Genetic Image Network for Image Classification

Shinichi Shirakawa, Shiro Nakayama, and Tomoharu Nagao

Graduate School of Environment and Information Sciences, Yokohama National University, 79-7, Tokiwadai, Hodogaya-ku, Yokohama, Kanagawa, 240-8501, Japan
shirakawa@nlab.sogo1.ynu.ac.jp, shiro@nlab.sogo1.ynu.ac.jp, nagao@ynu.ac.jp

Abstract. Automatic construction methods for image processing proposed till date approximate adequate image transformation from original images to their target images using a combination of several known image processing filters by evolutionary computation techniques. Genetic Image Network (GIN) is a recent automatic construction method for image processing. The representation of GIN is a network structure. In this paper, we propose a method of automatic construction of image classifiers based on GIN, designated as Genetic Image Network for Image Classification (GIN-IC). The representation of GIN-IC is a feed-forward network structure. GIN-IC transforms original images to easier-to-classify images using image transformation nodes, and selects adequate image features using feature extraction nodes. We apply GIN-IC to test problems involving multi-class categorization of texture images, and show that the use of image transformation nodes is effective for image classification problems.

1 Introduction

Various image processing and recognition techniques using evolutionary computation (EC) have been studied and their effectiveness is demonstrated [1]. The typical examples are template matching, image filter design, image segmentation, and image classification. In particular, genetic programming (GP) [2] has been frequently applied to image classification tasks [3–6]. Tackett primarily applied GP to solve image classification problems [3]. Parallel Algorithm Discovery and Orchestration (PADO) [4, 5], a graph-based GP method, was applied to object recognition problems. Zhang et al. used linear genetic programming (LGP) for multi-class image classification [6]. The results showed that this approach outperforms the basic tree-based GP approach. In addition, GP was used to develop useful texture-feature extraction algorithms [7, 8]. In these studies, the evolved features are either at par or outperform human-designed features.

In addition, automatic construction methods for image processing have been proposed [9–13]. They have constructed various types of image processing algorithms automatically. The genetic image network (GIN) is a recent automatic construction method for image processing [12, 13]. The representation of GIN is a network structure.

In general, it is difficult to select and extract image features because the appropriate image features depend on the images of a target problem. In this paper, we propose a method for automatic construction of image classifiers based on GIN, the Genetic Image Network for Image Classification (GIN-IC). The representation of GIN-IC is a feed-forward network structure. GIN-IC is composed of image transformation nodes, feature extraction nodes, and arithmetic operation nodes. GIN-IC transforms the input images to easier-to-classify images using the image transformation nodes and selects adequate image features using the feature extraction nodes. We apply GIN-IC to test problems of multi-class classification of texture images.

The next section of this paper is an overview of several related studies. In Section 3, we describe our proposed method, GIN-IC. Next, in Section 4, we apply the proposed method to the problem of texture classification and show several experimental results. Finally, in Section 5, we describe conclusions and future work.

2 Related Work

2.1 Genetic Programming and Graph-based Genetic Programming

Genetic Programming (GP) [2], an evolutionary computation technique, was introduced by Koza. GP evolves computer programs, which are usually tree structures, and searches for a desired program using a genetic algorithm (GA). Recently, many extensions and improvements to GP have been proposed. Parallel Algorithm Discovery and Orchestration (PADO) [4, 5] is a graph-based GP rather than a tree structure. PADO was applied to object recognition problems. Cartesian genetic programming (CGP) [14] was developed from a representation used for the evolution of digital circuits and it represents a program as a graph. In CGP, the genotype is an integer string that denotes a list of node connections and functions. This string is mapped into the phenotype of an index graph.

