP approach achieved better or the same results on three datasets and slightly worse results on the other four datasets. EvoCNN as a state-of-the-art deep learning method achieved the best results on some datasets, but it

required a large number of computational resources (must be trained on GPU) and training instances. The comparisons further demonstrated the effectiveness of IEGP for image classification. Compared with the previous EGP method, IEGP achieved significantly better results on most benchmark datasets. The comparisons between EGP and IEGP in the distributions of the training and test results showed that IEGP achieved better and more stable results than EGP. Compared with the FGP method proposed in Chapter 3, IEGP achieved better performance in most comparisons, indicating that the performance can be further improved by using ensembles for image classification.

Different from the previous two chapters, this chapter developed a new GP-based approach to automatically learn features and evolve ensembles for image classification. Ensemble classifiers can improve the accuracy over the single classifier on image classification. Although these approaches have achieved promising results on different types of image classification tasks, they may be computationally expensive, especially when the number of training instances is large. Therefore, the next chapter will develop a new GP-based approach with a new multi-population algorithm framework to further improve both the classification performance and the computational efficiency by creating an ensemble for image classification.

Chapter 6

Multi-Population GP with Knowledge Transfer and Ensembles for Fast Feature Learning

6.1 Introduction

The previous chapters developed new GP-based approaches that are able to improve the generalisation performance on different types of image classification tasks. Although existing GP-based feature learning approaches and the proposed approaches have achieved promising results on many image classification tasks, they are often computationally expensive due to a large number of expensive fitness evaluations. The evaluation of the features learned by GP is often based on the training set in the fitness evaluation process. It is known that many image classification datasets have a large number of training instances, e.g., MNIST [136] has 60,000 training images. The high computational cost is challenging for applying GP-based feature learning algorithms to such large datasets. Therefore, it is neces-

sary to develop a new GP-based approach that is computationally cheap for feature learning in image classification.

GP is a population-based approach that uses a number of individuals to search for the best solution. The large population can be split into several small populations and each small population can learn features from a small subset of the large training set. Thus, evaluating the individuals on a small subset of the training set is computationally cheaper than on the original training set. Therefore, this chapter proposes a multi-population GP-based approach to fast feature learning for image classification. With multiple populations, the knowledge learned by one population can be extracted and used to help the learning of the other population(s). An effective knowledge transfer method can be proposed to achieve this. However, learning from a small subset of the original training set, the GP-based approach may find solutions with poor generalisation performance. Therefore, to improve the generalisation performance, it is also necessary to investigate a new fitness function and a new ensemble formulation method for image classification.

6.1.1 Chapter Goals

The overall goal of this chapter is to develop a new approach to improving both the generalisation performance and computational efficiency of GP-based feature learning algorithms for image classification. To achieve this goal, a new feature learning approach with a new multi-population algorithm framework, a new fitness function and a new combination strategy for ensemble formulation is proposed. To reduce the computational cost, a new multi-population algorithm framework that uses multiple small populations to learn features from small subsets of the original training data is developed. During the evolutionary process, knowledge transfer is employed to improve the learning performance of the small populations. To improve the generalisation performance, a new fitness function and a new

combination strategy for ensemble formulation are proposed to create an effective ensemble for classification. The new approach uses the same program structure, the function set and the terminal set as the FGP approach (proposed in Chapter 4) for feature learning. The new approach is termed as fastFGP in short. Specifically, this chapter will investigate

- how a multi-population algorithm framework is developed in GP-based feature learning algorithms to improve computational efficiency;
- how the knowledge across different small populations is extracted and transferred to improve the learning performance;
- how an effective fitness function is proposed to better evaluate the individuals on small subsets of the training data;
- how an effective ensemble with weighted classifiers is created for image classification;
- whether the proposed fastFGP approach can achieve better classification performance and reduce the computational cost than the baseline GP-based feature learning approach (the FGP approach proposed in Chapter 4); and
- whether the fastFGP approach can achieve better classification performance than the other benchmark methods.

6.1.2 Chapter Organisation

The remainder of this chapter is organised as follows. Chapter 6.2 presents the proposed approach. Experiments are designed in Chapter 6.3. Chapter 6.4 discusses and analyses the experimental results. Further analysis of the proposed approach is presented in Chapter 6.5. Chapter 6.6 concludes this chapter.

6.2 The Proposed Approach

This section introduces the proposed approach, fastFGP, in detail. First, the overall algorithm is outlined. Second, it introduces the knowledge transfer and a new mutation operator. Then it describes the new log-loss-based fitness function and the fitness evaluation process. Finally, it presents the strategy that combines final solutions to create an ensemble for classifying images.

6.2.1 Algorithm Framework

The overall feature learning process is generally computationally expensive due to a large number of expensive fitness evaluations. Using a small number of training instances or a small population in GP can reduce the computational cost. However, using a small number of training instances for feature learning may cause the model/solution not to generalise on unseen data due to the high variations among images. In addition, a small population might not be sufficient for GP to find good solutions. Based on these considerations, a new feature learning framework is developed in GP-based feature learning algorithms to reduce the computational cost and to improve the generalisation performance of the solutions.

The overall framework of the proposed fastFGP approach is shown in Figure 6.1. In the new framework, the full training set $\mathcal{D}_{train}$ is split into N non-overlapping small subsets with preserving the class ratio to form subsets $\mathcal{D}_1, \ldots, \mathcal{D}_N$. Then $N + 1$ small populations, i.e., $P_0, P_1, \ldots, P_N$, are randomly initialised using a tree/program generation method based on the program structure, the function set, and the terminal set of FGP. The first population P_0 uses the full training data $\mathcal{D}_{train}$ for feature learning, which aims at finding the best solution on the full training data with high generalisation performance. The remaining N populations $P_1, \ldots, P_N$ use $\mathcal{D}_1, \ldots, \mathcal{D}_N$ for feature learning, respectively, which aim to find the best solutions on the non-overlapping small subsets of $\mathcal{D}_{train}$. Each population

solves one problem on the training set or a small subset. The best solutions for these problems, $\mathcal{D}_{train}, \mathcal{D}_1, \ldots, \mathcal{D}_N$, can be the same or different because these datasets have different instances. In this framework, each population in the new framework uses a small population size. The population size for each population is set as $[PS/(N+1)]$ (use the integer part if PS is indivisible by $N+1$), where PS is the commonly used population size for a GP method. Note that compared with other evolutionary algorithms, e.g., GAs and PSO, GP often has a larger population size, e.g., 500 used in FGP. Therefore $PS/(N+1)$ is 100 when $N = 4$ and $PS = 500$.

At each generation, P_0 is evaluated on the full training data $\mathcal{D}_{train}$ and P_i $(1 \leq i \leq N)$ is evaluated on the small subset $\mathcal{D}_i$ $(1 \leq i \leq N)$. During the evolutionary learning process, useful knowledge is extracted from and transferred between the $N+1$ populations to improve the learning performance. The extracted knowledge is stored in two archives and the knowledge transfer is achieved by using a new mutation operator. At each generation, the crossover operator and the new mutation operator are employed to create a new population for the next generation. Note that reproduction or elitism is not employed in the proposed approach since the best trees can be stored in two archives. When the termination condition is satisfied, the best solution of each population is returned. Based on the $N+1$ best solutions, a combination strategy is proposed to combine these solutions to create an ensemble for solving image classification, which aims to achieve high generalisation performance on the unseen dataset.

Computational Complexity Analysis and Comparisons

We compare the computational complexity of fastFGP and FGP to analyse whether the computational complexity is reduced in fastFGP. In GP-based feature learning algorithms, the main computational cost is from fitness evaluation. For simplification, we compare the computational complexity of FGP and fastFGP on population evaluation at one generation. At each generation, FGP evaluates PS individuals on $\mathcal{D}_{train}$, where the number of

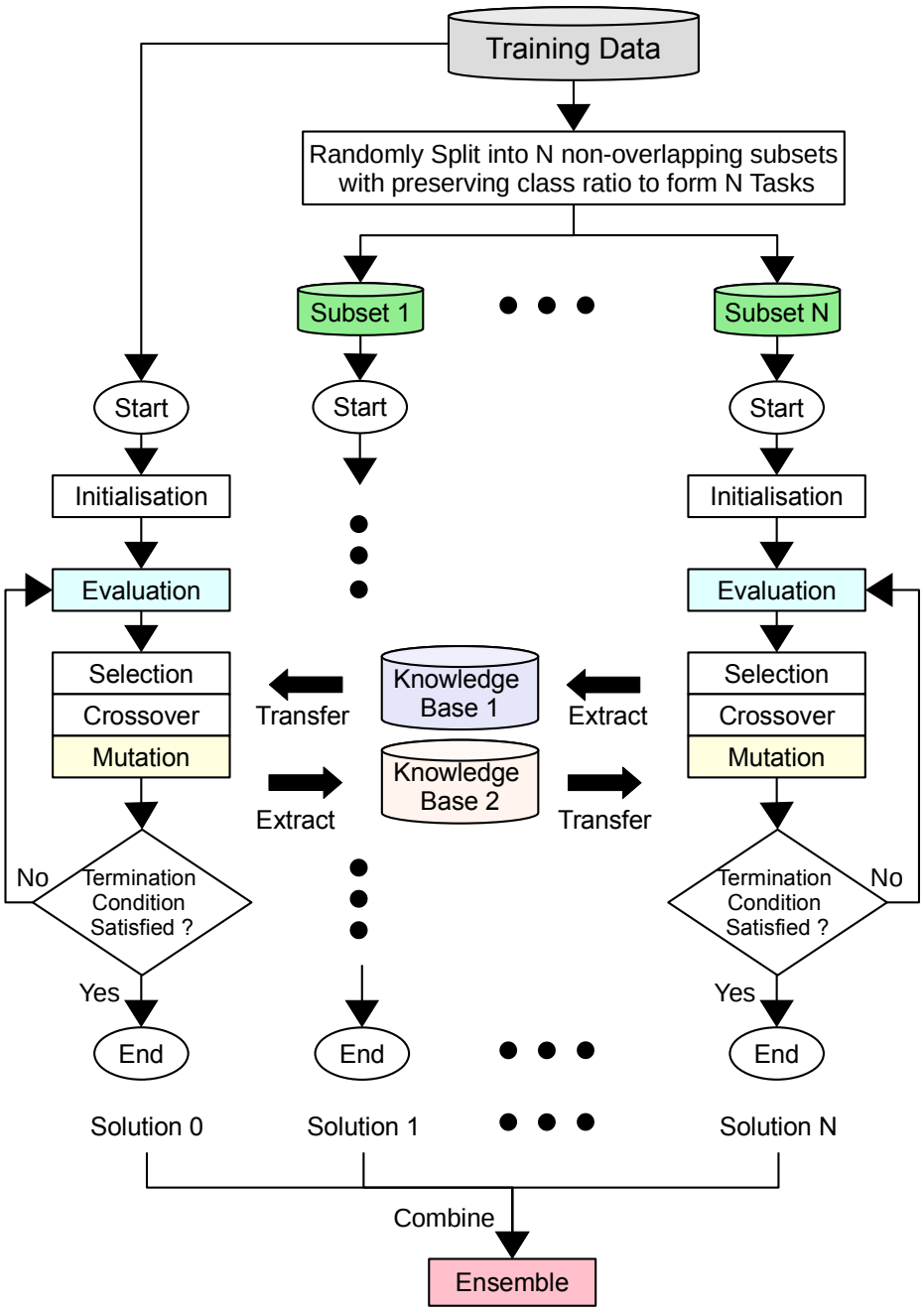


Figure 6.1: Overall framework of the proposed fastFGP approach.

instances is n. The fastFGP approach evaluates $[PS/(N+1)]$ individuals on $\mathcal{D}_{train}$ and $[PS/(N+1)]$ individuals on $\mathcal{D}_1, \ldots, \mathcal{D}_N$, respectively. In $\mathcal{D}_1, \ldots, \mathcal{D}_N$, the number of instances is $[n/N]$ (use the integer part if n is indivisible by N).

The FGP method uses a linear SVM for classification in the fitness evaluation process. The approximated complexity of SVM is $O(n^2 f)$ [54, 172], where n represents the number of training instances and f represent the number of features. The fastFGP method uses LR for classification in the fitness evaluation process and the complexity of LR is $O(nf^2)$. When $f \leq n$, the complexity of LR is smaller than the complexity of SVM. To analyse the effects of the N on the computational complexity, we assume that both FGP and fastFGP use a Linear SVM for fitness evaluation to achieve easy comparisons. We assume that the complexity of transforming n images into features using a GP individual is $O(n)$ and f is the same for all the individuals. Then the complexity of FGP at one generation is

$$O_{FGP} = O(PS \times (n^2 f + n)), \tag{6.1}$$

The complexity of fastFGP at one generation $O_{fastFGP}$ is

$$O_{fastFGP} = O(\frac{PS}{N+1}(n^2 f + n) + \frac{N \times PS}{N+1}((\frac{n}{N})^2 f + \frac{n}{N})). \tag{6.2}$$

Based on equations (6.1) and (6.2), we can have $O_{fastFGP} < O_{FGP}$ on the condition that $N > \frac{1}{2(nf+1)} + \sqrt{1 - \frac{4nf+3}{(2(nf+1))^2}}$. As $n > 1$, $f > 1$, and $2^2(nf+1))^2 > 4nf + 3$, it can be induced that when $N > 1$, we can have $O_{fastFGP} < O_{FGP}$.

In the fastfGP approach, we set $N = 4$, leading to the creation of five small populations. When $N = 4$, the complexity of fastFGP is $O_{fastFGP} = O(\frac{1}{5}PS(n^2 f+n)) + \frac{4}{5}PS((\frac{n}{4})^2 f + \frac{n}{4})$. After simplification, we have $O_{fastFGP} = O(PS(\frac{n^2 f+n}{4}))$, which is three times less than the complexity of FGP, i.e., $O(PS(n^2 f + n))$.

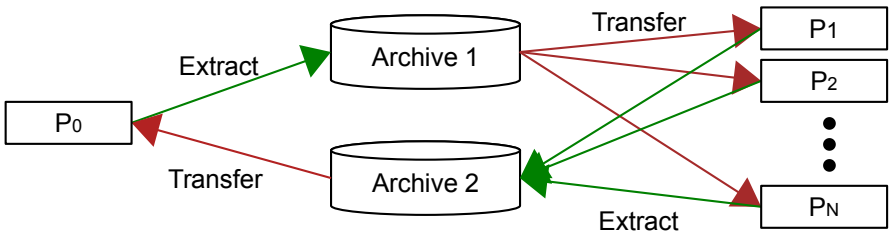


Figure 6.2: The knowledge transfer route in fastFGP crossing different small populations.

6.2.2 Knowledge Transfer and A New Mutation Operator

The fastFGP approach has $N+1$ small populations, which can learn different knowledge from various data $\mathcal{D}_{train}, \mathcal{D}_1, \ldots, \mathcal{D}_N$. Each dataset can be considered as a single task with different instances. Knowledge transfer crossing these tasks can help improve the learning performance on another task. But the key questions are what to transfer, how to transfer and when to transfer [170]. In the fastFGP approach, a new method is proposed to achieve knowledge transfer between the $N+1$ small populations at each generation, which is different from the current work on knowledge transfer in GP.

In FGP, the tree or subtrees represents the interrelationship between the input variables/images and the operators, which indicates the knowledge (building blocks) discovered from the task by learning. The knowledge can be extracted to reuse [114]. The fastFGP approach has multiple populations, which indicates that there are several possible sources to extract trees and reuse them. To achieve fast and effective knowledge transfer, the relationship between the datasets can be used to provide intuitive hints towards how to extract and use the knowledge across different populations. Based on the relationships $\mathcal{D}_i \subset \mathcal{D}_{train}$ $(1 \leq i \leq N)$ and $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$ $(1 \leq i \leq N; 1 \leq j \leq N; i \neq j)$, a simple knowledge transfer is defined in fastFGP, which is shown in Figure 6.2.

