

Using Restricted Loops in Genetic Programming for Image Classification

Gayan Wijesinghe
School of Computer Science
and Information Technology
RMIT University, GPO Box 2476V
Melbourne Victoria 3001
Email: gayan@cs.rmit.edu.au

Vic Ciesielski
School of Computer Science
and Information Technology
RMIT University, GPO Box 2476V
Melbourne Victoria 3001
Email: vc@cs.rmit.edu.au

Abstract—Loops are rarely used in genetic programming due to issues such as detecting infinite loops and invalid programs. In this paper we present a restricted form of loops that is specifically designed to be evolved in image classifiers. Particularly, we evolve classifiers that use these loops to perform calculations on image regions chosen by the loops. We have compared this method to another classification method that only uses individual pixels in its calculations.

These two methods are tested on two synthesised and one non-synthesised greyscale image classification problems of varying difficulty. The most difficult problem requires determining heads or tails of 320x320 pixel images of a US one cent coin at any angle. On this problem, the accuracy of the loops approach was 97.80% in contrast to the no-loop method accuracy of 79.46%. Use of loops also reduces overfitting of training data. We also found that loop methods overfit less when only a few training examples are available.

I. INTRODUCTION

Loops of the kind normally used in programming languages such as C and Java are rarely used in genetic programming (GP) because of two main problems: (1) Handling infinite loops and (2) The evolutionary process must keep the body of a loop consistent with the initialisation, termination and index updating components. In previous work [1], [2] we have had success with an approach that deals with these two problems by evolving loops with restricted syntax and semantics. In [3] we have used the approach to solve a difficult classification problem involving small binary images. In this paper we examine whether the approach will scale up to problems involving large greyscale images.

In the area of machine vision, traditional image classification methods rely on computing features of pixel values for classification. The selection and configurations of these features depend on the problem and therefore these classification methods that used them were highly domain-dependent. In order to produce a domain-independent approach, GP has been used to evolve formulations of individual raw pixel values into classifier functions. This method has been successful in being domain-independent but due to the searching power of GP and hidden idiosyncrasies present in the training data, the classifiers overfit the training data and perform poorly when tested on unseen data. As there is a large number of pixels in larger images, the task of finding a few points by which to classify a large number of images is a very hard

task. Even if such points could be found, the visual noise that exists in images complicates the problem further.

To prevent the classifiers from relying heavily on unreliable individual pixels, properties of entire regions in images can be considered. Evolving programs that perform operations on such self-chosen regions requires evolvable looping mechanisms. Domain knowledge of images can be used to derive constraints that makes these types of loops possible within GP.

By limiting the operations performed by the body and by constraining the chosen regions to be within the dimensions of the image that is being classified, loops have already been used in a restricted form to create functions of an image's regions for classification [3]. As idiosyncrasies are less likely to exist in entire regions and the effects of noise in individual pixels are dominated by visual features common to the regions, the use of loops can produce less overfitting, more general solutions.

Our objective in this paper is to explore the use of restricted loops in a range of difficult greyscale image classification problems and to determine whether there are any significant benefits when using loop for this task. We are particularly interested in answering the following research questions:

- 1) Can restricted loops improve the accuracy of classification?
- 2) Can restricted loops reduce overfitting of classifiers?
- 3) Can restricted loops require less training examples?

When answering the above questions, we use a simple circles and squares classification problem to first investigate how classifiers with loops can be used on objects in greyscale images and how they perform against ones that do not use loops. Then we use a synthesised character (CAPTCHA) classification problem to investigate the overfitting problem in evolved classifiers when using loops and no loops. A third problem of classifying greyscale images of coins is used to investigate how loops perform in non-synthesised problems as well as to understand how the loops method compare to the no loops method under different sized training and testing sets.

II. RELATED WORK

There are two main kinds of computer vision problems which involve objects – classification and detection. In classification problems it is necessary to discriminate between different kinds of objects, for example given full face images of a number of people whose names are known, the task is determine the name corresponding to any given image. In detection problems the task is to determine whether some object is present in a relatively larger image, for example, is the face of George Bush present in a photo of a group of politicians. Object detection generally involves classification as a sub-task. There has been considerable work in the last 15 years on methods for object classification and detection problems that use pixel values directly, or simple pixel statistics like mean and standard deviation of regions. This is in contrast to classical computer vision where a great deal of effort is expended on computing various features of the object which are then used in a subsequent classifier or detector. However, there is some work in using GP to evolve feature extractors, for example [4].

