Binary Image Classification: A Genetic Programming Approach to the Problem of Limited Training Instances

Harith Al-Sahaf harith.al-sahaf@ecs.vuw.ac.nz
School of Engineering and Computer Science,
Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand

Mengjie Zhang mengjie.zhang@ecs.vuw.ac.nz
School of Engineering and Computer Science,
Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand

Mark Johnston mark.johnston@msor.vuw.ac.nz
School of Mathematics, Statistics and Operations Research,
Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand

Abstract
In the Computer Vision and Pattern Recognition fields, image classification represents an important, yet difficult, task to perform. The remarkable ability of the human visual system, which relies on only one or a few instances to learn a completely new class or an object of a class, is a challenge to build effective computer models to replicate this ability. Recently, we have proposed two Genetic Programming (GP) based methods, One-shot GP and Compound-GP, that aim to evolve a program for the task of binary classification in images. The two methods are designed to use only one or a few instances per class to evolve the model. In this study, we investigate these two methods in terms of performance, robustness, and complexity of the evolved programs. Ten data sets that vary in difficulty have been used to evaluate these two methods. We also compare them with two other GP and six non-GP methods. The results show that One-shot GP and Compound-GP outperform or achieve comparable results to other competitor methods. Moreover, the features extracted by these two methods improve the performance of other classifiers with handcrafted features and those extracted by a recently developed GP-based method in most cases.

Keywords
Genetic Programming, Local Binary Patterns, One-shot Learning, Image Classification.

1 Introduction

Image classification represents a cornerstone in a broad range of domains such as *Computer Vision* and *Pattern Recognition*. Mainly, image classification aims at categorising images into different groups based on their contents. This task has received a lot of attention over the last few decades due to its importance and difficulty. Hence, a large number of methods have been proposed in the literature that aim at tackling different aspects of the image classification task. Some of those methods aim at addressing the accuracy problem (Lu and Weng, 2007), whilst others try to speed-up the training process (Fan et al., 2004). Moreover, some methods have been proposed to tackle different variations that can occur on instances of the same group such as illumination, rotation, and noise (Ojala et al., 2000; Guo et al., 2010).

The majority of those methods were not designed to build a model using a limited or small number of labelled instances. In other words, those methods were designed based on the assumption of having an abundant (to some extent) number of instances in order to build a model that can sufficiently generalise to unseen data. However, it is not always easy, feasible, or even possible to acquire a large number of labelled instances. Therefore, the limitation of having a few labelled instances to build or estimate the set of parameters of a given model needs to be addressed.

Jain and Chandrasekaran (1982) have partially investigated the problem of building or training a model using a few instances. Later, Raudys and Jain (1991) have studied this problem, and provided a list of recommendations and guidelines for practitioners: (1) the design parameters should be carefully selected when the number of instances is small, an example is the number of neighbours in the *k-nearest neighbour* (kNN) algorithm; (2) using powerful machines (e.g. a computer with a large amount of memory and a high processing speed), it is highly recommended to try a variety of feature extraction and selection methods and test their impact on different classification approaches; (3) in order to estimate the classification error, a sufficient number of instances is required in the test set; (4) the use of distinct instances in the training and test sets; (5) identifying and mitigating the problem of high error rate which can be due to the large number of features, the small number of training instances, the complexity of the model, inappropriate kernel width of nonparametric models, or the presence of the *outliers* in the case of parametric models; and (6) for feature selection, special attention should be given to selecting a good or the best subset of features.

Genetic Programming (GP) (Koza, 1992) is a well-known Evolutionary Computation (EC) algorithm inspired by the Darwinian principles of the natural selection and the concept of "survival of the fittest". GP aims at automatically exploring the solution space in order to evolve a computer program (solution) for a user-defined problem (Koza, 1992). GP has been widely used to tackle the problems of image classification (Smart and Zhang, 2003; Zhang and Smart, 2004; Zhang and Johnston, 2009; Downey and Zhang, 2009; Atkins et al., 2011; Abdulhamid et al., 2011; Al-Sahaf et al., 2014b,a), object detection (Zhang and Ciesielski, 1999; Zhang et al., 2004; Liddle et al., 2010), edge detection (Fu et al., 2011, 2012), and image descriptor (Perez and Olague, 2009; Hindmarsh et al., 2012; Olague and Trujillo, 2011; Albukhanajer et al., 2014). A large number of the GP-based methods operate in two stages where feature detection and extraction is performed first, and evolving a classifier takes place second. The process of designing a good set of features requires background knowledge from the domain. For example, discriminating between images of *benign* and *malignant* tumor cells needs to be carried out under the guidance of an expert in this domain. Other GP-based methods are capable of performing image classification using raw pixel values as input rather than pre-extracted features (Song and Ciesielski, 2004; Al-Sahaf et al., 2012b). However, neither the methods of the first category, nor those of the second category were designed to evolve a classifier using a few labelled instances.

Recently, we have proposed two GP-based methods for the task of binary classification in images namely *One-shot GP* (Al-Sahaf et al., 2013b) and *Compound-GP* (Al-Sahaf et al., 2013a). Those two methods use only one or a few labelled instances of each class to evolve computer programs that are capable of generalising to the unseen data.

To tackle the problem of having a limited number of labelled instances, some methods based on the concepts of *learning by knowledge transfer* have been proposed and this approach has been termed as *one-shot learning*. The aim is to use a large number of available instances of a related domain (the source domain) to the target domain, and

only one or a few labelled instances per class of the target domain, to build a model (Fei-Fei et al., 2006; Ishikawa and Mogi, 2011; Salakhutdinov et al., 2012).

Fei-Fei et al. (2006) proposed a Bayesian-based model for the problem of object recognition. In their system, *general knowledge* is extracted from previously learnt groups of objects using abundant instances, which is then used to form a prior probability density function. A posterior density is then produced by updating this knowledge given a small set of training instances in the target domain. The system was tested using a data set consisting of instances that fall into 101 different classes, and compared against two commonly used methods: *maximum likelihood* (ML), and *maximum a posteriori* (MAP). The results of their experiments show that their system significantly outperformed the other methods when the number of training instances is relatively small. Moreover, they investigated the effectiveness of using the knowledge extracted by their method on the performances of the two other methods. The results of the investigation suggested that a better performance has been achieved when both of the ML and MAP methods used the knowledge extracted by the new method.

Deformation matrices were used by Miller et al. (2000) for the problem of object classification. The system is trained using a large number of instances of a character (e.g. letter "A") and then try to make the system learn a different object (e.g. number "4") using a single (or few) instance. To achieve this goal, two learning scenarios were combined under a single framework: adopting the *transfer learning by model parameters* approach in order to use a reduced number of instances in the target domain; and frequently updating the system as more training images become available. The method is tested on binary image classification using two different training set sizes: 1000 instances; and only one instance. The results of their experiment show that, unlike other comparative methods, the performance did not drop significantly after reducing the number of training instances of the target domain from $1,000$ to 1.

Rodner and Denzler (2011) have modified the Randomised Decision Trees (RDF) (Liu et al., 2005; Dhurandhar and Dobra, 2008) learning algorithm in order to build a classifier using very few instances. Similar object classes are used to learn a prior distribution, which is then reused (transferred) to maximise a posteriori estimation of the model parameters. Evaluating the method on three data sets shows that a significant performance improvement have been achieved over RDF classifier of Geurts et al. (2006).

The task of detecting and extracting a set of good features is usually carried out by a domain-expert who in many cases is either hard to find or can be very costly. Feature descriptors or image descriptors play an essential role to detect and extract informative features such as shape, texture, scale or size, rotation, and colour (Szeliski, 2010).

The *local binary patterns* (LBP) (Ojala et al., 1996) are one of the well-known feature descriptors. This descriptor represents an essential part of the One-shot GP and Compound-GP methods. Therefore, LBP is discussed more in the next section.

Goals

Motivated by the remarkable ability of the human's brain to learn a new object from only one or a few examples, two new GP-based methods have been developed for the task of binary classification in image. We are precisely interested in addressing the following objectives.

- Comparing the performances of those two methods against both GP and non-GP methods using domain-specific handcrafted features;

- Investigating the capability of the evolved programs by those two methods to handle the rotation variation;
- Studying the goodness of the detected and extracted features by the evolved programs by those two methods via testing the impact of these features on the performance of different types of classifiers;
- Investigating the efficiency of the two methods by analysing the average time required to evolve a program, average time to evaluate an instance, and the program size; and
- Investigating the interpretability of the evolved programs by those two methods.

The rest of the paper is organised as follows. The concepts of LBP is reviewed in Section 2. Section 3 describes the One-shot GP and Compound-GP methods. The experiment settings, data sets, and baseline methods are discussed in Section 4. The results of the experiments are presented in Section 5. Section 6 provides an interpretation of some programs that were evolved by the One-shot GP and Compound-GP methods. Section 7 concludes this paper and recommends some future work directions.

2 Background

This section provides a brief introduction for the *local binary patterns*, which is a key component of the One-shot GP and Compound-GP methods.

2.1 Local Binary Patterns

Motivated by the method of Wang and He (1990), Ojala et al. (1994) proposed a simple dense feature descriptor called *local binary patterns* (LBP) that calculates a binary code for each pixel of an image based on the intensity value of the neighbouring pixels. Similar to any dense descriptor, LBP scans the image in a pixel-by-pixel basis, and a code of a specific length is generated based on the variation in the intensity level of equally spaced pixels located on the boundary of a circle centred on the current pixel. This operator is formally defined as:

$$LBP_{P,R}\left(x_c, y_c\right) = \sum_{p=0}^{P-1} s\left(v_p - v_c\right) 2^p \qquad s(a) = \begin{cases} 1 & \text{if } a \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$v_p = I\left(x_p, y_p\right) \tag{2}$$

$$x_p = x_c + R\cos\left(2\pi p/P\right) \tag{3}$$

$$y_p = y_c - R\sin\left(2\pi p/P\right) \tag{4}$$

where (x_c, y_c) are the coordinates of the central pixel of the current window, P is the number of neighbouring pixels, R is the radius of the circle (distance between the central pixel and any of the neighbouring pixels), and v_p and v_c are the intensity of the p^{th} neighbouring and central pixels respectively. The $s(\cdot)$ is a thresholding function that returns 1 if the argument is positive or zero, and 0 otherwise. Originally, LBP was designed to operate in a 3×3 window and denoted as $LBP_{8,1}$; therefore, the resulting binary code is of length 8 bits as demonstrated in Figure 1.

Despite the cost that results from scanning the image pixel-by-pixel, the LBP operator has been shown to be a powerful feature descriptor in a large number of studies in the *Computer Vision* and *Pattern Recognition* fields (Liu et al., 2012; Nguyen et al., 2013; Yang and Chen, 2013). As this operator attracted a large number of researchers

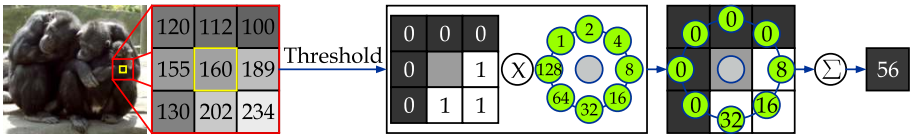


Figure 1: An example shows the required steps to generate the LBP code of a pixel.

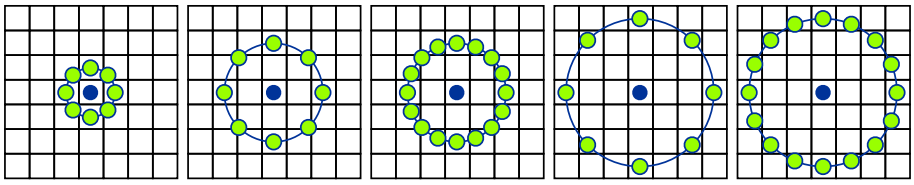


Figure 2: Demonstration of the circular local neighbouring pixel with different P (number of neighbours) and R (radius) settings. From left to right, $LBP_{8,1}$, $LBP_{8,2}$, $LBP_{16,2}$, $LBP_{8,3}$, and $LBP_{16,3}$.

over the last decade, it has received a lot of attention and numerous variants have been proposed (Pietikäinen et al., 2011), e.g., the size of the window, the method of thresholding the neighbouring pixels, and the length of the resulting code. Figure 2 shows five different LBP examples that vary in size (R) or number of neighbouring pixels (P).

2.2 Uniform Local Binary Patterns

The introduction of *uniform* LBP, denoted as $LBP^{u}_{P,R}$, represents a substantial restriction of the basic LBP operator (Ojala et al., 1996, 2002). The LBP code is called uniform when it has no more than two circular bitwise transforms, i.e., changing from 1 to 0 or from 0 to 1, which is determined using Equation (5). For example, patterns 11100111, 00111100, 11111110, and 00001110 are uniform; whereas patterns 10010111, 11100101, and 00011010 are not uniform.

$$LBP_{P,R}\left(x_c, y_c\right) \text{ is } \begin{cases} \text{uniform} & \text{if } \left(\left|s\left(v_{P-1}-v_c\right)-s\left(v_0-v_c\right)\right| + \sum_{p=1}^{P-1}\left|s\left(v_p-v_c\right)-s\left(v_{p-1}-v_c\right)\right|\right) \leq 2 \\ \text{non-uniform} & \text{otherwise} \end{cases} \quad (5)$$

Therefore, uniform patterns represent a subset of the overall space of an LBP code. It has been observed that over 85% of the LBP codes of an image are uniform (Ojala et al., 2002; Ahonen et al., 2006). There are two reasons to prefer uniform codes over basic LBPs: (1) omitting non-uniform codes can significantly reduce the number of possible patterns, which has large impact on reducing the length of the feature vector (more details in the following subsection); and (2) a variety of texture primitives can be detected using uniform codes such as line ends, edges, corners, and spots. Figure 3 presents some of these texture primitives, where 0 and 1 are these pixels having lower and higher values than the center pixel respectively.

2.3 Local Binary Patterns Histogram

The generated LBP codes of an image are used to form a frequency or spectrum histogram (Ojala et al., 2002) known as local binary patterns histogram ($LBP^{h}_{P,R}$). Each

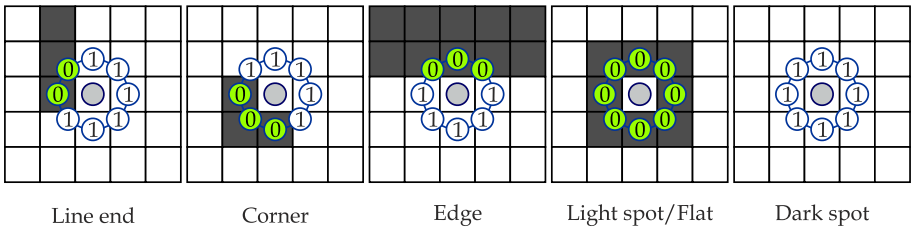


Figure 3: Some texture primitives that can be detected using uniform LBP codes.

bin of the $LBP^h_{P,R}$ accumulates the frequency of a single code. Therefore, the length of the $LBP^h_{P,R}$ depends on the total number of codes that can be represented, i.e., 2^P where P represents the number of neighbouring pixels (i.e. number of bits in the code). For example, if the LBP code is of length 8-bits and a unit radius ($LBP_{8,1}$), then 2^8 different codes can be represented starting at 0 = 00000000 and ending at 255 = 11111111; thus, the length of the $LBP^h_{8,1}$ in this case is 256 bins. $LBP^h_{P,R}$ is formally defined as:

$$LBP^h_{P,R}(b) = \sum_{i=1}^{M-1} \sum_{j=1}^{N-1} \delta\left(LBP_{P,R}\left(x_i, y_j\right), b\right) \quad b = 0, 1, \ldots, B-1 \tag{6}$$

where M and N are the width and height of the image respectively, (x_i, y_j) is the pixel at the (i, j) coordinates, and b and B are the b^{th} bin of the histogram and the total number of bins respectively. The $\delta(x, y)$ function, returns 1 if $x = y$, and 0 otherwise. The pixel at top-left corner of the image has the coordinates $(0, 0)$, whilst the pixel at the bottom-right corner of the image has the coordinates $(M-1, N-1)$. Moreover, the border pixels of the image are cropped to ensure that the sampling window does not exceed the image boundaries.

The $LBP^h_{P,R}$ is a feature vector of an image. The histogram can be either *global* (Chang et al., 2012) or a concatenation of *local* histograms (Ahonen et al., 2006; Tan et al., 2006; Pietikäinen et al., 2011). The former approach produces one histogram that consists of 2^P bins for each image as shown in Figure 4(a). The latter generates a number of sub-histograms that are computed from a specific (mostly non-overlapping) regions of the image, and concatenate those histograms to form the final feature vector. Therefore, the resulted feature vector of the second approach is of length $G \times 2^P$ where G is the number of regions as presented in Figure 4(b).

As mentioned earlier, omitting non-uniform codes has significant impact on the length of the resulted feature vector. The 8-bits length code produces a histogram consists of 256 bins; however, there will be only 58 bins if we consider only uniform codes (more details in (Ojala et al., 2000)). In order to also take non-uniform in to consideration, Ojala et al. (2002) have suggested to add an extra bin for all non-uniform codes. Therefore, the total number of bins of a 8-bit length code is 59, which means around 76% shorter histogram than the original case (256-bin).

3 Proposed Methods

The structure of the One-shot GP and Compound-GP methods[1], including the function and terminal sets, and the main components are explained in this section. This sec-

[1]The initial versions of the two methods have been presented in (Al-Sahaf et al., 2013b) and (Al-Sahaf et al., 2013a) with very limited investigations. This paper extend them with completely new results and thorough investigations.