2.2 Automatic Construction of Image Transformation

In image processing, it is difficult to select filters satisfying the transformation from original images to their target images. The Automatic Construction of Tree structural Image Transformation (ACTIT) system [9–11] has been proposed previously. ACTIT approximates adequate image transformation from original images to their target images by a combination of several known image processing filters. ACTIT constructs tree-structured image processing filters using GP [2]. The individual in ACTIT is a tree-structured image transformation. The terminal nodes of a tree are the original images, whereas the non-terminal nodes are several kinds of image processing filters. A root node represents an output image. Users provide training images, and the ACTIT system automatically constructs appropriate image processing procedures. 3D-ACTIT [10, 11] is an extended method that automatically constructs various 3D image processing procedures, and is applied to medical image processing and video image processing. Recently, two other extensions of ACTIT, Genetic Image Network (GIN) [12] and Feed Forward Genetic Image Network (FFGIN) [13], have been proposed. Instead of a tree representation, they are represented by a network structure. Their biggest difference from ACTIT is the structure of connections between image processing filters. In general, a network structure theoretically includes a tree structure (i.e., a network structure also represents a tree structure). Therefore, the descriptive ability of a network representation is higher than that of a tree structure. In GIN and FFGIN, a genotype is a string of integers that indicate image processing filter types and connections. Other studies show that GIN and FFGIN automatically construct a simple structure for complex image transformation using their network representation [12, 13].

3 Genetic Image Network for Image Classification (GIN-IC)

3.1 Overview

The image classifier GIN-IC consists of image transformation, feature extraction, and arithmetic operation components. The greatest advantage of GIN-IC is its image transformation (preprocessing) component, which influences image feature selection. In other words, GIN-IC generates and selects adequate image features by a combination of nodes.

3.2 Structure of GIN-IC

GIN-IC constructs an acyclic network-structured image classifier automatically. Figure 1 shows an example of the phenotype (feed-forward network structure) and genotype (string representing the phenotype) of the proposed method. One of the benefits of this type of representation is that it allows the implicit reuse of nodes in its network. The nodes of GIN-IC are categorized into five types: input nodes, image transformation nodes, feature extraction nodes, arithmetic operation nodes, and output nodes. Input nodes correspond to original images. Image transformation nodes execute image transformation using the corresponding well-known image processing filters. We prepare one-input one-output filters and two-inputs one-output filters in the experiments. Feature extraction nodes extract an image feature from input images. Arithmetic operation nodes execute arithmetic operations. Image classification is performed using the values of output nodes. The image classification procedure in GIN-IC is as follows:

1. Image transformation (preprocessing) of original images
2. Feature extraction from images
3. Arithmetic operation and image classification

In GIN-IC, these processes evolve simultaneously.

In GIN-IC, the feed-forward network structure of nodes is evolved, as shown in Figure 1. Numbers are allocated to each node beforehand. Increasingly large

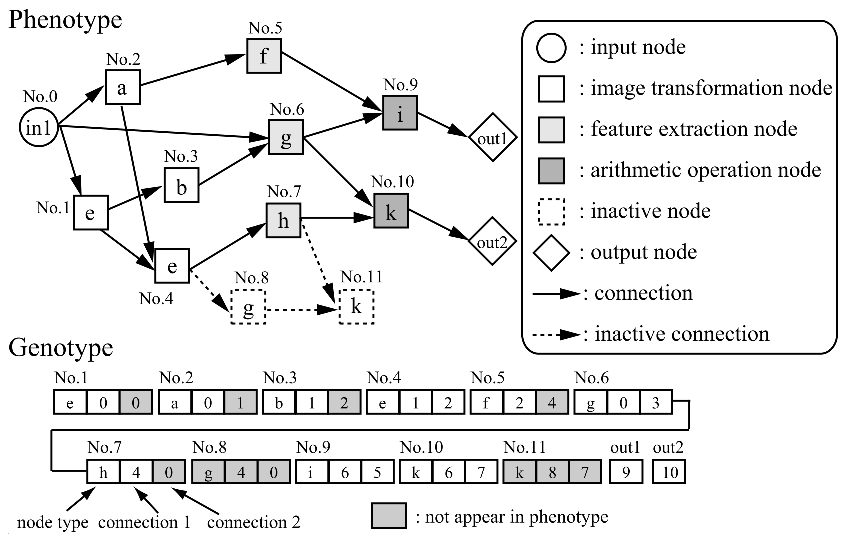

Fig. 1. Example of a structure in GIN-IC (phenotype) and the genotype, which denotes a list of node types and connections.

