An Automated Ensemble Learning Framework Using Genetic Programming for Image Classification

Ying Bi, Bing Xue, Mengjie Zhang
School of Engineering and Computer Science
Victoria University of Wellington, P.O. Box 600, Wellington, New Zealand
{Ying.Bi;Bing.Xue;Mengjie.Zhang}@ecs.vuw.ac.nz

ABSTRACT

An ensemble consists of multiple learners and can achieve a better generalisation performance than a single learner. Genetic programming (GP) has been applied to construct ensembles using different strategies such as bagging and boosting. However, no GP-based ensemble methods focus on dealing with image classification, which is a challenging task in computer vision and machine learning. This paper proposes an automated ensemble learning framework using GP (EGP) for image classification. The new method integrates feature learning, classification function selection, classifier training, and combination into a single program tree. To achieve this, a novel program structure, a new function set and a new terminal set are developed in EGP. The performance of EGP is examined on nine different image classification data sets of varying difficulty and compared with a large number of commonly used methods including recently published methods. The results demonstrate that EGP achieves better performance than most competitive methods. Further analysis reveals that EGP evolves good ensembles simultaneously balancing diversity and accuracy. To the best of our knowledge, this study is the first work using GP to automatically generate ensembles for image classification.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by classification**; • **Genetic programming**; • **Ensemble method**; • **Computer vision** → Image classification;

KEYWORDS

Genetic Programming, Ensemble Learning, Image Classification, Feature Learning, Machine Learning, Computer Vision

ACM Reference Format:
Ying Bi, Bing Xue, Mengjie Zhang. 2019. An Automated Ensemble Learning Framework Using Genetic Programming for Image Classification. In *Genetic and Evolutionary Computation Conference (GECCO '19), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3321707.3321750

GECCO '19, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6111-8/19/07…$15.00
https://doi.org/10.1145/3321707.3321750

1 INTRODUCTION

Ensemble learning is a popular topic in recent years [7]. An ensemble often consists of multiple *base/individual learners* to solve a problem [32]. Each *base/individual learner* is trained using a traditional machine learning algorithm. Generally, an ensemble can achieve better generalisation performance than a single learner [33]. However, to achieve a strong generalisation ability, the learners in an ensemble should be accurate and diverse. To obtain a good ensemble, many methods have been developed, including bagging and boosting methods [8, 13, 29]. However, in most existing ensemble methods, the selection of the base learners and the combination of them are often manually determined. The complementarity of the learners may not be well considered and addressed during this process. Therefore, this work develops a new approach to automating ensemble learning to address the above limitations.

Genetic programming (GP) aims at automatically evolving computer programs to solve problems [15] and is well-known for its flexible representation, good search ability and high interpretability of the solutions. Using GP to construct ensembles for classification is not new since each GP tree can be a classifier for binary or multi-class classification [9]. The bagging and boosting methods are often employed to construct an ensemble of GP trees, such as in [8, 13, 29]. The GP-based ensemble methods have shown promising results in classification [29]. However, to the best of our knowledge, there are no GP-based ensemble methods for image classification.

Image classification is a fundamental task in computer vision and machine learning. But it is very challenging due to high variations of images. Generally, the raw pixels of the images are often not meaningful so that feature extraction, which is able to extract informative features from images, is often needed. Recently, GP has been widely applied to feature learning and image classification [4, 17, 27]. The flexible representation allows GP to integrate many image-related operators, such as histogram of orientated gradient (HOG) [17], Gaussian filter, Sobel filter, and Gabor filter [27], to learn informative features for image classification. The features are more effective and discriminative than many hand-crafted features extracted by traditional feature extraction methods such as HOG [27]. However, most existing GP-based methods are only suitable for binary classification [1, 17] or a particular domain such as texture [2]. Therefore, a powerful GP-based method, which is able to solve different types of image classification tasks, is needed.

This work aims to develop a new GP-based ensemble method for image classification [1]. The new method will provide end-to-end solutions for image classification by performing feature learning, classification algorithm selection, classifier training, combination,

[1]The code is released on *https://github.com/yingbi7460/egp*

and prediction, automatically and simultaneously. To achieve this, a novel program structure, a new function set and a new terminal set are developed in EGP. Specifically, this study will answer the following questions.

(1) How to develop the program structure of the EGP approach?
(2) What are the function set and the terminal set for EGP?
(3) How to use the EGP approach for image classification?
(4) Can EGP achieve better performance than state-of-the-art methods on image classification data sets?
(5) Can EGP evolve ensembles with high accuracy and diversity?

2 BACKGROUND

This section briefly provides important concepts of GP. Then it reviews recent work related to this study and summarises current limitations.

2.1 GP and Strongly Typed GP

GP uses a tree-based representation, where each individual is represented by a tree. A GP tree consists of a root node, a number of internal nodes and leaf nodes. The leaf nodes are constructed by selecting terminals from a predefined terminal set. The terminal set often contains variables/features and constant parameters, where the variables/features are related to problems. The root node and internal nodes are built by selecting functions from a predefined function set, which often has a number of functions including arithmetic functions and domain-dependent functions.

In standard GP, the functions and the terminals deal with one data type. To deal with multiple data types, strongly typed GP (STGP) was proposed in [22]. In STGP, an input type and an output type of each function need to be specified, and an output type of each terminal needs to be specified. To construct a GP tree, the input type of a node should be the same as the output type of its child node. By such a constraint, STGP is very suitable for dealing with multiple tasks, where different functions or terminals can be used for each task. Because of the complexity of the image-related tasks, STGP is commonly used for image analysis, including feature extraction and image classification [4]. In order to deal with multiple tasks using a single GP tree, it is necessary to develop a program structure. For example, in [27], the program structure of GP has an input layer, a filtering layer, a pooling layer, and a concatenation layer. Each layer has its own functions that allow GP to automatically select them to build GP trees.