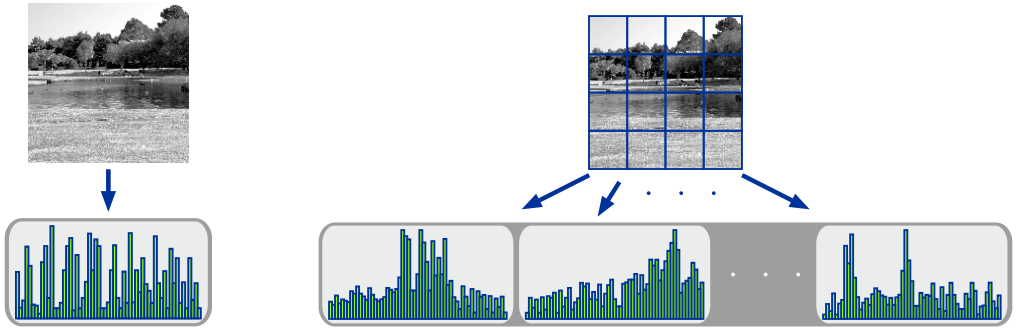


Figure 4: Examples of local binary patterns histograms generated from two different approaches: (a) one histogram per image; and (b) one histogram per region then concatenated.

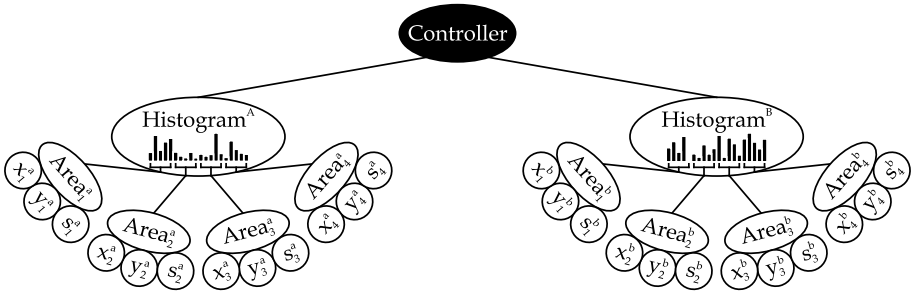


Figure 5: The general structure of a program evolved by the One-shot GP method.

tion also highlights the major similarities and differences between those two methods. In each method, the function and terminal sets are explained, followed by the fitness measure, training and testing/evaluation procedures.

3.1 One-shot GP

The program evolved by this method has a static structure (e.g. type and number of nodes); however, it is dynamic in terms of the position and size of the detected regions. Figure 5 shows a general structure of a program evolved by the One-shot GP method.

3.1.1 Function Set

In this method, the evolved program is made up of three non-terminal types of nodes of which each has its own restrictions and performs a distinct task. The first type is the *Controller* node, which only occurs at the root of the evolved program. Thus, each program has only one node of this type. The *Controller* node is responsible for predicting the class label of the instance being evaluated based on the results of its children. The second type is the *Histogram* node, which represents the type of the child nodes of *Controller*. The *Histogram* nodes are responsible for accumulating the results of its children to form a single $LBP^{h}_{P,R}$ feature vector. Each *Histogram* node corresponds to a single class, i.e., the number of this type of node depends on the total number of classes. The third and last type of the non-terminal node is the *Area* node. As the name suggests,

each *Area* node corresponds to a region of the instance being evaluated (the image), that is specified by the values of its children. The *Area* nodes represent the children of the *Histogram* nodes, and are responsible for performing the feature extraction task. Similar to the *Histogram* nodes, the number of the *Area* nodes is predefined; however, this number is not restricted by the number of the classes. In other words, this number is set experimentally and in our experiments, this number has been set to four.

3.1.2 Terminal Set

Similarly, the terminal set consists of three types of node which are the children of the *Area* nodes as shown in Figure 5. These nodes are *x-coordinate* (x for short), *y-coordinate* (y for short), and *window-size* ($size$ for short), which are all of type integer. The values of those nodes are randomly generated and represent a *square-shaped* window of size equals to $size$ and centered at a pixel of the coordinates (x, y). Therefore, this part of the evolved program tree is dynamic as the values of those nodes are randomly generated. The value of the x and y coordinates cannot be negative or greater than the image width and height respectively, i.e., $0 \leq x < M - 1$ and $0 \leq y < N - 1$ where M and N are, respectively, the width and height of the image. Moreover, the $size$ has been limited to be between 3 (i.e. 3×3 window) and $\min(M, N)/2$, where $\min(\cdot, \cdot)$ returns the minimum value of the arguments. The sampling windows are truncated if they exceed the boundaries of the image.

3.1.3 Fitness Measure

The fitness measure of the One-shot GP method aims at maximising the *between-class* distance, minimising the *within-class* distance, maximising the accuracy, and minimising the overlapping ratio of the detected regions to ensure the distinction of those regions, as shown in Equation (7).

$$Fitness_1 = \frac{D_{\mathrm{W}} + OVR}{D_{\mathrm{B}} + ACC_1} \tag{7}$$

$$ACC_1 = \frac{\Gamma}{|S_{\mathrm{N}}|} \tag{8}$$

$$OVR = \frac{1}{\sum_{i=1}^{G} \varphi_i} \sum_{i=1}^{G-1} \sum_{j=i+1}^{G} \bigcap (\varphi_i, \varphi_j) \tag{9}$$

$$D_{\mathrm{W}} = \sum_{i \in S_{\mathrm{N}}} \sum_{j \in S_{\mathrm{R}}} \eta(hist(i), hist(j)) \quad \{\forall i, j \mid class(i) = class(j)\} \tag{10}$$

$$D_{\mathrm{B}} = \sum_{i \in S_{\mathrm{N}}} \sum_{j \in S_{\mathrm{R}}} \eta(hist(i), hist(j)) \quad \{\forall i, j \mid class(i) \neq class(j)\} \tag{11}$$

Here ACC_1 is the performance (accuracy) of the wrapped kNN classifier, where Γ is the number of correctly classified instances and $|S_{\mathrm{N}}|$ is the total number of non-representative instances. OVR is the overlapping ratio between the detected regions, and D_{W} and D_{B} are, respectively, the within-class and between-class distances. Furthermore, S_{R} and S_{N} are the set of representative and non-representative (discussed below) instances respectively, G is the total number of regions, and the function $\bigcap(\cdot, \cdot)$ returns the overlapping or intersection (the number of shared pixels) between the arguments (i.e. regions). The $hist(i)$ and $class(i)$ functions are, respectively, returning the histogram and actual class label of the i^{th} instance. In order to prevent division by zero, the denominator of the fitness function is set to a very small value (0.0001) when both of the between-class distance and the accuracy of the wrapped classifier are zero.

The $\eta(\cdot,\cdot)$ function calculates the distance between two histograms, which is defined as shown in Equation (12).

$$\eta\left(\mathbf{A},\mathbf{Q}\right)=\frac{\Big(\mu\left(\mathbf{A}\right)-\mu\left(\mathbf{Q}\right)\Big)^{2}}{\sigma\left(\mathbf{A}\right)+\sigma\left(\mathbf{Q}\right)} \tag{12}$$

$$\mu\left(\mathbf{V}\right)=\frac{1}{|\mathbf{V}|}\sum_{v\in\mathbf{V}}v \tag{13}$$

$$\sigma\left(\mathbf{V}\right)=\sqrt{\frac{1}{|\mathbf{V}|-1}\sum_{v\in\mathbf{V}}\Big(v-\mu\left(\mathbf{V}\right)\Big)^{2}} \tag{14}$$

Here $\mathbf{A}$ and $\mathbf{Q}$ are the two histograms (e.g. sets of values), and $\mu(\cdot)$ and $\sigma(\cdot)$ return the mean and standard deviation of a histogram respectively.

The One-shot GP method uses some of the training set instances to be the basis for comparison and decision making, denoted as *representative instances* (S_{R}). Each representative instance is randomly selected from the training set; however, only **one** instance of each class is selected. The number of the representative instances is equal to the total number of classes. Each of the representative instances is assigned to one of the *Controller* node children (i.e. the *Histogram* nodes). The aim behind assigning an instance to a *Histogram* node is to make this node responsible for identifying instances of only one class (i.e. identifying instances having class label similar to that of the representative instance), which can be seen as a *one-verses-all* (Rifkin and Klautau, 2004) approach. The rest of the training set instances form the *non-representative set* (S_{N}) that is used to measure the performance of the evolved program during the training phase.

The training process consists of four steps. In the first step, the system is iterating over the list of the representative instances to extract the representative histograms, i.e., each *Histogram* node generates a single LBP histogram relying on the detected regions by the *Area* nodes and the assigned representative instance. The distance between the representative instances (D_{B}) is calculated using the generated representative histograms. In the second step, the overlapping ratio (OVR) of the detected regions is calculated using Equation (9). Third, the system uses the content of the non-representative set to measure the performance of the wrapped classifier (ACC_1), and calculates the within-class distance (D_{W}). To accomplish this third step, the system generates a set of histograms for each instance (one from each of the *Histogram* nodes), calculates the distance between each of the generated histograms and the corresponding representative histograms, and predicts a class label similar to that of the closest representative instance, i.e, *the-Nearest Neighbour* (1-NN) algorithm (Fix and Hodges, 1951). The fourth step, is to measure the goodness of the evolved program, which is achieved via passing the calculated distances (D_{B} and D_{W}), overlapping ratio (OVR), and accuracy (ACC_1) to the fitness function. It is important to notice that at least two instances of each class are required to evolve the model, where one of them is used as a representative instance and one (or more) is used to populate the non-representative set.

3.1.4 The Test/Evaluation Procedure

The testing phase is handled differently from the training phase. The main concern of the evaluation phase is to test the generalisation ability of the best evolved program on unseen (i.e. test set) data. Therefore, the proportion of the correctly classified instances to the total number of instances represents the final result of this phase.

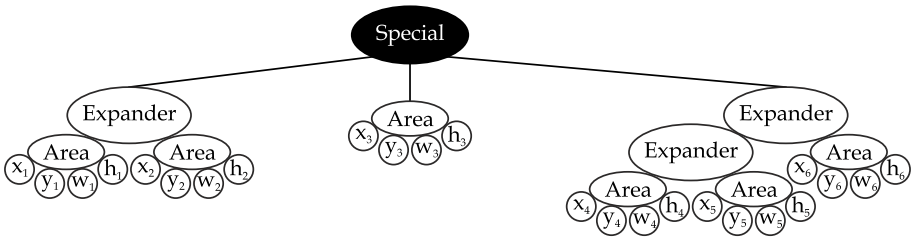


Figure 6: The general structure of a program evolved by the Compound-GP method.

In order to classify an instance, the system generates one histogram from each of the *Histogram* nodes based on the specified region by this node's *Area* nodes. Then the distances between the generated histograms and the corresponding representative histograms are calculated. The class label is predicted based on 1-NN. In other words, the class label of the closest representative histogram is assigned to the instance being evaluated.

3.2 Compound-GP

Figure 6 shows a general structure of an evolved program by Compound-GP.

3.2.1 Function Set

Similar to One-shot GP, the tree evolved by the Compound-GP method consists of three types of non-terminal nodes as presented in Figure 6. The first is the *Special* node that represents the root node similar to the *Controller* node of the One-shot GP method. The main roles of this node are to generate and save a number of *patch* objects based on the results of its children, and use the generated patch objects to train a number of wrapped classifiers. The patch object is made up of the mean and standard deviation values of a histogram generated from the instances being evaluated, along with the actual class label of the instance being evaluated. Unlike the *Controller* node of the One-shot GP method, the number of children of the *Special* node is predefined and has no relation with the number of the classes. Moreover, the children can be of different types such as *Expander* and *Area*. The evolved program by the One-shot GP method consists of only one wrapped classifier, whilst the evolved program by Compound-GP consists of four classifiers of two types for each child node (branch). The second type is the *Expander* node, which is responsible for allowing the system to evolve programs of different sizes by having chains of this node. An example is presented in Figure 6, where the *Special* node has three children that each has different tree size. The *Expander* node does not alter the results of its children, and only passes these results to its parent node. Hence, the appearance of this node in the program tree is optional. The third type of function is the *Area* node. Similar to the *Area* node of the One-shot GP method, this node resides near the leaves of the program tree. Each *Area* node represents a detected region of the image. However, the number of children of this node is different in One-shot GP and Compound-GP. In the former, this node has three children (x, y, and $size$); while it has four children (rectangle) in the case of the latter method.

3.2.2 Terminal Set

The terminal set consists of four integer-valued nodes: (1) *x-coordinate* (x for short); (2) *y-coordinate* (y for short); (3) *window-width* (w for short); and (4) *window-height* (h for short). Those four nodes represent a *rectangular* window of size $w \times h$ and centred at

a pixel with coordinates (x, y). The values of those four nodes are positive (including zero) and randomly selected from an associated predefined interval for each of them. The intervals of the x and y coordinates are $[0, M-1]$ and $[0, N-1]$ respectively; where M and N are the image width and height respectively. On the other hand, the values of w and h are selected from $[3, M]$ and $[3, N]$ respectively. Like One-shot GP, the sampling windows are truncated if they exceed the boundaries of the image.

3.2.3 Fitness Measure

The fitness function of Compoun-GP is composed of three main components as shown in Equation (15).

$$Fitness_2 = \frac{\textit{p-value} + OVR}{ACC_2} \tag{15}$$

$$ACC_2 = \sum_{i=1}^{C} \left(SVM_i^{\mathbf{S}} + SVM_i^{\mathbf{L}} \right) \tag{16}$$

Here *p-value* is the between groups difference that is calculated using the one-way *analysis of variance* (ANOVA), OVR is the overlapping ratio between the detected regions of the image using Equation (9), and ACC_2 is the total performance (accuracy) of the wrapped *support vector machines* (SVM) classifiers (Cortes and Vapnik, 1995) on the training set (more details below). C is the total number of children of the *Special* node, which is a fixed predefined value that was empirically set to 3 in our experiments. The $SVM_i^{\mathbf{S}}$ and $SVM_i^{\mathbf{L}}$ are the i^{th} SVM classifier that is trained using the list of single and multi-patch objects respectively.

The training process of the Compound-GP method is more complicated than that of the One-shot GP method. The process consists of eight steps as depicted in Figure 8. In the first step, the system iterates over the set of the detected regions (*Area* nodes) and calculates the ratio of overlapping (OVR) between those regions. Iterating over the instances of the training set and generating a number of LBP histograms for each instance (one from each $Area$ node), represents the second step. Each of those histograms is generated from an *Area* node based on the specified region by the values of the four children (x, y, w and h) of that node. In the third step, the statistics, i.e., mean and standard deviation, of each of those histograms are calculated, used along with the actual class label of the instance being evaluated to construct a patch object, and store this patch object in the *list of multiples* that is denoted as $\mathbf{L}$ as demonstrated in Figure 7. In the fourth step, those histograms (generated from different *Area* nodes) are concatenated with each other, the statistics of the resulted joined histogram are calculated, and a patch object is extracted using these statistics along with the actual class label of the instance being evaluated as shown in Figure 7. The patch object constructed in this fourth step is stored in the *list of singles* that is denoted as $\mathbf{S}$. Therefore, the result of the third and fourth steps is $G+1$ patch objects for each instance, where G is the total number of detected regions. Thus, the total number of patch objects in $\mathbf{L}$ is the total number of training instances $\times$ the number of Area nodes. Meanwhile, the $\mathbf{L}$ list consists of an equal number of patch objects to the number of instances in the training set (one object per instances). The use of both local and global features has been shown to have potential on improving the classifier performance (Lisin et al., 2005; Lim and Galoogahi, 2010). Therefore, in this study features generated from each of the detected regions (i.e. the $\mathbf{L}$ list) as well as those resulted from the combination of multiple regions (i.e. the $\mathbf{S}$ list) are used.

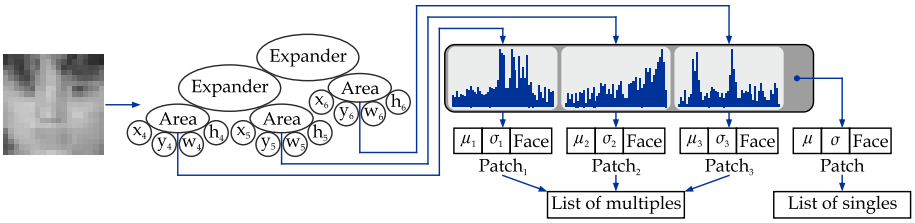


Figure 7: How the detected regions of an image are used to extract three *patch* objects and add each of them to the *list of multiples*, and to extract only one *patch* object from the concatenated histograms and add it to the *list of singles*.

It is important to notice that the patch objects that were generated from each of the *Special* node children are grouped together to form the **L** and **S** lists. In other word, each of the *Special* node children has one of each of those two lists (C children $\times$ 2 lists). Moreover, each of the *Special* node children has four classifiers of two types: (1) two SVM classifiers; and (2) two kNN (k=1) classifiers. Only the SVM classifiers are used during the training process, because the system is designed to evolve a program even when there is only one instance per class. The first SVM classifier is trained using the patch objects of the **S** list and denoted as $SVM^{\mathbf{S}}$. The second SVM classifier is trained using the objects of the **L** list and denoted as $SVM^{\mathbf{L}}$. Training those SVM classifiers represents the fifth step of the training procedure. In the sixth step, the performance of those SVM classifiers (ACC_2) that were trained using the lists of patches of the corresponding branch is measured using the patch objects resulted from other branches. In the seventh step, the system measures the distinction of the detected regions (*p-value*) via using the ANOVA test on the content (i.e. the mean and standard deviation values) of all **L** lists. The last step of the training process is to calculate the fitness function value using the results of the first, sixth, and seventh steps.

In summary, the results of the training phase are two trained SVM classifiers and two lists of patches for each sub-tree (branch) of the *Special* node. The lists of patches will be used as the knowledge base for the two kNN classifiers of each sub-tree. Figure 8 demonstrates the process of calculating the fitness function components of an evolved program during the training phase.