numbers are allocated, in order, to input nodes, image transformation nodes, feature extraction nodes, arithmetic operation nodes, and output nodes. Connections that cannot be executed are restricted at the genotype level; for instance, the feedback structure is restricted. The nodes take their input from the output of previous nodes in a feed-forward manner. Furthermore, the image transformation and feature extraction nodes are connected either with input nodes or image transformation nodes. The arithmetic operation nodes are connected either with the feature extraction nodes or other arithmetic operation nodes; that is, the connections of nodes obey these data type restrictions, which are based on the idea of strongly typed genetic programming [15].

To adopt an evolutionary method, GIN-IC uses genotype-phenotype mapping. This genotype-phenotype mapping method is similar to Cartesian Genetic Programming (CGP). The feed-forward network structure is encoded in the form of a linear string. The genotype in GIN-IC is a fixed length representation and consists of a string that encodes the node function ID and connections of each node in the network. However, the number of nodes in the phenotype can vary in a restricted manner, as not all of the nodes encoded in the genotype have to be connected. This allows the existence of inactive nodes. In Figure 1, node No. 8 and 11 are inactive nodes.

Because GIN-IC constructs a feed-forward network structured image classification procedure, it can represent multiple outputs. Therefore, GIN-IC enables easy construction of a multi-class image classification procedure using a single network structure.

3.3 Genetic Operator and Generation Alternation Model

To obtain the optimum structure, an evolutionary method is adopted. The genotype of the proposed method is a linear string. Therefore, it is able to use a standard genetic operator. In this paper, we use mutation as the genetic operator. The mutation operator affects one individual, as follows:

– Select several genes randomly according to the mutation rate Pm for each gene.
– Change the selected genes randomly under the constraints of the structure.

We use (1+4) Evolution Strategy ((1+4) ES) as the generation alternation model. The (1+4) ES procedure in the experiments works as follows:

1. Set generation counter $t = 0$. Generate an individual randomly as a parent M.
2. Generate a set of four offsprings, C, by applying the mutation operation to M.
3. Select the elite individual from the set $M + C$ (the offspring is selected if the same best fitness with the parent). Then replace M with the elite individuals.
4. Stop if a certain specified condition is satisfied; otherwise, set $t = t + 1$ and go to step 2.

Since GIN-IC has inactive nodes, a neutral effect on fitness is caused by genetic operation (called neutrality [16, 14]). In step 3, the offspring is selected if the same best fitness with the parent, then the searching point moves even if the fitness is not improved. Therefore, we believe that efficient search is achieved though simple generation alternation model. This (1+4) ES is also adopted and showed the effectiveness in CGP [14] which has feed-forward network structure as well as GIN-IC.

4 Experiments and Results

In this section, we apply GIN-IC to the problem of multi-class texture classification and confirm that using image transformation nodes is effective.

4.1 Settings of the Experiment

In this experiment, we apply GIN-IC to a simple test problem of multi-class classification of texture images. We use texture images from the publicly available database VisTex[1]. We use six classes in this experiment. We make 128 images with 64×64 pixels each by dividing two texture images of 512×512 pixels for each class. All images used in the experiment are grayscale images with a size of 64×64 pixels. The number of training images is 60 (10 images for each class). The training images used in the experiment are displayed in Figure 2.