2.2 Related Work

2.2.1 ***Ensemble Learning Using GP.*** For a classification problem, GP can evolve classifiers via the evolutionary learning process from a training data set. A common GP-based classifier often uses the features as the inputs and produces a float number as the output, which can be used to make classification decision for a binary classification task according to a predefined threshold. Considering GP as a base learner, existing ensemble methods such as bagging and boosting can be used to build an ensemble of GP-based classifiers. Karakatič and Podgorelec [14] developed a GPAB method for classification, where AdaBoost was used to update the weights for instances and for each GP-based classifier in the ensemble. Kammerer and Affenzeller [13] integrated a confidence measure in GP-based classifier ensembles to determine reject or accept predictions based on the confidence value and a predefined threshold. Tran et al. [29] employed bagging to build a set of interval GP (IGP) classifiers for classification on missing data. This ensemble method was more accurate than single IGP classier and other ensembles. Instead of running GP multiple independent times to obtain a set of classifiers using bagging, Dick et al. [8] proposed a new GP-based approach that was able to generate an ensemble from a single run. This method has improved the computational complexity of building GP-based ensembles.

Existing work has shown promising results of GP-based ensembles on classification or regression tasks. However, no GP-based ensemble methods address image classification tasks, which are challenging due to the high variations of images. Most existing GP-based ensemble methods need to run GP multiple times, which are time consuming [13, 29]. There are multiple ways of GP to obtain an ensemble as GP has a flexible representation. This paper will address the current limitations by further exploring the potential of GP on ensemble learning for image classification.

2.2.2 ***GP for Feature Learning and Image Classification.*** GP has been applied to feature learning for image classification using raw pixels as inputs. Based on whether a classification method is used for evaluation, the existing work can be broadly classified into two groups, which are using GP to construct classifiers and using GP to learn features from raw pixels. The first group uses GP to construct classifiers with simultaneously dealing with region detection, feature extraction and feature construction [1, 17]. The methods in this group often have a new program structure with the capability of dealing with multiple tasks in a single tree, and are naturally suitable for binary image classification. The second group uses GP to automatically learn a set of features and uses a machine learning classification method, e.g., KNN and SVMs, to perform classification [2, 27]. Compared with the methods in the first group, the methods in this group can deal with binary or multi-class image classification tasks, and are often more accurate. Therefore, this paper focuses on the methods in the second group.

In recent years, many image-related operators such as Gaussian filter, mean filter, Sobel filter, and HOG have been developed as functions in GP to extract/learn high-level features for effective classification [3, 17, 27]. These image-related operators were very helpful for learning important features [3, 17, 27]. Based on the framework of GP for feature learning in [3, 27], this paper uses a set of image-related operators for feature learning with the consideration of efficiency and effectiveness.

In [2, 27], a classification algorithm was employed to evaluate each individual of GP. However, how to select the classification algorithm and whether it is effective for classification using the learned features are often determined manually via trial and error. Instead of using a single classification algorithm, this paper proposes an ensemble method to achieve better classification performance. The new method is able to address the above limitations by using a set of image-related operators for feature learning, and automatically selecting suitable classification methods for classification, and find the combinations for optimising the classification performance.

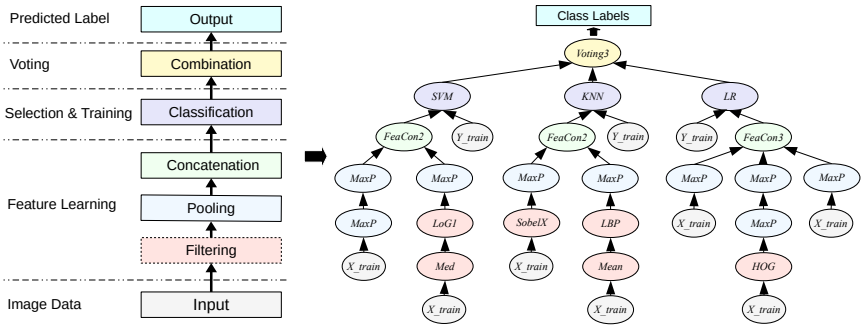

Figure 1: The program structure of EGP and an example program that can be evolved by EGP.

3 PROPOSED APPROACH

This section describes the proposed EGP approach in detail, including the overall algorithm, the new program structure, the new function set, the new terminal set, and the test process of EGP.

3.1 Overall Algorithm

The overall algorithm of EGP for image classification is described in Algorithm 1. The inputs of the EGP system are the commonly used parameter settings for GP and the training data, i.e., images X_train and labels Y_train. The aim of EGP is to search for the best individual, which produces the combined predictions/outputs (class labels) for a image classification task.

Algorithm 1: Framework of EGP

Input : Commonly used parameter settings for EGP; X_train: the training data; Y_train: the labels of training data.
Output : $Best\_Individual$: the best program tree.

$P_0 \leftarrow$ Initialise the population using the *ramped half-and-half* method based on a new program structure, a new function set and a new terminal set;
$g \leftarrow 0$;
while $g \leq G$ **do**
  Evaluate the fitness of each individual in P_g;
  for each individual p *in* P_g **do**
    if p *in* $Cache\_Table$ **then**
      $accy_p \leftarrow$ the fitness value of $Cache\_Table(p)$;
    else
      Feed X_train and Y_train to p;
      Obtain the predicted class labels $Y\_predict$;
      $accy_p \leftarrow$ calculate the accuracy according to Y_train and $Y\_predict$;
    end
    Assign $accy_p$ as fitness value to p;
  end
  Update $Best\_Individual$;
  Update $Cache\_Table$;
  $P_{g+1} \leftarrow$ Use genetic operators to generate a new population;
  $g \leftarrow g + 1$
end
Return $Best\_Individual$.