In fastFGP, two archives are built to store the extracted trees (including

subtrees) from different populations for transferring. These extracted trees are stored in two archives, i.e., $Archive1$ and $Archive2$. $Archive1$ is used to store $[PS/(N+1)]$ best trees extracted from P_0 and $Archive2$ is used to store $[PS/(N+1)]$ best trees extracted from $P_1, \ldots, P_N$. Specifically, $[PS/(N*(N+1))]$ best trees are extracted from $P_i (1 \leq i \leq N)$ to form $Archive2$. Note that we limit the storage of $Archive1$ and $Archive2$ to $PS/(N+1)$ trees in order to save memory. The trees in $Archive1$ are the best trees of P_0 of all the past generations. The trees in $Archive2$ are the best trees of $P_1, \ldots, P_N$ of all the past generations. The $Archive1$ and $Archive2$ are updated at each generation after fitness evaluations to store the best trees so far.

Algorithm 5: Knowledge-based Mutation Operator

Input : p: an individual for mutation; $Archive$: an archive storing the extracted trees for knowledge transfer ($Archive1$ or $Archive2$ depending on p from which population).

Output : A new offspring o after mutation.

1 $p_{kb} \leftarrow$ Select an individual from $Archive$ using tournament selection;

2 $Node\_types \leftarrow$ Find all the common types of the nodes in individuals p_{kb} and p; `// fastFGP is based on strongly typed GP`

3 $sub\_tree \leftarrow$ Randomly select a type from $Node\_types$ and select a subtree from p_{kb} based on the type;

4 $o \leftarrow$ Select the node with the same type in p and replace its branch with sub_tree.

The knowledge transfer happens in the mutation operation at each generation. Since the two archives, $Archive1$ and $Archive2$, can store best trees, only crossover and mutation operators are employed in fastFGP to generate new populations during the evolutionary process and elitism is not used/needed. Standard mutation operator randomly generates a new branch to replace the branch from a randomly selected node of the tree, which can add new genetic materials into the population. Instead of us-

ing randomly generated subtrees in mutation, a new mutation operator is proposed to use the branch of trees selected from the archives. The new mutation operator is called knowledge-based mutation operator, which is described in Algorithm 5. The new operator starts with selecting one best tree/individual p_{kb} from the archives using Tournament selection. Then the common types ($Node\_types$) of all the nodes in p_{kb} and p (the individual will be mutated) are found. The fastFGP approach is based on strongly typed GP and the type constraint needs to be satisfied when performing mutation. One type is randomly selected from $Node\_types$ and a subtree (sub_tree) is selected from p_{kb} based on the selected type. Then mutation is performed on p by randomly selecting a node with the same type and replacing the branch of p using the selected sub_tree. The new individual/tree has a subtree (sub_tree) from the archive, which indicates that the knowledge is successfully transferred from the archives to the current population.

6.2.3 Fitness Evaluation

In many GP-based feature learning algorithms [197], SVM is often used for classifying images. SVM builds multiple strong binary classifiers for a classification problem and cannot directly provide probabilities of prediction. Compared with SVM, LR, also known as softmax regression, can predict confidence scores/probabilities of different classes for each instance, which may be better for building an ensemble. The fastFGP approach uses multinomial LR for image classification. In multinomial LR, the probability of an instance x to belong to one of the $C-1$ classes is $P\{y=c|x\}$ and the probability of x to belong to class C is $P\{y=C|x\}$, which are formulated in Equation (6.3) and Equation (6.4), respectively.

$$P\{y=c|x\} = \frac{e^{w_c^T \cdot x+\beta_c}}{1+\sum_{c=1}^{C-1} e^{w_c^T \cdot x+\beta_c}}, c=1,2,3,\cdots,C-1, \tag{6.3}$$

$$P\{y = C|x\} = \frac{1}{1 + \sum_{c=1}^{C-1} e^{w_c^T \cdot x + \beta_c}}, \tag{6.4}$$

where w_c represents a $S \times 1$ vector of weights and β_c represents the bias of the linear model: $w_c^T \cdot x + \beta_c$. w_c^T represents the vector transpose. The weights $\{w_1, w_2, \ldots, w_{C-1}\}$ and biases $\{\beta_1, \beta_2, \ldots, \beta_{C-1}\}$ can be estimated from training data using maximum likelihood estimation.

The outputs of a LR classifier for an instance x are probabilities of different classes, such as $P\{y = 0|x\} = 0.1$ and $P\{y = 1|x\} = 0.9$. The commonly used log-loss (also known as cross-entropy loss) function is more suitable for evaluation than the accuracy metric. The accuracy metric only considers the total number of correctly classified instances, which, for example, consider whether $P\{y = 0|x\}$ is larger or smaller than $P\{y = 1|x\}$. However, the log-loss function can provide more information by quantifying the penalty if the predicted probability for the correct class is not one. In general, the smaller the loss, the better the classification performance. A zero log-loss indicates that all the instances have predicted probabilities of one in the correct class and zero in the other classes. In this chapter, we formulate the fitness function based on the log-loss function. Since log-loss is a minimisation problem, we transform it as a maximize problem by using negative log-loss. As in FGP, stratified k-fold cross-validation is employed to improve the generalisation performance of the solutions. Therefore, the fitness function for the fastFGP approach can be formulated as Equation (6.5).

$$\begin{aligned} \max F &= \max \frac{1}{K} \sum_{i=1}^{K} L_i \\ &= \max \frac{1}{K} \sum_{i=1}^{K} (\frac{1}{M_i} \sum_{m=1}^{M_i} \sum_{c=1}^{C} y_{m,c} \log p_{m,c}). \end{aligned} \tag{6.5}$$

where L_i represents the negative log-loss on the ith fold. $y_{m,c} \in \{0, 1\}$ represents the true label of instance m on class c. $p_{m,c} \in [0, 1]$ represents the predicted probability of instance m on class c and $\sum_{c=1}^{C} p_{m,c} = 1$. C rep-

resents the total number of classes of the dataset.K represents the number of K in K-fold cross-validation and M_i represents the total number of instances in fold i. The value of F is in the range $(-\infty, 0]$.

FGP uses a hash table to store a number of individuals to avoid evaluating the same individuals again. The fitness evaluation process of fastFGP also employs this strategy for fast evaluation. Since there are $N+1$ populations in fastFGP, $N+1$ hash tables (CT_n $(0 \leq n \leq N)$) are built to store individuals and their fitness values in the past generations, respectively. Based on the hash table and the new fitness function, the overall fitness evaluation process is shown in Algorithm 6.

Algorithm 6: Fitness Evaluation

Input : CT_n: the hash table of population n;
$\mathcal{D}_n = \{(x_1, y_1), \ldots, (x_m, y_m)\}$: the training data for population n ($\mathcal{D}_0$ is $\mathcal{D}_{train}$), p: the evaluated individual.
Output : $f(p)$.

```
if p in CT_n then
    f(p) ← the fitness value of CT_n(p);
else
    Use p to transform {x_1, ..., x_m} into features {f_1, ..., f_m};
    Standardise {f_1, ..., f_m} according to the mean and standard
     deviation values;
    Split {f_1, ..., f_m} and {y_1, ..., y_m} into K = 5 folds preserving
     the class ratio;
    for i = 1 to K do
        Fit four folds of data except for the ith fold into logistic
         regression to train a classifier;
        Test the classifier on the ith fold of data;
        L_i ← Calculate the values of log-loss on the ith fold
         according to Equation (6.5);
    end
    f(p) ← Calculate the average negative log-loss values;
end
Return f(p).
```

6.2.4 Ensemble Formulation for Classification

After the evolutionary process, fastFGP returns the best individual from each population. Therefore, $N + 1$ best individuals, i.e., $I_0, \ldots, I_N$, are returned. To achieve good generalisation performance on test data, an ensemble is created based on these individuals and their weights for classification. The outline of ensemble formulation is shown in Figure 6.3.

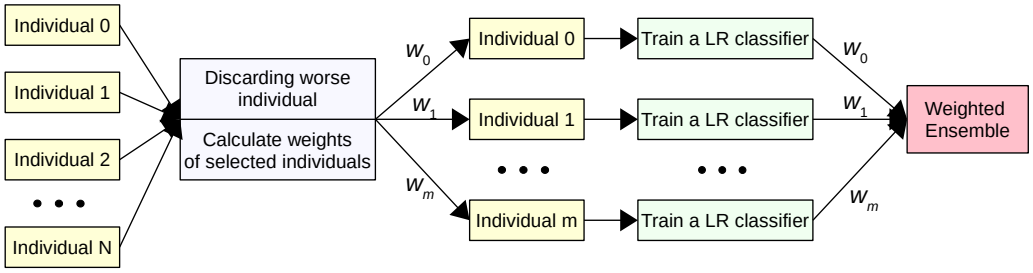


Figure 6.3: Outline of ensemble formulation.

First, the weights of individuals are calculated and some bad individuals are discarded according to the weights. The calculation of the weight is based on the performance (fitness value) of each individual on the full training data $\mathcal{D}_{train}$. The bad individuals are discarded in order to make the individual classifiers be not only diverse but also accurate. In this method, we discard individuals by comparing their fitness values (f) with the fitness value (f_0) of the best individual of the first population P_0. This is because the range of the fitness values (the negative log-loss values) varies with the datasets and f_0 is often higher than f. The discarding criterion will be $f < a * f_0$. In this approach, $a > 1$ and we set a as 2 to balance discarding and selecting individuals.

The main steps of ensemble formulation are summarised as follows:

Step 1: Evaluate each individual I_i $(1 \leq i \leq N)$ on $\mathcal{D}_{train}$ using the fitness function to obtain fitness values f_i. Note that I_0 has its fitness value f_0 on $\mathcal{D}_{train}$ so that it is not evaluated again;

Step 2: Discard bad individuals if $f_i < 2f_0$ $(1 \leq i \leq N)$ to obtain $m+1$ individuals with their fitness values $\{f_0, \ldots, f_m\}$ $(0 \leq m \leq N)$;

Step 3: To obtain the weights of each classifier. First, the negative (log-loss) values of f_i $(0 \leq i \leq m)$ are transformed to positive using $\sum_{i=0}^{m} f_i/f_i$ to obtain $\{pf_i\}_{i=1}^{m}$ (note that the rank of $\{f_i\}_{i=0}^{m}$ does not change). Second, pf_i is scaled into $[0, 1]$ using $pf_i / \sum_{i=0}^{m} pf_i$ to obtain $\{w_i\}_{i=0}^{m}$, where $\sum_{i=0}^{m} w_i = 1$;

According to the $m+1$ selected individuals and the weights, we can build $m+1$ classifiers using the corresponding individuals and create an ensemble based on the classifiers and the weights. Each classifier is built from an individual I_i $(0 \leq i \leq m)$. In this process, the individual I_i is used to transform images in $\mathcal{D}_{train}$ to features, i.e., $\mathcal{D}^i_{train}$. The features in $\mathcal{D}^i_{train}$ are standardised. Then $\mathcal{D}^i_{train}$ is fed into LR to train the classifier i. Repeating this process, the $m+1$ trained classifiers using $\mathcal{D}_{train}$ are obtained. The weight for classifier i is w_i $(w_i \in \{w_i\}_{i=0}^{m})$.

In the test process, the $m+1$ trained classifiers are combined to obtain an ensemble for classifying images in test set $\mathcal{D}_{test}$ using weighted voting [247]. In the test process, the test set $\mathcal{D}_{test}$ is transformed into features $\mathcal{D}^i_{test}$ using the corresponding individual I_i. The $\mathcal{D}^i_{test}$ is standardised according to $\mathcal{D}^i_{train}$. Then an instance x in $\mathcal{D}^i_{test}$ can be feed into a classifier i to obtain the predicted probabilities for x, i.e., $\{P^i(y=0|x), \ldots, P^i(y=C|x)\}$. Repeating this process, $m+1$ sets of predicted probabilities for instance x are obtained from the $m+1$ classifiers.

Based on the weights $\{w_i\}_{i=0}^{m}$ and the probabilities from each classifier, the weighted probabilities for instance x are calculated using Equation (6.6).

$$P(y=c|x) = \sum_{i=0}^{m} w_i \times P^i(y=c|x),\ c=0,\ldots,C, \tag{6.6}$$

Based on the weighted probabilities, we can obtain the class label c^* for instance x returned by Equation (6.7).

$$c^* = \underset{P(y=c|x)}{\operatorname{argmax}} \{P(y=0|x), \ldots, P(y=C|x)\}. \quad (6.7)$$

Finally, the class label for each instance in $\mathcal{D}_{test}$ and the overall classification accuracy on the test set can be calculated.

6.3 Experiment Design

A number of experiments have been conducted to show the effectiveness of the proposed approach. This section designs the experiments, including benchmark datasets, benchmark methods, and parameter settings.

6.3.1 Benchmark Datasets

In the experiments, 11 datasets are employed to examine the performance of the proposed approach. These datasets are the same as those employed in Chapters 4 and 5 except for the ORL dataset. The ORL dataset is very small, having 10 images per class. Such a small dataset cannot be used in the proposed framework and there is no need to reduce the computational cost of GP for feature learning from such a small dataset. The employed 11 datasets are FEI_1 [215], FEI_2 [215], KTH [153], FS [79], MB [132], MRD [132], MBR [132], MBI [132], Rectangle [132], RI [132], and Convex [132]. These datasets contain various representative image classification tasks, i.e., facial expression classification (FEI_1 and FEI_2), texture classification (KTH), scene classification (FS), digit recognition (MB, MRD, MBR, and MBI), and object classification (Rectangle, RI and Convex). The detailed information about these datasets are listed in Table 4.5.

6.3.2 Benchmark Methods

The majority of the benchmark methods are the same as those employed in Chapters 4 and 5. Besides, the FGP method proposed in Chapter 4

and the IEGP method proposed in Chapter 5 are employed for comparisons to show the effectiveness of the proposed fastFGP approach. Therefore, there are 14 benchmark methods on datasets 1-4, i.e., six classification algorithms using raw pixels (i.e., SVM, KNN, LR, AdaBoost, RF, and ERF), four SVMs using different pre-extracted features (i.e., uLBP+SVM, LBP+SVM, HOG+SVM, and SIFT+SVM), two CNNs with different architectures (i.e., CNN-5 and CNN-8), and two GP-based methods (i.e., EGP [38] and FGP). On datasets 5-11, there are 21 benchmark methods for comparisons, i.e., SVM+RBF [132], SVM+Poly [132], SAE-3 [183], DAE-b-3 [183], CAE-2 [183], SPAE [236], RBM-3 [183], ScatNet-2 [49, 53], RandNet-2 [53], PCANet-2 (softmax) [53], LDANet-2 [53], NNet [132], SAA-3 [132], DBN-3 [132], FCCNN [177], FCCNN (with BT) [177], SPCN [140], EvoCNN [208], EGP [38], IEGP, and FGP.

6.3.3 Parameter Settings

The parameter settings for fastFGP are based on the commonly used settings for GP [114]. In the FGP method, the population size is 500. To achieve fair comparisons, the total population size in fastFGP is 500. Based on preliminary experiments, N is set as 4 as it achieves a good balance between efficiency and effectiveness. With $N = 4$, fastFGP has five small populations with a size of 100, respectively. The crossover rate is 0.8 and the mutation rate is 0.2. The best individuals can be stored into the archives so that fastFGP does not need elitism. The selection method is tournament selection with size 7. The tree generation method is ramped half-and-half. The tree depth at the initialisation step is 2-6 and the maximum tree depth is 8. Note that fastFGP may evolve trees with depth larger than 8 because the type constraints are more important than the depth constraint in STGP. In the LR classification algorithm employed for fitness evaluation, stochastic average gradient (SAG) is used for optimising weights and bias as it is fast on large dataset [192]. The number of iterations in LR is set to 50 dur-

ing the evolutionary learning process to save computational cost and set as 100 in the test process. The other parameters for LR are based on the default settings of the algorithm implemented in *scikit-learn* [172].

