Hierarchical Feature Construction for Image Classification Using Genetic Programming

Masanori Suganuma, Daiki Tsuchiya, Shinichi Shirakawa and Tomoharu Nagao
Graduate School of Environment and Information Sciences, Yokohama National University
Yokohama, Japan
Email: suganuma-masanori-hf@ynu.jp

***Abstract*—In this paper, we design a hierarchical feature construction method for image classification. Our method has two feature construction stages: (1) feature construction by a combination of primitive image processing filters, and (2) feature construction by evolved filters. We verify the image classification performance of the proposed method on the MIT urban and nature scene dataset. The experimental results show that the two-stage feature construction improves the classification accuracy compared to single stage feature construction. In addition, the proposed method outperforms several existing feature construction methods.**

I. Introduction

Image understanding is an important task in computer vision because of its broad range of applications, such as image classification [1] [2] and object recognition [3] [4]. Many state-of-the-art hand-crafted feature descriptors such as the local binary pattern (LBP) [5], histogram of oriented gradients (HOG) [6], scale-invariant feature transform (SIFT) [7], and Gabor bank [8] have shown good classification performance. Although these hand-crafted descriptors can show good performance on certain image types, they may show poor performance on other types. For instance, LBP works well for texture classification tasks but may not be a good descriptor for scene classification tasks. Hence, a domain adaptive feature construction method is needed.

Deep learning [9] has been used to extract domain adaptive features for image classification. The greedy layer-wise training technique [10], which learns the feature extraction process in a hierarchical way, has been developed in the deep learning community. This method has demonstrated that layer-wise stacking of feature extractors achieves better performance compared to the single layer architecture. Recently, the multiple-layered architecture (i.e., the deep architecture) is setting the winning records in image classification competitions [11] [12].

Evolutionary algorithms, especially genetic programming (GP) [13], have been successfully used to generate domain adaptive features for image classification. To the best of our knowledge, there are two major types of studies: studies that use predefined features as inputs of GP [14] [15], and studies that use raw pixel values as inputs of GP [16] [17]. In the first type, the inputs of GP are domain specific or domain independent statistical features such as the average intensity of the central local region and the standard deviation of all pixels of the image. The second type uses the raw pixel intensity values as the inputs of GP. In [16], the image transformation program is evolved by Cartesian Genetic Programming (CGP) [18] [19], and several statistical moments extracted from the transformed image are used for inputs to the classifier. Al-Sahaf et al. [17] have proposed a two-tier approach that uses GP to select regions of an image and to extract pixel statistics features from the regions. This approach used simple pixel statistics from selected regions, which also produced easily interpretable solutions; a region encloses the eye of a face dataset, which suggests that the eye pixels are descriptive regions for face recognition. Shirakawa et al. [20] have proposed a genetic image network for image classification (GIN-IC). GIN-IC transforms original images using simple image processing filters and extracts statistical features from the transformed images. The GIN-IC evolves the combination of the image processing filters, the statistical features, and the classifier. Hirano et al. [21] have also proposed a feature extraction method using image transformation. These approaches show that the image transformation based on the combination of image processing filters is effective for image classification tasks.

Most GP-based works for evolving feature descriptors select and combine a predefined set of features based on statistical moments. In addition, they construct the feature descriptor by a single evolutionary computation process. Similarly to a deep learning methodology, a hierarchical architecture must perform better than a single stage architecture in the GP-based feature construction methods. In fact, Agapitos et al. [22] have proposed a way to evolve a hierarchical feature construction method with GP and shown that the hierarchical architecture improves the classification accuracy compared to the single stage architecture. The method has four layers: (1) filter bank layer; (2) transformation layer; (3) average pooling layer; (4) classification layer. The filter bank layer is a collection of random-value filters that are used as kernels in convolution operations. The transformation layer has one or two feature extraction stages, which transforms input images into feature maps with evolved programs by GP. In the pooling layer, an average operation is performed over local neighborhoods of a feature map with a window, and the classification layer is a regularized logistic regression classifier. In this method, the image transformation is first performed with random-