While there has been some experimentation with neural networks [5], the main focus of work using pixels and pixel statistics is on GP classifiers and detectors. In early work, for example [6] and [7], pixels were used directly as terminals (Figure 1a). This work was quite successful, but did not scale well to larger, more complex images. Subsequent work, such as [8], [9] and [10] used means and standard deviations of pixels in predefined regions. Some examples of the kinds of regions used are shown in Figure 1. This approach avoids the requirement to hand craft the feature extraction programs required by the classical approach, but is not totally domain independent as the region shapes need to be determined for each new domain. The work in [11] and [12] is an attempt to overcome this limitation by using GP to evolve a set of rectangular shapes and associated pixel statistics. The method was quite successful on a number of problems where the objects were very small relative to the size of the image.

An early application of GP was for classification problems. Different ways of representing evolved classifiers are described in [13]. The most common approach is to evolve numeric expression classifiers in which the program returns a numeric value which is mapped into a class label. In binary classification problems, such as the work in this paper, values less than zero typically indicate one class, while values greater than or equal to zero indicate the other class. More recent approaches involve ensembles [14] or use multi-objective methods [15], [16].

In some recent work [3], [2] numeric expression classifiers with a restricted form of looping were evolved. While the primary focus of the work was on evolving loops the approach was successful in evolving classifiers for a synthetic problem that involved discriminating noisy circles and squares [3]. The work was limited to small images. The work in this paper investigates whether the method will scale up to much larger images.

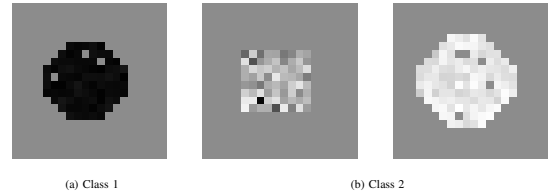


Fig. 2. The simple circles and squares problem (20x20 pixels)

III. SYNTAX AND SEMANTICS OF LOOPS

In our work, we use the RMITGP package which uses strong typing to ensure that all evolved programs are valid. Loops have the following syntax:

(LOOP Begin End Operation)

Begin and *End* are a pair of (x, y) coordinates in the image which define a bounded rectangle, thus infinite loops are not possible. The *Operation* is evolved to be either a sum operation or a subtract operation and is performed on each pixel in the rectangle. This result is then returned by *LOOP*. To ensure that valid programs are generated, domain knowledge and typing are used to restrict the chosen coordinates to be within the image.

IV. SIMPLE CIRCLES AND SQUARES PROBLEM

The simple circles and squares problem is a two class classification problem that has a total of 100 generated images of circles and squares. The problem contains 3 primary shapes, two circles and a square. A sample of these can be seen from Figure 2.

The task is to distinguish the shape in Figure 2a (class 1) from the square and the other circle in Figure 2b (class 2). Each shape is centered in the image and is the foreground of a single shaded background. In class 1, a few random pixels inside the circle is given a lighter colour. All the pixels within the shapes of the other two classes are assigned a random shade of grey. All sample images are 20x20 pixels in size and the shapes within the images are approximately half the size of the containing image.

Although the images are not high in detail, the similarities between the shapes, the sizes of these shapes and the large variations in pixel values make this problem of a difficulty level suitable for our initial investigation of using loops in image classifiers.

A. Experiments

Using this problem, we analyse the classification performances of different loop configurations in GP. In addition to having experiments for both methods (loops and without loops), our preliminary experiments showed that forcing the use of loops and applying a linear parsimony pressure if more than 1 loop is used within a program, produced much simpler solutions with few or only 1 loop. Therefore we experiment with this configuration along with a configuration (“free loops”) that will only enable loops but not apply any such parsimony pressure. Please refer to Figure 3 for a summary

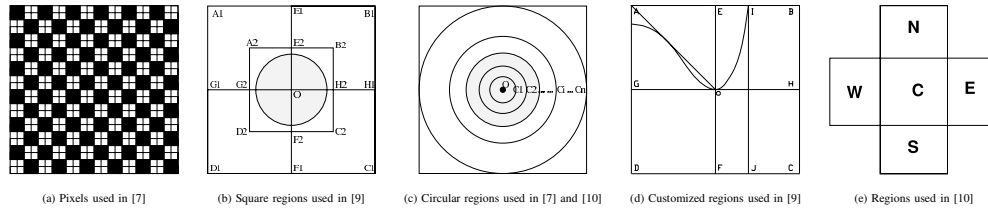


Fig. 1. Examples of using pixels and predefined regions

Method	Node Type	Name	Description
No loops	Functions	Plus Minus Point(x,y)	+ Operation - Operation Grey value at coordinate (x, y)
	Terminals	RandCoordinate	A random coordinate
Loops	Functions	Same as for no loops and, LOOP(Begin,End,Operation)	Execute body operation from (x_1, y_1) to (x_2, y_2)
	Terminals	Same as for no loops and, PlusOperation MinusOperation	Positive sum of pixels Negative sum of pixels