The parameter settings for the benchmark methods, i.e., SVM, KNN, LR, RF, AdaBoost and ERF, refer to [16, 235, 249], where the number of nearest neighbours is 1 in KNN, the number of trees is 500 and the maximum tree depth is 100 in RF and ERF. These algorithms are implemented based on the *scikit-learn* [172] package. The other parameters are the default ones in this package for simplification. The CNN-5 and CNN-8 methods are implemented on *Keras* [57]. ReLU is used as the activation function and softmax is used for classification at the final layer. Dropout is added after the pooling layer and the first fully connected layer with 0.25 and 0.5 probabilities, respectively, to avoid overfitting [201]. Note that CNN-5 and CNN-8 are just examples of CNNs to show whether the proposed approach can achieve better results than simple CNNs on datasets 1-4. The performance of CNNs on datasets 1-4 might be further improved by using deep CNNs with transfer learning or manually tuning on architectures, which are out of the scope of this chapter.

The fastFGP approach is implemented on the DEAP (*Distributed Evolutionary Algorithm in Python*) [86] package. To further speed up the fastFGP approach, the SCOOP package [105] is employed to use multiple cores for fitness evaluation. In the fitness evaluation, each individual is evaluated on one core and multiple individuals are evaluated at the same time according to the number of cores employed. In the experiments, four cores are used to run fastFGP. In general, the training time can be sped up by using more cores. On each dataset, fastFGP has been executed 30 independent runs according to the convention of the EC communities. The benchmark methods on datasets 1-4 have also been executed 30 independent runs for comparisons. The best and average results of the 30 runs on each dataset are reported. Note that all the reported results are on the test sets, which are unseen to the training and evolutionary processes.

6.4 Results and Discussions

This section presents the experimental results obtained by the proposed fastFGP approach and the benchmark methods on the 11 benchmark datasets. This section compares the fastFGP approach with the benchmarks methods in terms of classification accuracy and training time on these benchmark datasets.

6.4.1 Classification Results

Comparison with FGP

The classification results, i.e., maximum accuracy (Max), mean accuracy and standard deviation (Mean±St.dev), of fastFGP and FGP are listed in Table 6.1. The Wilcoxon rank-sum test with a 95% significance interval is used to compare fastFGP with FGP. In Table 6.1, the "+" or "−" symbols indicate that fastFGP achieves significantly better or worse results than FGP. The "=" symbol indicates that fastFGP achieves similar results to FGP.

Table 6.1 shows that fastFGP achieves significantly better results than FGP on five datasets and similar results on the remaining six datasets. The proposed fastFGP approach obtains higher maximum accuracy than FGP on ten datasets except for the MRD dataset. It achieves better mean accuracy than FGP on ten datasets except for the MBI dataset. More importantly, fastFGP improves the mean accuracy by 6.93% on FS, which is a challenging task of understanding natural scene images. The classification results show that fastFGP is more effective than FGP on the 11 different image classification tasks. Compared with FGP, the fastFGP approach uses the same individual representation, function set, terminal set, and parameter settings. The results show that the new components, i.e., the new algorithm framework, the new knowledge transfer method, the new fitness function, and the constructed ensemble, in fastFGP are effective for improving the performance of the baseline method (FGP) in image classi-

Table 6.1: Classification Accuracy (%) of fastFGP and FGP on the 11 Datasets

	FGP		**fastFGP**	
Dataset	Max	Mean±St.dev	Max	Mean±St.dev
FEI_1	98.00	94.47±2.67=	98.00	**94.80±2.61**
FEI_2	96.00	91.33±3.36=	**98.00**	**91.40±3.02**
KTH	98.79	96.07±1.13+	**99.09**	**97.84±0.78**
FS	74.48	71.59±1.74+	**80.23**	**78.52±1.09**
MB	98.82	98.70±0.06+	**98.95**	**98.85±0.05**
MRD	92.63	91.56±0.61+	92.58	**91.99±0.40**
MBR	93.46	92.66±0.43=	**93.73**	**92.79±0.41**
MBI	92.52	89.65±1.44=	**93.16**	89.49±0.92
Rectangle	100.0	99.88±0.11+	100.0	**99.99±0.02**
RI	93.90	92.66±0.62=	**95.11**	**93.05±1.03**
Convex	98.46	98.16±0.19=	**98.54**	**98.22±0.14**
Total		5+, 6=		

fication. With these new designs, the fastFGP approach can find the best solutions from each small population and construct an effective ensemble of diverse and effective classifiers using these solutions to achieve higher generalisation on different image classification datasets. The results indicate that the goal of improving the generalisation performance has been successfully achieved.

Comparison with Other Benchmark Methods on Datasets 1-4

The classification results of fastFGP and 13 benchmark methods on datasets 1-4 are listed in Tables 6.2 and 6.3. The "+", "−" and "=" symbols indicate that fastFGP is significantly better, worse or similar than/to the compared method.

Tables 6.2 and 6.3 show that the proposed fastFGP approach obtains significantly better results than most benchmark methods on the FEI_1, FEI_2, KTH, and FS datasets. Specifically, fastFGP is significantly better

than eight methods and similar to four methods out of the 13 benchmark methods on the FEI_1 dataset. On the FEI_2 dataset, fastFGP is significantly better than nine methods and similar to two methods. On the KTH and FS datasets, fastFGP performs significantly better than any of the benchmark methods. The FEI_1 and FEI_2 datasets are facial expression classification tasks, which are relatively easy so that many benchmark methods achieve good results, i.e., over 90% accuracy. The KTH and FS datasets are texture classification and scene classification tasks, which are more difficult than the FEI_1 and FEI_2 datasets. The benchmark methods achieve very low accuracy on these two datasets, i.e., the mean accuracy is less than 83% on the KTH dataset and less than 62% on the FS dataset, while fastFGP achieves 97.84% mean accuracy on the KTH dataset and 78.52% mean accuracy on the FS dataset. The comparisons show that fastFGP is effective for solving different types of image classification tasks of varying difficulty.

Comparison with Other Benchmark Methods on Datasets 5-11

The classification accuracy of the proposed fastFGP approach and 21 benchmark methods on datasets 5-11 are listed in Table 6.4. The datasets 5-11 have public training and test sets so that the accuracy of these benchmark methods on the test set is collected from the corresponding papers. On these datasets, we only compare the best results of these methods because most benchmark methods only reported the best results. In Table 6.4, the symbol "↑" indicates that the best accuracy obtained by fastFGP is better than that of the compared method.

From Table 6.4, we can find that the proposed fastFGP approach achieves better results than most of the effective methods on the seven datasets. Note that these datasets have been extensively exploited by these methods so that any improvement in accuracy is difficult. In addition, most benchmark methods are (deep) neural network-based methods, which are known as powerful methods for feature learning and image classifica-

Table 6.2: Classification Accuracy (%) of the FEI_1 and FEI_2 Datasets

	FEI_1		FEI_2	
Method	Max	Mean±St.dev	Max	Mean±St.dev
SVM	90.00	90.00±0.00+	88.00	88.00±0.00+
KNN	32.00	32.00±0.00+	8.00	8.00±0.00+
LR	92.00	92.00±0.00+	88.00	88.00±0.00+
RF	98.00	**97.07±1.01**–	90.00	89.20±1.13+
AdaBoost	80.00	78.67±1.32+	80.00	76.00±3.44+
ERF	94.00	93.27±0.98+	92.00	90.60±0.93=
uLBP+SVM	66.00	56.73±3.66+	68.00	62.53±3.52+
LBP+SVM	68.00	64.60±1.83+	74.00	69.80±0.00+
HOG+SVM	96.00	96.00±0.00=	82.00	82.00±0.00+
SIFT+SVM	56.00	56.00±0.00+	62.00	62.00±0.00+
CNN-5	98.00	95.40±1.30=	98.00	95.27±1.62–
CNN-8	98.00	95.33±1.32=	96.00	90.93±1.87=
EGP [38]	**100.0**	96.20±2.06=	**100.0**	**98.07±1.70**–
fastFGP	98.00	94.80±2.61	98.00	91.40±3.02
Overall		8+, 4=, 1–		9+, 2=, 2–

tion. Compared with these benchmark methods, the fastFGP approach, as a GP method, ranks first among all the methods on four datasets, i.e., MB, Rectangle, RI, and Convex, ranks second on two datasets, i.e., the MBR and MBI datasets, and ranks fourth on the remaining one dataset, i.e., MRD. On the MB dataset, fastFGP and LDANet-2 find the best accuracy of 98.95%. However, LDANet-2 only achieves 87.58% on the MBI dataset, which is a variant of the MB dataset by adding additional image background, and 83.80% on the RI dataset, which is a variant of Rectangle by adding additional background noise. This indicates that the performance of LDANet-2 is significantly affected by these additional noises in the MBI and RI datasets. Compared with LDANet-2, fastFGP is less affected by achieving 93.16% accuracy on the MBI dataset and 95.11% accuracy on the RI dataset. The EvoCNN method, which is a state-of-the-art deep learning method using EC to evolve architectures of CNNs, achieves

Table 6.3: Classification Accuracy (%) of the KTH and FS Datasets

	KTH		FS	
Method	Max	Mean±St.dev	Max	Mean±St.dev
SVM	46.97	44.59±2.83+	20.63	20.30±0.15+
KNN	34.24	34.24±0.00+	24.35	24.35±0.00+
LR	48.79	48.79±0.00+	23.49	23.49±0.00+
RF	60.00	57.81±0.83+	37.36	36.53±0.49+
AdaBoost	37.88	33.44±1.37+	17.47	13.04±1.47+
ERF	61.52	59.83±0.86+	37.94	37.15±0.36+
uLBP+SVM	78.79	73.29±4.18+	49.79	33.27±8.90+
LBP+SVM	83.64	82.71±0.51+	53.50	50.45±1.80+
HOG+SVM	57.27	55.96±0.64+	12.11	7.91±2.47+
SIFT+SVM	65.76	65.76±0.00+	60.92	60.92±0.00+
CNN-5	85.76	82.56±1.87+	50.14	48.03±1.16+
CNN-8	76.36	71.63±3.18+	49.16	46.79±1.01+
EGP [38]	87.88	77.53±5.17+	67.17	61.07±2.91+
fastFGP	**99.09**	**97.84±0.78**	**80.23**	**78.52±1.09**
Overall		13+		13+

the best results on the MRD, MBR and MBI datasets. These three datasets are difficult due to additional variations. EvoCNN requires the Graphics Processing Unit (GPU) implementation and uses extensive computational resources to find the best architectures for CNNs so that it can achieve better classification accuracy. In contrast, fastFGP is implemented on the Central Processing Unit (CPU) and the computational cost is affordable (or even less). Although fastFGP achieves slightly worse results on these three datasets, it is noticeable that it achieves better results than EvoCNN on the remaining four datasets. Compared with EGP, IEGP and FGP, fastFGP achieves better results on six datasets except for the MRD dataset, where IEGP achieves slightly better accuracy. Compared with IEGP, which is an ensemble method that automatically learns features and builds many different classifiers (e.g., SVM, RF and LR) to construct the ensemble, fastFGP only uses the LR classifier to construct the ensemble but achieves

Table 6.4: Classification Accuracy (%) of Datasets 5-11

Method	MB	MRD	MBR	MBI	Rectangle	RI	Convex
SVM+RBF [132]	96.97 (↑)	88.89 (↑)	85.42 (↑)	77.39 (↑)	97.85 (↑)	75.96 (↑)	80.87 (↑)
SVM+Poly [132]	96.31 (↑)	84.58 (↑)	83.38 (↑)	75.99 (↑)	97.85 (↑)	75.95 (↑)	80.18 (↑)
SAE-3 [183]	96.54 (↑)	89.70 (↑)	88.72 (↑)	77.00 (↑)	97.86 (↑)	75.95 (↑)	–
DAE-b-3 [183]	97.16 (↑)	90.47 (↑)	89.70 (↑)	83.32 (↑)	98.01 (↑)	78.41 (↑)	–
CAE-2 [183]	97.52 (↑)	90.34 (↑)	89.10 (↑)	84.50 (↑)	98.79 (↑)	78.46 (↑)	–
SPAE [236]	96.68 (↑)	89.74 (↑)	90.99 (↑)	86.76 (↑)	–	–	–
RBM-3 [183]	96.89 (↑)	89.70 (↑)	93.27 (↑)	83.69 (↑)	97.40 (↑)	77.50 (↑)	–
ScatNet-2 [49, 53]	98.73 (↑)	92.52 (↑)	87.70 (↑)	81.60 (↑)	99.99 (↑)	91.98 (↑)	93.50 (↑)
RandNet-2 [53]	98.75 (↑)	91.53 (↑)	86.53 (↑)	88.35 (↑)	99.91 (↑)	83.00 (↑)	94.55 (↑)
PCANet-2 (softmax) [53]	98.60 (↑)	91.48 (↑)	93.15 (↑)	88.45 (↑)	99.51 (↑)	86.61 (↑)	95.81 (↑)
LDANet-2 [53]	**98.95**	92.48 (↑)	93.19 (↑)	87.58 (↑)	99.86 (↑)	83.80 (↑)	92.78 (↑)
NNet [132]	95.31 (↑)	81.89 (↑)	79.96 (↑)	72.59 (↑)	92.84 (↑)	66.80 (↑)	67.75 (↑)
SAA-3 [132]	96.54 (↑)	89.70 (↑)	88.72 (↑)	77.00 (↑)	97.59 (↑)	75.95 (↑)	81.59 (↑)
DBN-3 [132]	96.89 (↑)	89.70 (↑)	93.27 (↑)	83.69 (↑)	97.40 (↑)	77.50 (↑)	81.37 (↑)
FCCNN [177]	97.57 (↑)	91.09 (↑)	93.55 (↑)	86.77 (↑)	–	–	–
FCCNN (with BT) [177]	97.32 (↑)	90.41 (↑)	93.03 (↑)	89.20 (↑)	–	–	–
SPCN [140]	98.18 (↑)	90.19 (↑)	94.16	90.45 (↑)	99.81 (↑)	89.40 (↑)	–
EvoCNN (best) [208]	98.82 (↑)	**94.78**	**97.20**	**96.47**	99.99 (↑)	94.97 (↑)	95.18 (↑)
EGP (best) [38]	97.19 (↑)	–	–	–	99.91 (↑)	–	93.97 (↑)
IEGP (best)	98.82 (↑)	94.28	93.59 (↑)	89.41 (↑)	**100**	94.88 (↑)	98.26 (↑)
FGP (best)	98.82 (↑)	92.63	93.46 (↑)	92.52 (↑)	**100**	93.90 (↑)	98.46 (↑)
fastFGP (best)	**98.95**	92.58	93.73	93.16	**100**	**95.11**	**98.54**
Rank	**1/22**	4/21	2/21	**2/21**	**1/19**	**1/18**	**1/14**

similar or even better performance than IEGP. The results on these seven datasets demonstrate that fastFGP is more effective than existing methods by achieving better results in almost all the comparisons.

6.4.2 Training Time

The fastFGP approach is sped up by the SCOOP [105] package to use four CPU cores for fitness evaluation. To achieve fair comparisons, we also run FGP on four cores using the SCOOP package. Due to the high computational cost of FGP, we only run it on the FEI_1, FEI_2, KTH, FS, and Rectangle datasets. The training time of fastFGP and FGP on the five datasets are shown in Figure 6.4. The computational time of fastFGP on the other

datasets are shown in Figure 6.5.

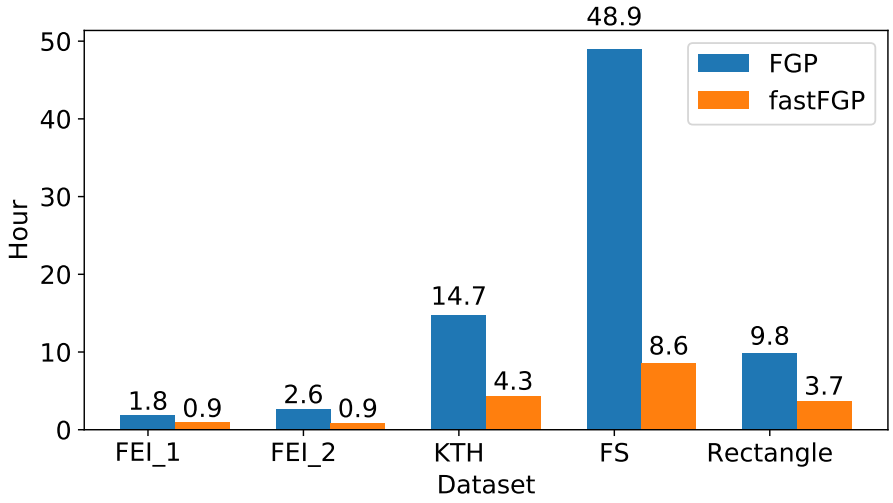


Figure 6.4: Training time (hour) of fastFGP and FGP on the FEI_1, FEI_2, KTH and Rectangle datasets.

