A One-shot Learning Approach to Image Classification using Genetic Programming

Harith Al-Sahaf, Mengjie Zhang, and Mark Johnston

Evolutionary Computation Research Group,
Victoria University of Wellington, PO Box 600, Wellington, New Zealand
{harith.al-sahaf,mengjie.zhang}@ecs.vuw.ac.nz
{mark.johnston}@msor.vuw.ac.nz

Abstract. In machine learning, it is common to require a large number of instances to train a model for classification. In many cases, it is hard or expensive to acquire a large number of instances. In this paper, we propose a novel genetic programming (GP) based method to the problem of automatic image classification via adopting a *one-shot learning* approach. The proposed method relies on the combination of GP and *Local Binary Patterns* (LBP) techniques to detect a predefined number of informative regions that aim at maximising the *between-class* scatter and minimising the *within-class* scatter. Moreover, the proposed method uses only two instances of each class to evolve a classifier. To test the effectiveness of the proposed method, four different texture data sets are used and the performance is compared against two other GP-based methods namely *Conventional GP* and *Two-tier GP*. The experiments revealed that the proposed method outperforms these two methods on all the data sets. Moreover, a better performance has been achieved by Naïve Bayes, Support Vector Machine, and Decision Trees (J48) methods when extracted features by the proposed method have been used compared to the use of domain-specific and Two-tier GP extracted features.

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

1 Introduction

The ability of recognising objects surrounding us represents one of the supreme tasks of human brains, specifically the visual system. Different parts of our bodies (i.e. eyes, hands, tongue, and brain) cooperate with each other in order to learn new objects. Humans are heavily relying on the visual system to capture the variety of object characteristics such as colour, shape, size, and distance. One study [3] shows that the brain of a six year child can recognise objects from more than 10^4 categories, and the learning process continues throughout life. Furthermore, the human brain has the ability to organise learnt objects into different informative groups.

Image classification is mainly concerned with the task of grouping images based on the similarity of its contents, which represents an important task in a variety of fields such as automatic face recognition, disease detection, and machine vision. The importance of this operation has attracted many researchers

over the last three decades; and a rich set of different methods have been proposed to the problems of object classification, detection, and recognition. However, performing image classification by machines remains difficult and not as easy as it is by humans.

Genetic Programming (GP) is an evolutionary computation method based on Darwinian principles of natural selection [10]. The promising results achieved using GP techniques to solve a variety of problems in different domains represent a major reason that motivated researchers to investigate those techniques even more over decades. However, the high computational cost of such techniques represents its major drawback.

Local Binary Patterns (LBP) aims at extracting image descriptors based on the relation between each pixel value in an image and its 8 (3×3 window) neighbouring pixels [16]. Since 1995, a number of LBP variants have been introduced and investigated in the literature. In Section 2 of this paper we will provide more details about this operator.

Generally, the task of learning or evolving models requires tuning a large number of parameters in order to capture features covering a diversity of different objects. It has been observed that a large number of training instances are required to adjust or estimate the models' parameters values [7], [22], [23], [24]. In many cases, the task of acquiring a large number of instances can be difficult, expensive or infeasible (e.g. ID-card identification and e-passport). Jain and Chandrasekaran [9] discussed the problem of the training set size in general. Raudys and Jain [20] investigated the effect of using a smaller training set on statistical pattern recognition and gave guidelines and recommendations for practitioners. Moreover, Duin [5] showed that one possible way to reduce the number of used training instances is via reducing the searching space size (i.e. number of features). The main difficulty of this approach is that it has to be handled by a domain expert with good background knowledge about the problem nature, which is in many cases hard and expensive.