Fig. 3. GP function and terminal sets for all experiments

GP Setting	Value
Runs	100
Population size	100
Max. generations	1000
Termination	0% error or >max gens.
System	RMITGP v1.5
Min. tree depth	1
Max. tree depth	6
Crossover rate	70%
Mutation rate	28%
Elitism rate	2%

Fig. 4. Simple circles and squares problem: GP setup

TABLE I
SIMPLE CIRCLES AND SQUARES PROBLEM: CLASSIFICATION ERRORS OF
FINAL SOLUTIONS

Method & Experiment	Training Error	Testing Error
Free loops	0.00%	0.08%
No loops	0.00%	0.51%
Parsimony pressure loops	0.48%	1.43%

This makes it clear that wanting programs with fewer loops can be unsatisfactory for problems that may require multiple loops.

V. CAPTCHA CHARACTER PROBLEM

The CAPTCHA problem is a 2-class classification problem where the classifier must determine whether a given image of an alphabetic character is A or B. Figure 5a shows examples of these two classes. We are interested in this problem due to the controllability of its difficulty as explained below.

This problem was derived from the task of creating CAPTCHA images. A CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) image contains distorted graphics of alphanumerics that only humans can easily recognise [17]. They are used on web pages to prevent malicious software robots and scripts from automatically filling in forms.

Our fundamental problem contains 150x150 pixel 500 generated images, each with character A or B. In each image, the foreground pixels are generated by selecting a random number from a normal distribution with a mean of 170 and the background with a mean of 130. Both have a variance of 30 and introduces noise into images.

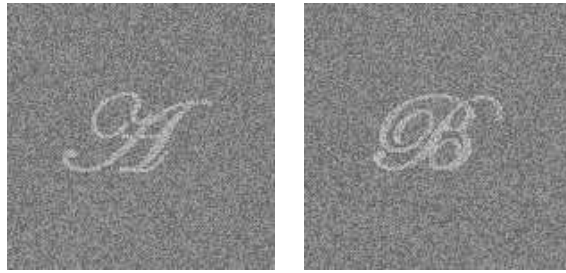
Noise in data increases the difficulty of evolving classifiers since regularities in data are distorted. For example, if our problem did not contain noise, a classifier can very easily be

of the functions and terminals used in the GP system and the configuration of the GP system is shown in Figure 4.

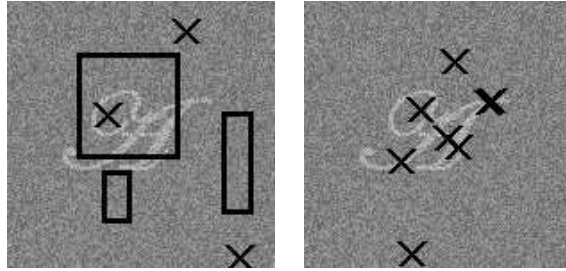
B. Experimental Results

As shown in Table I, the best performing configuration was the one that enabled loops but did not apply any parsimony pressure on loop usage (“free loops”). When we closely examined these solutions, we discovered that loops were there in 94% of the classifiers that obtained 0% error for both training and testing. The remaining few that did not contain any loops performed similarly to the “no loops” configuration.

In contrast, the configuration that forced the programs to have at least one loop and applied parsimony pressure on high loop usage had lower fitness. When their solutions were analysed, we discovered that the parsimony pressure caused almost all programs to be evolved with only 1 loop, making them sub-optimal. To support this, we calculated the loop usage in the “free loops” configuration and found that the solutions that obtained 0% error contained at least 2-3 loops.



(a) The two classes: "A" and "B"



(b) Regions used by loops

(c) Pixel points used by the no-loop method

Fig. 5. The CAPTCHA problem (150x150 pixels)

created to say whether the character is A or B by using a value of a single pixel which will only have a certain value when A is present. However, with noise, the value of this pixel will vary across every instance of A and B encountered. To overcome noise, the classifier needs to be trained on many examples to learn more general features of the classes and therefore it should not be affected by noise in unseen data.

Additionally we have created three variants of the problem with varying levels of difficulty. In the first variant, "AB-00", the character is centered in all the sample images. In the second variant, "AB-10", the character is randomly moved off the center by up to 10 pixels in each axis. The third variant, "AB-20" is similar to "AB-10" but the character is moved at most by 20 pixels.

As the number of possible locations of the object increases, it will be harder to find an accurate classifier since now, in addition to overcoming noise, the classifiers must also need to take general visual characteristics into account. For this, many pixels will need to be carefully selected if only individual pixels are used for classification. Even then, the performance on images unseen during training is not guaranteed to be the same. We expect that looping through entire regions will benefit the solutions.