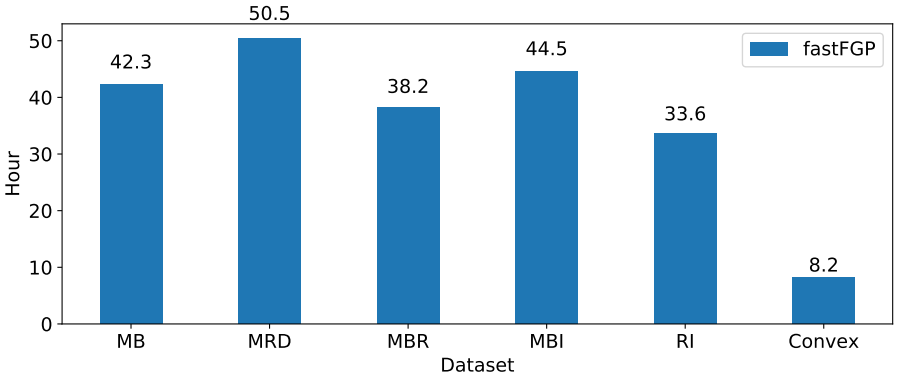


Figure 6.5: Training time (hour) of fastFGP on the MB, MRD, MBR, MBI, RI, and Convex datasets.

From Figure 6.4, it is clear that the proposed fastFGP approach is much faster than the FGP method in training on the five datasets. On the FEI_1 and FEI_2 datasets, fastFGP uses less than 1 hour. On the KTH dataset, FGP uses 14.68 hours for training, while fastFGP only uses 4.26 hours. On

the FS dataset, FGP uses 48.93 hours for training, while fastFGP only uses 8.58 hours. On the Rectangle dataset, fastFGP uses 3.66 hours, while FGP uses 9.78 hours. Therefore, it is clear that fastFGP uses much less training time than FGP. Specifically, fastFGP is at least twice faster than FGP. In addition, when FGP takes a long training time, the saving of fastFGP is even larger. As a result, fastFGP is less computationally expensive than FGP, which is consistent with the finding of the theoretical complexity analysis in Section 6.2.1. Compared with FGP, the fastFGP approach uses multiple populations to learn features from small subsets of the training set. Compared with FGP, fastFGP evaluates most of the individuals on the subsets with a smaller number of training instances at each generation, which significantly reduces the fitness evaluation time. Thus, the overall training time of fastFGP is much less than that of FGP.

From Figure 6.5, it is clear that fastFGP uses about 38 to 50 hours on the MB, MRD, MBR, and MBI datasets, 33.6 hours on the RI dataset, and 8.2 hours on the Convex dataset. The state-of-the-art method, EvoCNN, needs 2 to 3 days to run the experiments using two GPU cards on these benchmarks [208]. The training time of fastFGP is less or similar than/to that of EvoCNN if not considering other factors. However, it is noted that the computational capacity of GPU is much higher than that of CPU. The fastFGP approach could use much less training time than EvoCNN if it could be implemented on GPU, which is beyond the scope of this chapter and necessary to be investigated in the future. The comparisons can provide insights on what is the computational cost of fastFGP in contrast to the state-of-the-art deep learning method (i.e., EvoCNN).

Comparing the training time of fastFGP with both FGP and EvoCNN, it can be found that fastFGP can significantly reduce the training time of feature learning. The results show that the goal of improving the computational effectiveness has been successfully achieved.

6.4.3 Summary

To sum up, the results show that fastFGP is an effective and promising approach for feature learning to image classification. Compared with the original FGP method, fastFGP not only improves the classification performance but also significantly reduces the computational time. Compared with a large number of existing image classification algorithms, fastFGP achieves better results in most comparisons, which further indicates the effectiveness of fastFGP. The extensive experimental results on different datasets demonstrate that fastFGP can effectively solve different types of image classification tasks. The results indicate that fastFGP achieves high generalisation performance by constructing an effective ensemble of diverse and accurate classifiers using the solutions found by small populations from different subsets of training data. The comparisons of fastFGP with FGP and EvoCNN in training time show that the computational efficiency of fastFGP is improved by using small populations to learn features from small subsets of the training data.

6.5 Further Analysis

This section further analyses the ensembles constructed from the solutions of fastFGP to show why fastFGP can achieve good classification performance. Further analysis is conducted to analyse whether and why the knowledge transfer is effective in fastFGP.

6.5.1 Analysis on the Constructed Ensembles

The fastFGP approach outputs five best solutions, i.e., one solution from a small population, and creates an ensemble based on these solutions for testing. Note that N is set as four in fastFGP so that there are five small populations to return five solutions. In the constructed ensemble, there are at most five classifiers. Each classifier is obtained from one solution.

Therefore, the performance of the five classifiers on the test set and the weights of the five classifiers in the ensemble are analysed to show why fastFGP can achieve high generalisation performance.

Figure 6.6 shows the distributions of the classification accuracy (%) obtained by every single classifier and the constructed ensemble on the test set of MB. Note that the results are from the independent 30 runs. From Figure 6.6, it can be found that the classification performance of all the five classifiers is between 98.1% to 98.8%. Among all the five classifiers, the best one is the first one, which is obtained using the best solution found from the full training data. The other four classifiers are obtained using the best solutions found from the small subsets of the training data. This shows that using a small number of training instances could reduce the generalisation performance of the solutions. However, the differences in the accuracy between all the classifiers are not very big. One reason may be that the knowledge transfer between the multiple small populations improves generalisation performance.

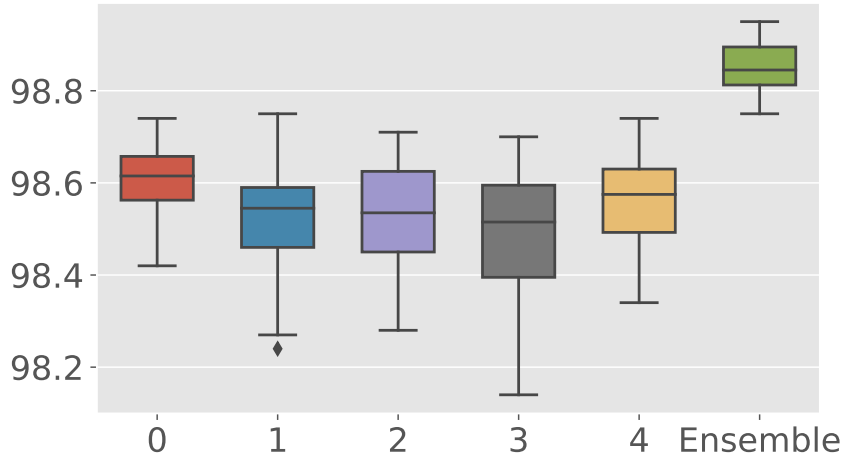


Figure 6.6: The distributions of classification accuracy (%) obtained by each single classifier and the constructed ensemble on the MB dataset.

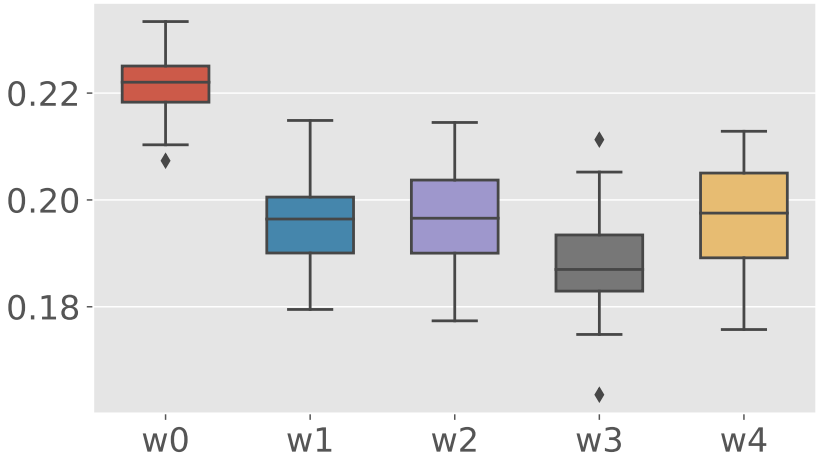


Figure 6.7: The distributions of weights of the five classifiers in the constructed ensemble on the MB dataset.

Comparing the results of the five classifiers with the ensemble in Figure 6.6, it is clear that the constructed ensemble achieves better generalisation performance than any of the five classifiers. More importantly, the lowest accuracy of the ensembles in the 30 runs is better than the highest accuracy of the five classifiers on the MB dataset. In general, to obtain a good ensemble, the classifiers should be diverse and accurate [249]. Since each classifier is trained using solutions from the different small populations, it is clear that the diversity of the classifiers in the ensemble is high. Since the accuracy of every single classifier is also high, the generalisation performance of the constructed ensemble is better than any of the single classifiers. It is noticeable that the accuracy of the constructed ensemble is better than any of the benchmark methods on the MB dataset although the best accuracy of the five classifiers is worse than some benchmark methods, such as FGP and RandNet-2.

Figure 6.7 shows the distributions of weights of the five classifiers in the ensembles on the MB dataset. The weights are calculated according to the performance of the solutions found by the small populations on the

full training data, which has been described in Section 6.2.4. From Figure 6.7, it is obvious that the weight of the first classifier is higher than those of the other four classifiers. From Figures 6.6 and 6.7, we can find that the classifier that has higher weights in the ensemble also obtains higher accuracy on the test set of MB. This shows that the developed combination strategy can combine the classifiers to obtain an effective ensemble with high generalisation performance.

6.5.2 Effectiveness of Knowledge Transfer

To analyse the effectiveness of knowledge transfer in fastFGP, the baseline method without knowledge transfer is used for comparison. Six different datasets comprising different various numbers of instances are used to conduct the experiments. The parameter settings for the fastFGP approaches with and without knowledge transfer (KT) are the same. The classification results of these two approaches are listed in Table 6.5. The "+" and "=" symbols show that fastFGP with knowledge transfer is significantly better and similar than/to the baseline method without knowledge transfer, respectively.

Table 6.5: Classification Accuracy (%) of fastFGP with and without Knowledge Transfer (KT) on Six Datasets

	fastFGP without KT		**fastFGP with KT**	
Dataset	Max	Mean±St.dev	Max	Mean±St.dev
FEI_1	98.00	93.47±2.83=	98.00	94.80±2.61
FEI_2	96.00	88.73±5.08+	**98.00**	**91.40±3.02**
KTH	98.79	98.16±0.43=	**99.09**	97.84±0.78
FS	79.95	78.63±0.74=	**80.23**	78.52±1.09
MB	98.91	98.86±0.04=	**98.95**	98.85±0.05
Convex	98.12	97.86±0.24+	**98.54**	**98.22±0.14**
Total		2+, 4=		

From Table 6.5, it can be found that fastFGP with knowledge transfer

achieves significantly better or similar performance than/to that without knowledge transfer on the six different datasets. On five of the six datasets, fastFGP with knowledge transfer achieves better maximum accuracy than without knowledge transfer. On the remaining dataset (FEI_1), fastFGP with knowledge transfer achieves the same maximum accuracy without knowledge transfer. On the FEI_1, FEI_2 and Convex datasets, fastFGP with knowledge transfer achieves better mean accuracy than without knowledge transfer. On the KTH, FS and MB datasets, fastFGP without knowledge transfer achieves better mean accuracy than with knowledge transfer, but the results of the 30 runs are not significantly different. From Table 6.5, it is noteworthy that fastFGP without knowledge transfer can achieve comparable performance on some datasets. One possible reason is that randomly generated subtrees in the mutation operation may improve the diversity of each small population. Thus, multiple diverse solutions may be found by small populations. Thus, an effective ensemble of diverse classifiers can be formulated to achieve high generalisation performance. Compared to fastFGP without knowledge transfer, in the fastFGP with knowledge transfer method, subtrees of the best solutions/trees found by small populations in fastFGP can be extracted and reused in the mutation operation to replace the selected subtrees. This can help to improve the learning performance of every single small population and find the best individual with better performance. Using multiple effective individuals, an effective ensemble can be constructed for image classification to achieve high generalisation performance. The results show that knowledge transfer can improve the learning performance of fastFGP.

6.6 Chapter Summary

In this chapter, a multi-population GP-based approach (fastFGP) with knowledge transfer and ensembles was developed to improve both generalisation performance and computational efficiency of GP-based feature learn-

ing algorithms for image classification. In the fastFGP approach, a new algorithm framework, where the training data and population are split into multiple small subsets/populations and each population learns features from a small subset of the original training set, was proposed to reduce the computational cost. A new knowledge transfer method, a new fitness function and a new combination strategy were developed to improve the generalisation performance of fastFGP. With these designs, fastFGP can build an ensemble using the best solutions from the different small populations for image classification. Extensive experiments were conducted to show the effectiveness of fastFGP on 11 different image classification tasks of varying difficulty.

The results showed that fastFGP not only achieves better generalisation performance but also has significantly a lower computational cost than the baseline GP-based feature learning algorithm (the FGP method). Compared with other effective methods, fastFGP achieved better performance in almost all the comparisons on 11 different image classification datasets. The results indicated that fastFGP is a promising and effective approach to feature learning for image classification. Compared with the state-of-the-art deep learning method, EvoCNN, fastFGP achieved better or comparable performance and used less computational resources. Further analysis of the constructed ensembles and the classifiers in the ensembles showed that the constructed ensembles obtained better performance than single individual classifiers. The analysis revealed the effectiveness of the developed ensemble formulation strategy. Further analysis showed that knowledge transfer improved the learning performance of fastFGP, leading to constructing an effective ensemble for image classification.

Different from the previous chapters, one of the goals of this chapter was to improve the computational efficiency of GP-based feature learning algorithms for image classification. This problem has seldom been investigated in the GP literature. Many other approaches, such as surrogate, can be used to address this problem, which will be investigated in the future.

Next chapter will conclude this thesis and point out a number of potential research directions for future work.

Chapter 7

Conclusions and Future Work

This thesis focuses on Genetic Programming (GP) for feature learning in image classification. The overall goal of this thesis was to further investigate and explore the potential capability of GP for image classification by developing a new GP-based approach to automatically learning effective features for different types of image classification tasks. This goal has been successfully achieved by developing a new GP-based method with image descriptors for global and/or local feature learning (FLGP in Chapter 3), a new GP-based method with many image-related operators for feature learning (FGP in Chapter 4), a new GP-based method for simultaneous feature learning and ensemble evolving (IEGP in Chapter 5), and a new GP-based method with multiple small populations and knowledge transfer for fast feature learning and ensemble formulation (fastFGP in Chapter 6), for various types of image classification tasks. The performance of the FLGP method was examined on eight different image classification datasets with relatively large image sizes because it can automatically select small regions from the large input images. The performance of the FGP and IEGP methods was examined on 12 different image classification datasets, including seven datasets with a large number of instances/images that have not been solved by existing GP-based methods. The performance of the fastFGP method was tested on 11 large datasets

out of these 12 datasets because one dataset is too small for conducting the experiments. These datasets having varying difficulty represent a wide range of image classification tasks, i.e., face classification, facial expression classification, texture classification, scene classification, and object classification. The classification results have clearly shown that these proposed methods can achieve better classification performance than a large number of competitive methods on these different types of image classification tasks.

The rest of this chapter provides the conclusions for each research objective of this thesis and the key findings of each chapter. The thesis is concluded by highlighting potential research directions for future work.