Motivated by humans remarkable ability of learning relatively new objects using one or few images, researchers have tried to replicate this functionality in machines and termed it as *one-shot learning* [6]. This problem has been broadly researched and numerous methods are proposed in the field of, but not limited to, machine vision. To stimulate the ability of humans to rapidly learn numerous types of regularities and generalisations, Yip and Sussman [26] proposed a novel method towards fast learning in the domain of morphology. The method exploits the characteristics of sparse representations and forced constraints by a plausible hardware mechanism. A Bayesian-based method is proposed by Fei-Fei and Fergus [6] investigating the problem of object categories using the one-shot learning approach. The aim of their study was to use only one or very few of images to learn much information about a category. Their results show the system can effectively use information gathered during the learning phase to discriminate between unseen instances. Lake et al. [12] proposed a generative model motivated by the concepts of one-shot learning. The method shows how obtaining knowledge from previously learnt characters can be relied on to infer the use of

strokes for different characters composition. A hierarchical Bayesian model has been developed in [21] that uses a single training example to learn informative information about a complete category.

In this paper, we propose a hybrid GP based method that adopts the one-shot learning approach to the problem of image classification. The proposed method relies on the combination of GP and a LBP operator to handle the task of automatic binary classification in images using raw pixel values. We are interested in investigating the following objectives:

- To develop a program structure that has the ability to capture informative information of the two classes;
- To find a suitable design of a fitness function that minimises the *within-class* scatter and maximises the *between-class* scatter;
- To test the performance of the system against other GP based methods for automatic image classification (i.e. Two-tier GP [2]), and conventional GP using hand-crafted domain-specific features; and
- To investigate the ability of the proposed method for feature extraction by comparing the features extracted by the proposed method with domain-specific features and those extracted by Two-tier GP on three commonly used methods: Naïve Bayes, Support Vector Machines and Decision Trees (J48).

The rest of this paper is structured as follows. Section 2 briefly explains the Local Binary Patterns (LBP) operator and LBP histogram (LBPH). A detailed description of the proposed method is given in Section 3. Experimental design, data sets, baseline methods, and parameter settings are described in Section 4. Results are shown in Section 5. Section 6 concludes the paper.

2 Local Binary Patterns

The Local Binary Patterns (LBP) operator was originally proposed by Ojala et al. [16] in which the authors aimed at calculating each pixel value of an image based on the values of its neighbouring pixels. The basic LBP operator works in a 3×3 window, which consists of three steps: (1) Assign the value of a neighbouring pixel to 0 if it is less than the centre value of the window and 1 otherwise; (2) the values of the resulted matrix are then multiplied by the power of two in a clockwise direction; and (3) the value of the centre pixel is then replaced by the summation of the resulted values as shown in Figure 1. Formally, the LBP operator can be defined as

$$LBP_{n,r}(x_c, y_c) = \sum_{j=0}^{n-1} 2^j s(V_j - V_c) \tag{1}$$

where r is the radius and n is the number of neighbouring pixels. The values of x_c and y_c represent the coordinate of the centre pixel of the current window. The j^{th} pixel value is denoted as V_j whilst the value of the centre pixel is V_c. The function $s(x)$ returns 1 if $x \geq 0$ and zero otherwise. A variety of LBP operators have been proposed in the literature that differ in the way of thresholding the neighbouring pixel values, the size of the window and radius, and calculating the

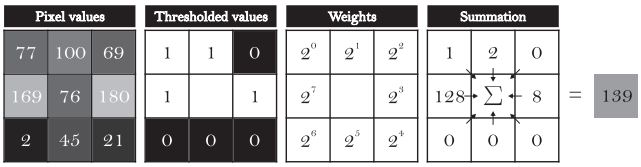

Fig. 1. An example of the required steps to extract an LBP code.

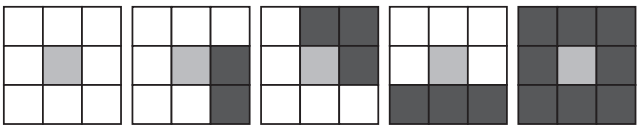

Fig. 2. Some examples of texture primitives of different uniform LBP codes.

final value (more details can be found in [18]). An important extension of basic LBP is known as *uniform patterns* [17] that is denoted as $LBP^{u2}_{n,r}$. A circularly traversed pattern is considered to be *uniform* if bits equal 1 are consequent. For example, if we have the code 00000000, then 00011000, 00110000, and 00111000 are all uniform codes. Uniform codes are important for two reasons: (1) the frequency of uniform codes is higher than non-uniform ones [1]; hence, omitting non-uniform patterns reduces the number of possible LBP codes significantly; and (2) uniform codes can be used to detect different texture primitives as shown in Figure 2.