A. Experiments

Using the CAPTCHA problem we investigate how evolving loops to perform calculations on entire regions can be more beneficial than using individual pixels for classifiers on problems with many degrees of variation. Particularly, we are interested in the generalisation performance or how well

GP Setting	Value
Runs	20 (CAPTCHA) 40 (Rotated coins)
Population size	100
Max. generations	1000
Termination	0% error or >max gens.
System	RMITGP v1.5
Min. tree depth	1
Max. tree depth	8
Crossover rate	70%
Mutation rate	28%
Elitism rate	2%

Fig. 6. CAPTCHA and rotated coins problem: GP system setups

the testing performance will match the training performance on these two configurations.

For the above, we extend our GP system setup from the simple circles and squares problem. Again, we use a method that uses loops as well as one that does not. However, the method that uses loops now only forces each classifier contain at least 1 loop but it does not apply any evolutionary pressure on the number of loops used as long as there is at least 1. For the no-loops method, we use the same method used in the simple circles and squares problem. For both methods, Figure 3 shows the GP functions and terminals that will be used and Figure 6 shows the configuration of the GP system and runs.

We perform experiments for each of the 3 CAPTCHA variants described earlier in the description of the problem. The classifiers will be evolved with and without loops and the average training and testing errors of the classifiers will be measured over 20 runs.

B. Experimental Results

Table II shows the final training and testing fitnesses. Figure 7 shows the training fitness improvement during the evolutionary process for the loops and no loops methods. Figure 8 shows the difference between testing and training fitnesses during the evolutionary process for both methods.

The following is an evolved loop classifier that produced 0% error for both training and testing in the AB-00 experiment. Figure 5b shows its regions.

(+ (+ (- (LOOP 121 63 136 118 MinusOperation) (- (LOOP 40 30 95 87 PlusOperation) (POINT 100 17))) (POINT 56 64)) (- (LOOP 68 96 54 123 MinusOperation) (POINT 130 144)))

The following is an evolved no-loop classifier that produced 0% error for both training and testing in the AB-00 experiment. Figure 5c shows the pixel points it chose for classification.

(+ (+ (+ (POINT 80 77) (POINT 86 82)) (POINT 84 34)) (- (- (POINT 65 61 (POINT 60 142)) (+ (POINT 54 90) (+ (POINT 103 57) (POINT 105 56))))))

TABLE II
CAPTCHA PROBLEM: CLASSIFICATION ERRORS OF FINAL SOLUTIONS

Experiment	GP Method	Training Error	Testing Error	Overfit
AB-00	Loops	0.03%	0.00%	-0.03%
	No loops	1.35%	3.48%	2.13%
AB-10	Loops	6.99%	7.21%	0.22%
	No loops	31.63%	43.91%	12.28%
AB-20	Loops	18.98%	21.46%	1.48%
	No loops	32.49%	50.40%	17.91%

There is a clear improvement of approximately 3.5% to over 36% when using loops in classifiers. Although the non-shifted experiment (AB-00) was easy to solve, there were no solutions in the shifted experiments, from either method.

Figure 7 shows that initially, the no-loop method has a slightly better fitness than the loops method but the latter quickly overtakes and continues to improve while the former tends to become stagnant. Although the fitness still seems to be increasing at the terminating point (100000 evaluations or 100 generations), analysis of the fitness curve gradients in the final 20 generations, we approximated that in order for the training fitnesses to reach 0% error, the loops method would require 1.62 and 7 million less evaluations (respectively) than the no loops method for the AB-10 and AB-20 experiments, provided the rates of improvement remain the same.

When considering the overfit (the gap between the testing and training fitness) during evolution, shown on Figure 8, we see the loop method's testing fitnesses being similar (AB-00 and AB-10) and relatively close (AB-20) to its training fitnesses. Only in the hardest problem (AB-20), the loops method has a non-closing constant overfit. In contrast, the no-loops method's overfit always increases rapidly in the first few generations and becomes stagnant (AB-00 and AB-10) or continues to increase (AB-20).

The above results show that using only individual pixels for classification leads to low testing accuracies and high levels of overfitting to training data. Evolving loops to perform operations on entire regions is much more desirable due to the method's better learning ability when dealing with harder problems such as this.

Although using loops produced better classifiers, the training on average consumed 6 times the training time of classifiers without loops. When running the hardest experiment (AB-20) on our hardware, the loop method consumed 2 hours and the no-loop method consumed 20 minutes, approximately per run, on average.