[1] http://vismod.media.mit.edu/vismod/imagery/VisionTexture/vistex.html

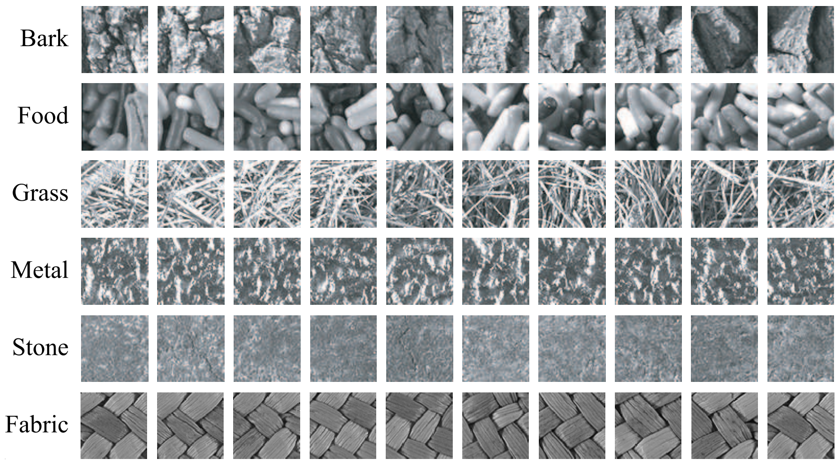

Fig. 2. Training images used in the experiment. There are six classes: Bark, Food, Grass, Metal, Stone, and Fabric.

Table 1. Parameters used in the experiment.

Parameter	*Value*
Generation alternation model	(1+4)ES
Number of generations	112500
Mutation rate (P_m)	0.02
Number of image transformation nodes	100 or 0
Number of feature extraction nodes	100
Number of arithmetic operation nodes	100
Number of output nodes	6

To take advantage of the multi-class image classification capabilities of GIN-IC, we prepared six output nodes, one for each class. The class represented by the output node with the largest value is considered to be the class of the input image.

We use the number of correct classifications of training images and the number of active nodes as a fitness function. The fitness function used in these experiments is described as follows:

$$f = N_c + \frac{1}{N_a} \tag{1}$$

where N_c is the number of correct classifications of training images, and N_a is the number of active nodes. A higher numerical value indicates better performance. If the classification performance is the same, the number of active nodes should be

Table 2. Discrimination rate for the test images using obtained classifiers (average over 10 different runs).

	Use image filters	Do not use image filters
Bark	70.7 %	53.4%
Food	86.1 %	44.1%
Grass	88.1 %	84.8%
Metal	76.7 %	92.5%
Stone	80.4 %	83.4%
Fabric	94.7 %	92.0%
Average	82.8 %	75.0%

small in this fitness function. We think N_c is more important than N_a. However, a weight coefficient may have to be given to N_c and N_a.

The parameters used by the proposed method are shown in Table 1. We assume that the number of image transformation nodes is either 0 (to verify the effect of using image transformation nodes) or 100. If the number of the image transformation nodes is 0, this method resembles classification techniques using the usual GPs. Results are given for ten different runs with the same parameter set.

We prepare simple and well-known image processing filters as the image transformation nodes in the experiment (26 one-input one-output filters and 9 two-input one-output filters), e.g., mean filter, maximum filter, minimum filter, Sobel filter, Laplacian filter, gamma correction filter, binarization, linear transformation, difference, logical sum, logical prod, and so on. Seventeen simple statistical values are used as feature extraction nodes, e.g., mean value, standard deviation, maximum value, minimum value, mode, 3 sigma in rate, 3 sigma out rate, skewness, kurtosis, and so on. The arithmetic operation nodes are 20 well-known arithmetic operations, e.g., addition, subtraction, multiplication, division, threshold function, piecewise linear function, sigmoid function, absolute value, equalities, inequalities, constant value, and so on. GIN-IC is expected to construct a complex image classifier using a combination of these nodes.

4.2 Results and Discussion

When the number of image transformation nodes was 100, GIN-IC achieved 100% classification accuracy for all training images. However, 100% accuracy in classifying training images was not obtained when the number of image transformation nodes is 0, suggesting the effectiveness of using image transformation nodes. We believe that the image transformation nodes change the original images to images adequate for classification.