The overall learning process of EGP is similar to the standard GP process except for that a $Cache\_Table$ is employed in EGP to reduce the time of evaluation as GP is known for computationally expensive on image data [25]. The $Cache\_Table$ stores the best individuals of the past generations and a search is conducted in the $Cache\_Table$ before evaluating each individual at each generation [25]. The size of $Cache\_Table$ can be any number but a tradeoff between the searching time of $Cache\_Table$ and the evaluation time of an individual needs to be considered. In EGP, we set the number of individuals in $Cache\_Table$ to $5 \times N$ as the population size of EGP is small, where N is the population size.

Fitness Function: The fitness function for EGP is the classification accuracy, which is calculated according to the EGP system output $Y\_predict$ and the real class labels Y_train. The classification accuracy is the most commonly used fitness function for GP on image classification [17, 27].

The main differences of EGP and existing GP methods on image classification are the program structure, the function set and the terminal set, which determine how each individual solves the task by producing a combined output under a tree-based representation. The following subsections will introduce these distinctive components of EGP.

3.2 New Program Structure

The EGP method is based on the STGP method, which has been introduced in Section 2.2.2. A new program structure is developed in EGP and it is important for specifying the type for each function. Figure 1 shows the logical flowchart of constructing the program structure and an example program that can be evolved by EGP.

As shown in Figure 1 with different colours, the new program structure compounds of the input, filtering, pooling, concatenation, classification, combination, and output processes. The inputs of EGP are the image data, X_train and Y_train, where X_train is the training set and Y_train is the class labels of X_train. The processes of filtering, pooling and concatenation belong to feature learning, where image data are transformed into a set of features. The process of classification including classification method selection and training. In this process, the input image features produced by the feature learning process are transformed into predicted class labels. The combination process receives the predicted class labels and produces combined predictions using voting functions. Finally, the outputs of EGP are the predicted class labels $Y\_predict$.

In the new program structure, the classification process is connected with the feature learning process, which means that the inputs for the classification functions in the evolved GP trees are different if their child branches are different. As the classifiers will be combined by the voting functions in the ensemble, this allows the classifiers to have high diversity, which is one of the most important issues in ensemble learning.

It is noticeable that the tree depth of the feature learning process and the tree depth of the combination process are flexible, and the tree depth of the classification process is fixed as 1. Based on this program structure, different processes have different functions, which form the new function set of EGP.

3.3 New Function Set

The new function set has three different types of functions according to the tasks, i.e., feature learning functions, classification functions, and combination functions.

Feature Learning Functions: The feature learning functions aim to transform the input data X_train into $X\_features$, which means that each image in the original data set is transformed into a set of features/numbers. There are many functions than can be used for this purpose. In EGP, the selection of feature learning functions are based on the existing work on GP for feature learning in [3, 27], where filtering, pooling and concatenation functions are employed. The new set of feature learning functions employed in EGP is listed in Table 1. The three concatenation functions aim to concatenate multiple vectors/images into a vector to obtain $X\_features$. The pooling function is the *MaxP* function, which performs max-pooling with a commonly used 2×2 kernel to each image in X_train.

Table 1: Feature learning functions

Type	Functions	Description
Concatenation	*FeaCon1, FeaCon2, FeaCon3*	Concatenate two vectors or two/three images into a vector, respectively
Pooling	*MaxP*	Conduct max-pooling with 2×2 kernel to each image
Filtering	*Gau, GauD, Gabor, Lap, LoG1, LoG2, Sobel, SobelX, SobelY, LBP, HOG, Med, Mean, Min, Max, Sqrt, W-Add, W-Sub, ReLU*	Performing the corresponding filtering operation to each image

Note that each function takes a number of images as input, performs corresponding operation to each image and returns a number of images/vectors.

The filtering functions are *Gau, GauD, Gabor, Lap, LoG1, LoG2, Sobel, SobelX, SobelY, LBP, HOG, Med, Mean, Min, Max, Sqrt, W-Add, W-Sub* and *ReLU*. The *Gau* and *GauD* functions are the Gaussian filter and the Gaussian derivative filter. *Gau* has a parameter, i.e., standard deviation σ, and *GauD* has three parameters, i.e., o_1, o_2 and σ. The o_1 and o_2 represents the orders of derivative along the X/Y axis, respectively. The *Gabor* function uses the Gabor filter with orientation θ and frequency f. The *Lap* function performs Laplacian filtering to each image. The *LoG1* and *LoG2* functions conduct Laplacian of Gaussian filtering with $\sigma = 1$ or 2 to each image. The *SobelX* and *SobelY* functions perform Sobel filtering along the X and Y axis to each image, respectively. The *Sobel* function conducts Sobel edge detection to each image. The *LBP* and *HOG* functions calculate LBP and HOG image for each image, respectively, and they may produce images containing informative features. The *Med, Mean, Max*, and *Min* functions perform median, average, max, and min filtering with 3×3 window to each image, respectively. The *Sqrt* function returns $\sqrt{p}$ for each p in an image and returns 1 if $p \leq 0$. The *W-Sub* and *W-Add* functions subtract or add two weighted images with the same or different sizes, i.e., $image_1 \times n_1 -/+ image_2 \times n_2$. If the sizes are not the same, the two functions will cut the two images based on the smaller width and height. The *ReLU* function is the rectified linear unit and it returns $max(0,\ p)$ for each p in an image.

Classification Functions: Any existing classification methods can be used as functions under the current framework of EGP. However, to narrow the search space, we select six different classification methods as classification functions of EGP based on [30]. The functions are logistic regression (LR), KNN, SVM, RF, extremely randomised forest (ERF), and AdaBoost, as listed in Table 2. Commonly used parameters are set for the six functions [2, 33].

EGP uses the evolutionary learning process to find the best individual that is able to perform feature transformation, classifier training, prediction and combination, simultaneously and automatically. Because the traditional classification algorithms need a learning process to learn a classifier, it is necessary to develop a new process for these functions that suits the current framework of the proposed method. Therefore, the six classification functions perform differently during the evolutionary learning process and after the evolutionary learning process, respectively.