978-1-5090-1897-0/16/$31.00

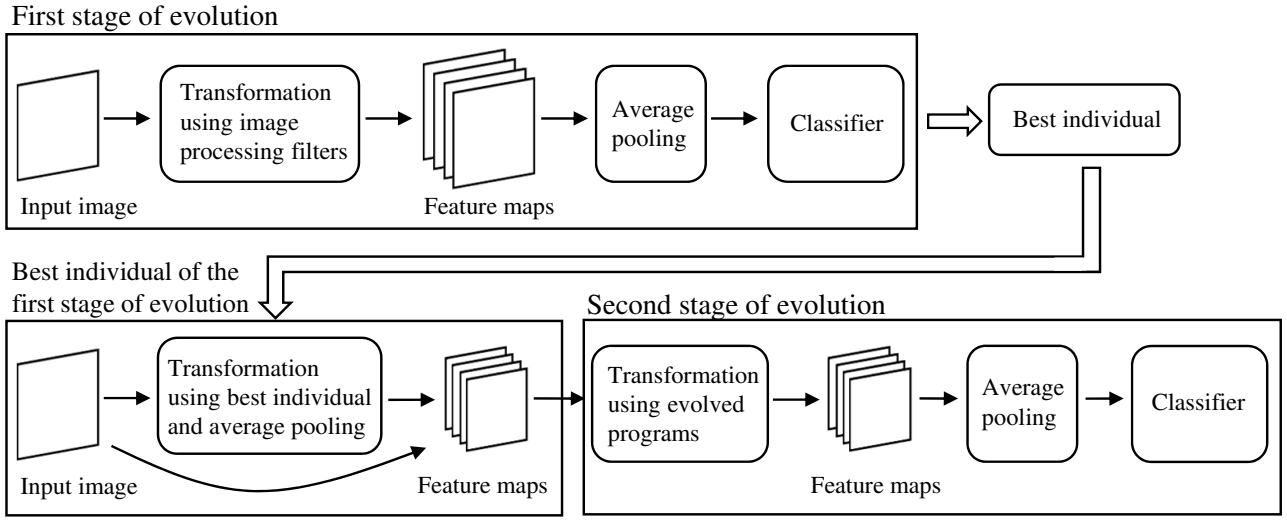

Fig. 1. Outline of the proposed method.

value filters. However, the image transformation based on the combination of image processing filters must perform better than the image transformation based on random-value filters, as was reported in [20] [21].

In this paper, we propose a hierarchical feature construction method based on GP and aim to improve the classification performance by incorporation of the filter bank in the evolution. The architecture of the proposed method has two feature construction stages: (1) a transformation of original images into feature maps based on the combination of image processing filters, and (2) a transformation of the feature maps into final feature maps using evolved programs. By converting 2-D final feature maps to 1-D vectors, we obtain the final feature descriptor. By incorporating the filter bank in the evolution, we aim to improve the classification performance. To evaluate the classification performance of the proposed method, we apply the proposed method on the MIT urban and nature scene dataset.

The rest of this paper is organized as follows. Section II describes the proposed method. Section III presents the experimental setup and results. Section IV concludes and proposes future works.

II. Hierarchical feature construction

The outline of the proposed method is shown in Fig. 1. The multilayer architecture is defined as follows:

1) Filtering layer.
2) Average pooling layer.
3) Transformation layer.
4) Classification layer.

A. Filtering layer

The filtering layer extracts the discriminative features from the input images using image processing filters. Fig. 2 shows the structure of the filtering layer and an example of the phenotype and the genotype. The structure of the layer is

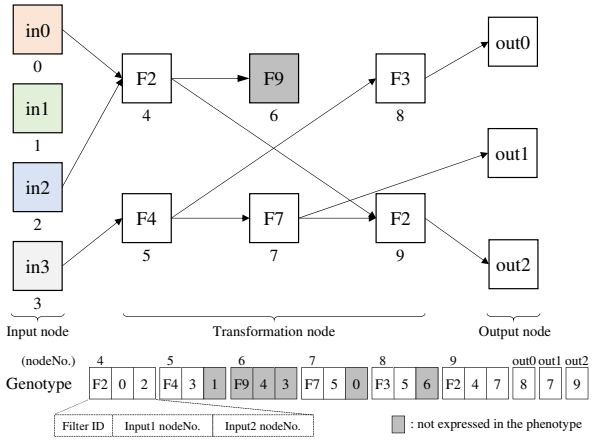

Fig. 2. Example of the filtering layer and its phenotype and genotype.