Next, we apply the obtained classifiers to test images. Test images (non-training images that are similar to training images) are not used in the evolutionary process. The number of test images is 708 (118 images for each class). The average classification performance over ten different runs for the test images

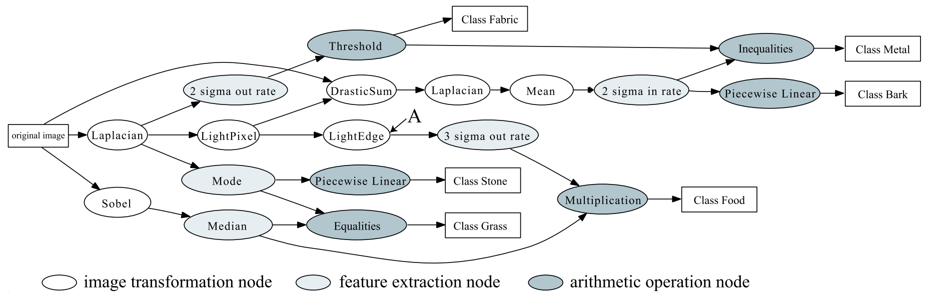

Fig. 3. Example of obtained structure using GIN-IC.

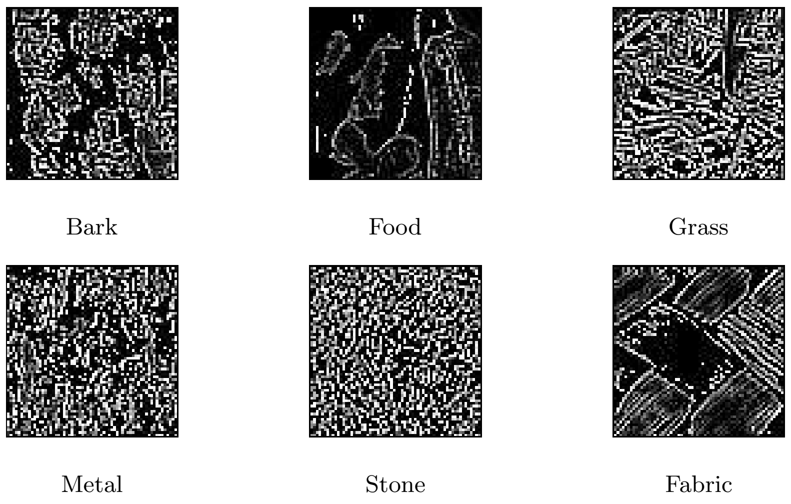

Fig. 4. Examples of output images at node A in Figure 3.

are shown in Table 2. The values of Table 2 are discrimination rates for each class. According to the results, the obtained classifiers affect test images as well as training images. The result also shows the effectiveness of using image transformation nodes. In particular, the discrimination rates of “Bark” and “Food” improve when image transformation nodes are used. The discrimination rates of “Metal” and “Stone” are worse on the other hand.

Figure 3 is an example of the obtained structure (feed-forward network-structured image classifiers) constructed by GIN-IC. GIN-IC constructs the structure using the image transformation nodes in its network. Examples of output images at node A are shown in Figure 4. From these output images, GIN-IC transforms the original images to other types of images and also transforms the features of the images. This shows that GIN-IC automatically constructs image

Table 3. Confusion matrix and discrimination rates using the structure of Figure 3 for the test images. Each column of the matrix represents a predicted class, while each row represents an actual class. Each class has 118 test images.

	Bark	Food	Grass	Metal	Stone	Fabric	Discrimination rate
Bark	**94**	0	0	13	11	0	79.7%
Food	1	**116**	0	0	0	1	98.3%
Grass	8	0	**97**	12	1	0	82.2%
Metal	0	0	0	**110**	8	0	93.2%
Stone	0	0	0	1	**117**	0	99.2%
Fabric	0	0	0	0	13	**105**	90.0%
Average discrimination rate							90.4%

transformation and feature extraction components through learning. Moreover, node A connects to the output node of "Food" and the output image of "Food" is characteristic of other class images.

Table 3 shows the confusion matrix and discrimination rates using the structure of Figure 3 for the test images. This classifier achieves a discrimination rate greater than 90%. We can observe that the classifier tends to mistake the "Bark" class for either the "Metal" or "Stone" class.

In the experiments, we only use simple statistical values as image features. We consider that the classification performance improves if the appropriate texture features are prepared.

