Two-tier Genetic Programming for Automatic Feature Extraction, Feature Selection and Image Classification

Harith Al-Sahaf
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
Email: harith.alsahaf@gmail.com

Kourosh Neshatian
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
Email: kourosh.neshatian@ecs.vuw.ac.nz

Mengjie Zhang
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
Email: mengjie.zhang@ecs.vuw.ac.nz

***Abstract*—Genetic Programming (GP), a flexible evolutionary learning paradigm, has been vastly used in image classification domain. In this domain, the learning process is usually performed using a selected subsets of *extracted features*. Feature extraction and selection are widely practised as pre-processing steps in order to cope with high levels of redundancy and lack of class separation in raw pixel information. These pre-processing, however, require domain-specific knowledge and often domain expert intervention. In this paper, we propose a *two-tire* GP approach that automatically performs feature extraction, implicit feature selection, and classification. The new approach does not depend on domain-specific features extracted by human; instead, a single evolved program performs feature extraction and classification in one phase. The result of our experiments show that the proposed approach performs better than the domain-specific pre-extracted features approach. It also outperforms previous proposed GP-based feature extraction and classification systems.**

***Index Terms*—genetic programming; feature extraction; feature selection; image classification.**

I. Introduction

There are three important steps in many computer vision systems [1]: i) *image capturing* which is the task of capturing the emitted or reflected light, by using a compound optical-sensory system; ii) *feature extraction* which is concerned with transforming (mapping) the captured image into a new input space such that the new space contains discriminative information regarding objects of interest; iii) *Classification* which is concerned with identifying images (or cut-outs) of different categories based on the extracted features and a pre-trained classifier. Image classification is an important task in computer vision with application ranging from medical applications to robotics and automation [2].

Raw image pixel values are seldom used directly for image classification task due to high data redundancy and large search space [3]. *Feature extraction and selection* refer to the process of transforming raw pixel data into a new (low-dimensional) domain in which classification algorithms can perform better. Image classification algorithms—among them Very often domain-expert intervention is required in order to extract useful features for image classification. Feature selection is required in order to find a sub-set of extracted features that have significant effect in distinction between different classes.

Genetic Programming (GP) is an Evolutionary Computing (EC) domain-independent algorithm that simulates biological evolution (Darwinian principle) to automatically solve a user-defined problem [4][5][6][7]. GP methods have been used broadly and in different ways to solve simple and complex problems such as classification[2][8][9], object detection[10][11], feature extraction[12][13][14], and feature selection[15][16][17]. The use of GP methods to perform feature extraction has also been investigated. Sherrah et al. [17] have proposed a GP-based feature extraction system using generalised linear machine. Their result show that their method yielded better performance than using domain-expert-designed features. A GP-based approach to performing feature extraction from multiple sources has been proposed by Szymanski et al. [13]. Their proposed system was able to achieve very good results. Lam and Ciesielski [18] have used a GP-based approach to developing algorithms for texture feature extraction. The evolved programs were able to achieve similar or slightly better results than domain-specific features.

A two-stage GP-based scheme has been proposed by Oechsle and Clark [19]. Their system performs the operation of feature extraction in the first stage, and classification in the second stage. Both of these stages were GP-based; however, the system requires human intervention to reformulate the extracted features in the first stage before it can be used to in the second stage. Atkins et al. [4] proposed a GP-based system that performs feature extraction and classification using a single evolved program. Their system has three tiers and performs image filtering in the first tier, feature extraction in the second tier, and classification in the third tier. The results of their experiments show that the system was able to achieve comparable performance to the domain-specific pre-extracted features.

The paper aims to develop a domain independent approach to the use of genetic programming for image classification tasks. Instead of using manually crafted, task specific features as inputs to the GP programs, this approach will directly use the raw image pixels as inputs. Unlike the existing approaches, this approach will design a novel program structure that consists of a feature extraction part and a classification part with the goal of evolving programs each of which can automatically perform image region selection, feature extraction and classification without human intervention. This approach will be examined and compared with the GP approach using manually crafted task specific features and a recent GP method [4] on four benchmark image classifications problems of varying difficulty. Specifically, we will investigate:

1) how such a program structure can be designed and developed in GP, including the design of the function set and terminal sets;
2) whether the new GP approach with raw image pixels can achieve better classification accuracies than GP using task specific features;
3) whether the new GP approach outperforms the recent GP methods on these problems;
4) whether the new GP approach can automatically evolve/leave good programs that can be easily interpreted for image classification;

The rest of this paper is organised as follows. The structure of the proposed system is discussed in details in section 2. The experiment design, GP parameters, datasets and fitness function are explained in section 3. The results and analysis of the conducted experiments are discussed in section 4. Finally, the conclusions of this paper work are shown in section 5.