In [13], a grey-scale invariant intensity-based descriptor is proposed named *NI-LBP*. The main difference between basic LBP and NI-LBP is that the latter uses the mean value of all pixels in a window as the threshold instead of only the value of the centre pixel. In this study, we use the same descriptor (NI-LBP) to extract the pattern of each pixel.

Traditionally, the frequencies of LBP codes appearance are used to form Local Binary Pattern Histogram (LBPH) [8]. For example, if we have an 8-bit codes then we can label 256 different labels starting from 0 (00000000) up to 255 (11111111). Each label of the LBPH is considered as a bin that accumulates the number of occurrences of a specific value or label. By omitting non-uniform codes, there will be only 59 bins (58 uniform codes plus one bin for all non-uniform ones). Let $LBP_{n,r}(i,j)$ identify the calculated LBP code of the $pixel(i,j)$ where $0 \leq i < N$ and $0 \leq j < M$ of an $N \times M$ image; then the histogram h of length L of the entire image can be formally defined as

$$h(l) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (LBP_{n,r}(x_i, y_j) = l) \qquad l = 0, 1, ..., L-1 \tag{2}$$

LBP histograms can either be calculated over the entire image or combine multiple histograms that are obtained from different areas (mostly non-overlapping). The latter approach is used in this study. The length of the histogram vector varies based on the number of areas as shown in Figure 3.

3 The New Method

The representation of a new *one-shot* GP-based method is described in this section. The design of the GP individual that extracts LBP histograms from a

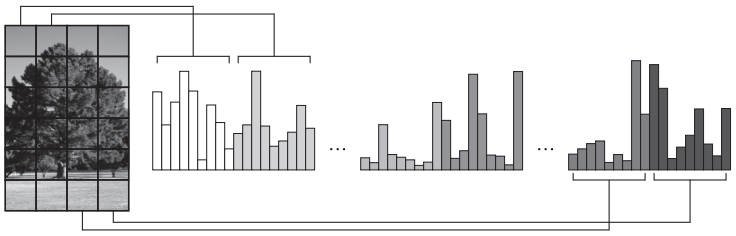

Fig. 3. An example of an LBP histogram that constructed as the combination of 24 sub-histograms (one per region).

raw image pixel value and calculates the similarity of two histograms occupies the first part of this section. The rest of the section gives the terminal and function sets, and the fitness function.

3.1 One-shot GP and Program Structure

An individual is made up of three types of non-terminal nodes: (1) controller node; (2) histogram node; and (3) area node. Figure 4 shows a general structure of an individual for binary classification. Each individual has a set of *controlling instances*, one for each class notated as $controller^X$ where X represents the class label of that instance. The training process starts by extracting the histogram of each controller instance depending on the detected areas by the current individual. Hence, in our case of binary classification we have $histogram^A$ and $histogram^B$ where A and B are the class labels of the two classes. The controller histogram will be compared with the histogram of each and every instance in the training and test sets. The system then iterates over all instances of the training set and for each instance there will be H histograms (equal to the maximum number of classes) one from each branch of the individual tree. The distance between each of the corresponding histograms (controller and instance resulted from the same branch) is then calculated and passed over to the *controller* node (more details of this step in the fitness function subsection). To meet the condition of *one-shot learning* (only one or a few number of instances), the training set is made up of two randomly selected instances of each class. One is used as a controller and the other is used to reflect the goodness of the evolved individual on unseen data (validation set).

The size of the evolved individual under this design is fixed in terms of depth and width of the tree. The number of branches (children of the controller node) increases only if more classes are added which will increase the size of the tree horizontally. The number of detected areas has the same impact and will result in a wider tree. Hence, the positions and sizes of the detected areas represent the only dynamic part of the evolved individual.