a feed-forward network structure that is similar to CGP and GIN-IC. Each node in the network has one or two connections to previous nodes or input nodes. The types and connections of nodes are optimized by the evolutionary algorithm. The nodes are categorized into input nodes, transformation nodes, and output nodes. Four input nodes correspond to the red, green, and blue components of the input RGB image and a gray scaled image. The transformation nodes execute the image transformation using the predefined image processing filters shown in Table I, which transform one or two 2-D images to a single 2-D image of the same size. The output nodes have feature maps from the original images through the transformation nodes. The output nodes pick up the transformed images by the image filters. We call the transformed output images *feature maps*.

B. Average pooling layer

The average pooling layer receives n feature maps from the previous filtering layer. The layer is the $w \times w \times n$

TABLE I
IMAGE PROCESSING FILTERS USED IN THE FILTERING LAYER

Function name	Description
Ave	Averaging filter with 3×3, 5×5 window
Max	Maximum filter with 3×3, 5×5 window
Min	Minimum filter with 3×3, 5×5 window
Sob	Sobel filter with 3×3, 5×5 window
Lap	Laplacian filter with 3×3, 5×5 window
Gau	Gaussian smooth filter with 3×3, 5×5 window
LoG	Laplacian of gaussian filter with 3×3, 5×5 window
Exp	Expansion processing
Con	Contraction processing
Gab0	Gabor filter with 7×7, 11×11 window with orientation of 0 degree
Gab45	Gabor filter with 7×7, 11×11 window with orientation of 45 degree
Gab90	Gabor filter with 7×7, 11×11 window with orientation of 90 degree
Gab135	Gabor filter with 7×7, 11×11 window with orientation of 135 degree
Add	Add input two images pixel by pixel
Sub	Subtract input two images pixel by pixel
Mul	Multiply input two images pixel by pixel
Div	Divide input two images pixel by pixel
Abs	Absolute subtraction of input two images pixel by pixel

TABLE II
FUNCTION SET USED IN THE TRANSFORMATION LAYER

Function	# Inputs	Description
$+$	2	Add two inputs
$-$	2	Subtract two inputs
$\times$	2	Multiply two inputs
$\div$	2	Divide two inputs
Max	4	The largest value of inputs
Min	4	The smallest value of inputs
Ave	4	The average value of inputs
log	1	Take the natural logarithm for a input
Sqrt	1	Extract a square root of a input
$\times 2.0$	1	Multiply a input by 2.0
$\times 0.5$	1	Multiply a input by 0.5
$\times 0.1$	1	Multiply a input by 0.1

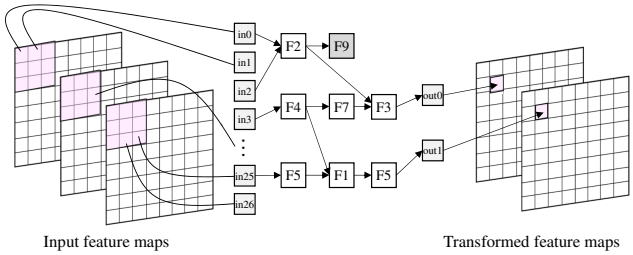

Fig. 3. Example of the transformation layer that transforms $8 \times 8 \times 3$ feature maps into $8 \times 8 \times 2$ feature maps using the evolved program. The inputs are the $3 \times 3 \times 3$ pixel intensity values surrounding a target pixel.

representation, which means that the $w \times w$-sized n feature maps are used. In the average pooling layer, an average operation is performed over the local neighborhoods of the $w \times w$ feature map with 2×2 window. As a result, the $w \times w \times n$ feature maps are reduced into $(w/2) \times (w/2) \times n$ feature maps.