II. The New Method

A. Individual program structure

The individual program consists of two tiers (layers): *aggregation* and *classification*. The nodes in these two tiers must appear in a specific order: classification nodes must be in the upper tier (including the root node), and aggregation nodes must be in the lower tier of an individual program tree. There is no restriction on the thickness of these tiers (the depth or the number of nodes in each tier) as long as the overall depth of a program tree is within the limit. On the other hand, the number and type of children of each node is restricted to the number and type of parameters that each node accepts. Figures 1 shows a simple individual grammar tree, and illustrates the roles of the two tires.

Aggregation functions occupy the lower tier of an individual program tree, which means they act on raw pixel values. The major task of this tier is to perform feature extraction. An aggregation function takes a part of an image (based on a shape and a window size) and converts it to a single floating double-precision value. Since the type of input and output values are different; no more than one aggregation function can be used in a hierarchy. Classification functions reside in the upper tier and sit on top of aggregation functions or other classification functions. The input and output of these functions are double-precision floating values. Therefore, the vertical thickness of

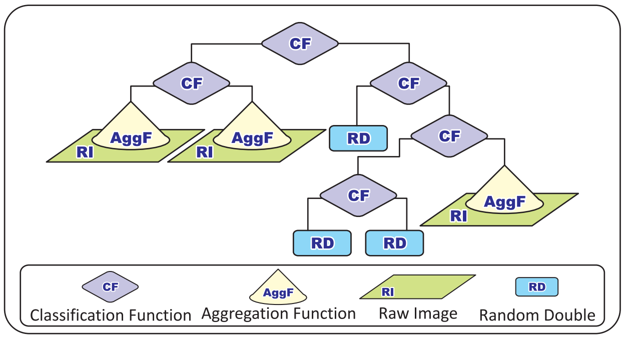


Fig. 1: An individual grammatical structure

this tier can vary and several classification functions may exist in one hierarchy. In this paper we only consider binary classification problems. For a given image, if the returned value at the root of a program tree is *negative*, the image is classified as belonging to class *A* (usually the background class); otherwise, the example is classified as belonging to class *B* (usually the object of interest).

B. Function Sets

1) Aggregation Functions: Table I lists the aggregation functions. All functions in this category have the same number and type of parameters. From left to right these parameters are: an image to extract feature from, two integers (x,y) representing the coordinate, the shape of the ROI area (window), and an integer represents the ROI (for example a rectangular window). All aggregation functions return a double-precision floating number to be used by a classification function. The window size is restricted based on the shape of the ROI. In the case of using *square*, *circular* and *rectangular* shapes the window size cannot be smaller than 3. However, in the case of *row* and *column* shapes the window size is restricted to one (the thickness) while the length is not restricted. The coordinate represents the centre of the ROI for square and circular shapes; it represents the starting point for the other three shapes. The coordinate values must reside in the image boundaries; x is in the interval [0,$Image_{width}$] and y in the interval [0,$Image_{height}$]. The window is truncated if it exceeds the image boundaries. Commonly-used statistical features such as *mean* and *standard deviation* [9][19][20] are implemented for aggregation. Other aggregation functions that haven proven effective in [4] such as *min*, *max* and *median* are also implemented.