3.2 Function Set

The function set in this study is *strongly-typed* [15], and is composed of three different types of nodes in which each type has its own functionality. The *controller* node represents the first type, which restricted to be the root of the individual

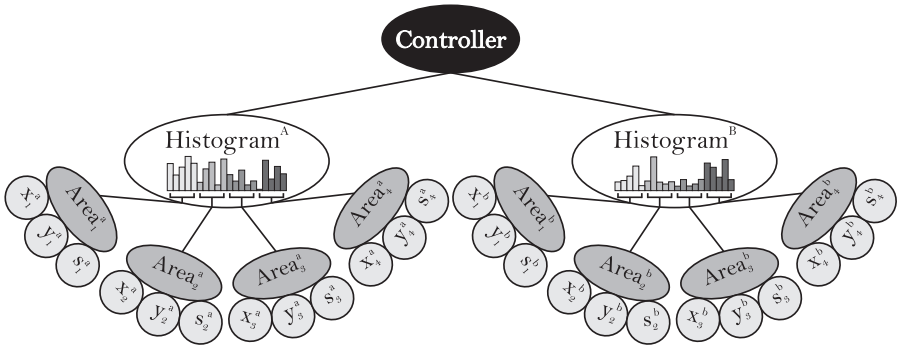

Fig. 4. The program structure of a *one-shot GP*, which consists of two *histogram* nodes that each has four *area* nodes.

tree. Hence, each individual has one and only one node of this type. The output of this node is a vector of double-precision values for each instance. The length of the resulted vector is fixed and equivalent to the number of classes. The type of input values of this node is the extracted LBP histograms from its children. The second type is the *histogram* node that represents the root node of each branch of the controller node children. *Histogram* nodes are responsible for extracting the LBP histograms by combining the received sub-histograms from each of its children. Each histogram node has a predefined number of *area* nodes that represent the third type of the function set nodes. Each *area* node detects an area of the image, calculates the LBP histogram, and passes it over to its parent (histogram node).

3.3 Terminal Set

The terminal set is also made of three nodes: 1) X-Coordinate; 2) Y-Coordinate; and 3) Window size. Each of these nodes is of type *integer* value and has its own restrictions. For example, *x-cord* and *y-cord* nodes take values in the range $[1, N-2]$ and $[1, M-2]$ respectively, where N and M represent the width and height of an image respectively. The *size* node is restricted to be in the range between 3 and 15.

3.4 Fitness Function

The fitness function used here aims at minimising the *within-class* scatter and maximising the *between-class* ones as shown in Equation (3). The fitness value is proportional to the distance of the same class instances and inversely proportional to distance between instances of different class. However, the denominator of the function is assigned to a small value (0.0001) if it was 0 in order to prevent the division by zero. Based on this design, the smaller the *within-class* distance or the greater the *between-class* distance, the better the fitness.

$$Fitness(I) = \sum_{j=1}^{T} \frac{\bar{\chi}^2(h_j^x, h_c^x)}{\bar{\chi}^2(\bar{h}_j^x, h_c^{\bar{x}})} \tag{3}$$

Here I is the current individual being evaluated, T is the total number of training instances, $\bar{\chi}^2(.)$ is the distance measure, h_j^x and $\bar{h}_j^x$ are the extracted

histograms of the j^{th} instance, and h_c^x and $h_c^{\bar{x}}$ are the two classes controller histograms. The motivation for this fitness function is to give the GP process the opportunity to detect more distinctive regions during the training process. Note that the use of *accuracy* makes the system suffer from the problem of over-fitting[1].

In [25], the χ^2 distance measurement shown in Equation (4) is used to measure the similarity between two histograms (bin-by-bin approach). To give the system more flexibility, a shape based version of this formula is used to make it shifting invariant (unbinned approach) via using the *mean* and *standard deviation* as shown in Equation (5) [19].

$$\chi^2(h^a, h^b) = \frac{1}{2}\sum_{i=0}^{\bar{B}-1}\frac{(h_i^a - h_i^b)^2}{h_i^a + h_i^b} \tag{4}$$

$$\bar{\chi}^2(h^a, h^b) = \frac{(\mu^a - \mu^b)^2}{\sigma^a + \sigma^b} \tag{5}$$

where h^a and h^b are the two histograms, $\bar{B}$ is the total number of bins, μ^a and μ^b are the mean of histogram h^a and h^b respectively, and σ^a and σ^b are the standard deviation of histogram h^a and h^b respectively.