3.2.4 The Test/Evaluation Procedure

The test phase is quite different and less complicated than the training phase. In order to test an instance, the system fed this instance to each sub-tree of the *Special* node and performs the following steps. First, the system generates an LBP histogram from each of the detect regions and uses it to construct a patch object. Second, the constructed patch objects are then fed to SVM and kNN classifiers that were trained using the **L** list and the predicted class label of each of them is reported. Third, the generated histograms in the first step are then concatenated and used to produce a single patch object, which is then fed to SVM and kNN classifiers that were trained using the **S** list and the predicted class labels are also reported.

After repeating the above three steps for each sub-tree of the *Special* node, the system will report $C \times 4$ class labels, where C represents the total number of children of the *Special* node, and 4 as there are four classifiers (two SVM and two kNN) associated with each child node. Via adopting the *voting* approach, the system will predict the class label that has the majority of the votes. However, having an even number of clas-

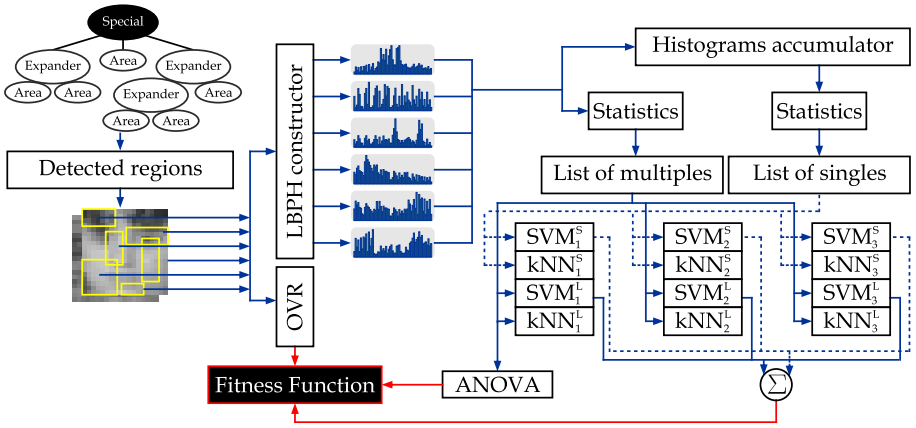


Figure 8: The process of calculating the fitness value of an evolved program by the Compound-GP method during the training phase.

sifiers may result in a situation where the votes are equally divided between the two classes. Hence, such a situation has been handled by relying on the closest instances (e.g. the smallest distance measured by all kNN classifiers).

4 Experimental Settings

In order to test the performance of the One-shot GP and Compound-GP methods, a series of experiments have been conducted that aim at investigating different aspects. Generally, those experiments can be divided into four groups: (1) comparing the performance of the One-shot GP and Compound-GP methods with the performance of the baseline methods; (2) checking the impact of the features extracted by each of the two methods (One-shot GP and Compound-GP) on the performance of a number of classifiers compared to the use of handcrafted and Two-tier GP extracted features; (3) investigating the ability of the two methods to handle the rotation variation; and (4) investigating the ability of the two methods to handle the scale variation.

This section provides more in-depth explanation of the above experiments. Moreover, the properties of the data sets that were used, parameter settings, baseline methods, and software characteristics are also discussed in this section.

4.1 Data Sets

The performance of the One-shot GP and Compound-GP methods has been evaluated using different types of data sets. The data sets can be categorised into four groups where three of them are textures and the fourth is object classification. Each data set in each group consists of only two classes (binary classification) of gray-scale images. The following subsections provide detailed discussion of each of those four groups.

4.1.1 Group A

The instances of the first group were taken from the *Kylberg Texture* data set (Kylberg, 2011). The *Kylberg Texture* data set consists of 28 classes as shown in Figure 9. This data set comes in two flavours: (1) without rotation; and (2) with rotation. The instances of both groups are gray-scale each of size 576×576 pixels.

Each class of the without rotation group consists of 160 unique instances. We have

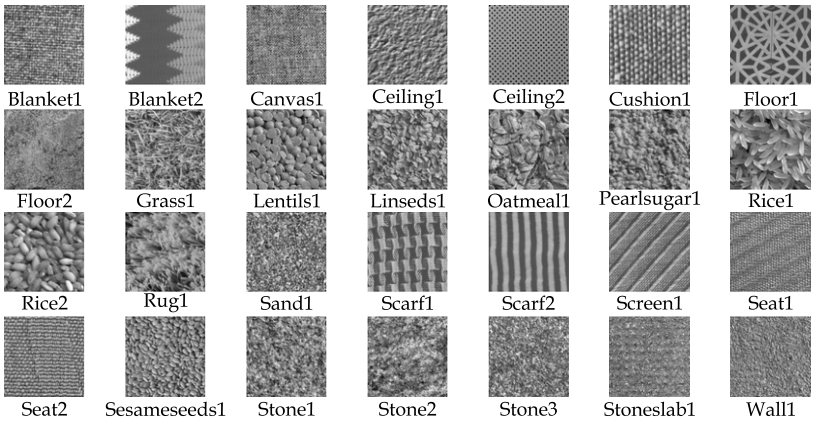


Figure 9: One sample of each of the 28 classes of the *Kylberg Texture* data set.

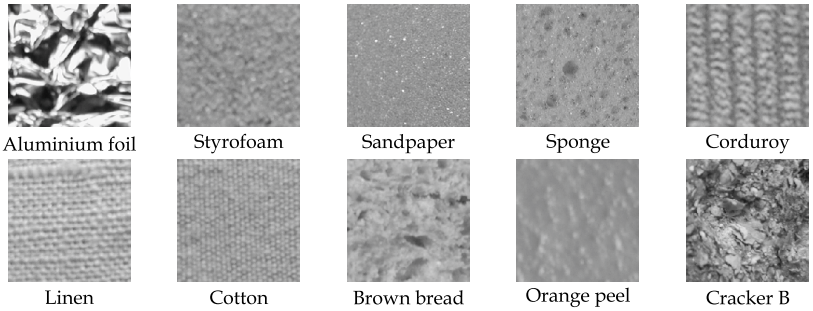


Figure 10: One sample of each class of the *KTH-TIPS* data set.

selected eight visually-close classes of this group (textures without rotation) to form four data sets for binary classification. *Textures-1* is the first set, made up of the *stones-lab1* and *wall1* classes. *Textures-2* is the second set, made up of the *rice2* and *sesameseeds1* classes. The *blanket1* and *canvas1* classes are selected to form the third set *Textures-3*. The fourth set is *Textures-4*, consists of the *linseds1* and *pearlsugar1* classes.

4.1.2 Group B

The data set of this group was taken from the *KTH-Textures under varying Illumination, Pose, and Scale* (KTH-TIPS) image data set (Bratko et al., 2006). The KTH-TIPS image data set consists of ten classes as depicted in Figure 10, where each consists of 81 instances of size 200×200 pixels.

We have selected only two classes that are visually-close to form the *Textures-5* data set in our experiments. This data set is more challenging than other texture data sets that were used in this study due to the scale variation of its instances which impose more difficulties on the model to handle.

4.1.3 Group C

The instances of the third group were also taken from the *Kylberg Texture* data set. However, the aim of this group's data sets is to test whether the One-shot GP and Compound-GP methods are invariant to rotation or not. Therefore, the classes of this group data sets were drawn from the *textures with rotation* classes of *Kylberg Texture*.

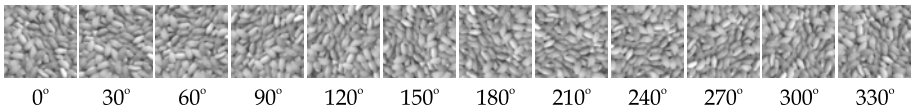


Figure 11: One sample from the *rice2* class of the *Kylberg Texture* data set rotated about the center in 12 different angles taken from the *textures with rotation* group.

Each class of the *textures with rotation* group, consists of $1,920$ instances that are the same 160 instances of the without rotation group rotated about the center in 12 different angles ($160 \times 12 = 1,920$). The instances are rotated between 0 and 330 degrees increments by 30 degrees. An example of an instance of the *rice2* class rotated at different angles is presented in Figure 11. For comparison purposes, the same images that were selected from the without rotation group of the *Kylberg Texture* data set to form *Textures-1*, *Textures-2*, *Textures-3*, and *Textures-4*, have been selected from the with rotation group to form *Textures-6*, *Textures-7*, *Textures-8*, and *Textures-9* data sets respectively.

4.1.4 Group D

Similar to *Group B*, this group consists of only one data set which is the *CBCL Faces* data set (Heisele et al., 2000). Unlike the data sets of the previous groups, Faces is not texture-based and the task is to discriminate between face and non-face instances. Therefore, the use of this data set will allow us to test the ability of the One-shot GP and Compound-GP methods to handle a different task other than texture classification.

Each instance of the *CBCL Faces* data set is of size 19×19 pixels, where the face instances were hand-aligned to be relatively in the center of the example. Originally, there are $2,429$ face and $4,548$ non-face instances in the training set, and 472 face and $23,573$ non-face instances in the test set. Clearly, the number of instances of the two classes is highly unbalanced; hence, we did not use the original division of the data and instead a nearly equal number of instances of each class has been selected. In other words, $6,000$ instances in total have been selected from the two classes that are $2,901$ faces and $3,099$ non-faces to form the *Faces* data set in our experiments.

4.2 Data Sets Preparation

Applying different image processing techniques as a preprocessing step can significantly affect the performance of the used model. Some of the well-known operations are the histogram equalisation, quantisation, and convolution operators such as Gaussian blurring. However, different data sets require different processing schemes, and can be even harder if the instances of the same data set were captured in an uncontrolled environment.

The total number of instances of each class has been divided equally between the training and test sets. Moreover, the original instances were used without applying any image preprocessing in order to investigate the ability of the One-shot GP and Compound-GP methods to handle the shifting of pixel values. However, each instance of the *Kylberg Texture* database was re-sampled (resized) to 57×57 pixels in our experiments in order to reduce the computational costs.

Apart from the Conventional-GP method, all other GP methods operate on raw pixel values, and automatically detect and extract features. However, Conventional-GP and all the non-GP methods require a prior step to detect and extract feature vectors, which needs to be handled by a domain-expert in order to design highly discriminative features. Thus, the features of all texture-based data sets were extracted from ten

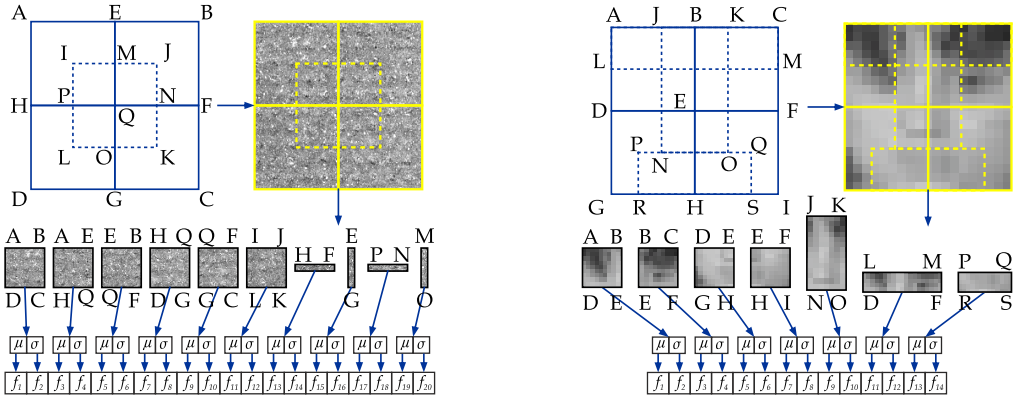


Figure 12: The regions of extracted features of the (a) texture, and (b) faces data sets.

regions (Zhang et al., 2003) as shown in Figure 12(a). The *mean* and *standard deviation* statistics of each of the four quadrants (AEQH, EBFQ, HQGD, and QFCG), the central quarter (IJKL), the horizontal lines (HF, and PN), the vertical lines (EG, and MO), and the entire image (ABCD) have been calculated to form a feature vector that consists of 20 values. Similarly, the *mean* and *standard deviation* of the eyes (LMFD), nose (JKON), mouth (PQSR), and the four quadrants (ABED, BCFE, DEHG, and EFIH) regions have been calculated for each of the *Faces* data set instances to construct a feature vector that consists of 14 values. The regions of the faces data set were designed based on the work of Bhowan et al. (2009) as shown in Figure 12(b).

4.3 Baseline Methods

In order to check the effectiveness of the proposed methods, a number of GP and non-GP methods have been evaluated on the used data sets.

4.3.1 GP-based methods

The One-shot GP and Compound-GP methods have been compared with two GP-based methods. The *Conventional-GP* is the first method that has a function set made up of the four arithmetic operators $+, -, \times$, and $\div$. Those operators have their regular meaning apart from the $\div$ operator, which is protected that returns zero if the second variable (dominator) is zero. The terminal set, on the other hand, consists of $rand$ which is a randomly generated double-precision float value between -1 and $+1$ (inclusive), and F_i where i represents the index of the feature. As mentioned earlier, this method relies on domain-specific handcrafted features. The fitness measure of Conventional-GP in both of the training and test phases is the *accuracy* that is formally defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{17}$$

where TP, TN, FP, and FN are the number of true positives, true negatives, false positives, and false negatives respectively.

The *Two-tier GP* (Al-Sahaf et al., 2012b) method is made up of two tiers that each has been specified to perform a specific task. The *upper* part of the program's tree

(first tier) represents the *classification* part that consists of the four arithmetic operators (similar to Conventional-GP) and *If-Then-Else*. Unlike other operators, the *If-Then-Else* operation has three children, which returns the value of the second child if the value of the first child is negative, and the value of the third child otherwise. The second tier, which occupies the lower part of the evolved program's tree, represents the *aggregation* part that consists of special nodes to perform the feature extraction task. Unlike the Conventional-GP method, the Two-tier GP method operates directly on the image raw pixel values and no preprocessing is required.

In both of the Conventional-GP and Two-tier GP methods, the program output space is divided into two parts that each corresponds to a group of instances of the same class label: negative; and positive including the zero value. Therefore, an instance is classified as belonging to a group (e.g. foreground) if the value of the root node is negative; otherwise, it is classified as belonging to another group (e.g. background).

4.3.2 Non-GP methods

The developed methods are compared to six non-GP methods.

- Naïve Bayes (NB) (John and Langley, 1995): a probabilistic classifier that designed based on the use of Bayes' theorem.
- Support Vector Machines (SVM) (Cortes and Vapnik, 1995): trained using the *sequential minimal optimisation* (SOM) algorithm that was invented by John Platt (Platt, 1999).
- Naïve Bayes/Decision Trees (NBTree) (Kohavi, 1996): a hybridised method that combines Decision Trees with Naive Bayes classifier. The latter represents the leaves of the tree.
- Adaptive Boosting M1 (AdaBoostM1) (Freund and Schapire, 1996): an adaptive method that relies on misclassified instances by previous classifiers to improve the subsequent ones.
- K* (KStar) (Cleary and Trigg, 1995): an instance-based method that uses an entropy-based distance measure to predict the class label of an instance based upon the similar instances of the training set.
- Non-Nested generalized (NNge) (Martin, 1995): an instance-based classifier that uses a non-nested exemplar, which works in a similar way to the nearest-neighbour method.

4.4 Evaluation

To evaluate each of the four GP-based methods (Conventional-GP, Two-tier GP, One-shot GP and Compound-GP), a specified number of instances of each class are randomly selected from the total number of instances available in the training set. The best evolved program using the selected instances at the end of the run is tested against the unseen data (test set). Due to the stochastic nature of GP, the same process has been independently executed 50 times using different *starting point* (seed value) each time, and only the average performance is reported. The non-GP methods on the other hand, have been trained using the exact same instances, but without repeating the execution multiple times (deterministic methods).

Moreover, due to the impact of the selected instances on the performance of the evolved classifier, the 50 runs of each GP-based method and the single execution of the non-GP methods, have been repeated 20 times using different *instances* each time.

Therefore, the total number of independent runs on a single data set is [4 GP methods $\times$ 50 runs $\times$ 20 repetitions] + [6 non-GP methods $\times$ 1 run $\times$ 20 repetitions] = $4,120$. The standard deviation over the 20 repetitions (i.e. 20 average performances) is reported.

The above procedure has further been repeated ten times using training sets of different *sizes* that starts from the minimum number (only one instance per class) and increases by one instance every time (the largest is ten instances per class). However, the smallest number of instances per class that can be used to evolve a program by the One-shot GP method is two (one representative and one or more non-representative); therefore, in the case of this method the above process has been repeated nine times instead of ten. This ten repetitions makes the total number of executions on each data set is [[$1,000$ runs $\times$ 3 methods $\times$ 10 sets] + [$1,000$ runs $\times$ 1 method $\times$ 9 sets] GP methods + [120 run $\times$ 10 sets] non-GP methods] = $40,200$.

4.5 Feature Extraction

The two new GP methods have their own mechanisms to perform feature extraction; hence, these methods can also be used for automatic feature extraction. The impact of the detected and extracted features by each of the One-shot GP and Compound-GP methods on the performance of six different classifiers is also investigated in this study. The classifiers are the non-GP baseline methods that were discussed in Section 4.3. To measure the goodness of the extracted features by each of those two methods (One-shot GP and Compound-GP), the handcrafted features (as discussed in Section 4.2) and those extracted by Two-tier GP (as discussed in (Al-Sahaf et al., 2012a)) are used.

In the case of the One-shot GP method, the *mean* and *standard deviation* values are calculated for each LBP histogram resulted from each *Histogram* node. As discussed in Section 3.1, each evolved program has two *Histogram* nodes; therefore, the feature vector of each instance consists of four values.

Similarly, the features extracted by the Compound-GP method represents the calculated statistics (mean and standard deviation) of the resulted LBP histograms. However, the program evolved by Compound-GP generates a number of histograms for each instance. Moreover, some histograms are generated from the *Area* nodes; whilst others result from the concatenation of histograms as described in Section 3.2. In other words, the patch objects of the $\mathbf{S}$ (list of singles) and $\mathbf{L}$ (list of multiples) are used as the extracted features. As it is hard to guess which of the two lists is better than the other, we decided to use each of them in isolation of the other, as well as the combination of the two.