TABLE I: Aggregation Functions Set

Function	Parameters	Description
AggMean	Image, X, Y, Shape, Size	Returns the mean value of the pixels determined by the window size, shape, and coordinates (x,y)
AggMed	Image, X, Y, Shape, Size	Similar to $AggMean$ except it returns the median value
AggStDev	Image, X, Y, Shape, Size	Similar to $AggMean$ except it returns the standard deviation value
AggMax	Image, X, Y, Shape, Size	Similar to $AggMean$ except it returns the maximum value
AggMin	Image, X, Y, Shape, Size	Similar to $AggMean$ except it returns the minimum value

2) Classification Functions: Table II lists the classification functions. Except for the conditional *IF* function that takes three parameters, all classification functions take two parameters. The type of input and output of these functions are double-precision floating point numbers. The $+$, $-$, $\times$ and $\div$ functions have their usual arithmetic meaning except for the $\div$ operator which is a protected division; that is, it returns zero if the divisor is zero. The conditional IF function returns the value of the second parameters if the value of the first parameter is *negative*; otherwise, it returns the value of the third parameter.

TABLE II: Classification Functions Set

Function	Parameters	Description
$+$	Double, Double	Performs the regular arithmetic addition operation
$-$	Double, Double	Performs the regular arithmetic subtraction operation
$\times$	Double, Double	Performs the regular arithmetic multiplication operation
$\div$	Double, Double	Performs a protected arithmetic division operation
IF	Double, Double, Double	Returns the value of the second parameter if the first parameter is less than zero, otherwise, it returns the value of the third parameter

C. Terminal Set

In order to meet the requirements of the proposed system, the terminals were designed differently and more specifically than other GP-based systems. Table III shows the terminal set. *Original* represents a 2D-array that contains the pixel values of the original image. *Random double* generates a random value in the interval [0,1]. *Shape* represents the shape of the aggregation window that can be one of five shapes: square, circular, rectangular, row and column. *AggWindow* is a restricted randomly generated integer value that determines the ROI. The restriction of this value can vary under different situation upon the window shape. For example, in all shapes except for rectangular, this value is restricted to be in the interval [3,$max(Image_{width}, Image_{height})$]; while in the case of using rectangular shapes, this value is ignored and the children of the rectangular node (*width* and *height*) will be used instead. The coordinate (x,y)—which are uniformly randomly generated—are used differently depending on the shape of ROI. For example, in the case of square and circular shapes these values represent the coordinates of the central pixel of the aggregation window. In the case of rectangular, row and column areas, these values represent the coordinates of the starting (top-left) pixel. However, the coordinates must reside inside the image boundaries, which means x must be in the interval [0,$Image_{width}$] and y in the interval [0,$Image_{height}$].

TABLE III: Terminals Set

Terminal	Type	Description
Original	Image	2D-array containing the image pixel values
RandDouble	Double	Returns a randomly generated double value in the interval [0,1]
Shape	String	Returns the shape of the $AggWindow$ {square, circular, rectangular, row, column}
Coords	Integer	A pair (X,Y) representing a coordinate (within the image)
AggWindow	Integer	Returns an integer value represents the aggregation window size, which can be any value in the interval [3,$max(m, n)$] where m and n are the width and height of the image respectively

D. Fitness Function

The program tree classification accuracy is used to measure the performance of the program (individual). Classification accuracy is calculated as the ratio of correctly classified examples to the total number of examples as illustrated in Equation 1. Individuals with higher accuracy (larger number of examples correctly classified) are preferable to individuals with lower accuracy.

$$Accuracy = \frac{No.\ Correctly\ Classified}{Total\ No.} \tag{1}$$

III. Experimental Design

The proposed system is implemented using the Evolutionary Computing Java-based (ECJ) package [21]. The ECJ package allows having different types of input and output for each nodes (Strongly-typed GP) which a requirement for the proposed system. Table IV lists the input and output types of nodes in different layers. To evaluate the performance of this approach, a number of experiments have been conducted. Four datasets were used in the conducted experiments. The results of the conducted experiments have been compared to the domain-specific pre-extracted features and Atkins et al. [4].

TABLE IV: GP Node Types

Type	Description
Double	The output of $Aggregation$ nodes, and the input/output of $Classification$ nodes
Integer	The type of $AggWindow$ and $Coords$ nodes
AggWindow	The input of $Aggregation$ nodes (determines the aggregation window size)
Coords	The input of $Aggregation$ nodes (determines the AggWindow centre/start pixel coordinates)
Image	The input of $Aggregation$ nodes