4 Experimental Design

The main focus of this section is to highlight the parameter settings and data sets that were used to evaluate the proposed method.

4.1 Data Sets

In order to test the effectiveness of the proposed method, four different texture image-based data sets are used. Six different classes of the *Kylberg Texture Dataset* [11] are selected to form three data sets in this paper. Originally, this data set is composed of 28 different classes that each consists of 160 instances. Each instance of this data set is a grey-scale image of size 576×576 pixels that was resampled to 57×57 pixel in our experiments. Furthermore, we equally divided the total number of instances of each class between the training and test sets. Hence, each set is made of 160 instances in total (2 classes $\times$ 80 instances).

The *stoneslab1* and *wall1* classes are picked to represents the first data set in this study and we named it as *Textures-1*. Figure 5(a) shows a sample of these two classes. *Textures-2* represents the second set that is made of the *rice2* and *sesameseeds* classes as shown in Figure 5(b). The training and test sets of the third set are made of the *blanket1* and *canvas1* class instances which we refer to as *Textures-3* in this paper and some of its instances are shown in Figure 5(c).

The instances of the fourth data set were taken from the *Columbia-Utrecht Reflectance and Texture* (CUReT) data set [4]. The CUReT data set is made of 61 classes in total that only *brown bread* and *sponge* classes are selected to form the fourth data set that we call it *Textures-4* in this study as shown in Figure 5(d). The size of each instance is a 200×200 pixel and there are 81 instances

[1] The GP process can easily detects areas that can be relied on to discriminate between a small number of instances. However, exposing such a model to a large set of unseen instances can result in a very poor performance.

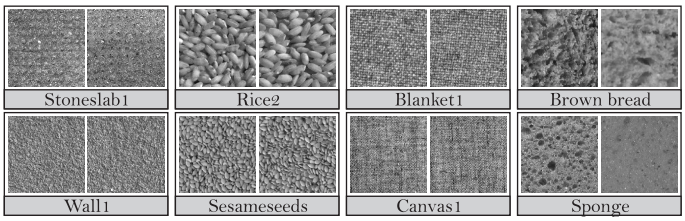

Fig. 5. Samples of the four texture data sets.

in each class. Hence, the training and test sets consist of 40 and 41 instances of each class respectively.

4.2 Baseline GP Methods

The performance of the proposed system has been compared against two other GP-based methods. The details of those two methods are discussed below.

Conventional GP. The conventional GP method is applied on a set of hand-crafted pre-extracted features. The *mean* and *standard deviation* of the entire image, the four quadrants, and the centre part of each instance have been calculated. The extracted 12 values are then stored in a text file which is later fed to a GP package to evolve a model. Then the evolved model is tested on the test set that was created in a similar way to the training set.

Two-tier GP. Al-Sahaf et al. [2] have developed a GP-based method for automatic feature extraction and selection, and image classification named it as *Two-tier GP*. The system automatically detects areas of different shapes and sizes, and uses different functions to extract the features from pixel values of those areas. The Two-tier GP method showed superior performance over all other competitive methods in that study [2]. Hence, we will compare the performance of the *one-shot GP* with this Two-tier GP method.

4.3 Parameter Settings

The GP parameters of the three methods in all conducted experiments are shown in Table 1. As shown in the table, some of the parameters are not applicable in the case of the proposed method due to the design restrictions discussed in Section 3. The rates of crossover, mutation and reproduction are 0.80, 0.19, and 0.01 respectively. The *tournament* selection method of size 7 is used to maintain population diversity.