7.1 Achieved Objectives

This thesis has achieved the following research objectives:

- Proposed a new GP-based approach with image descriptor for automatically learning global and/or local features for image classification. A new program structure allows GP to detect small regions from the relatively large input image, extract features using image descriptors from the detected regions or the input image, and combine the extracted features to form the output features for classification. A new function set with five commonly used image descriptors in the global and local scenarios and a new terminal set were developed. With these designs, the proposed approach can automatically select and combine existing image descriptors to extract rich and discriminative global and/or local features for different image classification tasks. A new fitness evaluation method was proposed to evaluate the GP individuals in order to improve the generalisation performance of the learned features. The proposed approach can significantly outperform other five GP-based methods,

eight traditional methods and three CNN methods in almost all the comparisons on eight different datasets of varying difficulty. The solutions/programs evolved by the proposed approach can provide potentially high interpretability. The analysis showed that the proposed approach can detect informative regions and find the most effective feature extraction functions to extract features from the regions/images.

- Proposed a new GP-based approach with a flexible program structure and a number of image-related operators for feature learning in image classification. A new flexible program structure with an input layer, filtering layers, pooling layers, a feature extraction layer, a feature concatenation layer, and an output layer was proposed. The new function set has a large number of image-related operators, i.e., filters, pooling operators, and image descriptors. With these designs, the proposed approach can learn three types of features, i.e., features from filtering and pooling, features from feature extraction, and the combination of features from filtering/pooling and feature extraction. The proposed approach can be easily applied to different image classification tasks to achieve good classification performance. The proposed approach can achieve better classification performance than a large number of effective algorithms on 12 benchmark datasets of varying difficulty. The solutions found by the proposed approach can be easily visualised as trees to show how and what features are extracted. The features learned by the proposed approach can be visualised to understand why the proposed approach is effective.

- Proposed a GP-based approach with a new representation to automatically and simultaneously learning features and evolving ensembles for image classification. This is the first approach that uses GP to learn features and evolve ensembles from raw images for classifica-

tion. A new representation with an input layer, a filtering & pooling layer, a feature extraction layer, a concatenation layer, a classification layer, a combination layer, and an output layer was proposed. A new function set with many different functions/operators for different subtasks was developed in the proposed approach. The proposed approach can automatically and simultaneously learn high-level features through multiple transformations, select effective classification algorithms, optimise the key parameters of the classification algorithms, and evolve effective ensembles for image classification. The diversity issue of the ensemble building can be automatically addressed by automating the ensemble building process in the proposed approach. The proposed approach can outperform a large number of benchmark methods on 12 different image classification datasets of varying difficulty. The solutions evolved by the proposed approach can provide potentially high interpretability, i.e., what types of features are learned, what classification algorithms are used to build the ensemble, and why they achieve good results.

- Proposed a multi-population GP-based approach with knowledge transfer and ensembles to achieve fast feature learning for effective image classification. A new multi-population algorithm framework was developed to split the training set into several non-overlapping subsets and use multiple small populations to learn features from the small subsets of the training set. A new knowledge transfer method was proposed to effectively extract useful knowledge from these small populations and use the extracted knowledge to improve the learning performance of the other small populations. A new fitness function based on logistic regression and log-loss was proposed to evaluate the fitness of the individual by providing more accurate information on the classification performance. A new ensemble formulation strategy was proposed to calculate the weights and create an effective weighted ensemble of classifiers to achieve

high generalisation performance for image classification. Compared with the FGP approach proposed in the second objective, which is the baseline GP-based feature learning algorithm, this proposed approach improved not only the generalisation performance but also the computational efficiency on 11 image classification datasets. The proposed approach can achieve better performance on 11 datasets than a large number of other effective benchmark methods, including many neural network-based methods and a state-of-the-art deep learning method, EvoCNN [208].

7.2 Main Conclusions

Overall, this thesis finds that GP can be effectively used for feature learning in image classification.

This section discusses the main conclusions for the four research objectives drawn from the four contribution chapters (Chapters 3 to 6).

7.2.1 GP with Image Descriptors for Global and/or Local Feature Learning

Chapter 3 proposes a new GP-based approach with image descriptors to learning global and/or local features from relatively large images for effective classification. This thesis finds that GP with image descriptors can learn effective global and/or local features to achieve better performance than other five GP-based algorithms, eight methods using different manually-extracted features, and three CNN-based methods on eight image classification datasets of varying difficulty. The proposed GP-based approach can learn a set of features that are more effective for classification than the single high-level feature constructed by the other GP methods. The features learned by the proposed GP-based approach are more effective than the eight types of manually extracted features for classifi-

cation. Compared with three CNN-based methods, the GP-based feature learning approach can evolve solutions of variable lengths and learn discriminative global and/or local features, which may be difficult for CNNs. The other findings of this chapter are summarised as follows.

GP Representation

This thesis shows how existing image descriptors can be integrated into GP to learn effective features for image classification by developing a GP representation. It is found that an effective individual representation allows GP to learn discriminative and rich features that can achieve promising performance for different types of image classification tasks. This thesis finds that the solutions of GP can integrate the processes of global feature extraction and/or local feature extraction into a single tree. With a well-designed representation, the solutions of GP can have variable depths or lengths to produce various numbers of features. Representative and well-known image descriptors, i.e., uniform Local Binary Patterns (uLBP) [169], Histogram of Oriented Gradients (HOG) [60], Scale-Invariant Feature Transform (SIFT) [220], Domain-Independent Features (DIF) [242], and Histogram (Hist), can be successfully employed as GP functions to achieve effective feature learning. The experimental results clearly show the benefit of developing GP representation on learning effective features for image classification.

Fitness Evaluation

It is found that an effective fitness evaluation process based on support vector machines and k-fold cross validation can guide the search of the algorithm towards to the optimal solution and improve the generalisation ability of the features learned by GP. The problem that the value ranges of the features extracted by different image descriptors (the feature extraction functions in GP) are different is addressed by the min-max normalisation

method in the fitness evaluation. The benefits of the new fitness evaluation process are clearly shown from the test results of the proposed approach.

Flexibility and Generality

This thesis shows that the GP-based feature learning approach can automatically learn various types and numbers of image features, indicating that it can find a (nearly) optimal set/number of features for solving a task. This is more effective and flexible than the other methods, e.g., the traditional feature extraction methods and the neural network-based methods, that often extract a fixed number of features. It is found that the GP-based approach can be easily applied to learn effective features to achieve promising results on different types of image classification tasks without human intervention. The classification results on different image classification tasks, i.e., facial expression classification, object classification, texture classification, scene classification, and painting classification, clearly show the generality and flexibility of the GP-based approach.

Interpretability

The solutions of the GP-based feature learning approach can provide potentially high interpretability. It is clearly found from the solutions of GP that informative regions can be detected from a large input image and the most effective feature extraction functions can be evolved to extract features from the detected regions or the input image. In addition, which image descriptors are evolved to extract the most effective features for classification can be easily found from the solutions of GP.

7.2.2 GP with Image-Related Operators for Feature Learning

Chapter 4 proposes a new GP-based approach with a flexible program structure and image-related operators to automatically learning features for image classification. The proposed approach achieves promising results on different types of image classification datasets, including the datasets having a large number of training and test instances. This thesis shows that the GP-based approach can achieve significantly better performance than the 11 commonly used methods on five small image classification datasets of varying difficulty. This thesis shows that the GP-based approach can achieve better classification performance than 18 existing effective methods on seven datasets with a large number of training and test instances. Compared with a state-of-the-art deep learning method, i.e., EvoCNN [208], the GP-based approach can achieve better performance on two large datasets and comparable performance on the remaining five large datasets. The other findings of this chapter are summarised as follows.

GP Representation

It is found that a flexible program structure comprising an input layer, filtering layers, pooling layers, a feature extraction layer, a concatenation layer, and an output layer can effectively integrate different functions (including image-related operators) and terminals into trees for feature learning. Compared with the GP representation developed in the previous objective (Chapter 3), the new GP representation includes the additional filtering layers and pooling layers and does not have the region detection layer. This design allows GP to uses multiple different image-related operators, i.e., image filters, pooling operator, and image descriptors, to automatically learn three typically different types of features from the input images, i.e., features from the feature extraction process, features from the

filtering and/or pooling process, and the combined features from the feature extraction and filtering/pooling processes. With a flexible program structure, GP can evolve shallow trees that contain a few functions or evolve deep trees with multiple layers of pooling and/or filtering. This thesis also finds that image-related operators can be used in GP as functional/internal nodes to achieve effective feature learning from different types of images, i.e., face images, texture images, scene images, digits images, and other object images.

Interpretability of Learned Features

It is found that the features learned by GP can provide informative insights on the tasks being tackled and why GP can achieve promising results. What features are learned by GP can be easily found from the solutions of GP because of the use of the image-related operator. By analysing the frequency of the image-related operators in the solutions evolved by GP, an overall picture of the dataset (task domain) can be obtained and it shows that the frequencies of the image-related operators vary with the dataset. By visualising the learned features, it is found that GP can find the optimal solutions to transform the data into a new feature space where the new data can be easily classified.

Number and Type of Learned Features

It is found the GP-based approach can learn various numbers of features from different datasets, which is effective and flexible for image classification. From the evolved solutions, it is found that GP learns three different types of features, i.e., features produced by feature extraction functions, features produced by filtering and/or pooling functions, the combination of features produced by feature extraction and filtering or pooling functions. From the complex images, the GP approach can learn features from filtering, pooling and feature extraction, which are invariant to particular

variations, such as rotation and illumination.

7.2.3 GP for Simultaneous Feature Learning and Ensemble Evolving

Chapter 5 proposes a GP-based approach with a new representation to automatically and simultaneously learning features and evolving ensembles for image classification. The effectiveness of the proposed approach is clearly shown from the classification results of 12 different image datasets. The proposed approach can achieve better classification performance than methods using raw pixels, methods using well-known features, two CNN-based methods, and the two GP methods on six small datasets. On seven datasets with a large number of training and test instances, the proposed approach can achieve better performance than all the benchmark methods on one dataset, is ranked second on three datasets, third on one dataset, and the fourth on the remaining one dataset. Compared with a state-of-the-art deep learning method, EvoCNN, the proposed approach achieves better or the same results on three datasets and slightly worse results on the remaining four datasets. The other findings of this chapter are summarised as follows.

GP Representation

This thesis shows that GP with a multi-layer individual representation can achieve automatic and simultaneous feature learning and ensemble evolving for image classification using a single tree. The new GP representation has seven layers with different functionalities, including feature learning and ensemble learning, which make it different from the current multi-layer representations of GP, such as in [197]. The new representation allows GP to produce solutions of ensembles from raw images without domain expertise. With feature extraction functions in the function set, it is found that GP can learn high-level features through multiple transforma-

tions that are invariant to certain variations such as rotation, which can improve the classification performance. With classification algorithms in the function set and the corresponding key parameters in the terminal set, the GP-based approach is able to automatically select suitable classification algorithms and optimise their key parameters to build an effective ensemble. With this design, it was found that an ensemble with effective and efficient classifiers can be automatically formulated to achieve promising results.

Automatic Ensemble Formulation

It is found that GP can automatically build effective ensembles for image classification. Commonly used classification algorithms, i.e., support vector machines, logistic regression, and random forest, can be automatically selected by GP to build effective ensembles. The key parameters of these classification algorithms can also be selected and optimised by GP to achieve better classification performance. More importantly, the diversity issue of the ensemble can be automatically addressed in the GP-based approach. It is found that the inputs to each classification algorithm (classifier) in the ensemble for the same dataset can be different when integrating the feature learning and ensemble evolving processes into a single GP tree. This increases the diversity of the constructed ensemble via input feature manipulation. Additionally, the parameters for the classification algorithms in the ensemble can be automatically selected and optimised, which encourages high diversity in the constructed ensemble. It is found that the GP-based approach can build effective ensembles of the same classifiers but different parameters, ensembles of ensembles, and ensembles of various classifiers. This shows the effectiveness and the flexibility of automatic construction of effective ensembles from raw images for classification.

7.2.4 Multi-Population GP with Knowledge Transfer and Ensembles for Fast Feature Learning

Chapter 6 proposes a multi-population GP-based approach with knowledge transfer and ensembles to improving both the generalisation performance and computational efficiency of GP-based feature learning algorithms for image classification. Compared with the baseline feature learning algorithm, the proposed approach uses the same program structure, function set and terminal set, but achieves better generalisation performance and higher computational efficiency. Compared with other effective methods, the proposed approach achieves better performance in almost all the comparisons on 11 different image classification datasets. Compared with the state-of-the-art deep learning method, EvoCNN, the proposed approach achieves better or comparable performance and uses less computational resources. The other findings of this chapter are summarised as follows.

Multi-Population Algorithm Framework

It is found that a well-designed multi-population algorithm framework of GP can improve both the generalisation performance and the computational efficiency of GP-based feature learning algorithms for different image classification tasks. The issue of high computational cost can be addressed by proposing a new multi-population algorithm framework. The new framework uses multiple small populations to find the best solutions on small subsets of the training set. By evaluating the small populations on a small number of training instances, the overall computational cost is reduced theoretically and empirically. Since each small population can find the best solution, a natural way to achieve high generalisation performance is to create an effective ensemble for classification using multiple solutions. The effectiveness and efficiency of the GP-based features learning approach with a multi-population algorithm framework is confirmed

via the comparisons with the baseline method without this framework and many other effective methods on different image classification tasks.

Knowledge Transfer

It is found that effective knowledge transfer can improve the learning performance of the GP-based feature learning algorithm with multiple populations. The knowledge transfer approach addresses the three main questions, i.e., what to transfer, how to transfer and when to transfer, of using transfer learning in GP. Based on the multi-population algorithm framework, the knowledge transfer route can be easily defined according to the relationships among the small populations. It is found that the subtrees of the best trees found in the past generations can be extracted from other small populations and used in the mutation operation to effectively improve the learning performance of the current small population. The comparisons with the baseline method without knowledge transfer clearly show the effectiveness of knowledge transfer in the multi-population GP-based feature learning approach on improving the classification performance.

Fitness Evaluation

It is found that the logistic regression classification algorithm and a log-loss-based fitness function can be used to effectively evaluate the learned features and guide the search of GP. Logistic regression can be used to build a soft classifier that predicts the probabilities of the instances in each class. Given the predicted probabilities, a log-loss-based fitness function provides more accurate information on how well the learned features perform in image classification versus only using the classification accuracy.

Ensemble Formulation

It is found that an effective ensemble can be formulated using the solutions found by the small populations of GP to achieve high generalisation performance in image classification. An ensemble formulation strategy can discard bad solutions of GP, keep the best solutions, build multiple classifiers, and combine the classifiers using reasonable weights. This thesis shows that the ensemble formulation strategy can give higher weights to the classifiers that can obtain higher test accuracy. It is found that an effective ensemble of classifiers can be constructed to achieve better performance than using individual classifiers for image classification.

7.3 Future Work

This section highlights key research directions for future work.

7.3.1 Employ Other Image-Related Operators in GP

This thesis has shown how image descriptors, image filters and other image-related operators were employed in GP to achieve effective feature learning for image classification. In the well-developed field of computer vision, there are many other image-related operators and descriptors, such as, frequency filters, Wiener Filter, Canny edge detector, Kirsch operator, morphological operations [2], that have not been employed in the proposed GP-based approaches in this thesis. Employing these operators in GP to learn other types of features for different image classification tasks is worthy to be further investigated. To achieve this, new program structures, new function sets and new terminal sets of GP are needed to effectively use different operators.

7.3.2 GP with Complex/Deep Structures

Many well-known deep CNNs (e.g., ResNet [100] and DenseNet [112]) have architectures with a large number of layers to learn effective representations of the data and have achieved promising classification results on large-scale image classification datasets. Compared with these deep CNNs, the evolved solutions of GP are simpler and have a smaller number of lengths or depths. The performance of GP with complex or deep structures as deep CNNs has not been extensively investigated. It is interesting to investigate whether GP with complex or deep structures can learn effective features to achieve comparable or even better performance than the well-known deep CNNs on large-scale and difficult image classification datasets.