4.6 Parameter Settings

In this study, four GP methods have been used. For comparison purposes, the settings of those methods' parameters are kept identical in all of the experiments as listed in Table 1. It is very important to notice that some of the parameters are not applicable in the case of One-shot GP due to the restrictions of the evolved program by this method.

4.7 Implementation

The GP-based methods have been implemented using the platform provided by the *Evolutionary Computation Java-based* (ECJ) package (Luke, 2013). The implementation of the non-GP methods on the other hand, have been taken from the *Waikato Environment for Knowledge Analysis* (WEKA) package (Hall et al., 2009).

Table 1: The GP Parameters of all experiments

Parameter	Value	Conventional-GP	Two-tier GP	One-shot GP	Compound-GP
Crossover Rate	0.80	✓	✓	✓	✓
Mutation Rate	0.19	✓	✓	✓	✓
Elitism Rate	0.01	✓	✓	✓	✓
Population size	200	✓	✓	✓	✓
Generations	20	✓	✓	✓	✓
Tree depth	2-10	✓	✓	✗	✓
Selection Type	Tournament	✓	✓	✓	✓
Tournament Size	7	✓	✓	✓	✓
Initial Population	Ramped half-and-half	✓	✓	✗	✓

5 Results and Discussions

The results of the experiments are reported and discussed in this section. This section is divided into four subsections that describe different aspects of the obtained results. The performances of the One-shot GP, Compound-GP, and all the baseline methods in terms of accuracy are presented in the first subsection. The goodness of the features extracted by each of the One-shot GP and Compound-GP methods are compared to the handcrafted areas and those extracted by the Two-tier GP in the second subsection. The third subsection, provides discussion on both of the training and test time of the four GP-based methods that are used in this study. Meanwhile, the fourth subsection shows the complexity of the evolved program in terms of the average size per generation of the four GP-based methods.

5.1 Accuracy

In order to check if the average performances of One-shot GP and Compound-GP are statistically significant compared to the performance of each of the baseline methods, a *Wilcoxon signed-ranks* test (Wilcoxon, 1945; Demšar, 2006) is used in this study. The methods are compared in pairs, and the *significance level* of the test was set to 5%. The $\downarrow$ and $\uparrow$ symbols indicate that the performance of the One-shot GP method compared to that of the other method is significantly worse and better respectively. Meanwhile, the $\Downarrow$ and $\Uparrow$ symbols appear if the performance of the Compound-GP method compared to that of the other method is significantly worse and better respectively. The method with the highest performance amongst all other comparative methods has its result made bold. However, in the case of having more than one method that have achieved 100% accuracy, we did not made any of those methods bold.

The results of this experiment are presented in tables in this section. Each table aggregates the results of data sets of one group. The first column of each table shows the name of the data set, and the total number of instances per class that were used in the training set are listed in the second column. Horizontally, each table is divided into two parts: the first lists the results of the non-GP methods; whilst the results of the GP-based methods are listed under the second part. The result of the One-shot GP method is missing from the first row of all tables (indicated by N/A symbol), as the minimum number of instances required by this method is two of each class.

5.1.1 Group A Data Sets

The results of the data sets of this group are presented in Table 2, which consists of the Textures-1, Textures-2, Textures-3, and Textures-4 data sets.

The first block of Table 2 shows the results of the methods on the Textures-1 data set. Compound-GP has scored second best performance on this data set after the Ad-

aBoostM1 method. The One-shot GP method is also showing good performance and scores third when the number of instances is less than six.

The results of Textures-2 data set are presented in the second block of Table 2. This data set represents one of the easiest; that most of the methods, apart from Conventional-GP and Two-tier GP, have scored above 99% when there were four or more instances of each class in the training set. However, the use of only one instance was enough to achieve 100.0% accuracy by the Compound-GP method, and 99.9% using two instances in the case of One-shot GP.

For Textures-3, all of the methods, apart from Two-tier GP, have achieved reasonably good accuracy above 80% when there were three or more instances of each class in the training set as shown in the third block of Table 2. Moreover, the Compound-GP method shows either the highest or in the top three ranked performance amongst other methods. Although the One-shot GP shows the second lowest performance amongst the comparative methods on this data set, the result shows that this method has achieved on average 79.3% accuracy using only two instances per class.

The results on Textures-4 data set are presented in the last block of Table 2. Apart from KStar, Conventional-GP, and Two-tier GP, all other methods have achieved on average over 80% accuracy. Moreover, the Compound-GP method has significantly outperformed all other methods on this data set. The Compound-GP method has achieved 95.1% accuracy using only one instance per class, which is significantly better than the highest achieved results by other methods even when there were 10 instances per class in the training set (NNge = 90.8%).

5.1.2 Group B Data Set

The results of the Textures-5 data set are presented in Table 3. This data set represents a more challenging task compared to all other texture-based data sets due to the variation in illumination, scale, and pose of its instances. The results show that One-shot GP and Compound-GP have significantly outperformed all other methods on this data set. Moreover, these two methods have achieved on average over 90% accuracy even when there is two instances in the training set. This show that the evolved programs by the two new methods are invariant (to some extent) to those variations.

5.1.3 Group C Data Sets

Table 4 presents the results on data sets of the third group, which consists of Textures-6, Textures-7, Textures-8, and Textures-9 data sets.

The results of Textures-6 data set (which represents the rotated version of Textures-1) show that Compound-GP and One-shot GP have respectively scored the first and third best performance as shown in the first block of Table 4. The two new methods have significantly outperformed all other methods, apart from AdaBoostM1 compared to One-shot GP, on this data set. Moreover, apart from KStar, all comparative methods show a significant drop in their performances compared to Textures-1; whilst the two new methods show nearly consistent performance on the two data sets.

Apart from AdaBoostM1, most of the methods have achieved similar accuracy on Textures-7 to Textures-2 as presented in the second block of Table 4. The One-shot GP and Compound-GP methods have achieved 100% accuracy even when the number of available instances is relatively small (less than three instances per class).

Similarly, the comparative methods show nearly consistent or slightly dropped performance on the rotated version Textures-8 of the Textures-3 data set as presented in the third block of Table 4.

The last block of Table 4 shows the results of Textures-9, which represents the ro-

Table 2: Results of the Group A data sets.

		Non-GP Methods						GP-based Methods			
	Size	AdaBoostM1	KStar	NB	NBTree	NNge	SVM	Conven.	Two-tier	One-shot	Compound
Textures-1	1	**99.4 ± 0.5** ⇓	61.1 ± 7.1 ⇑	57.0 ± 5.8 ⇑	63.7 ± 8.3 ⇑	52.5 ± 6.9 ⇑	52.5 ± 6.9 ⇑	54.1 ± 3.5 ⇑	51.0 ± 0.8 ⇑	N/A	96.5 ± 1.7
	2	**99.4 ± 0.3**↓⇓	67.3 ± 12.6↑⇑	68.7 ± 15.1↑⇑	61.3 ± 12.2↑⇑	63.4 ± 11.5↑⇑	62.6 ± 11.4↑⇑	55.4 ± 3.8↑⇑	51.9 ± 1.1↑⇑	90.7 ± 1.3	97.1 ± 1.0
	3	**99.4 ± 0.3**↓⇓	71.1 ± 12.3↑⇑	75.8 ± 17.7↑⇑	68.0 ± 18.2↑⇑	74.0 ± 15.8↑⇑	73.0 ± 14.4↑⇑	57.1 ± 3.7↑⇑	52.4 ± 1.1↑⇑	90.6 ± 1.1	97.5 ± 0.6
	4	**99.4 ± 0.3**↓⇓	70.0 ± 11.0↑⇑	82.2 ± 17.6↑⇑	79.5 ± 19.6↑⇑	83.3 ± 16.3↑⇑	80.0 ± 12.7↑⇑	58.3 ± 3.9↑⇑	53.2 ± 1.4↑⇑	91.0 ± 0.8	97.6 ± 0.4
	5	**99.4 ± 0.3**↓⇓	70.0 ± 8.2 ↑⇑	86.5 ± 15.8↑⇑	77.7 ± 17.4↑⇑	89.3 ± 14.9↑⇑	86.4 ± 11.8↑⇑	59.1 ± 3.8↑⇑	53.4 ± 1.2↑⇑	92.1 ± 0.5	97.6 ± 0.4
	6	**99.4 ± 0.3**↓⇓	69.0 ± 6.4 ↑⇑	90.8 ± 12.8↑⇑	90.1 ± 12.8↑⇑	94.5 ± 9.7 ↓⇑	89.8 ± 8.2 ↑⇑	60.2 ± 3.6↑⇑	53.2 ± 1.2↑⇑	92.9 ± 0.4	97.6 ± 0.3
	7	**99.4 ± 0.2**↓⇓	68.9 ± 5.1 ↑⇑	95.0 ± 7.9 ↓⇑	89.8 ± 9.0 ↑⇑	97.7 ± 2.6 ↓	94.0 ± 4.6 ↓⇑	60.7 ± 3.3↑⇑	53.7 ± 1.2↑⇑	93.2 ± 0.6	97.7 ± 0.2
	8	**99.4 ± 0.1**↓⇓	69.0 ± 4.4 ↑⇑	97.5 ± 1.6 ↓	88.0 ± 8.1 ↑⇑	98.9 ± 1.3 ↓⇓	94.7 ± 3.7 ↓⇑	61.2 ± 2.5↑⇑	53.9 ± 1.2↑⇑	93.4 ± 0.5	97.7 ± 0.2
	9	**99.4 ± 0.1**↓⇓	69.3 ± 4.2 ↑⇑	97.2 ± 1.8 ↓⇑	88.0 ± 9.6 ↑⇑	99.1 ± 0.9 ↓⇓	93.9 ± 4.4 ⇑	61.1 ± 2.3↑⇑	54.5 ± 0.9↑⇑	93.7 ± 0.5	97.7 ± 0.1
	10	**99.4 ± 0.1**↓⇓	70.8 ± 4.6 ↑⇑	96.3 ± 3.3 ↓⇓	88.4 ± 8.8 ↑⇑	99.1 ± 0.8 ↓⇓	94.4 ± 3.9 ↓⇑	61.1 ± 2.9↑⇑	54.5 ± 1.1↑⇑	94.0 ± 0.4	97.7 ± 0.1
Textures-2	1	52.3 ± 11.3 ⇑	97.4 ± 1.9 ⇑	74.7 ± 7.8 ⇑	96.3 ± 2.9 ⇑	69.7 ± 9.8 ⇑	69.7 ± 9.8 ⇑	53.8 ± 2.2 ⇑	50.3 ± 0.6 ⇑	N/A	**100.0 ± 0.0**
	2	79.6 ± 25.6↑⇑	98.2 ± 2.2↑⇑	92.1 ± 9.6↑⇑	99.8 ± 0.6↑⇑	95.0 ± 9.5↑⇑	95.6 ± 6.9↑⇑	59.7 ± 4.0↑⇑	51.1 ± 0.8↑⇑	100.0 ± 0.0	**100.0 ± 0.0**
	3	92.3 ± 18.9↑⇑	99.8 ± 0.3↑⇑	95.9 ± 6.6↑⇑	100.0 ± 0.0↓	99.1 ± 2.2↑⇑	98.8 ± 1.3↑⇑	65.6 ± 5.7↑⇑	51.7 ± 0.9↑⇑	100.0 ± 0.0	100.0 ± 0.0
	4	100.0 ± 0.0 ↓	100.0 ± 0.1 ⇑	99.1 ± 2.0↑⇑	100.0 ± 0.1 ⇑	99.9 ± 0.2↑⇑	99.5 ± 1.1↑⇑	70.1 ± 5.7↑⇑	52.2 ± 0.7↑⇑	100.0 ± 0.0	100.0 ± 0.0
	5	100.0 ± 0.0 ↓	100.0 ± 0.1 ⇑	99.3 ± 2.5↑⇑	100.0 ± 0.0↓	100.0 ± 0.0↓	99.8 ± 0.4↑⇑	73.0 ± 5.0↑⇑	52.3 ± 0.7↑⇑	100.0 ± 0.0	100.0 ± 0.0
	6	100.0 ± 0.0 ↓	100.0 ± 0.1 ⇑	99.8 ± 0.6↑⇑	100.0 ± 0.0↓	100.0 ± 0.0↓	99.9 ± 0.3↑⇑	74.7 ± 4.3↑⇑	52.4 ± 0.9↑⇑	100.0 ± 0.0	100.0 ± 0.0
	7	100.0 ± 0.0 ↓	100.0 ± 0.0↓	99.4 ± 2.0↑⇑	100.0 ± 0.0↓	100.0 ± 0.0↓	99.9 ± 0.3↑⇑	77.5 ± 3.3↑⇑	52.6 ± 1.1↑⇑	100.0 ± 0.0	100.0 ± 0.0
	8	100.0 ± 0.0 ↓	100.0 ± 0.0↓	99.5 ± 1.7↑⇑	100.0 ± 0.0↓	100.0 ± 0.0↓	99.9 ± 0.2↑⇑	78.0 ± 3.6↑⇑	53.0 ± 1.0↑⇑	100.0 ± 0.0	100.0 ± 0.0
	9	100.0 ± 0.0 ↓	100.0 ± 0.0↓	99.7 ± 0.7↑⇑	100.0 ± 0.0↓	100.0 ± 0.0↓	100.0 ± 0.1 ⇑	78.1 ± 3.2↑⇑	53.0 ± 0.9↑⇑	100.0 ± 0.0	100.0 ± 0.0
	10	100.0 ± 0.0 ↓	100.0 ± 0.0↓	99.9 ± 0.2↑⇑	100.0 ± 0.0↓	100.0 ± 0.0↓	100.0 ± 0.0↓	78.7 ± 3.8↑⇑	53.6 ± 0.8↑⇑	100.0 ± 0.0	100.0 ± 0.0
Textures-3	1	57.7 ± 24.9 ⇑	**90.8 ± 4.2**	74.8 ± 9.8 ⇑	90.8 ± 4.5	66.3 ± 14.2 ⇑	66.3 ± 14.2 ⇑	67.4 ± 4.5 ⇑	51.0 ± 1.1 ⇑	N/A	89.4 ± 8.7
	2	83.6 ± 16.7↓⇑	90.8 ± 4.8↓⇑	85.3 ± 9.7↓⇑	91.9 ± 3.5↓	90.3 ± 8.1 ↓⇑	87.7 ± 10.3↓⇑	80.4 ± 4.9 ⇑	53.0 ± 1.2↑⇑	79.3 ± 3.1	**92.0 ± 2.3**
	3	89.0 ± 4.6 ↓⇑	93.0 ± 3.4↓	87.2 ± 7.5↓⇑	91.5 ± 3.1↓	**93.3 ± 3.5** ↓⇓	92.8 ± 4.9 ↓	87.0 ± 2.7↓⇑	54.7 ± 1.6↑⇑	79.3 ± 2.4	92.2 ± 2.3
	4	90.3 ± 2.1 ↓⇑	93.0 ± 2.9↓	89.1 ± 6.3↓⇑	92.1 ± 2.1↓	93.0 ± 3.3 ↓	**94.3 ± 3.3** ↓⇓	89.2 ± 1.6↓⇑	55.8 ± 1.7↑⇑	80.4 ± 2.1	92.2 ± 2.1
	5	90.3 ± 2.2 ↓⇑	93.0 ± 3.0↓	89.4 ± 4.6↓⇑	92.3 ± 2.4↓	92.3 ± 2.7 ↓	**94.8 ± 2.1** ↓⇓	90.5 ± 0.9↓⇑	56.7 ± 1.8↑⇑	81.9 ± 1.7	92.4 ± 1.8
	6	90.4 ± 2.2 ↓⇑	93.6 ± 2.2↓⇓	89.2 ± 4.1↓⇑	91.6 ± 2.6↓	92.2 ± 2.8 ↓	**94.5 ± 2.3** ↓⇓	90.7 ± 0.9↓⇑	58.1 ± 1.4↑⇑	82.8 ± 1.2	92.3 ± 1.8
	7	90.3 ± 2.3 ↓⇑	93.8 ± 1.8↓⇓	88.0 ± 4.4↓⇑	91.7 ± 2.5↓	92.0 ± 2.6 ↓	**94.1 ± 2.6** ↓⇓	91.0 ± 0.9↓⇑	58.7 ± 1.5↑⇑	84.1 ± 1.4	92.4 ± 1.7
	8	90.5 ± 2.5 ↓⇑	93.8 ± 1.9↓⇓	88.9 ± 3.5↓⇑	92.4 ± 2.3↓	91.9 ± 2.5 ↓	**93.8 ± 2.3** ↓⇓	91.2 ± 1.1↓⇑	59.1 ± 1.4↑⇑	84.1 ± 1.3	92.4 ± 1.6
	9	90.7 ± 2.6 ↓⇑	**93.8 ± 2.0**↓⇓	89.1 ± 3.6↓⇑	92.3 ± 2.2↓	91.9 ± 2.1 ↓	93.6 ± 2.2 ↓⇓	91.4 ± 1.1↓⇑	60.1 ± 1.1↑⇑	84.6 ± 1.5	92.5 ± 1.6
	10	90.6 ± 2.6 ↓⇑	**93.6 ± 2.0**↓⇓	89.6 ± 3.2↓⇑	92.4 ± 2.4↓	92.0 ± 2.0 ↓⇑	93.0 ± 2.6 ↓	91.4 ± 1.1↓⇑	60.7 ± 1.0↑⇑	85.0 ± 1.6	92.8 ± 1.4
Textures-4	1	53.7 ± 7.0 ⇑	72.2 ± 7.5 ⇑	67.3 ± 7.0 ⇑	73.3 ± 7.2 ⇑	57.7 ± 16.6 ⇑	57.7 ± 16.6 ⇑	50.4 ± 1.3 ⇑	50.0 ± 0.5 ⇑	N/A	**95.1 ± 3.2**
	2	79.1 ± 13.6↑⇑	74.3 ± 7.4↑⇑	66.8 ± 8.4↑⇑	80.5 ± 6.6 ⇑	78.8 ± 8.0 ↑⇑	76.9 ± 8.2 ↑⇑	51.5 ± 1.6↑⇑	50.3 ± 0.5↑⇑	81.9 ± 4.1	**95.7 ± 2.3**
	3	87.6 ± 8.5 ↓⇑	76.0 ± 5.9↑⇑	74.8 ± 7.1↑⇑	84.0 ± 6.8 ↓⇑	81.2 ± 7.5 ⇑	81.2 ± 7.6 ⇑	52.0 ± 1.7↑⇑	50.5 ± 0.5↑⇑	79.6 ± 5.4	**96.0 ± 2.4**
	4	87.5 ± 8.1 ↓⇑	77.2 ± 6.1↑⇑	79.6 ± 7.9 ⇑	76.4 ± 13.3↑⇑	84.0 ± 7.2 ↓⇑	82.6 ± 6.8 ↓⇑	52.5 ± 1.6↑⇑	50.4 ± 0.6↑⇑	79.6 ± 5.3	**96.1 ± 2.1**
	5	87.5 ± 8.7 ↓⇑	76.8 ± 6.1↑⇑	83.8 ± 6.5↓⇑	84.2 ± 5.1 ↓⇑	85.5 ± 6.1 ↓⇑	85.5 ± 5.7 ↓⇑	52.7 ± 1.6↑⇑	50.3 ± 0.4↑⇑	81.7 ± 3.6	**97.1 ± 1.2**
	6	87.8 ± 6.5 ↓⇑	77.5 ± 6.4↑⇑	83.2 ± 6.3 ⇑	80.8 ± 12.0↑⇑	87.0 ± 5.8 ↓⇑	86.6 ± 4.8 ↓⇑	53.5 ± 1.7↑⇑	50.6 ± 0.5↑⇑	82.9 ± 3.7	**97.3 ± 0.9**
	7	87.5 ± 5.5 ↓⇑	77.6 ± 6.3↑⇑	83.4 ± 4.9 ⇑	86.3 ± 5.5 ↓⇑	89.5 ± 4.6 ↓⇑	87.7 ± 4.4 ↓⇑	54.4 ± 2.1↑⇑	50.7 ± 0.5↑⇑	84.4 ± 3.0	**97.6 ± 0.6**
	8	88.4 ± 4.1 ↓⇑	77.6 ± 6.9↑⇑	83.1 ± 4.6↑⇑	87.7 ± 4.6 ↓⇑	89.9 ± 4.2 ↓⇑	87.2 ± 3.5 ↓⇑	54.8 ± 1.2↑⇑	50.6 ± 0.5↑⇑	85.5 ± 2.7	**97.7 ± 0.6**
	9	88.3 ± 3.7 ↓⇑	77.4 ± 6.5↑⇑	84.1 ± 4.4↑⇑	86.6 ± 4.4 ⇑	90.6 ± 4.2 ↓⇑	87.9 ± 4.4 ↓⇑	55.1 ± 1.7↑⇑	50.6 ± 0.5↑⇑	86.4 ± 2.3	**97.7 ± 0.5**
	10	87.5 ± 4.0 ⇑	77.6 ± 6.1↑⇑	84.7 ± 3.7↑⇑	87.4 ± 4.7 ⇑	90.8 ± 4.0 ↓⇑	87.5 ± 5.0 ⇑	55.0 ± 2.0↑⇑	50.7 ± 0.6↑⇑	87.1 ± 1.7	**97.9 ± 0.4**