A. Datasets

Four datasets with increasing difficulty were used to evaluate the proposed system. The first dataset is the *Coins* dataset [22] which is considered very easy to distinguish between the examples of the two classes. This dataset consists of 384 grey-scale examples of size 55×55 that are equally divided into two classes: *Head* and *Tail*. The Second dataset is the MIT *Faces* dataset [23], which is considered easy. There are 60000 grey-scale examples of size 19×19 in this dataset that fall into two classes: *face* and *non-face*. The third dataset is *Microscopic Cells* [24], which is considered hard. The datasets originally consists of 18 classes; however only two of them were selected in this paper as we are investigating binary classification problems. There 1297 gray-scale images of size 25×25 in this dataset which fall into two classes: *Lymphocytes non activeés* and *Mésothéliales*. The fourth dataset is *Pedestrians* [25], which considered very hard. This dataset consists of 10002 grey-scale examples of size 18×36 that fall into two classes: *pedestrian* and *non-pedestrian*. Table V lists the number of examples in each class of the four datasets. Figures 2 and 3 list some examples of the two classes from each dataset.

TABLE V: The Four Datasets Structure

	Training Set			Test Set			Overall
	A	B	Total	A	B	Total	
Coins	96	96	192	96	96	192	384
Faces	1500	1500	3000	1500	1500	3000	6000
Micro. Cells	349	299	648	349	300	649	1297
Pedestrians	2500	2501	5001	2501	2500	5001	10002

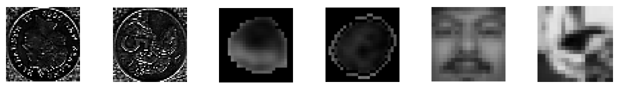

(a) (b) (c) (d) (e) (f)

Fig. 2: Example images from the Coins (a,b), Cells (c,d), and Faces (e,f) datasets

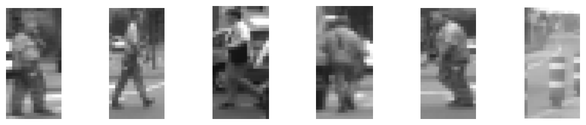
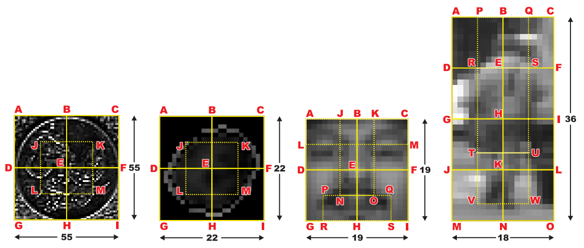

(a) (b) (c) (d) (e) (f)

Fig. 3: Example images from the Pedestrians dataset including pedestrians (a-c) and non-pedestrians (d-f)

B. Domain-Specific GP-based Approach (DS-GP)

To examine the goodness of the proposed approach, a GP-based approach have been used to perform the classification task using domain-expert pre-extracted features. A set of predefined areas have been designed differently for each dataset. Feature extraction operation is done by calculating the *mean* and *standard deviation* values of each area have been calculated. Then these features were fed to a GP-based system to build a classifier that can distinguish between the examples of the two classes.

There were five areas that have been used to extract ten features (mean and standard deviation of each area) of the *coins* and *cells* datasets: the four quadrants ABED, BCFE, DEHG, and EFIH; and the central square JKML as shown in Figure 4-a and 4-b. For *faces* dataset there were seven areas based on the work of Atkins et al. [4] and Bhowan et al. [26]: the quadrants ABED, BCFE, DEHG, and EFIH; and the areas ACML, JKON and PQSR that represents areas around the eyes, nose, and mouth respectively as shown in Figure 4c. Finally, there were eleven areas that have been defined for the *pedestrians* dataset based on the work of Atkins et al. [4]: the octets ABED, BCFE, DEHG, EFIH, GHKJ, HILK, JKMN and KLON; and the areas PQSR, RSUT and TUWV that represents areas around the head, torso and legs respectively as shown in Figure 4d.



(a) (b) (c) (d)

Fig. 4: Domain-specific pre-extracted features regions (a) Coins, (b) Cells, (c) Faces, and (d) Pedestrians datasets

C. Three-tier GP-based Approach (3T-GP)

The performance of the proposed system was compared to the GP-based system proposed by Atkins et al. [4] also. Mainly, as they have proposed a system that evolve single individual to perform the tasks of feature extraction and classification simultaneously; using *three-tier* structure. The three-tier structure consists of image filtering, aggregation, and classification layers. Functions of image filtering mainly aim to perform image enhancement such as, the use of *median* filter to remove the *salt and pepper* noise.