Table 2: Classification functions

Methods	Description
LR	Logistic regression
KNN	k-nearest neighbour. The number of neighbour is 1
SVM	Linear support vector machine. The penalty parameter C is 1.0
RF	Random forest. The number of trees is 500 and the maximum tree depth is 100 [33]
AdaBoost	Number of trees is 500 and the maximum tree depth is 100 [33]
ERF	Extremely randomised forest. The number of trees is 500 and the maximum tree depth is 100 [33]

The implementations of these functions are based on the *scikit-learn* package [23] and the other parameter settings are the default ones.

During the evolutionary learning process of EGP, each classification function uses k-fold cross-validation (CV) to obtain the predicted class labels $Y\_predict$ for the training set X_train. The number of k is set as 3 according to [33]. The consideration of using k-fold CV is to avoid overfitting and to obtain a strong generalisation ability of each trained classifier. In this process, the inputs of EGP are the training images X_train and the class labels Y_train. Each classification function takes $k - 1$ folds of the split training data to train the classifier and the remaining one fold is used for prediction. The predicted class labels for each fold are combined to form the outputs of the classification function.

Combination Functions: In ensemble learning, averaging and voting are the most commonly used combination methods for obtaining the final output of an ensemble [32]. EGP generates an ensemble using the classifiers trained from different or same classification algorithms so the ensemble may be a heterogeneous ensemble or a homogeneous ensemble. In this case, the voting method is more suitable than the averaging method for combination. Three different voting functions are developed in EGP. They are *Voting3, Voting5* and *Voting7*, which perform plurality voting for the input vectors (predicted labels). The three functions take three, five, and seven class labels as inputs, respectively, and return one class label. Each class label (discrete number) can be the output of a classification function or a combination function. This means that multiple combination functions can be evolved in a single EGP tree. This is the main reason that we only develop three voting functions requiring a small number of inputs.

3.4 Terminal Set

A new terminal set is developed in EGP. The types and the descriptions of all the terminals are listed in Table 3. The X_train terminal represents the training images. The Y_train terminal represents the class labels for X_train. The $\sigma, o_1, o_2, \theta, f, n_1$, and n_2 terminals are the parameters for different feature learning functions. They have predefined ranges, as listed in Table 3. Their values are randomly initialised according to their ranges and automatically evolved during the evolutionary learning process.

Table 3: Terminal set

Terminals	Type	Description
X_train	Array	The input image data with a number of grey-scale images (3D array, where the pixel values are in the range of $[0,\ 1]$)
Y_train	Array	The class labels for the training set. It is an array.
σ	Integer	The standard deviation of the Gaussian filter in the Gau and $GauD$ functions. Its range is in $[1,\ 3]$.
o_1, o_2	Integer	The orders of the Gaussian derivatives. They are randomly initialised from the range of $[0,\ 2]$
θ	Float	The orientation of the $Gabor$ function. It is in the range of $[0,\ 7\pi/8]$ with a step of $\pi/8$ [19]
f	Float	The frequency of the $Gabor$ function. It equals to $\frac{\frac{\pi}{2}}{\sqrt{2}^{\upsilon}}$, where υ is an integer in the range of $[0, 4]$ [19]
n_1, n_2	Float	The parameters for the *W-Add* and *W-Sub* functions. They are randomly generated from the range of $[0,\ 1)$

3.5 Test Process of EGP

After the evolutionary learning process of EGP, the best program is found and employed to predict class labels for an unseen data set, called the test set. The best program performs feature transformation for both the training set and the test set, then feed the training set and the class labels to the classification functions to train classifiers, and use the trained classifiers to predict class labels for the test set. In this process, the outputs of the classification function are actually the class labels of the test set. Each classification function performs general classification process using the training set and the test set. However, as the feature transformation and classification processes happen in a single GP tree, the inputs of each EGP tree are the training set, the test set and the class labels of the training set. To simplify the process, the training set and the test set are combined together to feed to the feature learning functions and then are used separately to feed the classification function to obtain the classifier and the prediction, respectively.

4 EXPERIMENT DESIGN

A number of experiments have been conducted to examine the effectiveness of EGP for image classification. This section designs the experiments.

4.1 Data Sets

Nine different data sets of varying difficulty are employed to conduct the experiments. The nine data sets include different types of image classification tasks, such as facial expression classification, texture classification, object classification, and scene classification. They are FEI_1 [28], FEI_2 [28], JAFFE [20], ORL [26], KTH [21], FS [10], MB [16], Rectangle [16], and Convex [16]. These data sets are binary or multi-class classification tasks. The FEI_1, FEI_2 and JAFFE data sets are facial expression classification tasks and they contain two or seven different facial expressions of different people. The ORL data set is to classify the face of 40 different people. The KTH data set is a texture classification task that to categorise images into 10 different groups. The FS data set contains 3859 natural scene images of 13 classes, including highway, street, coast, mountain, etc. The MB data set is a small subset of the famous digital recognition data set MNIST. The Rectangle data set is to recognise whether the width or the height of a rectangle in the image is bigger. The Convex data set is to classify images into the convex or non-convex sets. The details of the nine data sets are summarised in Table 4.

Table 4: Summary of the data sets

Data Sets	#Class	Image Size	Train Set	Test Set
FEI_1	2	60×40	150	50
FEI_2	2	60×40	150	50
ORL	40	50×55	240	160
JAFFE	7	55×55	140	73
KTH	10	50×50	480	330
FS	13	55×55	1300	2559
MB	10	28×28	12000	50000
Rectangle	2	28×28	1200	50000
Convex	2	28×28	8000	50000

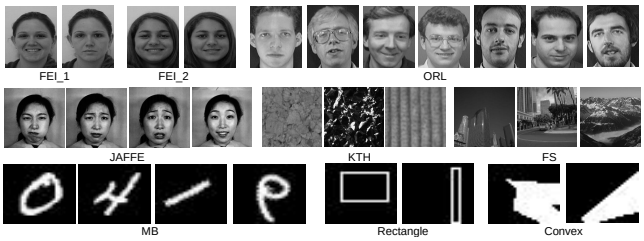

Figure 2: Example images of the nine data sets.