Table 3: Results of the group B data set.

		Non-GP Methods						GP-based Methods			
	Size	AdaBoostM1	KStar	NB	NBTree	NNge	SVM	Conven.	Two-tier	One-shot	Compound
Textures-5	1	70.3 ± 24.6 ⇑	66.2 ± 10.7 ⇑	67.9 ± 11.8 ⇑	63.7 ± 12.5 ⇑	64.9 ± 20.1 ⇑	64.9 ± 20.1 ⇑	58.5 ± 5.0 ⇑	49.8 ± 1.0 ⇑	N/A	**91.4 ± 14.3**
	2	73.4 ± 17.3↑⇑	71.5 ± 12.4↑⇑	71.5 ± 13.5↑⇑	69.8 ± 14.6↑⇑	73.4 ± 14.8↑⇑	69.0 ± 16.1↑⇑	55.9 ± 6.8↑⇑	51.1 ± 1.6↑⇑	91.3 ± 8.8	**91.6 ± 13.1**
	3	73.5 ± 15.1↑⇑	72.0 ± 14.2↑⇑	81.5 ± 10.3↑⇑	62.8 ± 15.7↑⇑	77.0 ± 13.1↑⇑	69.1 ± 18.1↑⇑	54.5 ± 6.4↑⇑	51.9 ± 1.7↑⇑	**91.4 ± 8.2**	91.2 ± 12.2
	4	74.9 ± 15.4↑⇑	72.3 ± 14.3↑⇑	83.9 ± 9.2 ↑⇑	62.0 ± 16.0↑⇑	76.0 ± 13.2↑⇑	69.7 ± 17.8↑⇑	54.9 ± 6.8↑⇑	52.2 ± 1.6↑⇑	**91.3 ± 7.6**	90.7 ± 11.7
	5	76.7 ± 16.2↑⇑	72.4 ± 14.0↑⇑	85.1 ± 7.8 ↑⇑	69.2 ± 17.2↑⇑	76.2 ± 12.5↑⇑	70.6 ± 17.4↑⇑	55.3 ± 6.6↑⇑	52.0 ± 2.2↑⇑	91.4 ± 7.1	**91.8 ± 10.6**
	6	76.8 ± 15.4↑⇑	72.8 ± 13.1↑⇑	85.2 ± 8.2 ↑⇑	69.2 ± 17.9↑⇑	77.6 ± 13.2↑⇑	71.5 ± 17.7↑⇑	54.8 ± 6.2↑⇑	52.5 ± 2.5↑⇑	**92.3 ± 6.7**	91.9 ± 10.6
	7	76.6 ± 15.4↑⇑	74.3 ± 13.1↑⇑	87.0 ± 6.5 ↑⇑	73.7 ± 17.0↑⇑	82.9 ± 7.9 ↑⇑	73.7 ± 16.9↑⇑	54.0 ± 6.8↑⇑	53.1 ± 2.2↑⇑	**92.7 ± 6.2**	92.5 ± 9.9
	8	77.2 ± 16.0↑⇑	75.4 ± 12.1↑⇑	87.2 ± 6.2 ↑⇑	76.0 ± 16.7↑⇑	84.3 ± 8.0 ↑⇑	74.6 ± 17.2↑⇑	52.9 ± 6.8↑⇑	53.2 ± 2.1↑⇑	**93.2 ± 6.3**	93.1 ± 9.2
	9	77.7 ± 16.3↑⇑	77.3 ± 11.6↑⇑	87.9 ± 4.3 ↑⇑	75.4 ± 17.3↑⇑	85.2 ± 7.9 ↑⇑	75.9 ± 16.3↑⇑	53.0 ± 6.8↑⇑	53.2 ± 2.1↑⇑	**93.6 ± 5.3**	93.5 ± 8.5
	10	77.7 ± 15.3↑⇑	77.6 ± 11.2↑⇑	88.5 ± 4.5 ↑⇑	78.2 ± 15.2↑⇑	85.4 ± 8.0 ↑⇑	77.6 ± 14.9↑⇑	52.0 ± 6.7↑⇑	53.3 ± 2.3↑⇑	94.0 ± 4.7	**94.1 ± 7.9**

tated version of the Textures-4 data set. While Compound-GP has maintained its performance, the performance of the One-shot GP method has greatly dropped compared to Textures-4. Similar to Compound-GP, other methods have also shown a nearly consistent performance on this data set. Noticeably, the AdaBoostM1 method shows a considerably better performance on this data set compared to the performance of this

Table 4: Results of the group C data sets.

		Non-GP Methods						GP-based Methods			
	Size	AdaBoostM1	KStar	NB	NBTree	NNge	SVM	Conven.	Two-tier	One-shot	Compound
Textures-6	1	93.6 ± 5.4 ⇑	61.9 ± 5.6 ⇑	53.1 ± 5.0 ⇑	61.5 ± 6.9 ⇑	49.5 ± 4.0 ⇑	49.5 ± 4.0 ⇑	50.4 ± 0.8 ⇑	50.2 ± 0.2 ⇑	N/A	**98.2 ± 1.0**
	2	94.2 ± 5.4 ⇑	67.7 ± 7.9↑⇑	65.2 ± 7.2↑⇑	63.4 ± 6.4 ↑⇑	55.4 ± 4.3↑⇑	55.1 ± 4.3↑⇑	50.7 ± 0.8↑⇑	50.3 ± 0.3↑⇑	93.7 ± 1.0	**98.7 ± 0.2**
	3	94.7 ± 5.3↓⇑	70.5 ± 4.8↑⇑	63.4 ± 7.6↑⇑	62.4 ± 7.2 ↑⇑	57.9 ± 3.7↑⇑	59.2 ± 5.2↑⇑	50.9 ± 0.8↑⇑	50.4 ± 0.2↑⇑	93.3 ± 0.7	**98.8 ± 0.2**
	4	95.2 ± 5.2↓⇑	72.4 ± 7.5↑⇑	62.2 ± 7.9↑⇑	65.0 ± 11.2↑⇑	59.1 ± 3.6↑⇑	60.2 ± 5.8↑⇑	51.5 ± 0.9↑⇑	50.5 ± 0.3↑⇑	93.5 ± 0.8	**98.8 ± 0.2**
	5	95.7 ± 5.1↓⇑	72.3 ± 6.0↑⇑	59.3 ± 6.4↑⇑	65.0 ± 11.1↑⇑	61.7 ± 4.0↑⇑	61.0 ± 5.8↑⇑	51.5 ± 1.2↑⇑	50.5 ± 0.4↑⇑	94.4 ± 0.4	**98.8 ± 0.2**
	6	96.3 ± 4.9↓⇑	73.4 ± 7.7↑⇑	56.8 ± 2.3↑⇑	65.8 ± 13.7↑⇑	64.7 ± 6.4↑⇑	64.0 ± 7.3↑⇑	51.5 ± 1.2↑⇑	50.6 ± 0.4↑⇑	95.0 ± 0.2	**98.8 ± 0.1**
	7	96.8 ± 4.6↓⇑	75.6 ± 7.1↑⇑	58.1 ± 2.7↑⇑	67.0 ± 14.3↑⇑	67.0 ± 6.5↑⇑	64.3 ± 6.8↑⇑	51.4 ± 1.5↑⇑	50.7 ± 0.3↑⇑	95.3 ± 0.3	**98.8 ± 0.1**
	8	97.3 ± 4.3↓⇑	78.7 ± 8.9↑⇑	58.5 ± 3.1↑⇑	67.5 ± 14.6↑⇑	69.5 ± 8.9↑⇑	66.0 ± 8.2↑⇑	51.5 ± 1.8↑⇑	50.6 ± 0.4↑⇑	95.5 ± 0.2	**98.8 ± 0.1**
	9	97.8 ± 3.8↓⇑	79.0 ± 9.1↑⇑	58.9 ± 3.2↑⇑	63.7 ± 16.8↑⇑	70.7 ± 10.0↑⇑	66.0 ± 7.6↑⇑	51.5 ± 1.8↑⇑	50.8 ± 0.4↑⇑	95.8 ± 0.3	**98.8 ± 0.1**
	10	98.3 ± 3.2↓⇑	81.3 ± 7.4↑⇑	60.2 ± 3.7↑⇑	65.6 ± 18.8↑⇑	70.6 ± 9.0↑⇑	66.4 ± 7.1↑⇑	51.5 ± 1.9↑⇑	51.0 ± 0.3↑⇑	96.0 ± 0.3	**98.8 ± 0.1**
Textures-7	1	49.6 ± 0.8 ⇑	97.6 ± 2.4 ⇑	73.3 ± 7.5 ⇑	97.8 ± 4.0 ⇑	66.5 ± 9.4 ⇑	66.5 ± 9.4 ⇑	56.8 ± 2.2 ⇑	51.1 ± 0.4 ⇑	N/A	**100.0 ± 0.0**
	2	67.6 ± 24.4↑⇑	99.2 ± 0.7↑⇑	99.1 ± 2.4↑⇑	99.9 ± 0.3↑⇑	97.6 ± 3.4↑⇑	93.2 ± 9.3↑⇑	66.7 ± 6.5↑⇑	51.4 ± 0.4↑⇑	100.0 ± 0.0	**100.0 ± 0.0**
	3	77.8 ± 25.2↑⇑	99.9 ± 0.2↑⇑	99.5 ± 0.9↑⇑	100.0 ± 0.0↓	100.0 ± 0.1 ⇑	99.9 ± 0.3↑⇑	76.1 ± 4.3↑⇑	52.1 ± 0.7↑⇑	100.0 ± 0.0	100.0 ± 0.0
	4	80.3 ± 24.8↑⇑	100.0 ± 0.0↓⇑	99.8 ± 0.3↑⇑	100.0 ± 0.0↓	100.0 ± 0.0↓	100.0 ± 0.0↓⇑	79.6 ± 2.0↑⇑	52.5 ± 0.7↑⇑	100.0 ± 0.0	100.0 ± 0.0
	5	82.7 ± 24.2↑⇑	100.0 ± 0.0↓	99.9 ± 0.2↑⇑	100.0 ± 0.0↓	100.0 ± 0.0↓	100.0 ± 0.0↓	82.0 ± 2.0↑⇑	53.1 ± 0.8↑⇑	100.0 ± 0.0	100.0 ± 0.0
	6	85.2 ± 23.2↑⇑	100.0 ± 0.0↓	99.1 ± 4.1↑⇑	100.0 ± 0.0↓	100.0 ± 0.0↓	100.0 ± 0.0↓	82.7 ± 2.3↑⇑	53.2 ± 0.8↑⇑	100.0 ± 0.0	100.0 ± 0.0
	7	87.7 ± 21.9↑⇑	100.0 ± 0.0↓	99.8 ± 0.6↑⇑	100.0 ± 0.0↓	100.0 ± 0.0↓	100.0 ± 0.0↓	83.8 ± 2.6↑⇑	53.6 ± 0.7↑⇑	100.0 ± 0.0	100.0 ± 0.0
	8	90.1 ± 20.3↑⇑	100.0 ± 0.0↓	100.0 ± 0.0↓	100.0 ± 0.0↓	100.0 ± 0.0↓	100.0 ± 0.0↓	85.1 ± 2.4↑⇑	54.1 ± 0.7↑⇑	100.0 ± 0.0	100.0 ± 0.0
	9	92.6 ± 18.1↑⇑	100.0 ± 0.0↓	100.0 ± 0.0↓	100.0 ± 0.0↑⇑	100.0 ± 0.0↓	100.0 ± 0.0↓	86.3 ± 2.3↑⇑	54.4 ± 0.6↑⇑	100.0 ± 0.0	100.0 ± 0.0
	10	95.1 ± 15.2↑⇑	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0↑⇑	100.0 ± 0.0	100.0 ± 0.0	86.7 ± 2.6↑⇑	54.8 ± 0.8↑⇑	100.0 ± 0.0	100.0 ± 0.0
Textures-8	1	57.7 ± 30.3 ⇑	**89.0 ± 4.7** ⇓	67.5 ± 8.8 ⇑	87.2 ± 6.2 ⇓	69.6 ± 12.5 ⇑	69.6 ± 12.5 ⇑	70.0 ± 2.1 ⇑	50.9 ± 0.5 ⇑	N/A	80.7 ± 8.1
	2	77.1 ± 26.4↓⇑	90.0 ± 3.7↓⇓	**91.0 ± 5.2**↓⇓	90.8 ± 4.6↓⇓	78.2 ± 9.9 ↓⇑	79.9 ± 11.7↓⇑	78.5 ± 2.8↓⇑	52.2 ± 0.7↑⇑	74.4 ± 4.7	84.4 ± 5.4
	3	**93.6 ± 1.4** ↓⇓	92.2 ± 3.4↓⇓	93.0 ± 2.5↓⇓	92.5 ± 1.6↓⇓	85.7 ± 6.7 ↓	85.4 ± 11.0↓	84.9 ± 2.5↓	53.4 ± 0.6↑⇑	75.4 ± 2.9	86.5 ± 4.6
	4	**93.7 ± 1.4** ↓⇓	92.7 ± 3.3↓⇓	93.3 ± 0.9↓⇓	93.5 ± 1.4↓⇓	89.3 ± 6.4 ↓⇓	88.0 ± 10.0 ↓	87.4 ± 1.3↓	54.1 ± 0.8↑⇑	75.9 ± 2.7	86.3 ± 4.3
	5	**93.8 ± 1.3** ↓⇓	93.0 ± 2.9↓⇓	92.4 ± 2.4↓⇓	93.7 ± 1.5↓⇓	91.1 ± 5.8 ↓⇓	89.7 ± 8.6 ↓⇓	88.7 ± 1.4↓⇓	55.0 ± 1.1↑⇑	77.2 ± 2.7	86.8 ± 2.4
	6	93.8 ± 1.3 ↓⇓	93.6 ± 2.6↓⇓	91.1 ± 3.5↓⇓	**94.0 ± 1.3**↓⇓	93.1 ± 4.9 ↓⇓	91.4 ± 7.4 ↓⇓	89.0 ± 1.4↓⇓	55.9 ± 1.1↑⇑	79.0 ± 2.3	87.8 ± 1.8
	7	93.9 ± 1.2 ↓⇓	93.7 ± 2.5↓⇓	93.0 ± 1.7↓⇓	94.1 ± 1.0↓⇓	**94.3 ± 3.9** ↓⇓	92.8 ± 6.4 ↓⇓	89.7 ± 1.1↓⇓	56.7 ± 1.1↑⇑	80.2 ± 2.6	87.1 ± 1.8
	8	93.9 ± 1.2 ↓⇓	93.9 ± 2.3↓⇓	92.6 ± 1.7↓⇓	94.0 ± 1.2↓⇓	**94.9 ± 3.7** ↓⇓	94.2 ± 5.1 ↓⇓	90.0 ± 1.0↓⇓	57.3 ± 0.9↑⇑	80.8 ± 2.3	87.4 ± 1.5
	9	93.9 ± 1.1 ↓⇓	94.1 ± 2.0↓⇓	89.1 ± 4.0↓⇓	94.3 ± 1.0↓⇓	95.4 ± 3.4 ↓⇓	**95.4 ± 3.8** ↓⇓	90.6 ± 1.0↓⇓	57.7 ± 1.1↑⇑	81.8 ± 2.2	87.5 ± 1.2
	10	94.0 ± 1.1 ↓⇓	94.1 ± 1.9↓⇓	91.1 ± 2.3↓⇓	94.4 ± 0.8↓⇓	95.9 ± 3.0 ↓⇓	**96.0 ± 3.3** ↓⇓	90.7 ± 0.9↓⇓	58.2 ± 0.8↑⇑	82.3 ± 1.9	87.5 ± 1.4
Textures-9	1	48.5 ± 9.4 ⇑	83.1 ± 5.7 ⇑	76.1 ± 6.9 ⇑	80.5 ± 6.1 ⇑	70.6 ± 15.2 ⇑	70.6 ± 15.2 ⇑	50.4 ± 1.0 ⇑	50.0 ± 0.3 ⇑	N/A	**90.5 ± 5.0**
	2	91.4 ± 14.0↓	83.8 ± 6.9↓⇑	71.5 ± 8.6 ⇑	86.1 ± 4.3↓⇑	90.3 ± 3.2 ↓⇑	90.1 ± 2.1 ↓⇑	51.3 ± 1.1↑⇑	50.1 ± 0.3↑⇑	71.4 ± 6.4	**92.0 ± 2.5**
	3	**97.0 ± 0.8**↓⇓	89.7 ± 2.9↓⇑	78.6 ± 10.1↓⇑	85.1 ± 8.0↓⇑	92.3 ± 3.3 ↓	92.1 ± 1.9 ↓	52.8 ± 2.1↑⇑	50.4 ± 0.2↑⇑	69.6 ± 5.5	92.4 ± 2.0
	4	**97.0 ± 0.8**↓⇓	91.3 ± 3.4↓⇑	71.6 ± 8.5 ↓⇑	89.7 ± 7.5↓⇑	94.0 ± 3.2 ↓⇓	92.9 ± 1.9 ↓	54.4 ± 2.7↑⇑	50.5 ± 0.4↑⇑	69.0 ± 4.6	92.3 ± 1.5
	5	**97.1 ± 0.9**↓⇓	92.2 ± 2.0↓⇑	74.4 ± 11.1↓⇑	91.1 ± 6.1↓⇑	94.9 ± 2.3 ↓⇓	93.5 ± 1.7 ↓	56.3 ± 3.5↑⇑	50.5 ± 0.4↑⇑	71.1 ± 3.9	93.0 ± 1.6
	6	**97.2 ± 0.9**↓⇓	93.2 ± 2.3↓	78.7 ± 9.7 ↓⇑	93.4 ± 3.2↓	95.5 ± 1.5 ↓⇓	94.1 ± 1.4 ↓⇓	57.6 ± 3.8↑⇑	50.7 ± 0.4↑⇑	73.3 ± 2.9	93.1 ± 1.5
	7	**97.2 ± 1.0**↓⇓	93.0 ± 2.6↓	81.4 ± 9.7 ↓⇑	92.9 ± 4.2↓	95.8 ± 1.4 ↓⇓	94.6 ± 1.4 ↓⇓	58.2 ± 3.1↑⇑	50.7 ± 0.5↑⇑	75.8 ± 2.8	93.3 ± 1.5
	8	**97.3 ± 1.0**↓⇓	93.1 ± 1.9↓	84.1 ± 9.5 ↓⇑	93.3 ± 4.1↓	96.2 ± 1.1 ↓⇓	94.7 ± 1.5 ↓⇓	59.6 ± 3.3↑⇑	50.7 ± 0.6↑⇑	77.4 ± 2.7	93.6 ± 1.5
	9	**97.4 ± 1.0**↓⇓	93.2 ± 1.8↓	84.8 ± 11.9↓⇑	93.8 ± 3.0↓	96.4 ± 0.9 ↓⇓	95.2 ± 1.2 ↓⇓	59.4 ± 3.3↑⇑	50.8 ± 0.5↑⇑	79.4 ± 2.8	93.5 ± 1.2
	10	**97.4 ± 1.0**↓⇓	93.2 ± 2.0↓	86.6 ± 9.8 ↓⇑	94.4 ± 2.3↓⇓	96.4 ± 0.9 ↓⇓	95.6 ± 1.0 ↓⇓	60.4 ± 3.3↓⇑	50.9 ± 0.4↓⇑	80.4 ± 2.8	93.6 ± 1.1