VI. ROTATED COINS PROBLEM

The "rotated coins problem" is a 2-class problem where the classifier must correctly classify faces of a coin as showing "heads" or "tails" (Figure 9a). A single image contains a single coin face and a total of 720 images have been generated by rotating each face by 1 degree increments. Each image is also 320x320 (102400) pixels in size and have 256 greylevels. Therefore, the problem contains approximately 74 million pixels. We are interested in this problem because we have a complete set of examples covering each state possible

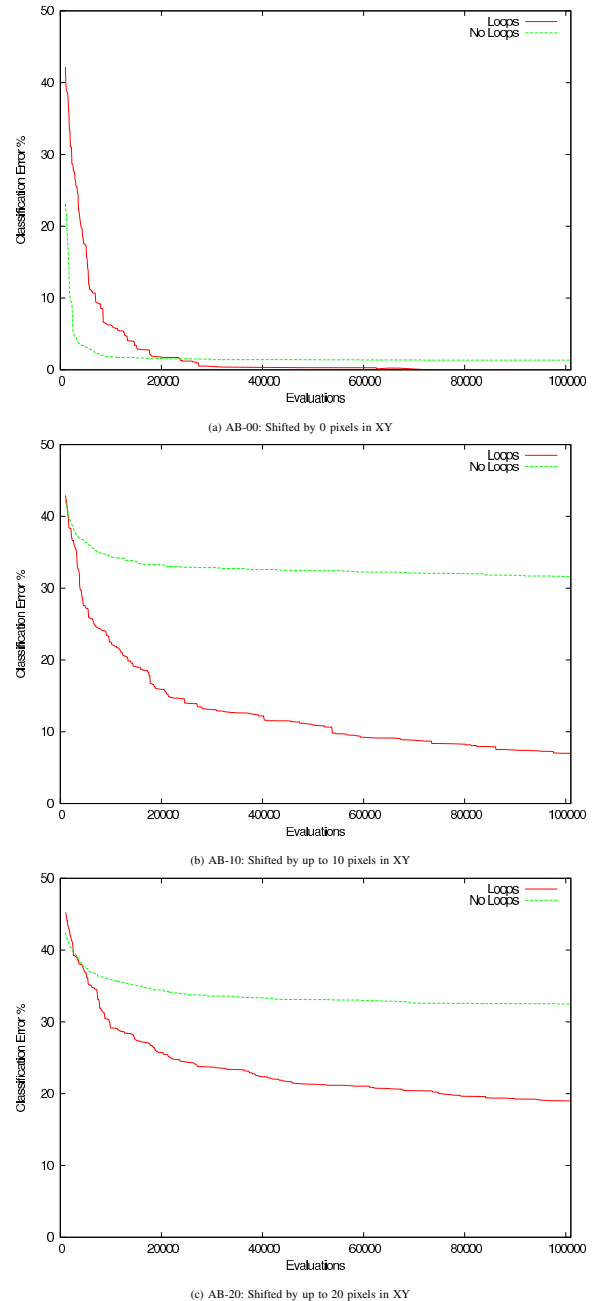
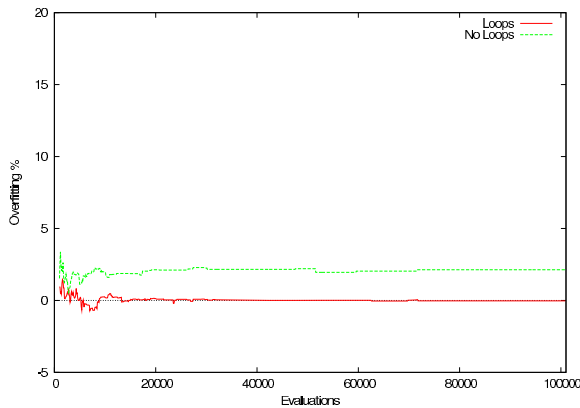
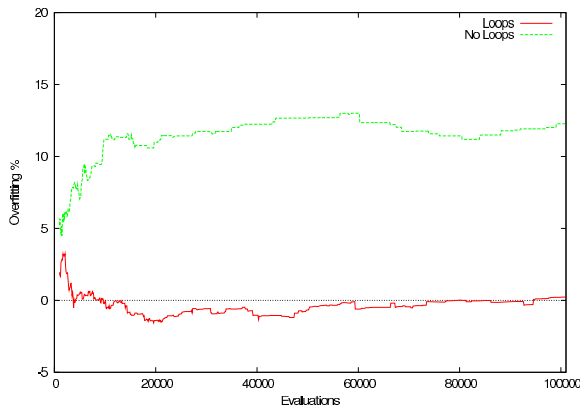