7.3.3 Grammar Guided GP for Feature Learning

This thesis has only employed Strongly Typed GP (STGP) [161] for feature learning in image classification. Based on STGP, a specific program structure can be defined to build a GP tree to perform specific tasks. The Grammar Guided GP (GGGP) algorithm [155, 225] can also achieve this by defining a specific grammar to generate context-sensitive or context-free languages for specific tasks. The potential of GGGP in feature learning has not been (extensively) investigated in this thesis and the other literature. Since GGGP is different from STGP, the initialisation method, the definition of grammar and the genetic operators should be further investigated. Future work can focus on these directions.

7.3.4 Computationally Cheap Evaluation Methods for Fast Feature Learning

One of the problems of GP for feature learning is the high computational cost, especially when the number of training instances is large. Many well-

known image classification datasets have a large number of training instances. It is time-consuming to apply GP to learn features from large datasets. Employing computationally cheap evaluation methods in GP for feature learning is necessary and important for applying GP to learning features to solve image classification tasks with a large number of instances. Well-known methods, such as surrogate and instance selection, can be used to address this issue.

7.3.5 Island-Based GP for Fast Feature Learning

The idea of the use of small populations in the proposed method in Chapter 6 is similar to some ideas of island-based evolutionary algorithms [51, 156]. Island-based evolutionary algorithms may be alternatives to improve the efficiency and effectiveness of the GP-based feature learning algorithms. However, it is necessary to investigate how the population are distributed into small populations and how these small populations communicate at each generation. There is still a large research space to explore how an island-based GP algorithm is developed to achieve fast feature learning for image classification.

7.3.6 Ensemble Learning

This thesis has shown that using an ensemble of classifiers achieved better performance than using a single classifier for image classification. However, there are many different ways to build ensembles for image classification by using different image features, classifiers and combination methods. Furthermore, how to create an effective ensemble with accurate and diverse classifiers is still an open issue in this field. Automated ensemble learning is a promising area that can automatically address these issues in the future.

7.3.7 Transfer Learning

Transfer learning can be used to further improve the learning performance of GP on feature learning for image classification. Although this thesis proposes a new knowledge transfer method in GP to improve its learning performance, the potential of knowledge transfer in GP has not been extensively investigated. It is interesting to further investigate what to transfer, when to transfer and how to transfer in the GP-based feature learning algorithms for image classification.

7.3.8 Multi-Objective Feature Learning

Feature learning can be formulated as a multi-objective optimisation problem of maximising the objective of the classification performance and minimising the objective of the number of learned features. A small number of features can shorten the classification time and provide potentially high interpretability. However, most feature learning algorithms, including the proposed GP-based approaches, only maximise the objective of the classification performance without considering the number of learned features. Therefore, it is necessary to investigate new approaches to find a set of non-dominated solutions for multi-objective feature learning. Future work can start by exploring GP using well-known evolutionary multi-objective algorithm frameworks, such as NSGA-II [62], SPEA2 [251], and MOEA/D [243], to address multi-objective feature learning.

7.3.9 Learning Features from Colour Images

The approaches proposed in this thesis learn features from grey-scale images rather than colour images. Typically, colour images are represented by red, green and blue channels, which provide richer information than grey-scale images. The proposed approaches can be easily extended to learn features from colour images by changing the terminal set. However,

it is computationally expensive. Future work can focus on using GP to learn features from colour images and improving its computational efficiency for image classification.

7.3.10 Few-Shot Learning

In many real-world applications, it is impossible or very difficult to collect a large number of labelled training data. Few-shot learning can be used to learn features using a small number of training instances. A few GP methods have been developed for few-shot learning in texture classification, such as [11, 16]. There are multiple different ways to achieve few-shot learning [221]. The potential of GP for few-shot learning, including one-shot learning and zero-shot learning, has not been extensively investigated in image classification.

7.3.11 GPU Implementation of STGP

One main reason for the success of deep learning in large-scale image classification is the acceleration of the Graphics Processing Unit (GPU). However, very few GP or EC frameworks implemented on GPU to speed up the evolutionary process. A recent example is the Karoo GP [202] method implemented in Python. The Karoo GP is based on Tensorflow and can be used to solve symbolic regression and classification. But it cannot support the acceleration of strongly typed GP, which is the most commonly used version of GP on image data. The performance of the GP method, particularly GP with deep and complex structures, is limited to the current image classification benchmark datasets due to the high computational cost. If these GP methods can be implemented in GPU and benefit from the acceleration, it is possible to develop new effective and efficient GP method for large-scale image classification.

7.3.12 GP for Other Computer Vision Tasks

This thesis is concerned mainly with using GP for feature learning in image classification as it is a fundamental task in computer vision. Besides image classification, solving other tasks in computer vision, e.g., object detection and image segmentation, may also need feature learning. The applications of the GP-based feature learning algorithms to these tasks are limited. Future work can explore how GP is used to learn effective features for solving other tasks, e.g., object detection and image segmentation.

7.4 Chapter Summary

This chapter summarised the objectives achieved in this thesis in GP-based feature learning for image classification. The major conclusions for the four achieved objectives were described in detail and the key research directions of future work were highlighted. This thesis explored the potential of GP for feature learning in image classification and showed a promise. It is worth to further explore this area by addressing the remaining issues.

Bibliography

[1] The british machine vision association and society for pattern recognition. *http://www.bmva.org/visionoverview* (February 20, 2017).

[2] ACHARYA, T., AND RAY, A. K. *Image Processing: Principles and Applications*. John Wiley & Sons, 2005.

[3] AFZALI, S., AL-SAHAF, H., XUE, B., HOLLITT, C., AND ZHANG, M. Genetic programming for feature selection and feature combination in salient object detection. In *Proceedings of International Conference on the Applications of Evolutionary Computation (Part of EvoStar)* (2019), Springer, pp. 308–324.

[4] AGAPITOS, A., LOUGHRAN, R., NICOLAU, M., LUCAS, S., O'NEILL, M., AND BRABAZON, A. A survey of statistical machine learning elements in genetic programming. *IEEE Transactions on Evolutionary Computation 23*, 6 (2019), 1029–1048.

[5] AGAPITOS, A., O'NEILL, M., NICOLAU, M., FAGAN, D., KATTAN, A., BRABAZON, A., AND CURRAN, K. Deep evolution of image representations for handwritten digit recognition. In *Proceedings of IEEE Congress on Evolutionary Computation* (2015), pp. 2452–2459.

[6] AHVANOOEY, M. T., LI, Q., WU, M., AND WANG, S. A survey of genetic programming and its applications. *TIIS 13*, 4 (2019), 1765–1794.

[7] AIN, Q. U., AL-SAHAF, H., XUE, B., AND ZHANG, M. Generating knowledge-guided discriminative features using genetic programming for melanoma detection. *IEEE Transactions on Emerging Topics in Computational Intelligence* (2020), 1–16. DOI: 10.1109/TETCI.2020.2983426.

[8] AIN, Q. U., XUE, B., AL-SAHAF, H., AND ZHANG, M. Genetic programming for skin cancer detection in dermoscopic images. In *Proceedings of IEEE Congress on Evolutionary Computation* (2017), pp. 2420–2427.

[9] AIN, Q. U., XUE, B., AL-SAHAF, H., AND ZHANG, M. Genetic programming for multiple feature construction in skin cancer image classification. In *Proceedings of International Conference on Image and Vision Computing New Zealand* (2019), IEEE, pp. 1–6.

[10] AL-SAHAF, H. *Genetic Programming for Automatically Synthesising Robust Image Descriptors with A Small Number of Instances*. PhD thesis, Victoria University of Wellington, New Zealand, 2017.

[11] AL-SAHAF, H., AL-SAHAF, A., XUE, B., JOHNSTON, M., AND ZHANG, M. Automatically evolving rotation-invariant texture image descriptors by genetic programming. *IEEE Transactions on Evolutionary Computation 21*, 1 (2017), 83–101.

[12] AL-SAHAF, H., BI, Y., CHEN, Q., LENSEN, A., MEI, Y., SUN, Y., TRAN, B., XUE, B., AND ZHANG, M. A survey on evolutionary machine learning. *Journal of the Royal Society of New Zealand 49*, 2 (2019), 205–228.

[13] AL-SAHAF, H., SONG, A., NESHATIAN, K., AND ZHANG, M. Extracting image features for classification by two-tier genetic programming. In *Proceedings of IEEE Congress on Evolutionary Computation* (2012), pp. 12291–12301.

[14] AL-SAHAF, H., SONG, A., NESHATIAN, K., AND ZHANG, M. Two-tier genetic programming: Towards raw pixel-based image classification. *Expert Systems with Applications 39*, 16 (2012), 12291–12301.

[15] AL-SAHAF, H., XUE, B., AND ZHANG, M. A multitree genetic programming representation for automatically evolving texture image descriptors. In *Proceedings of Asia-Pacific Conference on Simulated Evolution and Learning* (2017), Springer, pp. 499–511.

[16] AL-SAHAF, H., ZHANG, M., AL-SAHAF, A., AND JOHNSTON, M. Keypoints detection and feature extraction: A dynamic genetic programming approach for evolving rotation-invariant texture image descriptors. *IEEE Transactions on Evolutionary Computation 21*, 6 (2017), 825–844.

[17] AL-SAHAF, H., ZHANG, M., AND JOHNSTON, M. Binary image classification: A genetic programming approach to the problem of limited training instances. *Evolutionary Computation 24*, 1 (2016), 143–182.

[18] ALAHI, A., ORTIZ, R., AND VANDERGHEYNST, P. Freak: Fast retina keypoint. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (2012), pp. 510–517.

[19] ALMEIDA, A. E., AND TORRES, R. D. S. Remote sensing image classification using genetic-programming-based time series similarity functions. *IEEE Geoscience and Remote Sensing Letters 14*, 9 (2017), 1499–1503.

[20] ALPAYDIN, E. *Introduction to Machine Learning*. MIT Press, 2014.

[21] ALRASHIDI, M. R., AND EL-HAWARY, M. E. A survey of particle swarm optimization applications in electric power systems. *IEEE Transactions on Evolutionary Computation 13*, 4 (2009), 913–918.

[22] ANDREOPOULOS, A., AND TSOTSOS, J. K. 50 years of object recognition: Directions forward. *Computer Vision and Image Understanding 117* (2013), 827–891.

[23] ANTONIE, M.-L., ZAIANE, O. R., AND COMAN, A. Application of data mining techniques for medical image classification. In *Proceedings of the Second International Conference on Multimedia Data Mining* (2001), Springer, pp. 94–101.

[24] ARLOT, S., CELISSE, A., ET AL. A survey of cross-validation procedures for model selection. *Statistics Surveys 4* (2010), 40–79.

[25] ASA, B.-H., AND JASON, W. A user's guide to support vector machine. In *Data Mining Techniques for the Life Sciences*, vol. 609. Humana Press, 2010, pp. 223–239.

[26] ASUNCION, A., AND NEWMAN, D. Uci machine learning repository, 2007.

[27] ATKINS, D., NESHATIAN, K., AND ZHANG, M. A domain independent genetic programming approach to automatic feature extraction for image classification. In *Proceedings of IEEE Congress on Evolutionary Computation* (2011), pp. 238–245.

[28] BÄCK, T., FOGEL, D. B., AND MICHALEWICZ, Z. *Handbook of Evolutionary Computation*. CRC Press, 1997.

[29] BÄCK, T., HAMMEL, U., AND SCHWEFEL, H.-P. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation 1*, 1 (1997), 3–17.

[30] BALLARD, D. H., AND BROWN, C. M. Computer vision. *Prenice-Hall, Englewood Cliffs, NJ* (1982).

[31] BANZHAF, W. Genetic programming for pedestrians. In *Proceedings of International Conference on Genetic Algorithms* (1993), p. 628.

[32] BAY, H., TUYTELAARS, T., AND VAN GOOL, L. Surf: Speeded up robust features. In *Proceedings of European Conference on Computer Vision* (2006), Springer, pp. 404–417.

[33] BENGIO, Y., COURVILLE, A., AND VINCENT, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 8 (2013), 1798–1828.

[34] BEYER, H.-G., AND SCHWEFEL, H.-P. Evolution strategies–a comprehensive introduction. *Natural Computing 1*, 1 (2002), 3–52.

[35] BI, Y., XUE, B., AND ZHANG, M. An automatic feature extraction approach to image classification using genetic programming. In *Proceeding of the 21st European Conference on Applications of Evolutionary Computation* (2018), pp. 421–438.

[36] BI, Y., XUE, B., AND ZHANG, M. A Gaussian filter-based feature learning approach using genetic programming to image classification. In *Proceedings of the 31st Australasian Joint Conference on Artificial Intelligence* (2018), Springer, pp. 251–257.

[37] BI, Y., XUE, B., AND ZHANG, M. A survey on genetic programming to image analysis. *Journal of Zhengzhou University (Engineering Science) 39*, 06 (2018), 3–13.

[38] BI, Y., XUE, B., AND ZHANG, M. An automated ensemble learning framework using genetic programming for image classification. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2019), ACM, pp. 365–373.

[39] BI, Y., XUE, B., AND ZHANG, M. An evolutionary deep learning approach using genetic programming with convolution operators for image classification. In *Proceedings of IEEE Congress on Evolutionary Computation* (2019), IEEE, pp. 3197–3204.

[40] BI, Y., XUE, B., AND ZHANG, M. An effective feature learning approach using genetic programming with image descriptors for image classification [research frontier]. *IEEE Computational Intelligence Magazine 15*, 2 (2020), 65–77.

[41] BI, Y., XUE, B., AND ZHANG, M. Evolving deep forest with automatic feature extraction for image classification using genetic programming. In *Proceedings of The Sixteenth International Conference on Parallel Problem Solving from Nature* (2020), Springer, pp. 3–18.

[42] BI, Y., XUE, B., AND ZHANG, M. Genetic programming with a new representation to automatically learn features and evolve ensembles for image classification. *IEEE Transactions on Cybernetics* (2020. DOI: 10.1109/TCYB.2020.2964566).

[43] BI, Y., XUE, B., AND ZHANG, M. Genetic programming with image-related operators and a flexible program structure for feature learning to image classification. *IEEE Transactions on Evolutionary Computation* (2020. DOI:10.1109/TEVC.2020.3002229).

[44] BI, Y., ZHANG, M., AND XUE, B. Genetic programming for automatic global and local feature extraction to image classification. In *Proceedings of IEEE Congress on Evolutionary Computation* (2018), pp. 1–8.

[45] BOSCH, A., ZISSERMAN, A., AND MUNOZ, X. Image classification using random forests and ferns. In *Proceedings of IEEE 11th International Conference on Computer Vision* (2007), pp. 1–8.

[46] BOSER, B. E., GUYON, I. M., AND VAPNIK, V. N. A training algorithm for optimal margin classifiers. In *Proceedings of The Fifth Annual Workshop on Computational Learning Theory* (1992), ACM, pp. 144–152.

[47] BRAMEIER, M. F., AND BANZHAF, W. *Linear Genetic Programming*. Springer Science & Business Media, 2007.

[48] BREIMAN, L. Random forests. *Machine Learning 45*, 1 (2001), 5–32.

[49] BRUNA, J., AND MALLAT, S. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 8 (2013), 1872–1886.

[50] CAI, L., ZHU, J., ZENG, H., CHEN, J., CAI, C., AND MA, K.-K. HOG-assisted deep feature learning for pedestrian gender recognition. *Journal of the Franklin Institute 355*, 4 (2018), 1991–2008.

[51] CALÉGARI, P., GUIDEC, F., KUONEN, P., AND KOBLER, D. Parallel island-based genetic algorithm for radio network design. *Journal of Parallel and Distributed Computing 47*, 1 (1997), 86–90.

[52] CARUANA, R. Multitask learning. In *Learning to Learn*. Springer, 1998, pp. 95–133.

[53] CHAN, T.-H., JIA, K., GAO, S., LU, J., ZENG, Z., AND MA, Y. PCANet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing 24*, 12 (2015), 5017–5032.