Each training set of FEI_1, FEI_2, ORL, JAFFE, KTH, and FS is constructed by randomly selected a number of images from each class. Then the test set contains the remaining images. For FEI_1 and FEI_2, 75 images per class are used for training and 25 images per class are for testing, respectively. For ORL, 6 images per class are used for training and the remaining 4 images per class are used for testing. For JAFFE, about 21 images per class form the training set and the remaining images form the test set. For the KTH data set, 48 images per class form the training set and the remaining images form the test set. Since the FS data set is large, 100 images per class are used for training and the remaining images are used for testing. MB, Rectangle and Convex are commonly used benchmark data sets and have been well-split into the training and test sets. The training and test sets are directly used in the experiments. Several example images from the nine data sets are shown in Figure 2.

4.2 Benchmark Methods

To comprehensively demonstrate the effectiveness of EGP, a large number of state-of-the-art methods are employed for comparisons. On the FEI_1, FEI_2, ORL, JAFFE, KTH, and FS data sets, 12 competitive methods are well chosen and implemented for comparisons. On the MB, Rectangle and Convex data sets, 14 recent methods that have the reported results are used for comparisons.

4.2.1 Benchmark methods on FEI_1, FEI_2, ORL, JAFFE, KTH, and FS. The 12 benchmark methods are six classification algorithms using raw pixels, four SVM methods using different features and two CNNs. The six classification algorithms are SVM, KNN, LR, RF, AdaBoost, and ERF, which are the same as that used in EGP. The six methods take the raw pixel values of each image as inputs and train classifiers for classification. The goal of comparisons between them and EGP is to show whether the feature learning process and the combination in EGP are effective. The four SVM methods using different features are uLBP+SVM, LBP+SVM, HOG+SVM, and SIFT+SVM, which represent traditional image classification methods [3]. In each method, the features are extracted using a commonly used feature extraction method: i.e., uniform LBP (uLBP), LBP, HOG, and SIFT [3]. The extracted features are feed to a linear SVM for classification. The goal of comparisons is to show whether EGP can beat the traditional methods using different features. The two CNN methods are CNNs with five layers (CNN-5) and eight layers (CNN-8), respectively. As CNNs are well-known for image classification, the aim of the comparisons is to show whether EGP can achieve better performance than state-of-the-art methods.

4.2.2 Benchmark methods on MB, Rectangle and Convex. As the three data sets are the commonly used benchmark data sets, different methods have been proposed to obtain a good classification performance. The best reported results of 14 existing methods are used for comparisons. These methods are SVM+RBF [16], SVM+Poly [16], SAE-3 [24], DAE-b-3 [24], CAE-2 [24], SPAE [31], RBM-3 [24], ScatNet-2 [5, 6], PCANet-2 (softmax) [6], LDANet-2 [6], NNet [16], SAA-3 [16], DBN-3 [16], and SPCN [18]. Note that in some methods such as SVM+RBF and SVM+Poly, the parameters are tuned using the training set and then the best model is used to obtain the classification accuracy of the test set.

4.3 Parameter Settings

The parameter settings for EGP are based on the commonly used settings for GP [12]. In EGP, the population size is 100 and the number of generations is 50. A small population size is employed to reduce the computational cost. The rates for elitism, crossover and mutation are 0.01, 0.8 and 0.19, respectively. The selection method is Tournament selection with size 7. The *ramped half-and-half* method is used for population initialisation. The maximum tree depth is 8 and the minimum tree depth is 2. In EGP, the type constraint has a higher priority than the depth constraint so that the tree depth may over 8.

The implementation of EGP is based on the DEAP (*Distributed Evolutionary Algorithm in Python*) [11] package, which is the most popular package for implementing GP. The classification methods in EGP and in the benchmark methods are implemented using the popular package *scikit-learn* [23]. To avoid the experimental bias, the experiments of EGP and the compared methods on each data set run 30 times independently. Note that on the MB, Rectangle and Convex data sets, the benchmark methods have the reported results so that there is no need to run these methods.

5 RESULTS AND DISCUSSIONS

This section discusses and compares the experimental results of the EGP method and the benchmark methods on the nine data sets. The discussions are presented according to the different types of benchmark methods employed on the nine data sets.

5.1 Results on FEI_1, FEI_2, ORL, JAFFE, KTH, and FS

The classification results of EGP and the 12 benchmark methods are listed in Table 5. Each block in Table 5 lists the maximum accuracy (Max), mean accuracy and standard deviation (Mean±St.dev) on each data set. The best accuracy on each data set is highlighted in bold. For each data set, a commonly used non-parametric test for multiple comparisons: *Friedman* test, is used for statistical test with a 5% significance level. To compare the proposed EGP method with a benchmark method, a post hoc test using the *Holm* method is employed to the determine the differences between group means. In Table 5, the symbols "+" and "−" indicate that EGP is significantly better and worse than the benchmark method in terms of the classification results. The final row of each block in the table summarises and highlights the overall results of the significance test.