5 Conclusions and Future Work

In this paper, we propose a new method for automatic construction of an image classifier that evolves feed-forward network structured image classification programs. The image classifier of the proposed method consists of image transformation, feature extraction, and arithmetic operation components. The greatest advantage of the proposed method is the presence of the image transformation (preprocessing) component. We applied the proposed method to the problem of texture classification and confirmed that it obtained the optimum solution. From the experimental results, the obtained classifier is effective in texture classification. We are, however, recognizing that GIN-IC should be compared with other classifiers in order to evaluate the quality of GIN-IC. In future work, we will apply the proposed method to other problems of image classification and object recognition, particularly to larger problems and other types of problems. Moreover, we will introduce the use of images' color information for classification.

References

1. Cagnoni, S., Lutton, E., Olague, G., eds.: Genetic and Evolutionary Computation for Image Processing and Analysis. Volume 8 of EURASIP Book Series on Signal Processing and Communications. Hindawi Publishing Corporation (2007)

2. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press (1992)
3. Tackett, W.A.: Genetic programming for feature discovery and image discrimination. In: Proceedings of the 5th International Conference on Genetic Algorithms (ICGA-93), Morgan Kaufmann (1993) 303–309
4. Teller, A., Veloso, M.: Algorithm evolution for face recognition: What makes a picture difficult. In: International Conference on Evolutionary Computation, Perth, Australia, IEEE Press (1995) 608–613
5. Teller, A., Veloso, M.: PADO: A new learning architecture for object recognition. In Ikeuchi, K., Veloso, M., eds.: Symbolic Visual Learning. Oxford University Press (1996) 81–116
6. Zhang, M., Fogelberg, C.G.: Genetic programming for image recognition: An LGP approach. In: Applications of Evolutionary Computing, EvoWorkshops2007. Volume 4448 of LNCS., Valencia, Spain, Springer Verlag (2007) 340–350
7. Lam, B., Ciesielski, V.: Discovery of human-competitive image texture feature extraction programs using genetic programming. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO-2004), Part II. Volume 3103 of LNCS., Seattle, WA, USA, Springer-Verlag (2004) 1114–1125
8. Aurnhammer, M.: Evolving texture features by genetic programming. In: Applications of Evolutionary Computing, EvoWorkshops2007. Volume 4448 of LNCS., Valencia, Spain, Springer Verlag (2007) 351–358
9. Aoki, S., Nagao, T.: Automatic construction of tree-structural image transformation using genetic programming. In: Proceedings of the 1999 International Conference on Image Processing (ICIP-99). Volume 1., Kobe, Japan, IEEE (1999) 529–533
10. Nakano, Y., Nagao, T.: 3D medical image processing using 3D-ACTIT; automatic construction of tree-structural image transformation. In: Proceedings of the International Workshop on Advanced Image Technology (IWAIT-2004), Singapore (2004) 529–533
11. Nakano, Y., Nagao, T.: Automatic construction of moving object segmentation from video images using 3D-ACTIT. In: Proceedings of The 2007 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2007), Montreal, Canada (2007) 1153–1158
12. Shirakawa, S., Nagao, T.: Genetic image network (GIN): Automatically construction of image processing algorithm. In: Proceedings of the International Workshop on Advanced Image Technology (IWAIT-2007), Bangkok, Thailand (2007)
13. Shirakawa, S., Nagao, T.: Feed forward genetic image network: Toward efficient automatic construction of image processing algorithm. In: Advances in Visual Computing: Proceedings of the 3rd International Symposium on Visual Computing (ISVC 2007) Part II. Volume 4842 of LNCS., Springer (2007) 287–297
14. Miller, J.F., Thomson, P.: Cartesian genetic programming. In: Genetic Programming, Proceedings of EuroGP'2000. Volume 1802 of LNCS., Edinburgh, Springer-Verlag (2000) 121–132
15. Montana, D.J.: Strongly typed genetic programming. Evolutionary Computation **3**(2) (1995) 199–230
16. Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press (1994)