D. The New Approaches

1) With no Filtering (WNF): In this approach the same system as that of Atkins et al. [4] is used except that the filtering tier has been removed. There were two reasons behind removing this tier: firstly, reducing the search space to force the system to try different areas (extract different features) rather than try different enhancement methods; secondly, investigating the effect of image filtering on the performance of the system to perform feature extraction and classification tasks.

2) Row and Column Terminals (RCT): Similar to the previous approach, the filtering tier is not used in this approach. However, the window shape (used by aggregation functions) is of *square* shape. The *AggWindow* represents the size of the window (e.g. when *AggWindow* is 5, the square is of size 5×5). Other shapes in this category are rows and columns. The thickness of *row* and *column* shapes was restricted to one pixel. However, the length can be in the interval [0,$Image_{width}$] and [0,$Image_{height}$] for row and column respectively. Also, the x and y (the aggregation functions second and third children) represent the coordinates of the first pixel instead the centre of the window.

3) Different Window Shape Terminal (DWST): In this approach the *square*, *row* and *column* shapes were combined and two other window shapes were added: *circular* and *rectangular*. The motivation behind introducing these terminals was based on the improvement of the system performance from the use of row and column window shapes. The *circular* shape is determined using the *Bresenham* circle algorithm. The restriction of the newly introduced shapes were similar to the square window shape. However, the x and y represent the coordinates of the top-left corner of the rectangular window. Also, the rectangular shape does not rely on the *AggWindow* value; instead it has two values: *width* and *height*. Like square, the size of the circular and rectangular window (diameter or thickness) cannot be smaller than three pixels.

E. GP Parameters

The GP parameters used in all the experiments are shown in Table VI. An individual program tree cannot be smaller than 2 (the depth of the tree) or larger than 10. The problem of early convergence is mitigated by using large population size (1024 individuals) which as a trade-off has significant effects on increasing computational cost. Also, population diversity was maintained using the tournament selection method. The program stops if the *ideal* accuracy (100%) is achieved, or the maximum number of generation is reached (50 generations).

The same experiment has been repeated 30 times for each method and for each dataset (30 $\times$ 5 methods $\times$ 4 datasets = 600 runs in total). Different random *seeds* were used for each of the 30 repetitions; however the same set of random seeds were used in all of the conducted experiments (to make paired comparison possible). At the end of each run, the best evolved program was evaluated against the training and testing sets independently. Also, at the end of the 30 runs; the minimum, maximum, average and standard deviation accuracy of the best individuals were gathered.

TABLE VI: GP Parameters

Parameter	Value
Generations	50
Population Size	1024
Crossover Rate	0.80
Mutation Rate	0.19
Elitism Rate	0.01
Tree Depth	2-10
Selection Type	Tournament
Tournament Size	7
Builder Type	Ramped HALF-AND-HALF

IV. Results and Analysis

A. Results

Table VII shows the statistics gathered from applying DS-GP, 3T-GP and the new methods to the Coins dataset. Looking at the estimated statistics, DS-GP outperforms all other methods. Conducting a hypothesis test though, reveals that there is no statistically significant differences between the performance of DS-GP and that of the new method. The new methods, however, is significantly better than 3T-GP.

TABLE VII: Results of the Coins dataset[1]

	Training (%)				Testing (%)			
	Min	Max	Mean	St.Dev	Min	Max	Mean	St.Dev
DS-GP	100.0	100.0	100.0	0.00	99.48	100.0	99.95	0.16
3T-GP	75.00	100.0	93.40§	7.15	73.96	100.0	93.61§	6.91
WNF	97.40	100.0	99.81‡	0.52	98.44	100.0	99.81	0.42
RCT	94.79	100.0	98.49§‡	1.31	96.35	100.0	98.72§	1.16
DWST	98.96	100.0	99.62‡	0.47	96.88	100.0	99.36	0.93

TableVIII shows the results of 30 independent runs on *Faces* dataset. On average, the new methods outperform both of the DS-GP and 3T-GP. There is a statistically significant difference between the performance of the new methods and that of DS-GP. Also, the new methods perform extremely better than 3T-GP.