Table 5: Classification results (%) of EGP and the compared methods on FEI_1, FEI_2, ORL, JAFFE, KTH, and FS

Methods	Max	Mean±St.dev	Max	Mean±St.dev
Data Set	**FEI_1**		**FEI_2**	
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	**100.0**	96.20±2.06	**100.0**	**98.07±1.70**
	Overall	**9+, 1−**	**Overall**	**12+**
Data Set	**ORL**		**JAFFE**	
SVM	94.38	94.38±0.00+	93.94	**91.06±0.73**−
KNN	94.38	94.38±0.00+	71.21	71.21±0.00+
LR	93.75	93.75±0.00+	89.39	89.39±0.00−
RF	93.12	92.33±0.63+	75.76	72.48±1.99+
AdaBoost	59.38	52.27±4.00+	53.03	47.93±2.68+
ERF	97.50	96.71±0.59	77.27	73.89±1.72+
uLBP+SVM	87.50	87.42±0.21+	31.82	26.87±3.30+
LBP+SVM	88.12	87.52±0.20+	33.33	28.84±2.05+
HOG+SVM	91.25	91.25±0.00+	81.82	80.30±0.40+
SIFT+SVM	93.75	93.75±0.00+	33.33	33.33±0.00+
CNN-5	96.88	95.29±1.06+	**95.45**	90.96±2.68−
CNN-8	95.00	93.04±1.09+	90.91	84.54±4.33
EGP	**99.38**	**97.44±1.26**	92.42	84.24±4.28
	Overall	**11+**	**Overall**	**8+, 3−**
Data Set	**KTH**		**FS**	
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**
	Overall	**10+, 2−**	**Overall**	**11+**

Compared with SVM, KNN, LR, RF, AdaBoost, and ERF, which are traditional classification methods using raw pixels, EGP achieves significantly better or similar performance in 33 comparisons out of the total 36 (6×6) comparisons. On the six data sets, EGP is significantly better than or similar to KNN, AdaBoost and ERF. Compared with the six methods, the overall learning process of EGP allows it to automatically extract informative features, select the best classification algorithm and find the suitable combination of classifiers for effective classification. Thus, the classification performance of EGP is better than each single classification algorithm using raw pixels in most comparisons.

Compared with uLBP+SVM, LBP+SVM, HOG+SVM, and SIFT+SVM, which use different features, EGP achieves significantly better or similar performance in 23 comparisons out of the 24 (4×6) comparisons. The four methods are traditional image classification methods. EGP is more accurate than the four methods on the six data sets. Table 5 shows that the performance of the four competitive methods varies with the data sets. For example, LBP+SVM achieves 82.71% mean accuracy on KTH but only achieves 28.84% mean accuracy on JAFFE. This confirms the difficulty of feature extraction since successfully extracting informative features requires domain knowledge and human intervention. EGP has a feature learning process to automatically extract a set of features so that it is more effective and adaptive than the four methods.

Compared with CNN-5 and CNN-8, the EGP method obtains significantly better or similar results in 10 comparisons out of the 12 (2×6) comparisons. Compared with CNN-5, EGP achieves better or similar performance on four data sets. Specifically, EGP improves the mean accuracy by 2% on ORL and by 13.04% on FS. The EGP methods is more accurate than CNN-8 as EGP achieves better performance than CNN-8 on five data sets and similar performance to CNN-8 on one data set. Among the three methods, EGP finds the maximum accuracy on five data sets except for JAFFE. The overall comparisons reveal the effectiveness of EGP on image classification.

Summary: Table 5 shows that EGP significantly outperforms the 12 competitive methods in 61 comparisons out of the 72 (12×6) comparisons. On five data sets except for JAFFE, EGP achieves the maximum accuracy of all the methods. On the FEI_2, ORL, and FS data sets, EGP achieves similar performance to or significantly better performance than the 12 competitive methods. On the JAFFE data set, EGP is better than eight methods except for SVM, LR, CNN-5, and CNN-8. The results show that raw pixels of JAFFE are effective for a linear classifier. While EGP may not generate such a classifier because of the combination process. On the KTH data set, EGP is significantly better than 10 methods except for LBP+SVM and CNN-5 in mean accuracy. KTH is a texture classification task so that the LBP features are effective for classification. But EGP finds the maximum accuracy on KTH. On the large data set, FS, with a number of natural images, the proposed EGP method is more accurate than any of the benchmark methods. In summary, EGP is an effective and promising approach for image classification.

5.2 Results on MB, Rectangle and Convex

The results of EGP and 14 state-of-the-art methods on the MB, Rectangle and Convex data sets are listed in Table 6. Since the results of the 14 competitive methods are the best results, the maximum accuracy of EGP are used for comparisons. To better show the overall performance of EGP, the mean accuracy and the standard deviation are also listed in Table 6. The "+" denotes that EGP is better than the compared method in terms of the maximum accuracy. The overall comparisons are summarised in the final row of Table 6. Note that the results of the three data sets are from the corresponding references and some results have not been reported so that there are 13 competitive methods on Rectangle and 8 methods on Convex.

Table 6: Classification accuracy (%) of EGP and the compared methods on the MB, Rectangle and Convex data sets

Methods	MB	Rectangle	Convex
SVM+RBF [16]	96.97+	97.85+	80.87+
SVM+Poly [16]	96.31+	97.85+	80.18+
SAE-3 [24]	96.54+	97.86+	–
DAE-b-3 [24]	97.16+	98.01+	–
CAE-2 [24]	97.52	98.79+	–
SPAE [31]	96.68+	–	–
RBM-3 [24]	96.89+	97.40+	–
ScatNet-2 [5, 6]	98.73	99.99	93.50+
PCANet-2(softmax) [6]	98.60	99.51+	95.81
LDANet-2 [6]	98.95	99.86+	92.78+
NNet [16]	95.31+	92.84+	67.75+
SAA-3 [16]	96.54+	97.59+	81.59+
DBN-3 [16]	96.89+	97.40+	81.37+
SPCN [18]	98.18	99.81+	–
EGP (Max)	97.19	99.91	93.97
EGP (Mean)	96.59	99.41	91.50
EGP (St. dev)	0.24	0.15	1.52
Overall	9+	12+	8+

On the MB data set, the proposed EGP method achieves 97.19% maximum accuracy, which is better than nine methods and worse than five methods. The maximum accuracy of all these methods is 98.95%, which is 1.76% higher than the maximum accuracy obtained by EGP. On the Rectangle data set, EGP is better than 12 competitive methods and is worse than 1 method. EGP achieves 99.91% accuracy, which is slightly less than the maximum accuracy 99.99% achieved by ScatNet-2. On the Convex data set, EGP achieves better results than eight methods and worse results than one method. The maximum accuracy is 95.81% achieved by PCANet-2(softmax) and the maximum accuracy achieved by EGP is 93.97%. From the results, it is obvious that EGP can achieve a good classification performance, which is very close to the best one reported.