[54] CHANG, C.-C., AND LIN, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2*, 3 (2011), 27.

[55] CHAPELLE, O., HAFFNER, P., AND VAPNIK, V. N. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks 10*, 5 (1999), 1055–1064.

[56] CHEN, X., ONG, Y.-S., LIM, M.-H., AND TAN, K. C. A multi-facet survey on memetic computation. *IEEE Transactions on Evolutionary Computation 15*, 5 (2011), 591–607.

[57] CHOLLET, F., ET AL. Keras. `https://keras.io`, 2015.

[58] COELLO, C. C. Evolutionary multi-objective optimization: a historical view of the field. *IEEE Computational Intelligence Magazine 1*, 1 (2006), 28–36.

[59] COVER, T., AND HART, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory 13*, 1 (1967), 21–27.

[60] DALAL, N., AND TRIGGS, B. Histograms of oriented gradients for human detection. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2005), vol. 1, pp. 886–893.

[61] DAUME III, H., AND MARCU, D. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research 26* (2006), 101–126.

[62] DEB, K., PRATAP, A., AGARWAL, S., AND MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation 6*, 2 (2002), 182–197.

[63] DICKINSON, S. J. Object representation and recognition. *What is cognitive science 7* (1999), 172–207.

[64] DINIZ, J. B., CORDEIRO, F. R., MIRANDA, P. B., AND DA SILVA, L. A. T. A Grammar-based genetic programming approach to optimize convolutional neural network architectures. In *Proceedings of Anais do XV Encontro Nacional de Inteligência Artificial e Computacional* (2018), SBC, pp. 82–93.

[65] DITTIMI, T. V., AND SUEN, C. Y. Mobile phone based ensemble classification of deep learned feature for medical image analysis. *IETE Technical Review* (2019), 1–12.

[66] DORIGO, M. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.

[67] DORIGO, M., BIRATTARI, M., AND STUTZLE, T. Ant colony optimization. *IEEE Computational Intelligence Magazine 1*, 4 (2006), 28–39.

[68] DORIGO, M., MANIEZZO, V., AND COLORNI, A. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26*, 1 (1996), 29–41.

[69] DRUZHKOV, P., AND KUSTIKOVA, V. A survey of deep learning methods and software tools for image classification and object detection. *Pattern Recognition and Image Analysis 26*, 1 (2016), 9–15.

[70] DUCHI, J., HAZAN, E., AND SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research 12* (2011), 2121–2159.

[71] EBERHART, R., AND KENNEDY, J. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science* (1995), IEEE, pp. 39–43.

[72] EFRON, B., AND TIBSHIRANI, R. J. *An Introduction to the Bootstrap*. CRC press, 1994.

[73] ELSKEN, T., METZEN, J. H., AND HUTTER, F. Neural architecture search: A survey. *Journal of Machine Learning Research 20*, 55 (2019), 1–21.

[74] ELSON, J., DOUCEUR, J. R., HOWELL, J., AND SAUL, J. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In *Proceedings of ACM Conference on Computer and Communications Security* (2007), vol. 7, pp. 366–374.

[75] ESPEJO, P. G., VENTURA, S., AND HERRERA, F. A survey on the application of genetic programming to classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 40*, 2 (2010), 121–144.

[76] EVANS, B., AL-SAHAF, H., XUE, B., AND ZHANG, M. Evolutionary deep learning: A genetic programming approach to image classification. In *Proceedings of IEEE Congress on Evolutionary Computation* (2018), pp. 1–6.

[77] EVANS, B. P. Population-based ensemble learning with tree structures for classification. Master's thesis, Victoria University of Wellington, New Zealand, 2019.

[78] EVANS, B. P., AL-SAHAF, H., XUE, B., AND ZHANG, M. Genetic programming and gradient descent: A memetic approach to binary image classification. *arXiv preprint arXiv:1909.13030* (2019).

[79] FEI-FEI, L., AND PERONA, P. A bayesian hierarchical model for learning natural scene categories. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2005), vol. 2, pp. 524–531.

[80] FINLEY, A. O., AND MCROBERTS, R. E. Efficient k-nearest neighbor searches for multi-source forest attribute mapping. *Remote Sensing of Environment 112*, 5 (2008), 2203–2211.

[81] FOGEL, D. B. Introduction to evolutionary computation. *Evolutionary Computation 1* (2000), 1–3.

[82] FOGEL, L. J. Autonomous automata. *Industrial Research 4*, 2 (1962), 14–19.

[83] FOGEL, L. J., OWENS, A. J., AND WALSH, M. J. *Artificial Intelligence Through Simulated Evolution*. John Wiley, 1966.

[84] FOLEGO, G., GOMES, O., AND ROCHA, A. From impressionism to expressionism: Automatically identifying van gogh's paintings. In *Proceedings of IEEE International Conference on Image Processing* (2016), pp. 141–145.

[85] FORCÉN, J., PAGOLA, M., BARRENECHEA, E., AND BUSTINCE, H. Combination of features through weighted ensembles for image classification. *Applied Soft Computing 84* (2019), 105698.

[86] FORTIN, F.-A., DE RAINVILLE, F.-M., GARDNER, M.-A., PARIZEAU, M., AND GAGNÉ, C. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research 13*, Jul (2012), 2171–2175.

[87] FREUND, Y., AND SCHAPIRE, R. E. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of European Conference on Computational Learning Theory* (1995), Springer, pp. 23–37.

[88] FU, W. *Feature Extraction in Edge Detection using Genetic Programming*. PhD thesis, Victoria University of Wellington, New Zealand, 2014.

[89] FU, W., JOHNSTON, M., AND ZHANG, M. Automatic construction of gaussian-based edge detectors using genetic programming. In *Proceedings of European Conference on the Applications of Evolutionary Computation* (2013), Springer, pp. 365–375.

[90] FU, W., JOHNSTON, M., AND ZHANG, M. Low-level feature extraction for edge detection using genetic programming. *IEEE Transactions on Cybernetics 44*, 8 (2014), 1459–1472.

[91] GALAR, M., FERNANDEZ, A., BARRENECHEA, E., BUSTINCE, H., AND HERRERA, F. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42*, 4 (2012), 463–484.

[92] GARDNER, M. W., AND DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment 32*, 14 (1998), 2627–2636.

[93] GOLDBERG, D. E. Genetic algorithms in search, optimization, and machine learning, 1989. *Reading: Addison-Wesley* (1989).

[94] GOLDBERG, D. E., AND DEB, K. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms 1* (1991), 69–93.

[95] GUO, Y., LIU, Y., OERLEMANS, A., LAO, S., WU, S., AND LEW, M. S. Deep learning for visual understanding: A review. *Neurocomputing 187* (2016), 27–48.

[96] HARALICK, R. M. Statistical and structural approaches to texture. *Proceedings of the IEEE 67*, 5 (1979), 786–804.

[97] HARALICK, R. M., SHANMUGAM, K., ET AL. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 6 (1973), 610–621.

[98] HARTIGAN, J. A., AND WONG, M. A. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 28*, 1 (1979), 100–108.

[99] HASSABALLAH, M., ABDELMGEID, A. A., AND ALSHAZLY, H. A. Image features detection, description and matching. In *Image Feature Detectors and Descriptors*. Springer, 2016, pp. 11–45.

[100] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.

[101] HE, K., ZHANG, X., REN, S., AND SUN, J. Identity mappings in deep residual networks. In *Proceedings of European Conference on Computer Vision* (2016), Springer, pp. 630–645.

[102] HERBRICH, R., AND GRAEPEL, T. A PAC-Bayesian margin bound for linear classifiers. *IEEE Transactions on Information Theory 48*, 12 (2002), 3140–3150.

[103] HILBE, J. M. *Logistic Regression Models*. CRC press, 2009.

[104] HJELMÅS, E., AND LOW, B. K. Face detection: A survey. *Computer Vision and Image Understanding 83*, 3 (2001), 236–274.

[105] HOLD-GEOFFROY, Y., GAGNON, O., AND PARIZEAU, M. Once you SCOOP, no need to fork. In *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment* (2014), ACM, p. 60.

[106] HOLLAND, J. H. Outline for a logical theory of adaptive systems. *Journal of the ACM (JACM) 9*, 3 (1962), 297–314.

[107] HOLLAND, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1992.

[108] HOLLAND, J. H., BOOKER, L. B., COLOMBETTI, M., DORIGO, M., GOLDBERG, D. E., FORREST, S., RIOLO, R. L., SMITH, R. E., LANZI, P. L., STOLZMANN, W., ET AL. What is a learning classifier system? In *International Workshop on Learning Classifier Systems* (1999), Springer, pp. 3–32.

[109] HU, J., SHEN, L., AND SUN, G. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 7132–7141.

[110] HUANG, C.-L., AND DUN, J.-F. A distributed PSO–SVM hybrid system with feature selection and parameter optimization. *Applied Soft Computing 8*, 4 (2008), 1381–1391.

[111] HUANG, D., SHAN, C., ARDABILIAN, M., WANG, Y., AND CHEN, L. Local binary patterns and its application to facial image analysis: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 41*, 6 (2011), 765–781.

[112] HUANG, G., LIU, Z., VAN DER MAATEN, L., AND WEINBERGER, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 4700–4708.

[113] HUANG, T. Computer vision: Evolution and promise. In *Imaging Science and Technology, Evolution and Promise, 5th International Conference on High Technology* (1996), pp. 1–4.

[114] IQBAL, M., XUE, B., AL-SAHAF, H., AND ZHANG, M. Cross-domain reuse of extracted knowledge in genetic programming for image classification. *IEEE Transactions on Evolutionary Computation 21*, 4 (2017), 569–587.

[115] IRWIN-HARRIS, W., SUN, Y., XUE, B., AND ZHANG, M. A graph-based encoding for evolutionary convolutional neural network architecture design. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC)* (2019), pp. 546–553.

[116] JI, X., CUI, Y., WANG, H., TENG, L., WANG, L., AND WANG, L. Semisupervised hyperspectral image classification using spatial-spectral information and landscape features. *IEEE Access 7* (2019), 146675–146692.

[117] KARABOGA, D., AND BASTURK, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization 39*, 3 (2007), 459–471.

[118] KENNEDY, J., AND EBERHART, R. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks* (1995), vol. 4, pp. 1942–1948.

[119] KIM, J.-H. Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Computational Statistics & Data Analysis 53*, 11 (2009), 3735–3745.

[120] KIRA, K., RENDELL, L. A., ET AL. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Tenth National Conference on Artificial Intelligence* (1992), vol. 2, pp. 129–134.

[121] KOZA, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge, Massachusetts, London, England, 1992.

[122] KOZA, J. R. *Genetic Programming II: Automatic Discovery of Reusable Subprograms*. The MIT Press, 1994.

[123] KRIG, S. Feature learning and deep learning architecture survey. In *Computer Vision Metrics*. Springer, 2016, pp. 375–514.

[124] KRISHNAPURAM, B., CARIN, L., FIGUEIREDO, M. A., AND HARTEMINK, A. J. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 6 (2005), 957–968.

[125] KRIZHEVSKY, A. Learning multiple layers of features from tiny images. Master's thesis, University of Tront, 2009.

[126] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (2012), pp. 1097–1105.

[127] KROGH, A., AND VEDELSBY, J. Neural network ensembles, cross validation, and active learning. In *Proceedings of Advances in Neural Information Processing Systems* (1995), pp. 231–238.

[128] KUMAR, A., KIM, J., LYNDON, D., FULHAM, M., AND FENG, D. An ensemble of fine-tuned convolutional neural networks for medical image classification. *IEEE Journal of Biomedical and Health Informatics 21*, 1 (2016), 31–40.

[129] KUNCHEVA, L. I., AND WHITAKER, C. J. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning 51*, 2 (2003), 181–207.

[130] LA CAVA, W., AND MOORE, J. H. Learning feature spaces for regression with genetic programming. *Genetic Programming and Evolvable Machines* (2020), 1–35.

[131] LA CAVA, W., SILVA, S., DANAI, K., SPECTOR, L., VANNESCHI, L., AND MOORE, J. H. Multidimensional genetic programming for multiclass classification. *Swarm and Evolutionary Computation 44* (2019), 260–272.

[132] LAROCHELLE, H., ERHAN, D., COURVILLE, A., BERGSTRA, J., AND BENGIO, Y. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th International Conference on Machine Learning* (2007), ACM, pp. 473–480.

[133] LARRAÑAGA, P., AND LOZANO, J. A. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Springer Science & Business Media, 2001.

[134] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature 521*, 7553 (2015), 436–444.

[135] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[136] LeCun, Y., Cortes, C., and Burges, C. J. The MNIST database of handwritten digits, 1998. *URL http://yann. lecun. com/exdb/mnist 10* (1998), 34.

[137] Lee, K.-C., Ho, J., and Kriegman, D. J. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5 (2005), 684–698.

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

[139] Leutenegger, S., Chli, M., and Siegwart, R. Y. BRISK: Binary robust invariant scalable keypoints. In *Proceedings of IEEE International Conference on Computer Vision* (2011), pp. 2548–2555.

[140] Li, H., and Gong, M. Self-paced convolutional neural networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (2017), pp. 2110–2116.

[141] Liang, Y., Zhang, M., and Browne, W. N. Feature construction using genetic programming for figure-ground image segmentation. In *Proceedings of the 20th Asia Pacific Symposium on Intelligent and Evolutionary Systems* (2017), Springer, pp. 237–250.

[142] Liang, Y., Zhang, M., and Browne, W. N. Genetic programming for evolving figure-ground segmentors from multiple features. *Applied Soft Computing 51* (2017), 83–95.

[143] LIU, C., AND WECHSLER, H. A gabor feature classifier for face recognition. In *Proceedings of Eighth IEEE International Conference on Computer Vision* (2001), vol. 2, pp. 270–275.

[144] LIU, C., AND WECHSLER, H. Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition. *IEEE Transactions on Image Processing 11*, 4 (2002), 467–476.

[145] LIU, H., AND MOTODA, H. *Feature Extraction, Construction and Selection: A Data Mining Perspective*, vol. 453. Springer Science & Business Media, 1998.

[146] LIU, L., SHAO, L., LI, X., AND LU, K. Learning spatio-temporal representations for action recognition: A genetic programming approach. *IEEE Transactions on Cybernetics 46*, 1 (2016), 158–170.

[147] LOWE, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2 (2004), 91–110.

[148] LU, D., AND WENG, Q. A survey of image classification methods and techniques for improving classification performance. *International Journal of Remote Sensing 28*, 5 (2007), 823–870.

[149] LU, J., BEHBOOD, V., HAO, P., ZUO, H., XUE, S., AND ZHANG, G. Transfer learning using computational intelligence: a survey. *Knowledge-Based Systems 80* (2015), 14–23.

[150] LYONS, M., AKAMATSU, S., KAMACHI, M., AND GYOBA, J. Coding facial expressions with gabor wavelets. In *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition* (1998), pp. 200–205.

[151] MA, L., LI, M., MA, X., CHENG, L., DU, P., AND LIU, Y. A review of supervised object-based land-cover image classification. *ISPRS Journal of Photogrammetry and Remote Sensing 130* (2017), 277–293.

[152] MAATEN, L. V. D., AND HINTON, G. Visualizing data using t-SNE. *Journal of Machine Learning Research 9* (Nov 2008), 2579–2605.

[153] MALLIKARJUNA, P., TARGHI, A. T., FRITZ, M., HAYMAN, E., CAPUTO, B., AND EKLUNDH, J.-O. The kth-tips2 database. *Computational Vision and Active Perception Laboratory (CVAP), Stockholm, Sweden* (2006).

[154] MARSLAND, S. *Machine Learning: An Algorithmic Perspective*. CRC press, 2015.

[155] MCKAY, R. I., HOAI, N. X., WHIGHAM, P. A., SHAN, Y., AND O'NEILL, M. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines 11*, 3-4 (2010), 365–396.

[156] MERELO, J. J., MORA, A. M., FERNANDES, C. M., ESPARCIA-ALCAZAR, A. I., AND LAREDO, J. L. Pool vs. island based evolutionary algorithms: an initial exploration. In *Proceedings of 2012 Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (2012), IEEE, pp. 19–24.