TABLE VIII: Results of the Faces dataset[1]

	Training (%)				Testing (%)			
	Min	Max	Mean	St.Dev	Min	Max	Mean	St.Dev
DS-GP	88.00	92.83	90.84	1.21	90.57	94.03	92.70	0.97
3T-GP	71.93	94.37	84.08§	6.72	80.43	95.50	89.40§	3.25
WNF	85.97	97.47	93.79†‡	2.51	85.40	96.63	92.93‡	2.97
RCT	88.97	97.90	95.64†‡	1.95	89.53	97.90	95.00†‡	1.96
DWST	92.23	97.93	95.24†‡	1.56	89.43	97.27	94.39†‡	1.90

Table IX shows the statistics obtained from applying the best evolved individuals; of the investigated methods against the *Cells* dataset. All methods of the new approach and the 3T-GP show better performance over the DS-GP approach. The significant test shows that the performance of these methods was extremely significant better than the DS-GP. Also, the RCT and DWST worst evolved individuals show 3-4% better performance than DS-GP best evolved individual. The significant test shows significant gap between the performance of the new methods and 3T-GP.

TABLE IX: Results of the Cells dataset

	Training (%)				Testing (%)			
	Min	Max	Mean	St.Dev	Min	Max	Mean	St.Dev
DS-GP	76.23	90.59	85.01	4.26	83.76	91.37	87.61	2.03
3T-GP	86.57	95.99	92.42†	2.82	85.36	97.07	93.24†	2.65
WNF	87.50	96.76	93.76†	1.99	86.59	98.00	95.07†	2.10
RCT	93.52	96.76	95.70†‡	0.75	95.69	97.84	96.80†‡	0.63
DWST	93.98	97.07	95.64†‡	0.75	95.83	97.69	96.73†‡	0.62

The statistics of the 30 best evolved program of each method against the *Pedestrians* dataset are listed in Table X. The DS-GP shows better performance on the unseen data over the 3T-GP and WNF methods. Also, the results of 3T-GP and WNF show the occurrence of *over-fitting* problem. However, RCT and DWST outperformed the performance of DS-GP method. The significant test shows that RCT and DWST have extremely significant better performance over DS-GP, 3T-GP and WNF methods.

TABLE X: Results of the Pedestrians dataset

	Training (%)				Testing (%)			
	Min	Max	Mean	St.Dev	Min	Max	Mean	St.Dev
DS-GP	72.05	86.58	81.03	2.80	75.52	87.66	80.16	2.76
3T-GP	70.81	88.54	81.99	3.97	63.33	81.28	74.87§	4.75
WNF	80.54	87.26	84.66†	1.46	76.86	82.46	79.57‡	1.26
RCT	76.14	90.88	85.42†‡	3.61	74.37	91.04	83.43†‡	3.79
DWST	81.04	91.80	86.13†‡	2.97	75.16	92.28	84.90†‡	4.26

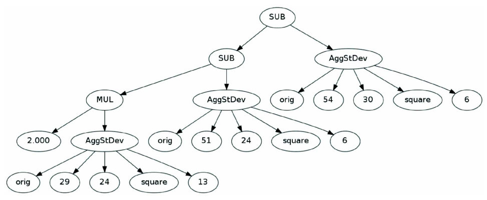
B. Analysis and Discussion

The individual tree representation of an evolved program by the DWST method on the Coins datatset is shown in Figure 5a. The evolved program scored 100% accuracy on both of the training and testing sets, which indicates that the system was able to extract effective features and find a good classifier at the same time without any prior knowledge or domain-expert intervention. Figure 5b-d show the regions of the extracted features, where the program focuses on the middle area of the coin (the head side is darker as it contains fewer details than the tail side) and the right edge (the head side has text around the edge while the tail side does not).

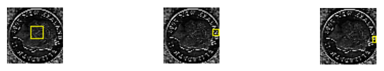
Figure 6a shows the tree representation of an evolved program on the Faces dataset that scored 97.23% and 97.27% accuracy on the training and testing sets respectively. Figure 6b-h show the feature regions, which indicate that the new method has been able to detect some similar areas to the domain-expert designed features (such as, the eyes and the nose). On the other hand, the system selects areas such as, the right cheek and forehead that domain-expert did not consider them as effective areas.