4.4 Evaluation

Two different set of experiments are conducted that each aims at testing a different objective. In the first set, the focus was toward investigating the performance of the proposed method against the two baseline (GP) methods. Hence, each of the three methods (two baseline methods and the proposed one) has been evaluated on the four data sets described at the beginning of this section. Only four instances (two of each class which represents the smallest number based on the current design) are randomly selected to play the role of training set. The best evolved individual of each run is then tested against the unseen instances (test

Table 1. The GP Parameters of all experiments

Parameter	Value	Conve-GP	Two-tier GP	One-shot GP
Generations	20	✓	✓	✓
Population Size	1000	✓	✓	✓
Tree min-depth	2	✓	✓	✗
Tree max-depth	10	✓	✓	✗
Crossover min-depth	2	✓	✓	✗
Crossover max-depth	10	✓	✓	✗
Mutation min-depth	2	✓	✓	✗
Mutation max-depth	10	✓	✓	✗
Initial Population	Ramped half-and-half	✓	✓	✗

set) and the accuracy is reported. In the case of both baseline methods, the value of 0 is used to divide the results space such that all negative values and 0 are considered representing one class while the other class is considered having positive values. However, the instance is evaluated as belonging to the branch having a smaller distance of the evolved program in the case of the proposed method.

This process is repeated for 30 independent runs using the same training and test sets. The average performance of the best evolved programs of the 30 runs on the test set is then recorded. The use of different instances to evolve the model has large impact on the performance of the resulted program. Hence, the entire procedure is repeated 10 times ($10 \times 30 = 300$ runs) using different instances in the training set each time while test set kept the same. At the end of all 10 repetitions the highest and lowest average performances are reported, and the mean and standard deviation statistics are calculated as shown in Table 2. The same procedure is used to evaluate all three methods using exactly the same training and test instances each time.

In the case of the second experiment, the best evolved program at the end of each of the 30 runs of the first experiment is used to extract features of the detected areas in the case of the proposed and Two-tier GP methods. Hence, there will be 10 different individuals as the first experiment is repeated 10 times. In the case of Two-tier GP method, the calculated values of the *aggregation* nodes are used to represents the extracted features similar to the work in [2]. In the case of one-shot GP method, the calculated differences between the two controller histograms and resulted histograms of each instance are considered to be the extracted features. This process is also repeated 10 times for each individual due to having 10 different training sets (as mentioned in above). It is important to notice that, extracted features by any two individuals of the test set are different due to different detected areas in terms of position, size, and/or number. In addition to feature sets extracted by those two methods, domain-specific features are also used in this experiment (as stated earlier). The extracted features by each of the three methods are then fed to three different classification methods, i.e., Naïve Bayes (NB), Support Vector Machines (SVM) and Decision Trees (J48). The goal of this set of experiments is to investigate the capability of the new method for automatic feature extraction.

The proposed method, as well as the two other GP-based methods, were all implemented using the platform provided by *Evolutionary Computing Java-based* (ECJ) package [14]. This is mainly because implementing strongly-typed GP in this package is relatively easy.

Table 2. Comparison between conventional GP, Two-tier GP, and One-shot GP

	Conventional GP (%)			Two-tier GP (%)			One-shot GP (%)		
	Min	Max	$\bar{x} \pm s$	Min	Max	$\bar{x} \pm s$	Min	Max	$\bar{x} \pm s$
Texture-1	52.73	66.37	57.10 ± 4.43	49.29	53.56	51.52 ± 1.23	82.37	92.23	**87.76** ± 3.65 †§
Texture-2	47.92	63.98	55.50 ± 4.76	50.29	53.33	51.30 ± 1.05	98.88	99.81	**99.38** ± 0.34 †§
Texture-3	50.77	66.48	57.75 ± 4.85	48.29	54.06	51.62 ± 1.69	64.92	82.50	**76.95** ± 6.09 †§
Texture-4	44.72	63.21	56.35 ± 5.94	50.41	55.08	53.05 ± 1.49	54.11	93.25	**81.12** ± 14.64 †§

5 Results and Discussions

This section highlights the results of the experiments. Here, t-tests have been used to compare the difference between the performances of the proposed method and each of the two baseline methods. The "†" and "§" signs in the tables appear if the performance of the proposed method is significantly different than conventional GP and Two-tier GP methods respectively. The bolded numbers in the tables represent the highest mean value amongst the three methods.