[157] MIKOLAJCZYK, K., AND SCHMID, C. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 10 (2005), 1615–1630.

[158] MILLER, J. F., AND THOMSON, P. Cartesian genetic programming. In *Proceedings of European Conference on Genetic Programming* (2000), Springer, pp. 121–132.

[159] MITCHELL, T. M. Machine learning. WCB, 1997.

[160] MOHRI, M., ROSTAMIZADEH, A., AND TALWALKAR, A. *Foundations of Machine Learning*. MIT press, 2018.

[161] MONTANA, D. J. Strongly typed genetic programming. *Evolutionary Computation 3*, 2 (1995), 199–230.

[162] MOSCATO, P., ET AL. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech Concurrent Computation Program, C3P Report 826* (1989), 1–68.

[163] MOTODA, H., AND LIU, H. Feature selection, extraction and construction. *Communication of IICM (Institute of Information and Computing Machinery, Taiwan) 5*, 67-72 (2002), 2.

[164] NANDI, R., NANDI, A. K., RANGAYYAN, R. M., AND SCUTT, D. Classification of breast masses in mammograms using genetic programming and feature selection. *Medical and Biological Engineering and Computing 44*, 8 (2006), 683–694.

[165] NARENDRA, P. M., AND FUKUNAGA, K. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, 9 (1977), 917–922.

[166] OHI, M., LI, Y., CHENG, Y., AND WALZ, T. Negative staining and image classification–powerful tools in modern electron microscopy. *Biological Pocedures Online 6*, 1 (2004), 23–34.

[167] OJALA, T., PIETIKAINEN, M., AND HARWOOD, D. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition* (1994), vol. 1, pp. 582–585.

[168] OJALA, T., PIETIKÄINEN, M., AND HARWOOD, D. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition 29*, 1 (1996), 51–59.

[169] OJALA, T., PIETIKAINEN, M., AND MAENPAA, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 7 (2002), 971–987.

[170] PAN, S. J., AND YANG, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering 22*, 10 (2010), 1345–1359.

[171] PASSINO, K. M. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems 22*, 3 (2002), 52–67.

[172] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[173] PINZ, A. Object categorization. *Foundations and Trends® in Computer Graphics and Vision 1*, 4 (2005), 255–353.

[174] POLI, R., LANGDON, W. B., AND MCPHEE, N. F. *A Field Guide to Genetic Programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`", 2008. (With contributions by J. R. Koza).

[175] PRICE, S. R., AND ANDERSON, D. T. Genetic programming for image feature descriptor learning. In *Proceedings of IEEE Congress on Evolutionary Computation* (2017), pp. 854–860.

[176] PRICE, S. R., ANDERSON, D. T., AND PRICE, S. R. GOOFeD: Extracting advanced features for image classification via improved genetic programming. In *Proceedings of IEEE Congress on Evolutionary Computation* (2019), pp. 1596–1603.

[177] QIAN, G., AND ZHANG, L. A simple feedforward convolutional conceptor neural network for classification. *Applied Soft Computing 70* (2018), 1034–1041.

[178] QUINLAN, J. R. *C4. 5: Programs for Machine Learning*. Elsevier, 2014.

[179] RAINA, R., BATTLE, A., LEE, H., PACKER, B., AND NG, A. Y. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine learning* (2007), ACM, pp. 759–766.

[180] RAWAT, W., AND WANG, Z. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation 29*, 9 (2017), 2352–2449.

[181] REAL, E., MOORE, S., SELLE, A., SAXENA, S., SUEMATSU, Y. L., TAN, J., LE, Q. V., AND KURAKIN, A. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning* (2017), pp. 2902–2911.

[182] RECHENBERG, I. Cybernetic solution path of an experimental problem. *Royal Aircraft Establishment Library Translation 1122* (1965).

[183] RIFAI, S., VINCENT, P., MULLER, X., GLOROT, X., AND BENGIO, Y. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of International Conference on Machine Learning* (2011), Omnipress, pp. 833–840.

[184] ROBERTS, M. E. The effectiveness of cost based subtree caching mechanisms in typed genetic programming for image segmentation. In *Proceedings of Workshops on Applications of Evolutionary Computation* (2003), Springer, pp. 444–454.

[185] RODRIGUEZ, J. D., PEREZ, A., AND LOZANO, J. A. Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 3 (2009), 569–575.

[186] RODRIGUEZ-COAYAHUITL, L., MORALES-REYES, A., AND ESCALANTE, H. J. Structurally layered representation learning: towards deep learning through genetic programming. In *Proceedings of European Conference on Genetic Programming* (2018), Springer, pp. 271–288.

[187] RODRIGUEZ-COAYAHUITL, L., MORALES-REYES, A., AND ESCALANTE, H. J. Convolutional genetic programming. In *Proceedings of Mexican Conference on Pattern Recognition* (2019), Springer, pp. 47–57.

[188] RODRIGUEZ-COAYAHUITL, L., MORALES-REYES, A., AND ESCALANTE, H. J. Evolving autoencoding structures through genetic programming. *Genetic Programming and Evolvable Machines 20*, 3 (2019), 413–440.

[189] RUSSEL, S., NORVIG, P., ET AL. *Artificial Intelligence: A Modern Approach*. Pearson Education Limited, 2013.

[190] RYAN, C., FITZGERALD, J., KRAWIEC, K., AND MEDERNACH, D. Image classification with genetic programming: Building a stage 1 computer aided detector for breast cancer. In *Handbook of Genetic Programming Applications*. Springer, 2015, pp. 245–287.

[191] SAMARIA, F. S., AND HARTER, A. C. Parameterisation of a stochastic model for human face identification. In *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision* (1994), pp. 138–142.

[192] SCHMIDT, M., LE ROUX, N., AND BACH, F. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming 162*, 1-2 (2017), 83–112.

[193] SCHÖLKOPF, B., BURGES, C. J., AND SMOLA, A. J. *Advances in Kernel Methods: Support Vector Learning*. MIT press, 1999.

[194] SERGYAN, S. Color histogram features based image classification in content-based image retrieval systems. In *Proceedings of the 6th International Symposium on Applied Machine Intelligence and Informatics* (2008), IEEE, pp. 221–224.

[195] SHA, D., AND HSU, C.-Y. A hybrid particle swarm optimization for job shop scheduling problem. *Computers & Industrial Engineering 51*, 4 (2006), 791–808.

[196] SHAO, J., ZHANG, X., DING, Z., ZHAO, Y., CHEN, Y., ZHOU, J., WANG, W., MEI, L., AND HU, C. Good practices for deep feature fusion. `https://www.youtube.com/watch?v=NaoVOOhVC3w&t=215s`, 2016. Online; accessed 20-October-2016.

[197] SHAO, L., LIU, L., AND LI, X. Feature learning for image classification via multiobjective genetic programming. *IEEE Transactions on Neural Networks and Learning Systems 25*, 7 (2014), 1359–1371.

[198] SHARIF, M., NAZ, F., YASMIN, M., SHAHID, M. A., AND REHMAN, A. Face recognition: A survey. *Journal of Engineering Science & Technology Review 10*, 2 (2017), 166–177.

[199] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[200] SMART, W. Genetic programming for multiclass object classification. Master's thesis, Victoria University of Wellington, New Zealand, 2005.

[201] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I., AND SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15*, 1 (2014), 1929–1958.

[202] STAATS, K., PANTRIDGE, E., CAVAGLIA, M., MILOVANOV, I., AND ANIYAN, A. TensorFlow enabled genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2017), pp. 1872–1879.

[203] STORN, R. On the usage of differential evolution for function optimization. In *Proceedings of Biennial Conference of the North American Fuzzy Information Processing Society* (1996), IEEE, pp. 519–523.

[204] STORN, R., AND PRICE, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization 11*, 4 (1997), 341–359.

[205] SUCAR, L. E. Bayesian classifiers. in: Probabilistic graphical models: Principles and applications. *Advances in Computer Vision and Pattern Recognition* (2015), 41–62.

[206] SUGANUMA, M., SHIRAKAWA, S., AND NAGAO, T. A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2017), pp. 497–504.

[207] SUGANUMA, M., TSUCHIYA, D., SHIRAKAWA, S., AND NAGAO, T. Hierarchical feature construction for image classification using genetic programming. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics* (2016), pp. 001423–001428.

[208] SUN, Y., XUE, B., ZHANG, M., AND YEN, G. G. Evolving deep convolutional neural networks for image classification. *IEEE Transactions on Evolutionary Computation 24*, 2 (2019), 1–14.

[209] SUN, Y., XUE, B., ZHANG, M., YEN, G. G., AND LV, J. Automatically designing CNN architectures using the genetic algorithm for image classification. *IEEE Transactions on Cybernetics* (2020), 1–15. DOI: 10.1109/TCYB.2020.2983860.

[210] SUTTON, R. S., AND BARTO, A. G. *Reinforcement Learning: An Introduction*. MIT press, 2018.

[211] SZEGEDY, C., IOFFE, S., VANHOUCKE, V., AND ALEMI, A. A. Inception-v4, Inception-ResNet and the impact of residual connections on learning. In *Proceedings of Thirty-first AAAI Conference on Artificial Intelligence* (2017), pp. 4278–4284.

[212] SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V., AND RABINOVICH, A. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1–9.

[213] SZELISKI, R. *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010.

[214] TANG, M., AND YAO, X. A memetic algorithm for VLSI floor planning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 37*, 1 (2007), 62–69.

[215] THOMAZ, C. E. FEI face database. *online: http://fei.edu.br/˜cet/facedatabase.html* (Mar 2012).

[216] TORREY, L., AND SHAVLIK, J. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques 1* (2009), 242–264.

[217] TRAN, B., XUE, B., AND ZHANG, M. Genetic programming for multiple-feature construction on high-dimensional classification. *Pattern Recognition 93* (2019), 404–417.

[218] TRAN, C. T., ZHANG, M., ANDREAE, P., XUE, B., AND BUI, L. T. An effective and efficient approach to classification with incomplete data. *Knowledge-Based Systems 154* (2018), 1–16.

[219] ULUSOY, I., AND BISHOP, C. M. Comparison of generative and discriminative techniques for object detection and classification. In *Toward Category-Level Object Recognition*. Springer, 2006, pp. 173–195.

[220] VEDALDI, A., AND FULKERSON, B. VLFeat: An open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM International Conference on Multimedia* (2010), pp. 1469–1472.

[221] WANG, Y., AND YAO, Q. Few-shot learning: A survey. *arXiv preprint arXiv:1904.05046* (2019).

[222] WEISE, T. Global optimization algorithms-theory and application. *Self-published 2* (2009).

[223] WEISS, K., KHOSHGOFTAAR, T. M., AND WANG, D. A survey of transfer learning. *Journal of Big Data 3*, 1 (2016), 9.

[224] WESTON, J., WATKINS, C., ET AL. Support vector machines for multi-class pattern recognition. In *Proceedings of the 7th European Symposium on Artificial Neural Networks* (1999), vol. 99, pp. 219–224.

[225] WHIGHAM, P. A., ET AL. Grammatically-based genetic programming. In *Proceedings of the Workshop on Genetic Programming: From Theory to Real-world Applications* (1995), vol. 16, pp. 33–41.

[226] WISTUBA, M., RAWAT, A., AND PEDAPATI, T. A survey on neural architecture search. *arXiv preprint arXiv:1905.01392* (2019).

[227] WRIGHT, J., YANG, A. Y., GANESH, A., SASTRY, S. S., AND MA, Y. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31*, 2 (2008), 210–227.

[228] WU, Y., SU, Q., MA, W., LIU, S., AND MIAO, Q. Learning robust feature descriptor for image registration with genetic programming. *IEEE Access 8* (2020), 39389–39402.

[229] XIA, J., GHAMISI, P., YOKOYA, N., AND IWASAKI, A. Random forest ensembles and extended multiextinction profiles for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing 56*, 1 (2018), 202–216.

[230] XIE, L., AND YUILLE, A. Genetic CNN. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 1379–1388.

[231] XING, X., JI, K., ZOU, H., CHEN, W., AND SUN, J. Ship classification in TerraSAR-X images with feature space based sparse representation. *IEEE Geoscience and Remote Sensing Letters 10*, 6 (2013), 1562–1566.

[232] XU, Y., LI, Z., YANG, J., AND ZHANG, D. A survey of dictionary learning algorithms for face recognition. *IEEE Access 5* (2017), 8502–8514.

[233] XUE, B. *Particle Swarm Optimisation for Feature Selection in Classification*. PhD thesis, Victoria University of Wellington, New Zealand, 2013.

[234] XUE, B., ZHANG, M., BROWNE, W. N., AND YAO, X. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation 20*, 4 (2015), 606–626.

[235] YOUNG, S., ABDOU, T., AND BENER, A. Deep super learner: A deep ensemble for classification problems. In *Proceedings of Canadian Conference on Artificial Intelligence* (2018), Springer, pp. 84–95.

[236] YU, T., GUO, C., WANG, L., XIANG, S., AND PAN, C. Self-paced autoencoder. *IEEE Signal Processing Letters 25*, 7 (2018), 1054–1058.

[237] YU, Z., ZHANG, Y., YOU, J., CHEN, C. P., WONG, H.-S., HAN, G., AND ZHANG, J. Adaptive semi-supervised classifier ensemble for

high dimensional data classification. *IEEE Transactions on Cybernetics*, 99 (2017), 1–14.

[238] YUSUP, N., ZAIN, A. M., AND HASHIM, S. Z. M. Overview of PSO for optimizing process parameters of machining. *Procedia Engineering 29* (2012), 914–923.

[239] ZEILER, M. D., AND FERGUS, R. Stochastic pooling for regularization of deep convolutional neural networks. In *Proceedings of the 1st International Conference on Learning Representations,* (2013), pp. 1–9.

[240] ZHANG, H., LI, J., HUANG, Y., AND ZHANG, L. A nonlocal weighted joint sparse representation classification method for hyperspectral imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 7*, 6 (2013), 2056–2065.

[241] ZHANG, M., AND CIESIELSKI, V. Genetic programming for multiple class object detection. In *Proceedings of Australian Joint Conference on Artificial Intelligence* (1999), vol. 1747, Springer, pp. 180–192.

[242] ZHANG, M., CIESIELSKI, V. B., AND ANDREAE, P. A domain-independent window approach to multiclass object detection using genetic programming. *EURASIP Journal on Advances in Signal Processing 2003*, 8 (2003), 841–859.

[243] ZHANG, Q., AND LI, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation 11*, 6 (2007), 712–731.

[244] ZHAO, Z.-Q., XU, S.-T., LIU, D., TIAN, W.-D., AND JIANG, Z.-D. A review of image set classification. *Neurocomputing 335* (2019), 251–260.

[245] ZHAO, Z.-Q., ZHENG, P., XU, S.-T., AND WU, X. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems 30*, 11 (2019), 3212–3232.

[246] ZHOU, B., LAPEDRIZA, A., KHOSLA, A., OLIVA, A., AND TORRALBA, A. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 40*, 6 (2017), 1452–1464.

[247] ZHOU, Z.-H. *Ensemble Methods: Foundations and Algorithms*. Chapman and Hall/CRC, 2012.

[248] ZHOU, Z.-H., AND FENG, J. Deep forest: towards an alternative to deep neural networks. In *Proceedings of International Joint Conferences on Artificial Intelligence* (2017), AAAI Press, pp. 3553–3559.

[249] ZHOU, Z.-H., AND FENG, J. Deep forest. *National Science Review 6*, 1 (2018), 74–86.

[250] ZHU, Y., YAO, Y., WU, Z., CHEN, Y., LI, G., HU, H., AND XU, Y. GP-CNAS: Convolutional neural network architecture search with genetic programming. *arXiv preprint arXiv:1812.07611* (2018).

[251] ZITZLER, E., LAUMANNS, M., AND THIELE, L. SPEA2: Improving the strength pareto evolutionary algorithm. *TIK-report 103* (2001).