Observably, the proposed methods have comparable performance to the DS-GP system on easier problems. As the difficulty of the dataset increases, however, the proposed

[1]The significant test (T-test) signs §, †, and ‡ represent significantly worst than DS-GP, significantly better than DS-GP, and significantly better than 3T-GP respectively.

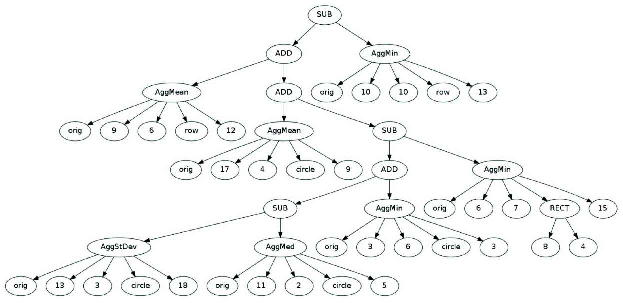


(a)

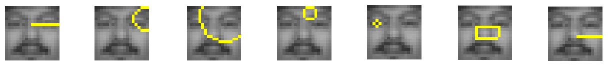


(b) (c) (d)

Fig. 5: Sample program (Coins) (a) Tree (b-d) Feature regions



(a)



(b) (c) (d) (e) (f) (g) (h)

Fig. 6: Sample program (Faces) (a) Tree (b-h) Feature regions

approach significantly outperforms DS-GP. Our results and analysis demonstrate the ability of the new system to perform feature extraction, implicit feature selection, and classification all by itself without the need for domain-expert pre-designed features. It is also observable that the domain-expert pre-extracted features were not as effective as we thought. Better solution can be obtained (evolved) by allowing the new system to select different features. For example, in the Pedestrians dataset the position of the body and legs may vary (e.g. off to the either side of the input window). Figure 7 shows feature areas of an individual scored 89.20% on the test set that dynamically finds the position of of the body, legs and feet; and takes the shifting of these parts into account.

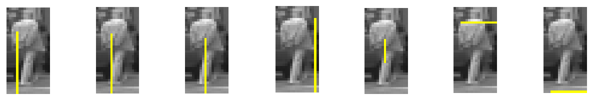


Fig. 7: Sample program (Pedestrians) feature regions

Sometimes the new system may evolve unnecessarily large programs . These programs can be simplified by removing the unnecessary parts. For example the individual program tree shown in Figures 5a is manually simplified in order to make it more human-readable (knowledge extraction). The simplification is only algebraic and thus the behaviour of the program (its accuracy/fitness) is unchanged.

V. Conclusions

In this paper, we proposed a two-tier GP-based approach to automatically evolving programs that perform feature extraction, implicit feature selection, and classification in image datasets. The performance of the proposed approach has been compared to both of the domain-specific pre-extracted features and three-tire GP-based approaches. Four datasets with increasing levels of difficulty have been used to evaluate the performance of the proposed system. Our results show that the new system has comparable performance to DS-GP and 3T-GP systems on easy datasets. The new system outperforms both of DS-GP and 3T-GP systems on hard datasets. As a byproduct, well-performing program trees can also be simplified in order to extract domain knowledge.

References

[1] N. Sebe, I. Cohen, A. Garg, and T. S. Huang, *Machine Learning in Computer Vision*. Springer, 2005.

[2] P. G. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 40, no. 2, pp. 121–144, 2010.

[3] Z. Pan, A. G. Rust, and H. Bolouri, "Image redundancy reduction for neural network classification using discrete cosine transforms," in *IJCNN (3)*, 2000, pp. 149–154.

[4] D. Atkins, K. Neshatian, and M. Zhang, "A domain independent genetic programming approach to automatic feature extraction for image classification," in *IEEE Congress on Evolutionary Computation*, 2011, pp. 5–8.

[5] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming*. lulu.com, 2008.

[6] T. Hiroyasu, S. Fujita, A. Watanabe, M. Miki, M. Ogura, and M. Fukumoto, "Comparison of gp and sap in the image-processing filter construction using pathology images," in *Image and Signal Processing (CISP), 2010 3rd International Congress on*, vol. 2, October 2010, pp. 904 –908.