Table 6 shows that the mean accuracy achieved by EGP is similar to the maximum accuracy and the standard deviation is very small. It can be concluded that EGP can achieve better or comparable performance to the state-of-the-art algorithms on the well-known benchmark data sets. On the three data sets, EGP is better than SVM+RBF, SVM+Poly, SAE-3, DAE-b-3, SPAE, RBM-3, NNet, SAA-3, and DBN-3 in terms of the maximum accuracy. EGP is worse than ScatNet-2, PCANet-2(softmax), LDANet-2, and SPCN in some comparisons, which may be because these competitive methods use a number of filters to extract invariant and discriminative features. In terms of the tasks, it seems that EGP is more effective for the binary classification tasks, Rectangle and Convex, than for the multi-class classification tasks, MB. This may due to the voting employed functions for combination in EGP. The voting functions

may be more effective for binary classification. The results and comparisons show the potential of EGP in image classification with a large number of training and testing instances.

6 FURTHER ANALYSIS

This section further analyses the programs evolved by EGP to reveal why it achieves good performance on image classification and whether the evolved ensembles can address accuracy and diversity simultaneously.

As EGP achieves the best results on the ORL data set of all the benchmark methods, an example program of EGP on this data set is visualised in Figure 3. This program achieves 99.375% classification accuracy on the test set of ORL. This program contains feature learning functions: *MaxP, Max, ReLU, LoG2, Med, LoG1*, and *FeaCon2*; classification functions: *ERF, SVM* and *RF*; and combination function: *Voting3*. The inputs of the program are X_data and Y_train, where X_data is actually the concatenations of the images of the training set and the test set and Y_train is the class labels of the training set. It is noticeable that the class labels of the test set are unseen to the overall test process, which is very important for avoiding the experimental bias.

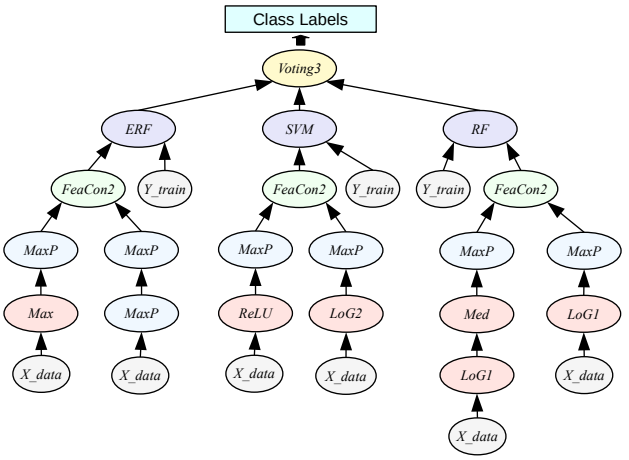

Figure 3: An example program of EGP on the ORL data set.

Splitting from the root node, the example program has three different child branches. The first child branch (the left one) uses ERF to train a classifier based on the features that described from the *FeaCon2* function. The trained ERF classifier is able to predict class labels for a test set. Therefore, the performance of this branch is tested on the test set. This branch achieves 96.875% accuracy on the test set. The second branch (the middle one) selects SVM to train a classifier using features that are different from the first branch as the child branches are different. The SVM classifier using the features produced by its child branches achieves 98.75% accuracy on the test set. The third branch (the right one) employs RF to train a classifier based on the features produced by the *FeaCon2* function with two different child branches. This branch achieves 94.375% accuracy on the test set of ORL.

By the further analysis on the example program, four keypoints of EGP can be summarised as follows:

(1) EGP automatically constructs ensembles of different classifiers. The performance of the ensembles is expected to be better than each single classifier. For example, the example program in Figure 3 achieves 99.375% accuracy, while each classifier in the example program only achieves 96.875%, 98.75% and 94.375% accuracy, respectively.
(2) EGP automatically selects the best classification function/method to build the ensemble for a target task. From Table 5, it is obvious that ERF is the best classifier on the ORL data set and AdaBoost is the least effective classifier on the ORL data set. Thus, the example program in Figure 3 has the ERF classification function and not the AdaBoost classification function.
(3) EGP automatically generates different high-level features to train each classification function, which further improves the diversity of the trained classifiers in the ensemble. From Figure 3, it is obvious that different classification functions have different inputs from their child nodes. These inputs are high-level features transformed from raw pixels using different feature learning functions such as filtering functions and pooling functions.
(4) EGP produces end-to-end solutions for image classification with a high generalisation ability. Each EGP solution takes the images and the class labels of the training set as inputs and produces the class labels for an unseen data set.

7 CONCLUSIONS AND FUTURE WORK

The goal of this study was to develop an automated ensemble learning approach using GP for image classification. The goal was successfully achieved by developing the new EGP approach with a novel program structure, a new function set and a new terminal set. The new approach was evaluated on nine different image classification data sets of varying difficulty and compared with a large number of commonly used methods. The experimental results show that EGP achieves better performance than most competitive methods. Further analysis shows that EGP is able to evolve good ensembles with high interpretability for image classification.

This is the first study using GP to automatically generate ensembles for image classification. The multiple tasks including feature learning, classification algorithm selection, classifier training and combination are integrated into a single EGP program. The results suggest that EGP automatically generates different high-level features to train each classification function, selects the best classification functions, and finds the optimal combination of the classifiers to predict the class labels with high accuracy.