method on Textures-4.

5.1.4 Group D Data Set

Table 5 shows the results of the experiment on the Faces data set. Both of One-shot GP and Compound-GP have achieved better accuracy than all other methods on this data set when the number of instances is smaller than four per class. Meanwhile, the Two-tier GP and NB start to compete when the number of training instances increases. Apart from NB, the Compound-GP method has significantly outperformed all other methods on this data set. The One-shot GP method, on the other hand, has significantly outperformed NBTree, SVM, and Conventional-GP in all the nine different sizes, and some cases compared to AdaBoostM1, KStar, NNge, and the Two-tier GP methods.

5.1.5 Summary

The results show that both One-shot GP and Compound-GP have successfully evolved programs using only a few instances that can generalised well to the unseen data. Compared to non-GP and GP-based methods, One-shot GP and Compound-GP have

Table 5: Results of the group D data set.

		Non-GP Methods						GP-based Methods			
	Size	ABoostM1	KStar	NB	NBTree	NNge	SVM	Conven.	Two-tier	One-shot	Compound
Faces	1	56.1 ± 8.2 ⇑	61.2 ± 9.0 ⇑	57.3 ± 7.6 ⇑	61.7 ± 9.3 ⇑	57.6 ± 9.0 ⇑	57.7 ± 9.2 ⇑	55.2 ± 2.7 ⇑	55.9 ± 1.9 ⇑	N/A	$\mathbf{67.8 \pm 10.0}$
	2	55.4 ± 10.0 ↑⇑	60.8 ± 9.2 ↑⇑	64.9 ± 8.8	59.1 ± 11.6 ↑⇑	61.4 ± 11.8 ↑⇑	62.8 ± 12.3 ↑⇑	57.6 ± 3.4 ↑⇑	60.9 ± 2.9 ↑⇑	67.6 ± 6.3	$\mathbf{68.6 \pm 9.7}$
	3	57.6 ± 9.6 ↑⇑	63.6 ± 9.0 ↑⇑	67.4 ± 11.0 ⇑	58.3 ± 11.4 ↑⇑	64.2 ± 12.9 ↑⇑	64.9 ± 12.3 ↑⇑	58.6 ± 3.3 ↑⇑	63.2 ± 2.7 ↑⇑	68.1 ± 4.6	$\mathbf{70.4 \pm 5.2}$
	4	63.3 ± 10.8 ↑⇑	65.3 ± 8.5 ↑⇑	$\mathbf{71.3 \pm 10.2}$ ↓	57.6 ± 10.7 ↑⇑	64.8 ± 13.1 ↑⇑	63.4 ± 16.0 ↑⇑	59.6 ± 3.4 ↑⇑	65.1 ± 2.6 ↑⇑	68.1 ± 4.9	70.6 ± 5.4
	5	64.5 ± 11.4 ↑⇑	67.0 ± 8.2 ⇑	$\mathbf{73.2 \pm 10.2}$ ↓	65.0 ± 10.8 ↑⇑	66.2 ± 12.3 ↑⇑	63.9 ± 12.1 ↑⇑	60.7 ± 4.7 ↑⇑	66.7 ± 2.5 ↑⇑	68.6 ± 4.1	72.8 ± 2.5
	6	66.9 ± 11.1 ↑⇑	68.8 ± 7.6 ⇑	$\mathbf{73.6 \pm 9.2}$ ↓	60.0 ± 11.2 ↑⇑	68.2 ± 11.6 ⇑	63.6 ± 11.5 ↑⇑	61.5 ± 4.5 ↑⇑	68.1 ± 2.0 ⇑	69.3 ± 3.7	73.5 ± 2.3
	7	66.3 ± 11.2 ↑⇑	69.6 ± 7.5 ⇑	$\mathbf{75.4 \pm 7.6}$ ↓	63.8 ± 9.2 ↑⇑	69.0 ± 12.3 ⇑	61.8 ± 12.5 ↑⇑	61.7 ± 4.8 ↑⇑	69.2 ± 1.3 ⇑	69.4 ± 3.7	74.9 ± 2.6
	8	68.6 ± 10.8 ⇑	70.1 ± 7.3 ⇑	$\mathbf{76.9 \pm 6.9}$ ↓⇓	66.1 ± 9.3 ↑⇑	70.4 ± 11.2 ⇑	62.6 ± 12.8 ↑⇑	61.9 ± 4.6 ↑⇑	70.2 ± 1.7 ⇑	70.0 ± 3.1	75.4 ± 2.1
	9	69.8 ± 8.7 ⇑	70.3 ± 7.1 ⇑	$\mathbf{77.2 \pm 6.7}$ ↓⇓	61.4 ± 8.9 ↑⇑	71.8 ± 9.9 ↓⇑	62.3 ± 10.8 ↑⇑	62.0 ± 4.2 ↑⇑	71.1 ± 1.6 ⇑	70.4 ± 2.5	76.0 ± 1.7
	10	72.2 ± 7.7 ↓⇑	70.3 ± 6.9 ⇑	$\mathbf{77.5 \pm 6.9}$ ↓⇓	66.0 ± 9.5 ↑⇑	73.2 ± 9.3 ↓⇑	62.7 ± 9.7 ↑⇑	62.6 ± 3.6 ↑⇑	71.1 ± 1.7 ⇑	70.8 ± 2.4	76.6 ± 1.3

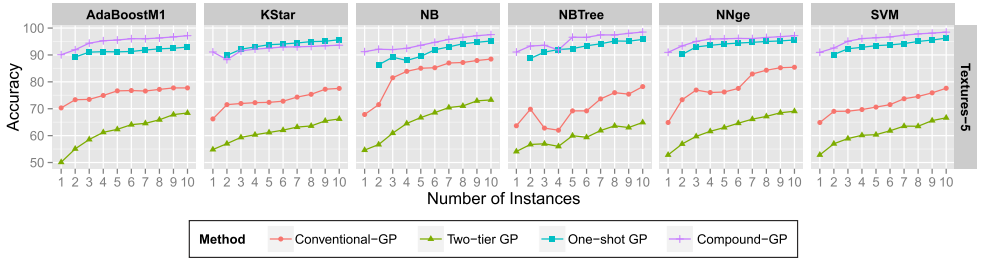


Figure 13: The average performance of the experimented methods on the Textures-5 data set using four different sets of features.

achieved significantly better or comparable results in most cases. However, in some situations the two new methods, especially One-shot GP, show significantly worse performance, e.g., on the Textures-1 data set compared to the performance of AdaBoostM1.

5.2 Feature Extraction

As highlighted in Section 4, the goodness of the detected and extracted features by the One-shot GP and Compound-GP methods have been investigated via feeding those features to six different classifiers (the non-GP methods that were used in the first experiment). The performance of those classifiers on the handcrafted features and those extracted by the Two-tier GP method have been compared to the use of One-shot GP and Compound-GP extracted features.

The result of each of the six classifiers on each of the data sets using four different sets of features is represented by a single line chart. Moreover, the results of the all six classifiers are aligned on a single row for each data set. The *y-axes* and *x-axes* of each chart represent the average accuracy and number of instances per class in the training set respectively. Due to the page limit, only the results on the data set of the second group are presented here and the rest are presented in Section A.1 of Appendix A.

On the Textures-5 data set, the features extracted by the One-shot GP and Compound-GP methods have significantly improved the performance of all the six classifiers over both handcrafted and Two-tier GP features as presented in Figure 13. As highlighted in Section 4, this data set are more challenging than other texture-based data sets due to illumination, scale, and pose variations of its instances.

5.3 Training and Testing Time

In this study, we also measured the average training and testing times in order to highlight the cost of evolving a program by each of the One-shot GP and Compound-GP methods. The results are presented in blocks of line charts that each shows the average time required to evolve or evaluate a program by each of the four GP methods. Each row of blocks represents either the training or testing phase, while each column presents the results of a single data set as presented in Figure 14. Similar to the plots of the previous section, the *x-axes* represents the number of instances per class in the training set; and *y-axes* represents the time in seconds. Due to the page limit, only the results on the data sets of the first group are presented here and the other figures are presented in Section A.2 of Appendix A.

The results show that both of the One-shot GP and Compound-GP methods takes significantly longer time to evolve and evaluate/test a program than Conventional-GP and Two-tier GP. Moreover, Conventional-GP is the fastest method amongst the four GP method, followed by the Two-tier GP with a slightly slower speed. This large gap between the two new and baseline methods was expected due to the following reasons:

- **Feature detection and extraction:** Apart from Conventional-GP, the other three GP-based methods perform feature detection and extraction at the lower part (near the leaves) of the evolved program. Performing those operations increases the complexity of the evolved program in terms of memory and computation costs. Moreover, in the case of the Two-tier GP, there are only four simple operations (minimum, maximum, mean, and standard deviation) that can be used to perform feature extraction. However, this operation is more complicated in the case of both One-shot GP and Compound-GP methods. The complication results from the calculation of the LBP code of each pixel in each of the detected regions, which require applying a threshold operation, checking if the calculated code is uniform or not, and accumulate it with other codes to form a histogram.
- **Wrapped classifiers:** In the case of both Conventional-GP and Two-tier GP the aim is to evolve a GP-based classifier. Thus, both of those methods do not have any wrapped classifier that needs to be trained. Meanwhile, the One-shot GP and Compound-GP methods have different number and type of wrapped classifiers. The One-shot GP method uses a simple kNN classifier; whereas Compound-GP consists of two SVM and two kNN classifiers for each of the *Special* node (the root of the evolved program's tree) children. Training and evaluating those wrapped classifiers introduce extra complexities that need to be handled.
- **Fitness function:** The fitness function of both Conventional-GP and Two-tier GP is simple that does not require extra calculations. However, the fitness function of both One-shot GP and Compound-GP is complex and requires calculating more parameters such as the overlapping ratio of the detected regions (OVR).
- **Termination criteria:** In the case of the One-shot GP and Compound-GP, the system is forced to proceed until the maximum number of generations is reached. Meanwhile, the Conventional-GP and Two-tier GP can terminate once an *ideal* program has been found.

5.4 Program Size

Here we investigate the complexity of the program evolved by each of the GP-based methods by average program size. The results are presented in line plots, where each

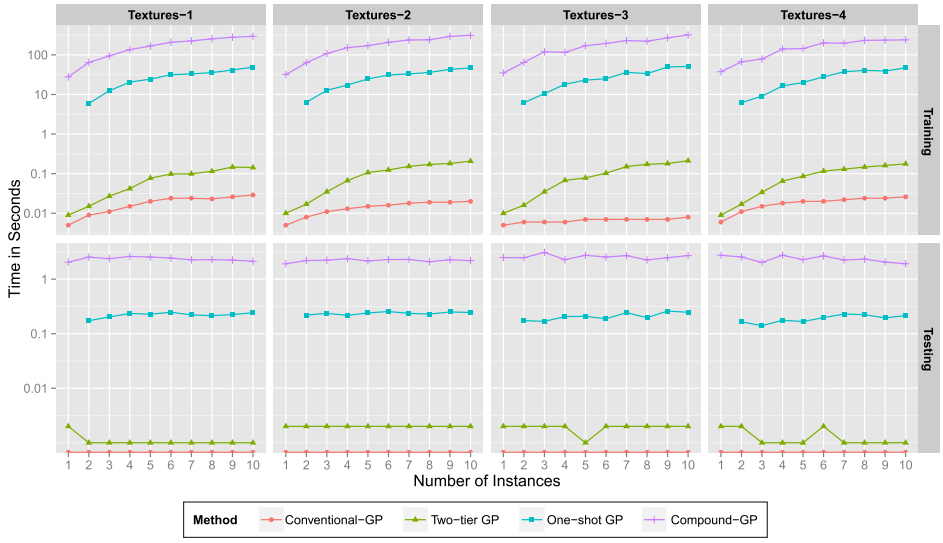


Figure 14: The average training and test time of the four GP-based methods on the Textures-1, Textures-2, Textures-3, and Textures-4 data sets.

row corresponds to one of the data sets, and each column represents the number of instances per class in the training set as presented in Figure 15. The *y-axes* and *x-axes* represent the number of nodes and generation respectively. Each block contains four lines one for each of the four GP methods. Due to the page limit, only the results on the data sets of the first group are presented here and the other figures are presented in Section A.3 of Appendix A.

The results of Conventional-GP and Two-tier GP show that when there are fewer than four instances per class, these two methods terminate early and before the maximum number of generations is reached. This was expected as both of those methods rely on the accuracy alone as a goodness measure. However, this does not occur in the case of One-shot GP and Compound-GP due to the other components of the fitness measures of these two methods that force the system to proceed.

The results show that One-shot GP has a constant program size that is neither affected by the size of the training set nor by the generation (progress of the run). This was expected due to the restriction of the program size of the One-shot GP method. The size is 35 nodes that are: one *Controller*; two *Histogram*; eight *Area*; and twenty four terminal nodes.