[7] J. Heaton, *Introduction to Neural Networks with Java*, 2nd ed. Heaton Research Inc., 2008.

[8] W. A. Tackett, "Genetic programming for feature discovery and image discrimination," in *ICGA*, 1993, pp. 303–311.

[9] W. R. Smart and M. Zhang, "Classification strategies for image classification in genetic programming," in *Proceeding of Image and Vision Computing Conference*, 2003, pp. 402–407.

[10] M. Zhang and M. Lett, "Genetic programming for object detection: Improving fitness functions and optimising training data," *IEEE Intelligent Informatics Bulletin*, vol. 7, no. 1, pp. 12–21, 2006.

[11] Y. Lin and B. Bhanu, "Object detection via feature synthesis using mdl-based genetic programming," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 35, no. 3, pp. 538–547, 2005.

[12] T. Kowaliw, W. Banzhaf, N. N. Kharma, and S. Harding, "Evolving novel image features using genetic programming-based image transforms," in *IEEE Congress on Evolutionary Computation*, 2009, pp. 2502–2507.

[13] J. J. Szymanski, S. P. Brumby, P. Pope, D. Eads, D. Esch-Mosher, M. Galassi, N. R. Harvey, H. D. W. McCulloch, S. J. Perkins, R. Porter, J. Theiler, A. C. Young, J. J. Bloch, and N. David, "Feature extraction from multiple data sources using genetic programming," in *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery VIII*, ser. SPIE, S. S. Shen and P. E. Lewis, Eds., vol. 4725, August 2002, pp. 338–345. [Online]. Available: http://citeseer.ist.psu.edu/540967.html

[14] H. Guo and A. K. Nandi, "Breast cancer diagnosis using genetic programming generated feature," *Pattern Recognition*, vol. 39, no. 5, pp. 980–987, 2006.

[15] J. Sherrah, R. E. Bogner, and A. Bouzerdoum, "Automatic selection of features for classification using genetic programming," in *ANZIIS*, 1996, pp. 284–287.

[16] M. G. Smith and L. Bull, "Genetic programming with a genetic algorithm for feature construction and selection," *Genetic Programming and Evolvable Machines*, vol. 6, no. 3, pp. 265–281, 2005.

[17] J. Sherrah, R. E. Bogner, and A. Bouzerdoum, "Automatic selection of features for classification using genetic programming," in *ANZIIS*, 1996, pp. 284–287.

[18] B. T. Lam and V. Ciesielski, "Discovery of human-competitive image texture feature extraction programs using genetic programming," in *GECCO (2)*, 2004, pp. 1114–1125.

[19] O. Oechsle and A. F. Clark, "Feature extraction and classification by genetic programming," in *ICVS*, 2008, pp. 131–140.

[20] M. Zhang, V. Ciesielski, and P. Andreae, "A domain-independent window approach to multiclass object detection using genetic programming," *EURASIP J. Adv. Sig. Proc.*, vol. 2003, no. 8, pp. 841–859, 2003.

[21] S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, E. Popovici, K. Sullivan, J. Harrison, J. Bassett, R. Hubley, A. Chircop, J. Compton, W. Haddon, S. Donnelly, B. Jamil, J. Zelibor, E. Kangas, F. Abidi, H. Mooers, and J. O'Beirne, "Ecj: A java-based evolutionary computation research system," 2010, [Online]. Available: http://cs.gmu.edu/ eclab/projects/ecj/.

[22] W. D. Smart and M. Zhang, "Using genetic programming for multiclass classification by simultaneously solving component binary classification problems," in *EuroGP*, 2005, pp. 227–239.

[23] K. Sung, "Learning and example selection for object and pattern detection," Ph.D. dissertation, Massachusetts Institute of Technology, 1995.

[24] O. Lezoray, A. Elmoataz, and H. Cardot, "A color object recognition scheme: application to cellular sorting," *Machine Vision and Applications*, vol. 14, pp. 166–171, 2003.

[25] S. Munder and D. M. Gavrila, "An experimental study on pedestrian classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1863–1868, 2006.

[26] U. Bhowan, M. Zhang, and M. Johnston, "Genetic programming for classification with unbalanced data," in *EuroGP*, 2010, pp. 1–13.