Table 2 shows the average performance of the 10 repetitions (each with 30 independent runs) gained from the first experiment on the four data sets. The proposed method has significantly outperformed both conventional GP and Two-tier GP methods on all of the data sets. Moreover, the proposed method, in its worst case, shows better performance than the best performance of the Two-tier GP method on all experimented data sets. However, in the case of conventional GP this property does not hold on the Textures-3 and Textures-4 data sets, but the best performance of the proposed one-shot GP is 16% and 30% higher than the conventional GP and Two-tier GP respectively.

The performance statistics of the NB, SVM and DT (J48) on the four data sets using three different sets of features are shown in Table 3. The three classifiers show significantly better performance on all data sets using the features extracted by the proposed method over using those extracted by the Two-tier GP. However, this property holds for NB and SVM in Textures-1 and Textures-4 data sets in the case of comparing the use of domain-specific features and the features extracted by the proposed method. In Textures-3 data set, the three classification methods have achieved better results using domain-specific features over features extracted by the other two methods, which is opposite to the case of Textures-4. We can observe that the features extracted by the new method have positive influence on the performances of both NB and SVM (scored first in three out of four data sets). This is also true in the case of DT (J48) in the Textures-4 data set. In all other cases, these methods have the second rank using the set of extracted features by the proposed method.

6 Conclusions

In this paper, a one-shot learning approach has been adopted to the problem of automatic image classification. The proposed method uses the combination of GP and LBP techniques to evolve a classifier. Moreover, the fitness function has been designed to maximises the distance of *between-class* and minimises the *within-class* distance. We used four texture data sets to evaluate the performance of the proposed method. The conventional GP and Two-tier GP methods have been used as competitive methods. Two experiments have been conducted

Table 3. The performance of Naïve Bayes, Support Vector Machine, and Decision Trees (J48) classification methods on the four texture data sets using domain-specific features, and features extracted by each of the Two-tier GP and One-shot GP methods

		Domain-specific (%)			Two-tier GP (%)			One-shot GP (%)			
		Min	Max	$\bar{x} \pm s$	Min	Max	$\bar{x} \pm s$	Min	Max	$\bar{x} \pm s$	
Tex-1	NB	53.75	92.50	70.00 ± 12.18	47.50	66.25	55.56 ± 5.65	83.75	96.25	**91.07** ± 4.29	†§
	SVM	47.50	94.38	74.64 ± 13.67	50.00	65.63	55.25 ± 5.44	88.13	98.75	**92.69** ± 3.72	†§
	J48	66.25	97.50	**88.13** ± 10.87	46.88	64.38	53.82 ± 6.55	51.88	90.00	79.00 ± 12.64	§
Tex-2	NB	96.25	100.0	98.50 ± 1.36	46.88	69.38	55.13 ± 6.35	93.75	100.0	**98.69** ± 2.07	§
	SVM	83.75	100.0	96.75 ± 4.98	48.75	58.13	52.50 ± 3.85	93.13	100.0	**99.13** ± 2.15	§
	J48	43.13	92.50	**71.50** ± 15.78	50.00	58.75	53.63 ± 3.59	53.75	96.88	69.63 ± 16.16	§
Tex-3	NB	61.25	92.50	**79.00** ± 12.39	36.25	63.75	53.57 ± 8.06	60.00	85.00	76.82 ± 8.56	§
	SVM	55.00	96.25	**85.75** ± 12.47	55.00	65.63	55.44 ± 6.05	78.75	86.25	82.94 ± 2.65	§
	J48	56.88	93.75	**76.25** ± 12.12	39.38	67.50	53.50 ± 8.17	52.50	86.25	69.56 ± 12.25	§
Tex-4	NB	42.68	82.93	61.22 ± 11.82	46.34	71.95	64.27 ± 8.81	36.59	98.78	**82.44** ± 19.22	†§
	SVM	42.68	51.71	61.59 ± 16.61	50.00	74.39	60.00 ± 7.53	67.07	100.0	**88.17** ± 10.94	†§
	J48	14.63	86.59	60.37 ± 19.89	29.27	76.83	55.98 ± 14.67	50.00	95.12	**75.00** ± 16.90	§