The Compound-GP method, on the other hand, evolves programs of different sizes, only restricted by the maximum-depth of the tree parameter. The results show that on average the method starts with a large program that gets reduced in later stages (subsequent generations). The main reason of this behaviour is the overlapping ratio (OVR) component of the fitness function, which forces the system to detect distinctive regions with minimal overlapping.

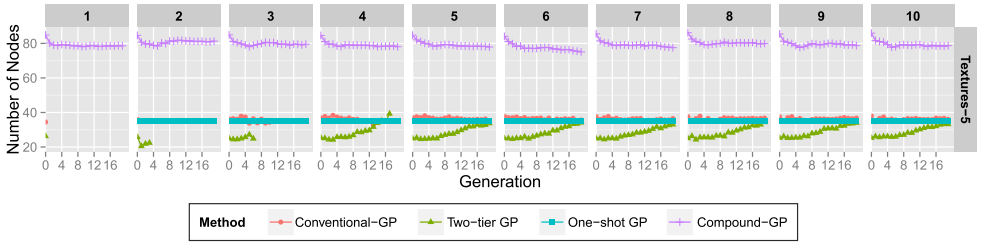


Figure 15: The average program size per generation of the four GP-based methods on the Textures-5 data set.

6 Further Analyses

Two evolved programs by each of the One-shot GP and Compound-GP methods are investigated in this section. The first example is taken from those programs that were evolved on one of the texture-based data sets, whereas the second example is a program that was evolved to discriminate between face and non-face instances.

6.1 One-shot GP Examples

The first example of an evolved program by the One-shot GP method on the Textures-2 data set is presented in Figure 16. The tree representation of this program is shown in Figure 16(a) that is made up of one *Controller*, two *Histogram*, and eight *Area* nodes. The position of each of those eight regions is highlighted in Figure 16(b) on one example of each class and the enlarged cut-outs are presented below each image. This program was evolved using the minimum allowed number of instances (two per class) by One-shot GP, and has achieved 100% accuracy on the unseen data of this data set. A closer look on the enlarged cut-outs reveals that the regions of the *rice* instance has less texture than that of the corresponding *sesameseeds* instance.

The example presented in Figure 17 shows an evolved program by the One-shot GP method on the Faces data set. This program has achieved 78.3% accuracy using only two instances of each class in the training set. The program detects the regions around both eyes, eyebrows, and cheeks. Those regions (especially around the eyes and cheeks) are similar to those were designed by a domain-expert, which shows the ability of the system to automatically detect such important regions.

6.2 Compound-GP Examples

The tree representation of an evolved program by the Compound-GP method on the Textures-2 data set is shown in Figure 18(a). This program was evolved using only one instance of each class, and has scored 100% accuracy on the unseen data of this data set. The detected regions by this program are shown in Figure 18(b). The evolved program shows that regions of different sizes and shapes have been detected, and the cut-outs of the two instances show clear differences in the texture especially the small ones.

Figure 19 shows an evolved program by the Compound-GP method on the Faces data set. The program has achieved 78.0% accuracy using only one instance per class. The tree representation shows that nine regions have been detected by this program as depicted in Figure 19(a). These regions are highlighted in Figure 19(b), which shows that some interesting regions such as the mouth, three regions around the left eye and

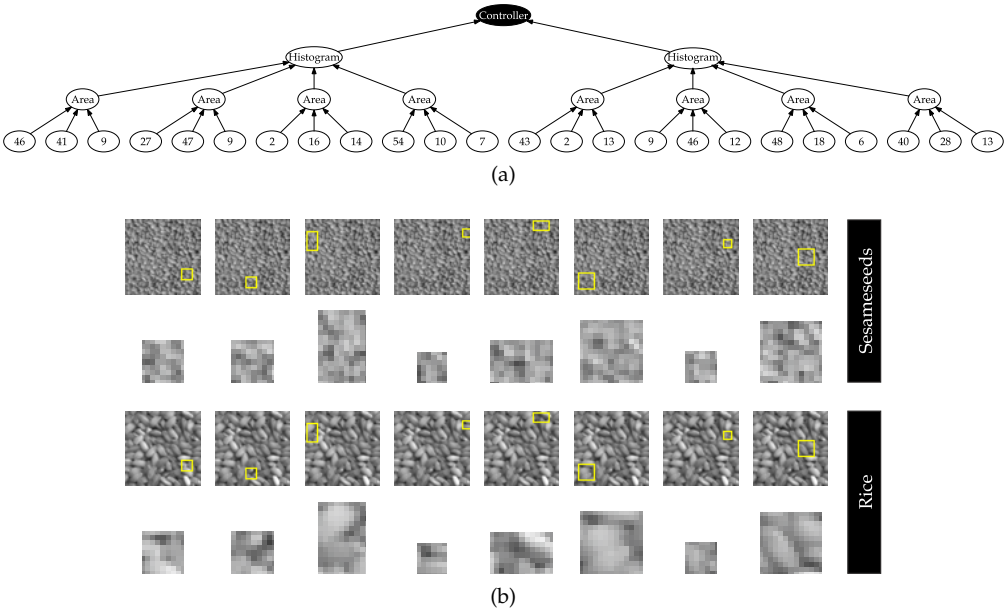


Figure 16: An evolved program by the One-shot GP method on the Textures-2 data set (a) the tree representation, and (b) the detected regions.

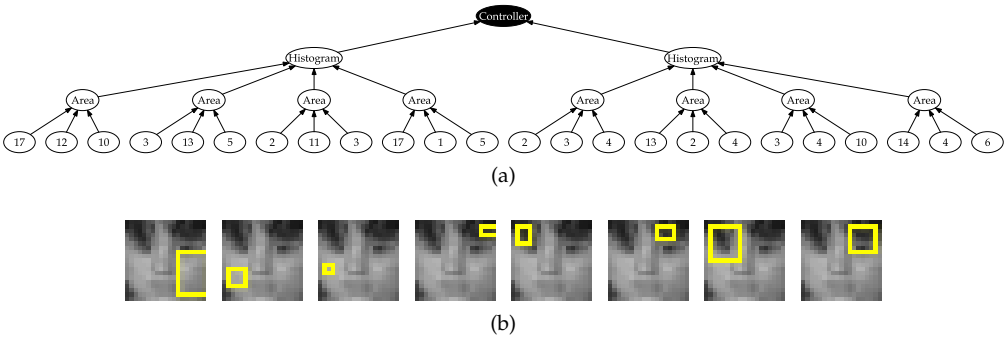
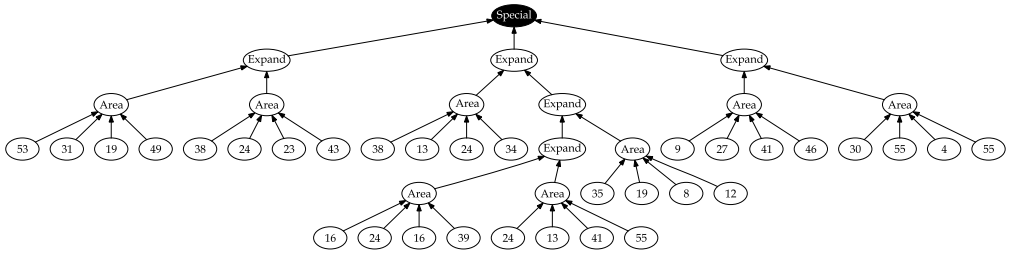


Figure 17: An evolved program by the One-shot GP method on the Faces data set (a) the tree representation, and (b) the detected regions.

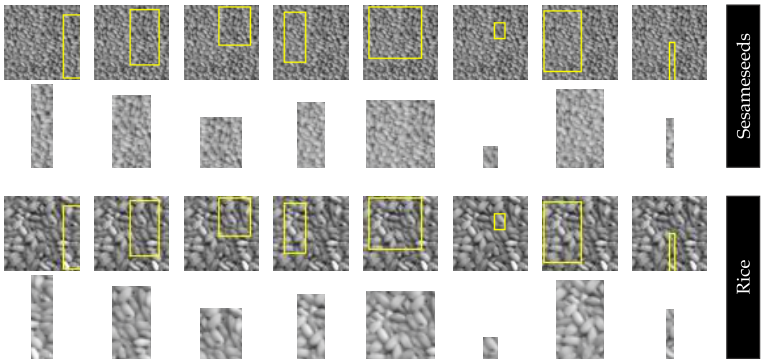
forehead, two regions detecting the left cheek, and three detecting the right cheek. This example shows that the detected regions have common features with those that were designed by a domain-expert.

6.3 Comments on the Number of Examples

An important question that is applicable to both methods is *why using a few instances can be sufficient to evolve a model with reasonably good performance*. The answer to this question can be the similarity between instances belonging to the same class. That is, instances belonging to one class must have distinctive features to those of other classes; otherwise, they must be grouped together. For example, the instances of each texture-based class have a special repetitive pattern or structure, where detecting this structure

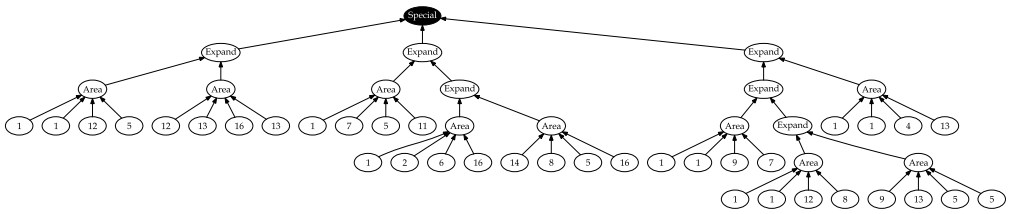


(a)

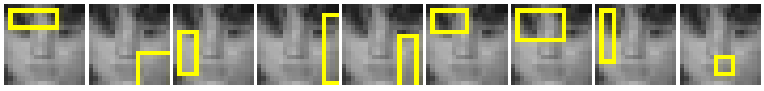


(b)

Figure 18: An evolved program by the Compound-GP method on the Textures-2 data set (a) the tree representation, and (b) the detected regions.



(a)



(b)

Figure 19: An evolved program by the Compound-GP method on the Faces data set (a) the tree representation, and (b) the detected regions.

represents the main task of the *Texture synthesis* field (Pietroni et al., 2010; Galerne et al., 2011). Similarly, all instances of the *Faces* data set have similar features in relativity fixed positions (eyes, nose, cheeks and so on). Therefore, allowing the system to detect distinctive regions from one or a few instances can be sufficient to generalise to unseen data. Therefore, the assumption is that the same pattern or structure is shared by all instances of one class.

7 Conclusions and Future Work

In this study, the One-shot GP and Compound GP methods have been investigated. The main aim of those two methods is to evolve a GP-based program for the task of binary classification in images using only one or a few instances per class. Moreover, neither of the two methods relies on the concept of *learning by knowledge transfer* approach where some instances from related domains can be used to assist the process of evolving a model. Using ten data sets of different flavours, and compared against two other GP and six non-GP methods, the results revealed that both of these new methods (One-shot GP and Compound-GP) were able to evolve good programs that are capable of performing image classification. Furthermore, the evolved programs have either outperformed all other competitor methods, or achieved comparable performance (i.e. accuracy) to the best of those other methods in most of the studied cases.

Some of the data sets that were used in this study are aimed at testing the ability of the evolved programs by One-shot GP and Compound-GP to handle illumination, rotation, and scale variations. Generally, the results show that the evolved programs by those two methods have maintained their performances or slightly dropped when one or more of those variations occurred. In other words, both of those new methods are capable to evolve (to some extent) illumination, rotation, and scale invariant programs.

The impact of the detected and extracted features by One-shot GP and Compound-GP on the performance of a variety of classifiers such as AdaBoostM1, NB, NBTree, SVM, KStar, and NNeg has also been investigated in this study. The results were compared against the use of handcrafted features and those were extracted by the Two-tier GP (Al-Sahaf et al., 2012a) method. The results show that the features of One-shot GP and Compound-GP have positive impact on the performance of most of the experimented classifiers in a large number of cases. However, in other cases those two methods (One-shot GP and Compound-GP) features have slightly degraded the performance of some of those classifiers.

In order to address the interpretability aspect, two examples of the evolved programs by each of the One-shot GP and Compound-GP methods have been discussed in detail in this study. The discussions revealed that the evolved programs are easy to interpret and unlike other methods that build a black-box like model.

Despite the good ability of the One-shot GP and Compound-GP methods to evolve good programs using a small number of examples, a closer look on the complexity of the evolved programs by these two methods reveal their drawbacks. The methods take a considerably longer time (especially Compound-GP) to evolve a good program than other GP-based methods specifically Conventional-GP and Two-tier GP. Moreover, the size of the evolved program by Compound-GP is larger than those evolved by Conventional-GP, Two-tier GP, and One-shot GP. The size of the evolved program by One-shot GP, is fixed and introduces an extra parameter that needs to be set. Although the features extracted by either of these two methods, i.e., One-shot GP and Compound-GP, are not biased to a specific classifier (as the results suggest), it may vary when the wrapped classifiers are replaced by other classifiers.

Future Work

In the future we plan to investigate the possibility of extending both of the One-shot GP and Compound-GP methods to perform multi-class image classification using a limited number of instances of each class in the training set. Reducing the complexity of the evolved programs by these two methods, represents another direction that are worth investigating in the future. We also plan to study the impact of using different classifiers

(i.e. wrapped) on the performance of the evolved programs, and the goodness of the extracted features by the One-shot GP and Compound-GP methods.

References

Abdulhamid, F., Neshatian, K., and Zhang, M. (2011). Image recognition using genetic programming with loop structures. In *Proceedings of the 26th International Conference on Image and Vision Computing New Zealand*, volume 29, pages 553–558.

Ahonen, T., Hadid, A., and Pietikäinen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041.

Al-Sahaf, H., Song, A., Neshatian, K., and Zhang, M. (2012a). Extracting image features for classification by two-tier genetic programming. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE.

Al-Sahaf, H., Song, A., Neshatian, K., and Zhang, M. (2012b). Two-tier genetic programming: Towards raw pixel-based image classification. *Expert Systems with Applications*, 39(16):12291–12301.

Al-Sahaf, H., Zhang, M., and Johnston, M. (2013a). Binary image classification using genetic programming based on local binary patterns. In *28th International Conference of Image and Vision Computing New Zealand (IVCNZ)*, pages 220–225.

Al-Sahaf, H., Zhang, M., and Johnston, M. (2013b). A one-shot learning approach to image classification using genetic programming. In *Proceedings of the 26th Australasian Joint Conference on Artificial Intelligence*, volume 8272 of *Lecture Notes in Computer Science*, pages 110–122. Springer.

Al-Sahaf, H., Zhang, M., and Johnston, M. (2014a). Genetic programming evolved filters from a small number of instances for multiclass texture classification. In *Proceedings of the 29th International Conference on Image and Vision Computing New Zealand*, pages 84–89. ACM.

Al-Sahaf, H., Zhang, M., and Johnston, M. (2014b). Genetic programming for multiclass texture classification using a small number of instances. In *Proceedings of the 10th International Conference on Simulated Evolution and Learning*, volume 8886 of *Lecture Notes in Computer Science*, pages 335–346. Springer-Verlag.

Albukhanajer, W., Briffa, J., and Jin, Y. (2014). Evolutionary multiobjective image feature extraction in the presence of noise. *IEEE Transactions on Cybernetics*, PP(99):1–12.

Atkins, D. L., Neshatian, K., and Zhang, M. (2011). A domain independent genetic programming approach to automatic feature extraction for image classification. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 238–245. IEEE.

Bhowan, U., Zhang, M., and Johnston, M. (2009). Genetic programming for image classification with unbalanced data. In *Proceedings of the 24th International Conference Image and Vision Computing New Zealand*, pages 316–321.

Bratko, A., Cormack, G. V., Filipic, B., Lynam, T. R., and Zupan, B. (2006). Spam filtering using statistical data compression models. *Journal of Machine Learning Research*, 6:2673–2698.

Chang, C.-W., Ho, C.-C., and Chen, J.-H. (2012). ADHD classification by a texture analysis of anatomical brain MRI data. *Frontiers in Systems Neuroscience*, 6(66):1–13.

Cleary, J. G. and Trigg, L. E. (1995). K*: An instance-based learner using an entropic distance measure. In *Proceedings of the 12th International Conference on Machine Learning*, pages 108–114. Morgan Kaufmann.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

Dhurandhar, A. and Dobra, A. (2008). Probabilistic characterization of random decision trees. *Journal of Machine Learning Research*, 9:2321–2348.

Downey, C. and Zhang, M. (2009). Multiclass object classification for computer vision using linear genetic programming. In *Proceedings of the 24th International Conference on Image and Vision Computing New Zealand*, pages 73–78. IEEE.

Fan, Z.-G., Wang, K.-A., and Lu, B.-L. (2004). Feature selection for fast image classification with support vector machines. In Pal, N., Kasabov, N., Mudi, R., Pal, S., and Parui, S., editors, *Neural Information Processing*, volume 3316 of *Lecture Notes in Computer Science*, pages 1026–1031. Springer.

Fei-Fei, L., Fergus, R., and Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611.

Fix, E. and Hodges, J. (1951). Discriminatory analysis-nonparametric discrimination: Consistency properties. Technical Report 4, USAF School of Aviation Medicine, Randolf Field, Texas.

Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann.

Fu, W., Johnston, M., and Zhang, M. (2011). Genetic programming for edge detection based on accuracy of each training image. In *Proceedings of the 24th International Conference on Advances in Artificial Intelligence*, pages 301–310. Springer-Verlag.

Fu, W., Johnston, M., and Zhang, M. (2012). Automatic construction of invariant features using genetic programming for edge detection. In *AI 2012: Advances in Artificial Intelligence*, volume 7691 of *Lecture Notes in Computer Science*, pages 144–155. Springer.