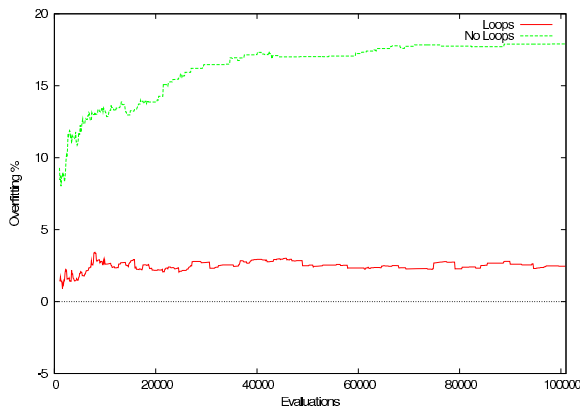
Fig. 7. CAPTCHA problem: The fitness improvement over evaluations for easy (a) to hard (c) experiments.



(a) AB-00: Shifted by 0 pixels in XY



(b) AB-10: Shifted by up to 10 pixels in XY

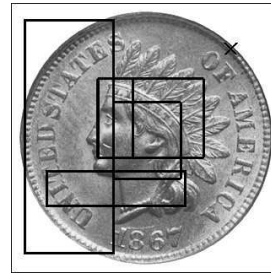


(c) AB-20: Shifted by up to 20 pixels in XY

Fig. 8. CAPTCHA problem: Overfitting amounts for easy (a) to hard (c) experiments during evolution.



(a) The two classes: "Heads" and "tails"



(b) Regions used by loops



(c) Pixel points used by the no-loop method

Fig. 9. The rotated coins problem (320x320 pixels)

rotated state to investigate the effects of training and testing set sizes.

We expect this to be a harder problem than the CAPTCHA problem due to the finer details present in non-synthesised data, the larger image size, and the rotations. Rotations about the center, in contrast to shifting off the center, makes it hard even for a classifier with loops since the object is confined to a smaller area.

A. Experiments

We use this problem to investigate how classifiers that use loops compare against ones that do not support loops in a non-synthesised, relatively large, machine-vision classification problem. Also we wish to compare the overfitting and generality of both methods under varying sizes of training sets to answer our 3rd research question.

The same sets of functions and terminals are used as before (Figure 3) and the GP system setup (Figure 6) is identical to that of the CAPTCHA problem. However, results are now calculated by averaging 40 runs.

B. Experimental Results

The results from our experiments are shown on Tables III and IV.

There is a significant improvement of over 5%-13% in training and over 15%-23% in testing when using loops against the no-loops method, as shown on Table III. When using loops with 70% of images for training, many solutions were found with 0% training and testing errors.

Figure 9b and Figure 9c show evolved classifiers from each method found by the experiment that used 70% data

TABLE III
 ROTATED COINS PROBLEM: CLASSIFICATION ERRORS OF FINAL
 SOLUTIONS ON VARIED TRAINING/TESTING SET SIZES

Experiment	GP Method	Training Error	Testing Error
70% Train/30% Test	Loops	1.75%	2.20%
	No loops	15.16%	20.54%
40% Train/60% Test	Loops	4.63%	5.72%
	No loops	14.24%	21.58%
10% Train/90% Test	Loops	4.79%	7.38%
	No loops	10.53%	30.52%

for training. The following is the loop classifier that uses the regions indicated in Figure 9b.

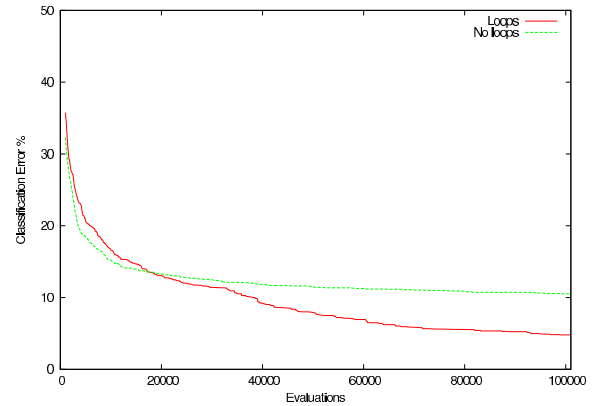
(- (- (POINT 260 54) (LOOP 206 240 42 199 MinusOperation))
 (- (- (LOOP 228 89 103 183 MinusOperation) (LOOP 121 296 16
 20 MinusOperation)) (+ (LOOP 121 208 201 117 PlusOperation)
 (LOOP 144 89 102 183 MinusOperation))))

The no-loops runs used the entire 1000 generations and did not find any solutions with a 0% error. Figure 10 shows the average training error for all the experiments over 1000 generations (100,000 program evaluations). It can be seen that the fitness curves of the 3 no-loops experiments are flattening out while having relatively high levels of error. This indicates that any reasonable amount of further evolution is unlikely to reduce the error levels to 0% for the no-loops methods. A similar observation can be made about the loops methods that use 40% and 10% training images. However, the loops method with 70% is seen to be gradually improving and has the potential to reach 0% if evolved further.