that aim at investigating different objectives. The performance of the proposed method is investigated in the first experiment and compared to each of the two baseline methods. The results of this experiment show superior performance of the new method over the other two competitive methods. The second experiment aims at investigating the effectiveness of the extracted features by the proposed method on the performance of Naïve Bayes, Support Vector Machines, and Decision Trees (J48) classification methods. The resulted performances are also compared against the use of both domain-specific features and features extracted by the Two-tier GP method. The results of this experiment show that significantly better or at least comparable performance can be achieved when features extracted by the proposed method are used.

References

1. T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, 2006.
2. H. Al-Sahaf, A. Song, K. Neshatian, and M. Zhang. Extracting image features for classification by two-tier genetic programming. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2012.
3. I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.
4. K. J. Dana, B. van Ginneken, S. K. Nayar, and J. J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, 1999.
5. R. P. Duin. Small sample size generalization. In *Proceedings of the Ninth Scandinavian Conference on Image Analysis*, volume 2, pages 957–964, Uppsala (Sweden), 1995.
6. L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.
7. R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 264–271, June 2003.
8. S. Hegenbart, S. Maimone, A. Uhl, A. Vécsei, and G. Wimmer. Customised frequency pre-filtering in a local binary pattern-based classification of gastrointestinal

images. In *Medical Content-Based Retrieval for Clinical Decision Support*, volume 7723 of *Lecture Notes in Computer Science*, pages 99–109. Springer, 2012.
9. A. Jain and B. Chandrasekaran. Dimensionality and sample size considerations in pattern recognition practice. In *Classification Pattern Recognition and Reduction of Dimensionality*, volume 2, pages 835–855. Elsevier, 1982.
10. J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.
11. G. Kylberg. 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, 2011.
12. B. M. Lake, R. Salakhutdinov, J. Gross, and J. B. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, pages 2568–2573, Austin, TX, 2011.
13. L. Liu, L. Zhao, Y. Long, G. Kuang, and P. Fieguth. Extended local binary patterns for texture classification. *Image and Vision Computing*, 30(2):86–99, 2012.
14. S. Luke. *Essentials of Metaheuristics*. Lulu, second edition, 2013. http://cs.gmu.edu/∼sean/book/metaheuristics/.
15. D. J. Montana. Strongly typed genetic programming. *Evolutionary Computation*, 3(2):199–230, 1995.
16. T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29(1):51–59, 1996.
17. T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.
18. M. Pietikäinen, A. Hadid, G. Zhao, and T. Ahonen. Local binary patterns for still images. In *Computer Vision Using Local Binary Patterns*, volume 40 of *Computational Imaging and Vision*, pages 13–47. Springer London, 2011.
19. F. C. Porter. Testing Consistency of Two Histograms. *ArXiv e-prints*, pages 1–35, 2008.
20. S. J. Raudys and A. K. Jain. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):252–264, 1991.
21. R. Salakhutdinov, J. B. Tenenbaum, and A. Torralba. One-shot learning with a hierarchical nonparametric bayesian model. *Journal of Machine Learning Research - Proceedings Track*, 27:195–206, 2012.
22. H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *Proceedings of Computer Vision and Pattern Recognition*, pages 1746–1759. IEEE Computer Society, 2000.
23. P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceeding of Computer Vision and Pattern Recognition*, pages 511–518. IEEE Computer Society, 2001.
24. M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *European Conference Computer Vision*, volume 2, pages 101–108. Springer Berlin Heidelberg, 2000.
25. J. Xie, D. Zhang, J. You, and D. Zhang. Texture classification via patch-based sparse texton learning. In *IEEE International Conference on Image Processing (ICIP)*, pages 2737–2740, 2010.
26. K. Yip and G. J. Sussman. Sparse representations for fast, one-shot learning. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 521–527. AAAI Press / The MIT Press, 1997.