In the future, the parameters of the classification algorithms will be developed as terminals to allow GP to automatically evolve and optimise their values. To achieve this and to improve the search efficiency, an effective local search operator may be needed for GP. It would be also interesting to use effective feature learning and classification algorithms such as CNNs to replace the current feature learning and classification processes of EGP to see whether the performance can be further improved.

REFERENCES

[1] Harith Al-Sahaf, Andy Song, Kourosh Neshatian, and Mengjie Zhang. 2012. Two-tier genetic programming: Towards raw pixel-based image classification. *Expert Systems with Applications* 39, 16 (2012), 12291–12301.

[2] Harith Al-Sahaf, Mengjie Zhang, Ausama Al-Sahaf, and Mark Johnston. 2017. 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.

[3] Ying Bi, Bing Xue, and Mengjie Zhang. 2018. A Gaussian Filter-Based Feature Learning Approach Using Genetic Programming to Image Classification. In *Australasian Joint Conference on Artificial Intelligence*. Springer, 251–257.

[4] Ying Bi, Bing Xue, and Mengjie Zhang. 2018. A survey on genetic programming to image analysis. *Journal of Zhengzhou University (Engineering Science)* 39, 06 (2018), 3–13.

[5] Joan Bruna and Stéphane Mallat. 2013. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1872–1886.

[6] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. 2015. PCANet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing* 24, 12 (2015), 5017–5032.

[7] Boyuan Chen, Harvey Wu, Warren Mo, Ishanu Chattopadhyay, and Hod Lipson. 2018. Autostacker: A compositional evolutionary learning system. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 402–409.

[8] Grant Dick, Caitlin A Owen, and Peter A Whigham. 2018. Evolving bagging ensembles using a spatially-structured niching method. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 418–425.

[9] Pedro G Espejo, Sebastián Ventura, and Francisco Herrera. 2010. 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.

[10] Li Fei-Fei and Pietro Perona. 2005. A bayesian hierarchical model for learning natural scene categories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 2. 524–531.

[11] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research* 13 (jul 2012), 2171–2175.

[12] Muhammad Iqbal, Bing Xue, Harith Al-Sahaf, and Mengjie Zhang. 2017. Cross-domain reuse of extracted knowledge in genetic programming for image classification. *IEEE Transactions on Evolutionary Computation* 21, 4 (2017), 569–587.

[13] Lukas Kammerer and Michael Affenzeller. 2018. Confidence-based ensemble modeling in medical data mining. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 163–164.

[14] Sašo Karakatič and Vili Podgorelec. 2018. Building boosted classification tree ensemble with genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 165–166.

[15] John R Koza. [n. d.]. *Genetic programming: on the programming of computers by means of natural selection*. MIT press, Cambridge.

[16] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. 2007. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th International Conference on Machine learning*. ACM, 473–480.

[17] Andrew Lensen, Harith Al-Sahaf, Mengjie Zhang, and Bing Xue. 2016. Genetic programming for region detection, feature extraction, feature construction and classification in image data. In *European Conference on Genetic Programming*. Springer, 51–67.

[18] Hao Li and Maoguo Gong. 2017. Self-paced convolutional neural networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 2110–2116.

[19] Chengjun Liu and Harry Wechsler. 2001. A Gabor feature classifier for face recognition. In *Eighth IEEE International Conference on Computer Vision*, Vol. 2. 270–275.

[20] Michael Lyons, Shigeru Akamatsu, Miyuki Kamachi, and Jiro Gyoba. 1998. Coding facial expressions with gabor wavelets. In *Third IEEE International Conference on Automatic Face and Gesture Recognition*. 200–205.

[21] P Mallikarjuna, Alireza Tavakoli Targhi, Mario Fritz, Eric Hayman, Barbara Caputo, and Jan-Olof Eklundh. 2006. The kth-tips2 database. *Computational Vision and Active Perception Laboratory, Stockholm, Sweden* (2006), 1–10.

[22] David J Montana. 1995. Strongly typed genetic programming. *Evolutionary Computation* 3, 2 (1995), 199–230.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[24] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. 2011. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Omnipress, 833–840.

[25] Mark E Roberts. 2003. The effectiveness of cost based subtree caching mechanisms in typed genetic programming for image segmentation. In *Workshops on Applications of Evolutionary Computation*. Springer, 444–454.

[26] Ferdinando S Samaria and Andy C Harter. 1994. Parameterisation of a stochastic model for human face identification. In *Proceedings of the Second IEEE Workshop on Applications of Computer Vision*. 138–142.

[27] Ling Shao, Li Liu, and Xuelong Li. 2014. Feature learning for image classification via multiobjective genetic programming. *IEEE Transactions on Neural Networks and Learning Systems* 25, 7 (2014), 1359–1371.

[28] Carlos Eduardo Thomaz. 2012. FEI face database. *online: http://fei.edu.br/~cet/facedatabase.html* (2012).

[29] Cao Truong Tran, Mengjie Zhang, Bing Xue, and Peter Andreae. 2018. Genetic Programming with Interval Functions and Ensemble Learning for Classification with Incomplete Data. In *Australasian Joint Conference on Artificial Intelligence*. Springer, 577–589.

[30] Steven Young, Tamer Abdou, and Ayse Bener. 2018. Deep Super Learner: A Deep Ensemble for Classification Problems. In *Advances in Artificial Intelligence: 31st Canadian Conference on Artificial Intelligence*. Springer, 84–95.

[31] Tingzhao Yu, Chaoxu Guo, Lingfeng Wang, Shiming Xiang, and Chunhong Pan. 2018. Self-paced autoencoder. *IEEE Signal Processing Letters* (2018).

[32] Zhi-Hua Zhou. 2012. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC.

[33] Zhi-Hua Zhou and Ji Feng. 2018. Deep forest. *National Science Review* 6, 1 (2018), 74–86.