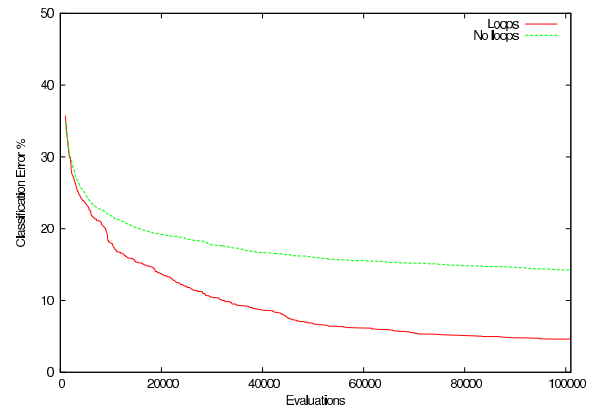
The overfitting amounts for each method are shown in Table IV. According to Tables III and IV, the no-loops method increases the training accuracy but does not improve the testing accuracy. This overfit increases as the number of training examples become fewer. However, with loops, only a negligible training improvement and a slight decrease in testing accuracy is shown when the number of examples is reduced. This shows that even when a lower number of examples is available, the loop method is more robust and is less likely to overfit than the no-loops method. In the loops experiment that used 70% of training data, over 60% of the found solutions were with 0% error in both training and testing.

Similarly to the CAPTCHA experiments, using loops required much longer to evolve than it did without loops. On the same hardware used for the previous experiments, a run of the loop experiment consumed 5 hours and the no-loop experiment consumed 22 minutes approximately, on average. Considering the size of the coins problem, which has over 6 times more pixels than the previous problem, evolving loops required only 2.5 times more CPU time.

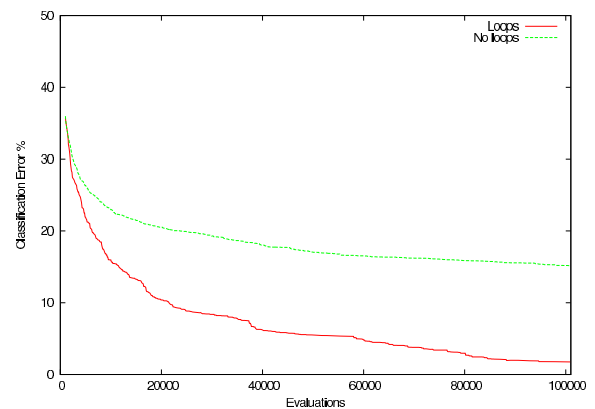
An intuitive next step for the above experiments is to enable values returned by loops to be used as starting or finishing coordinates of other loops. This, in effect, allows loops to consider different regions of an image for classification, based on the grey intensities of another region. We



(a) 10% training data



(b) 40% training data



(c) 70% training data

Fig. 10. Rotated coins problem: The fitness improvement over evaluations on different training/testing set sizes

TABLE IV
 ROTATED COINS PROBLEM: OVERFITTING AMOUNTS OF FINAL
 SOLUTIONS ON VARIED TRAINING/TESTING SET SIZES

Training/Testing Set Sizes	GP Method	Overfit (Testing-Training)
504/216 (70%/30%)	Loops	0.55%
	No loops	5.38%
288/432 (40%/60%)	Loops	1.09%
	No loops	7.34%
72/648 (10%/90%)	Loops	2.59%
	No loops	19.99%

investigated on this and found that, although GP discovered classifiers that performed as well as the ones found by the method that did not enable loops within the coordinates of other loops, none of the better solutions contained this added feature. The ones that did had an average testing error of about 50%, suggesting that no learning took place. We conjecture that if such unique secondary characteristics exist in an image, they could be directly used as the primary basis for the classification, instead. Also this method consumed over 10 hours on average per run, due to the massive increase in search space size.

VII. CONCLUSIONS

In this paper we have demonstrated how restricted loops can be successfully used to evolve better classifiers as compared to classifiers without loops. We have used these methods on two synthesised problems and one non-synthesised greyscale image classification problem of various difficulty levels.

First, we have shown in our experiments, that loops allowed image classifications to be based on regional characteristics rather than on individual pixels, which increased classification accuracies by 15% to 23% in our hardest problem. Then we have shown that loop methods are more robust since regional operations overcome idiosyncrasies in training data that lead no-loop classifiers to overfit. Finally, we discovered that even when only a small number of training examples were available, the use of loops obtained higher levels of accuracy while maintaining a lower overfit to training data. Due to operations on entire regions, the loop methods required 2 to 5 hours during training whereas the no-loop methods required around 20 minutes to achieve the above benefits.