Galerne, B., Gousseau, Y., and Morel, J. M. (2011). Random phase textures: Theory and synthesis. *IEEE Transactions on Image Processing*, 20(1):257–267.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63(1):3–42.

Guo, Z., Zhang, L., and Zhang, D. (2010). Rotation invariant texture classification using LBP variance (LBPV) with global matching. *Pattern Recognition*, 43(3):706–719.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: An update. *SIGKDD Explorations Newsletter*, 11(1):10–18.

Heisele, B., Poggio, T., and Pontil, M. (2000). Face detection in still gray images. A.I. memo 1687, Center for Biological and Computational Learning, MIT.

Hindmarsh, S., Andreae, P., and Zhang, M. (2012). Genetic programming for improving image descriptors generated using the scale-invariant feature transform. In *Proceedings of the 27th Conference on Image and Vision Computing New Zealand*, pages 85–90. ACM.

Ishikawa, T. and Mogi, K. (2011). Visual one-shot learning as an 'anti-camouflage device': A novel morphing paradigm. *Cognitive Neurodynamics*, 5(3):231–239.

Jain, A. and Chandrasekaran, B. (1982). Dimensionality and sample size considerations in pattern recognition practice. In *Proceedings of the Classification Pattern Recognition and Reduction of Dimensionality*, volume 2, pages 835–855. Elsevier.

John, G. H. and Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pages 338–345. Morgan Kaufmann.

Kohavi, R. (1996). Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 202–207. AAAI Press.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

Kylberg, G. (2011). The Kylberg texture dataset v. 1.0. External report (Blue series) 35, Centre for Image Analysis, Swedish University of Agricultural Sciences and Uppsala University, Uppsala, Sweden.

Liddle, T., Johnston, M., and Zhang, M. (2010). Multi-objective genetic programming for object detection. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8.

Lim, K.-L. and Galoogahi, H. (2010). Shape classification using local and global features. In *Proceedings of the 4th Pacific-Rim Symposium on Image and Video Technology*, pages 115–120. IEEE Computer Society.

Lisin, D. A., Mattar, M. A., Blaschko, M. B., Learned-Miller, E. G., and Benfield, M. C. (2005). Combining local and global image features for object class recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 3, pages 47–55. IEEE Computer Society.

Liu, F. T., Ting, K. M., and Fan, W. (2005). Maximizing tree diversity by building complete-random decision trees. In Ho, T. B., Cheung, D. W.-L., and Liu, H., editors, *Proceedings of the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, volume 3518 of *Lecture Notes in Computer Science*, pages 605–610. Springer.

Liu, L., Zhao, L., Long, Y., Kuang, G., and Fieguth, P. (2012). Extended local binary patterns for texture classification. *Image and Vision Computing*, 30(2):86–99.

Lu, D. and Weng, Q. (2007). A survey of image classification methods and techniques for improving classification performance. *International Journal of Remote Sensing*, 28(5):823–870.

Luke, S. (2013). *Essentials of Metaheuristics*. Lulu, second edition.

Martin, B. (1995). Instance-based learning: Nearest neighbor with generalization. Master's thesis, University of Waikato, Hamilton, New Zealand.

Miller, E., Matsakis, N., and Viola, P. (2000). Learning from one example through shared densities on transforms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 464–471.

Nguyen, D. T., Ogunbona, P. O., and Li, W. (2013). A novel shape-based non-redundant local binary pattern descriptor for object detection. *Pattern Recognition*, 46(5):1485–1500.

Ojala, T., Pietikäinen, M., and Harwood, D. (1994). Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, volume 1, pages 582–585. IEEE.

Ojala, T., Pietikäinen, M., and Harwood, D. (1996). A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29(1):51–59.

Ojala, T., Pietikäinen, M., and Mäenpää, T. (2000). Gray scale and rotation invariant texture classification with local binary patterns. In *Computer Vision - ECCV 2000*, number 1842 in Lecture Notes in Computer Science, pages 404–420. Springer.

Ojala, T., Pietikäinen, M., and Mäenpää, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987.

Olague, G. and Trujillo, L. (2011). Evolutionary-computer-assisted design of image operators that detect interest points using genetic programming. *Image Vision Computing*, 29(7):484–498.

Perez, C. B. and Olague, G. (2009). Evolutionary learning of local descriptor operators for object recognition. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pages 1051–1058. ACM.

Pietikäinen, M., Hadid, A., Zhao, G., and Ahonen, T. (2011). Local binary patterns for still images. In *Computer Vision Using Local Binary Patterns*, volume 40 of *Computational Imaging and Vision*, pages 13–47. Springer.

Pietroni, N., Cignoni, P., Otaduy, M., and Scopigno, R. (2010). Solid-texture synthesis: A survey. *IEEE Computer Graphics and Applications*, 30(4):74–89.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. In Schölkopf, B., Burges, C. J. C., and Smola, A. J., editors, *Advances in Kernel Methods*, pages 185–208. MIT Press.

Raudys, S. J. and Jain, A. K. (1991). Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):252–264.

Rifkin, R. and Klautau, A. (2004). In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141.

Rodner, E. and Denzler, J. (2011). Learning with few examples for binary and multiclass classification using regularization of randomized trees. *Pattern Recognition Letters*, 32(2):244–251.

Salakhutdinov, R., Tenenbaum, J. B., and Torralba, A. (2012). One-shot learning with a hierarchical nonparametric Bayesian model. In *ICML Unsupervised and Transfer Learning*, volume 27 of *JMLR Proceedings*, pages 195–206. JMLR.org.

Smart, W. R. and Zhang, M. (2003). Classification strategies for image classification in genetic programming. In *Proceedings of the International Conference on Image and Vision Computing New Zealand*, pages 402–407, Palmerston North, New Zealand. Massey University.

Song, A. and Ciesielski, V. (2004). Texture analysis by genetic programming. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2092–2099. IEEE Press.

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer-Verlag, 1st edition.

Tan, X., Chen, S., Zhou, Z.-H., and Zhang, F. (2006). Face recognition from a single image per person: A survey. *Pattern Recognition*, 39(9):1725–1745.

Wang, L. and He, D.-C. (1990). Texture classification using texture spectrum. *Pattern Recognition*, 23(8):905–910.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

Yang, B. and Chen, S. (2013). A comparative study on local binary pattern (LBP) based face recognition: LBP histogram versus LBP image. *Neurocomputing*, 120:365–379.

Zhang, M., Andreae, P., and Bhowan, U. (2004). A two phase genetic programming approach to object detection. In *Knowledge-Based Intelligent Information and Engineering Systems*, volume 3215 of *Lecture Notes in Computer Science*, pages 224–231. Springer.

Zhang, M. and Ciesielski, V. (1999). Genetic programming for multiple class object detection. In *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence*, volume 1747 of *Lecture Notes in Computer Science*, pages 180–192. Springer.

Zhang, M., Ciesielski, V., and Andreae, P. (2003). A domain-independent window approach to multiclass object detection using genetic programming. *EURASIP Journal on Advances in Signal Processing*, 2003(8):841–859.

Zhang, M. and Johnston, M. (2009). A variant program structure in tree-based genetic programming for multiclass object classification. In *Evolutionary Image Analysis and Signal Processing*, volume 213 of *Studies in Computational Intelligence*, pages 55–72. Springer.

Zhang, M. and Smart, W. (2004). Multiclass object classification using genetic programming. In *Applications of Evolutionary Computing*, volume 3005 of *Lecture Notes in Computer Science*, pages 369–378. Springer.

A Appendix

A.1 Feature Extraction

A.1.1 Group A Data Sets

Apart from AdaBoostM1, the use of One-shot GP and Compound-GP extracted features show significant improvement in the performance of the classifiers when there are fewer than six instances per class in the training set.

On Textures-2, on the other hand, all of the six classifiers have achieved perfect performance using the features extracted by the One-shot GP and Compound-GP methods. Apart from KStar and NBTree, it is clear that the new features have large positive impact on the performance of the other four classifiers when there are less than four instances per class in the training set.

The results of the experiment on Textures-3 show that the features of Compound-GP has slightly dropped the performance of all the six methods, whilst the features of One-shot GP show large negative impact on the performance of those classifiers. However, all the six classifiers have achieved significantly better performance using the features of the two new methods compared to those extracted by the Two-tier GP.

The features extracted by the One-shot GP show improvement in the performance of all the classifiers apart from the AdaBoostM1 classifier with handcrafted features. The features extracted by the Compound-GP method, on the other hand, show positive impact and have significantly improved the performance of all the six classifiers over the use of both the handcrafted and Two-tier features.

A.1.2 Group C Data Sets

The results show that the six classifiers have maintained their performances when the features of One-shot GP and Compound-GP are used on Textures-6 compared to Textures-1. Meanwhile, apart from AdaBoostM1, the use of handcrafted features have significant negative impact on the performance of those classifiers.

Similarly, the features extracted by both One-shot GP and Compound-GP show insensitivity to rotation on Textures-7. Also, a nearly consistent performance has been achieved by all those six classifiers when the handcrafted and Two-tier features are used.

Comparing the performance achieved of the six classifiers on Textures-8 and Textures-3, the classifiers show consistent or slightly improved performance when the handcrafted features are used; while the Compound-GP features have dropped the performance of almost all the six classifiers. The features extracted by One-shot GP, on the other hand, show similar behaviour to the handcrafted features; however, the gap between the two is significant. The inconsistency in the performance that the Compound-GP features show was expected due to having the same behaviour in the results of the first experiment on this data set.

On Textures-9, compared to Textures-4, the six classifiers have maintained their performances when the One-shot GP and Compound-GP extracted features were used. The handcrafted features show positive improvement in the performance of all the six classifiers compared to that was achieved on Textures-4.

A.1.3 Group D Data Set

Apart from NB, the features from the One-shot GP and Compound-GP methods have improved the performance of the other five classifiers on this data set. Although the Two-tier GP extracted features show improvement in the performance of those classifiers, this improvement is still not as good as those introduced by the two new methods (One-shot GP and Compound-GP). The handcrafted features show, in most cases, the worst performance on this data set.

A.1.4 Summary

The results of this experiment show that the feature extracted by the two new methods have improved the performance of the six classifiers in most cases. However, those features are also showing an inconsistency in other cases especially Textures-3 and its rotated version Textures-8. Most importantly, the features detected by the One-shot-GP and Compound-GP methods are not biased to a specific classifier, and are invariant to rotation, illumination, and scale variations to some extent.

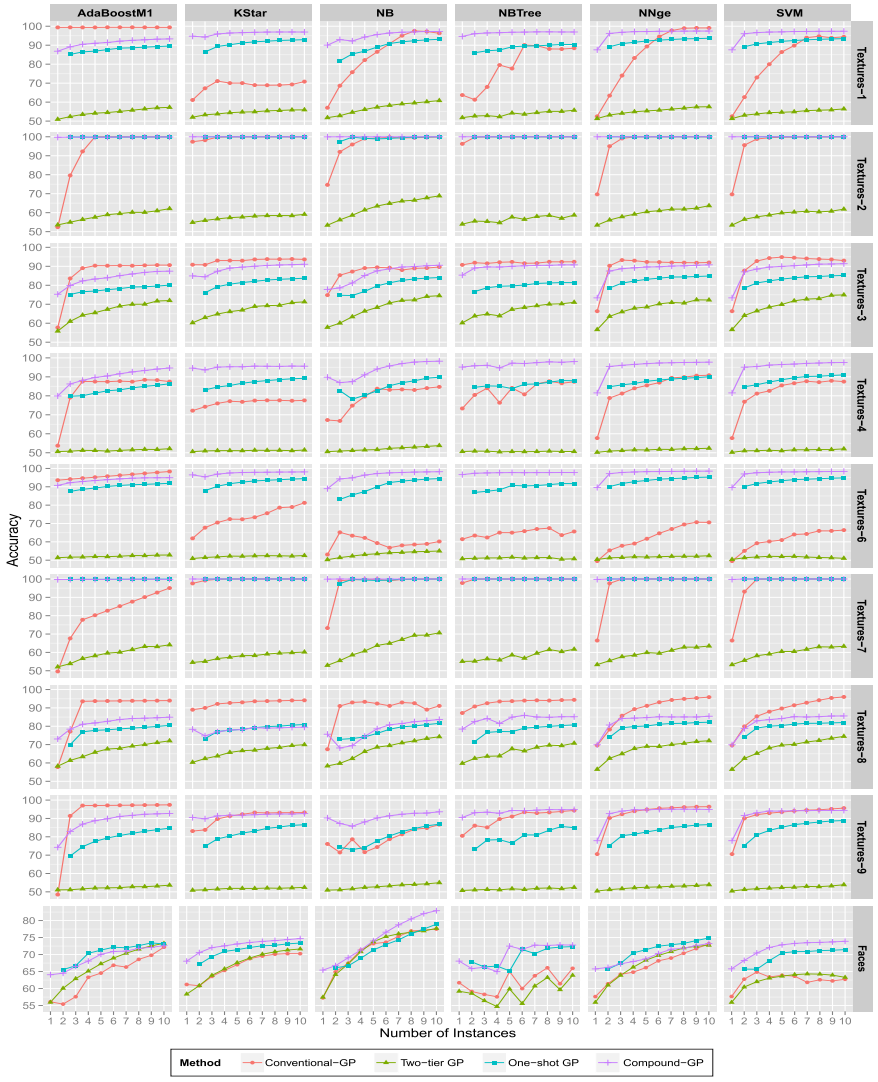


Figure 20: The average performance of the experimented methods on the data sets of the first, second, and fourth group using four different sets of features.

A.2 Training and Testing Time

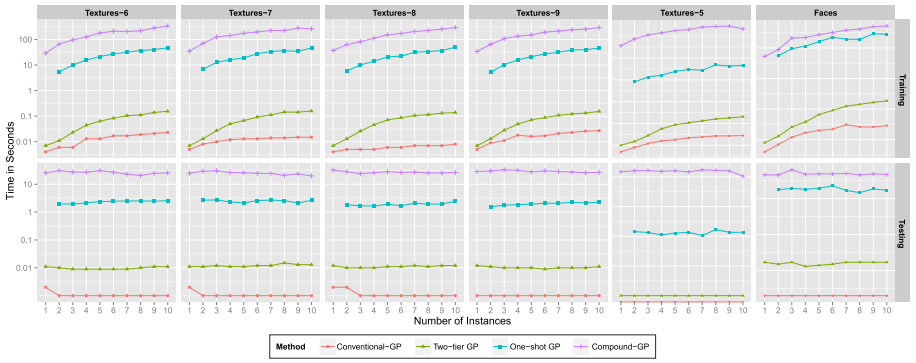


Figure 21: The average training and test time of the four GP-based methods on the data sets of the second, third, and fourth group.

A.3 Program Size

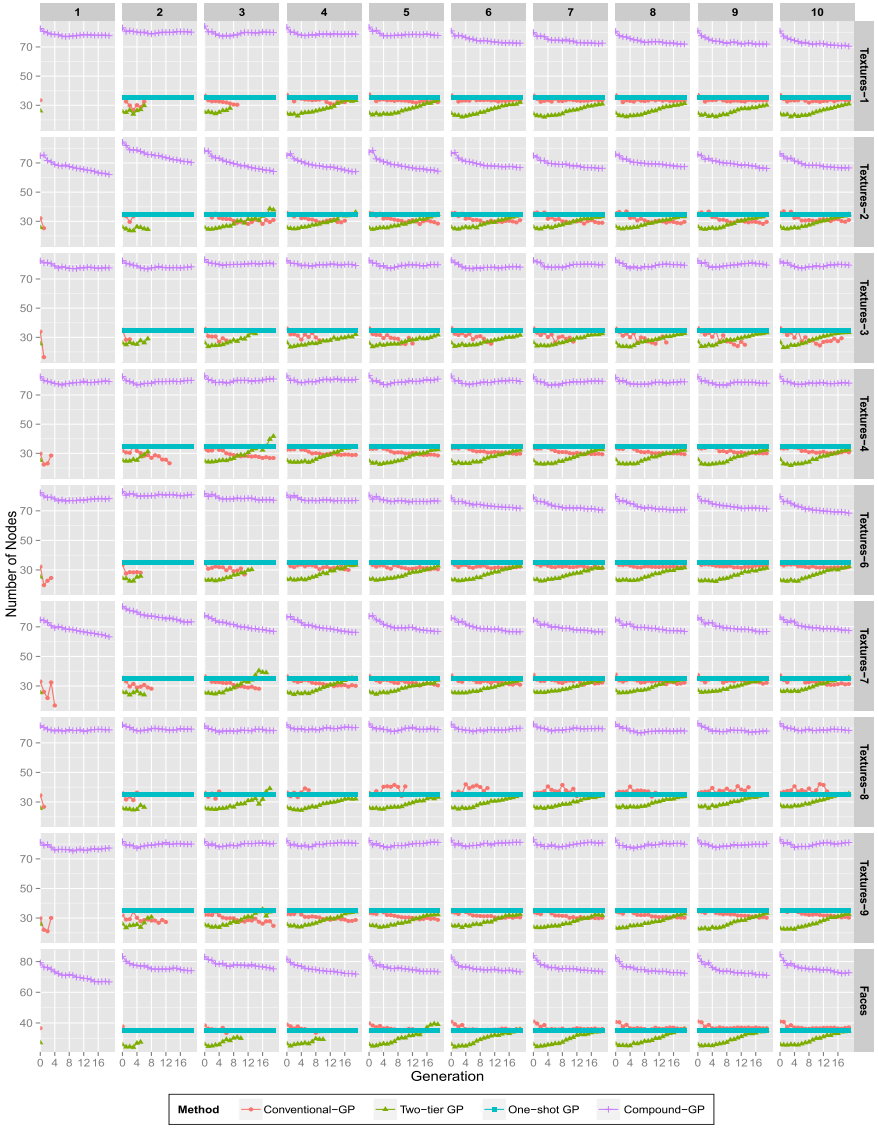


Figure 22: The average program size per generation of the four GP-based methods on the data sets of the first, second, and fourth group.