Although it could be argued that these restricted loops are merely genetic programming functions that operate on regions, our constructs have a starting point, an ending point and a body, all of which can vary, similar, in principle, to human-written loops.

As further work, we will investigate the use of more complex loop bodies and loops nested in loop bodies on image classification problems. If successful, we will also investigate on how indexing can be used to evolve programs that are similar to ones created by human programmers.

REFERENCES

- [1] V. Ciesielski and X. Li, "Experiments with explicit for-loops in genetic programming," in *Proceedings of the 2004 Congress on Evolutionary Computation (CEC2004)*, G. Greenwood, Ed., vol. 1. IEEE, Jun. 2004, pp. 494–503.
- [2] X. Li and V. Ciesielski, "An analysis of explicit loops in genetic programming," in *Proceedings of the 2005 Congress on Evolutionary Computation (CEC2005)*. IEEE Press, Sep. 2005, pp. 2522–2529.
- [3] —, "Using loops in genetic programming for a two class binary image classification problem," in *Proceedings of the 2004 Australian Joint Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence 3339*, G. Webb and X. Yu, Eds. Springer, Dec. 2004, pp. 898–909.
- [4] Y. Zhang and P. I. Rockett, "Evolving optimal feature extraction using multi-objective genetic programming: A methodology and preliminary study on edge detection," in *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2005, pp. 795–802.
- [5] M. V. Shirvaikar and M. M. Trivedi, "A network filter to detect small targets in high clutter backgrounds," *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 252–257, Jan 1995.
- [6] D. Howard, S. C. Roberts, and R. Brankin, "Evolution of ship detectors for satellite sar imagery," in *Proceedings of the Second European Workshop on Genetic Programming*. London, UK: Springer-Verlag, 1999, pp. 135–148.
- [7] M. Zhang and V. Ciesielski, "Neural networks and genetic algorithms for domain independent multiclass object detection," *International Journal of Computational Intelligence and Applications*, vol. 4, no. 1, pp. 77–108, Mar. 2004.
- [8] V. Ciesielski, A. Innes, J. Mamutil, and S. John, "Landmark detection for cephalometric radiology images using genetic programming," *International Journal of Knowledge Based Intelligent Engineering Systems*, vol. 7, no. 3, pp. 164–171, Jul. 2003.
- [9] M. Zhang, V. Ciesielski, and P. Andraea, "A domain-independent window approach to multiclass object detection using genetic programming," *EURASIP Journal on Applied Signal Processing, Special Issue on Genetic and Evolutionary Computation for Signal Processing and Image Analysis*, vol. 2003, no. 8, pp. 841–859, Jul. 2003.
- [10] D. Howard, S. C. Roberts, and C. Ryan, "Pragmatic genetic programming strategy for the problem of vehicle detection in airborne reconnaissance," *Pattern Recognition Letters*, vol. 27, no. 11, pp. 1275–1288, Aug. 2006, evolutionary Computer Vision and Image Understanding.
- [11] M. E. Roberts and E. Claridge, "Cooperative coevolution of image feature construction and object detection," in *Parallel Problem Solving from Nature - PPSN VIII*, ser. Lecture Notes in Computer Science, X. Y. et al., Ed., vol. 3242. Birmingham, UK: Springer-Verlag, 18-22 Sep. 2004, pp. 902–911.
- [12] —, "A multistage approach to cooperatively coevolving feature construction and object detection," in *Applications of Evolutionary Computing, EvoWorkshops 2005: Proceedings*, ser. Lecture Notes in Computer Science, F. R. et al., Ed., vol. 3449. Springer, 2005, pp. 396–406.
- [13] T. Loveard and V. Ciesielski, "Representing classification problems in genetic programming," in *Proceedings of the 2001 Congress on Evolutionary Computation*, J.-H. Kim, Ed. Seoul, South Korea: IEEE, May 2001, pp. 1070–1077.
- [14] G. Folino, C. Pizzuti, and G. Spezzano, "Improving cooperative gp ensemble with clustering and pruning for pattern classification," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2006, pp. 791–798.
- [15] A. McIntyre and M. Heywood, "MOGE: GP classification problem decomposition using multi-objective optimization," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2006, pp. 863–870.
- [16] M. Lemczyk and M. Heywood, "Pareto-coevolutionary genetic programming classifier," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM Press, 2006, pp. 945–946.
- [17] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: Using hard ai problems for security," in *EUROCRYPT*, 2003, pp. 294–311.