C. Transformation layer

The transformation layer receives n feature maps from the average pooling layer and the down-sampled four original images (red, green, and blue component of an RGB-image and a gray scale image), which is the $(w/2) \times (w/2) \times (n+4)$ representation. The original images are used to avoid the loss of important information through the previous filtering layer processing. In this layer, the output pixels are calculated by only using the local pixels in the input feature maps. The calculation process is defined by the feed-forward network structure like CGP and optimized by the evolutionary algorithm. In other words, the inputs of the evolved program are the pixel values of the $s \times s$ local patches in a given pixel. The d output images, the $1 \times 1 \times d$ representation, are obtained by the processing in this layer, i.e., the $n+4$ feature maps are transformed into d feature maps, which is a $(w/2) \times (w/2) \times d$ representation. The mathematical functions displayed in Table II are used as the function set for the evolved programs. Fig. 3 shows an example of the transformation layer.

D. Classifier layer

The classifier layer receives d feature maps, which is a $h \times h \times d$ representation. By converting 2-D feature maps to 1-D vectors, we obtain the final feature descriptor, which has $h \times h \times d$ features for classification. We use a linear support vector machine (SVM) as the classifier. The cost parameter c is set to 1. For the SVM solver, we use the implementation of multi-core LIBLINEAR (version 2.1-4) [23].

E. Training protocol

The training procedure in the proposed method consists of two-stage evolutionary optimization. In the first stage of evolution, we evolve the filtering layer F using the architecture of $I \rightarrow F \rightarrow P \rightarrow C$, where I is the input image, F is the filtering layer, P is the average pooling layer, and C is the classification layer. The filtering layer generates the $w \times w$-sized n feature maps, where w is the size of the original image. The average layer reduces the resolution of the representation to $(w/2) \times (w/2)$-sized n feature maps, and the feature maps are used for classification. We employ the classification accuracy for validation images as the fitness of each individual in the first stage of evolution. At the end of the first evolution stage, we obtain the best filter combination F_{best} that generates the $w \times w$-sized n feature maps.

In the second stage of evolution, we evolve the transformation layer T using the architecture of $F_{\text{best}} \rightarrow P \rightarrow T \rightarrow P \rightarrow C$, where F_{best} is the feature map generated in the first evolution stage, T is the transformation layer, P is the average pooling layer, and C is the classification layer. The transformation layer receives the down-sampled evolved feature maps F_{best} (consisting of n feature maps) and the down-sampled four original images. The transformation layer generates the $(w/2) \times (w/2)$-sized d feature maps, and the average layer reduces the resolution of the representation to $(w/4) \times (w/4)$-sized d feature maps, which are used for classification. We also employ the classification accuracy for validation images as the fitness similar to the first stage

Fig. 4. Sample images from the MIT urban and nature scene dataset.

TABLE III
PARAMETER SETTINGS FOR THE PROPOSED METHOD

Parameter	Value
Num. of generations (Filtering layer)	3000
Num. of generations (Transformation layer)	4000
Generation alternation model	MGG [24]
Population size	50
Children size	10
Crossover rate	1.0
Ratio for uniform crossover	0.8
Mutation rate	0.1

TABLE IV
PARAMETER SETTING FOR GPIP

Parameter	Value
Num. of generations	500
Generation alternation model	MGG [24]
Population size	50
Children size (genotype 1)	50
Children size (genotype 2)	10
Crossover rate	1.0
Mutation rate	0.05

of evolution. In this way, we obtain the best program that generates the $(w/2) \times (w/2)$-sized d feature maps.

III. EXPERIMENT AND RESULTS

A. Data sets

We test the proposed method with the MIT urban and nature scene dataset[1]. This dataset has eight categories (coast & beach, forest, highway, city center, mountain, open country, street, and tall building), and all of the images are 256×256 pixels. In this paper, each image in the dataset is resized from its original size to 128×128 pixels by linear interpolation, and then, we randomly choose 50 images from each category as the learning set for training SVM, 50 images from each category as the evaluation set for evaluating the individuals in the evolutionary optimization, and $1,488$ images as the test set. Some image examples of this dataset are shown in Fig. 4.

B. Experiment setup

Parameter setting for the proposed method is shown in Table III. Both the filtering layer and the transformation layer use the same genetic operators, the uniform crossover and the mutation for each gene. We implemented the filtering layer, the transformation layer, and the average pooling layer of our proposed method on the graphic processing unit (GPU) using the compute unified device architecture (CUDA) [2], and hence, each pixel is independently processed. We run our experiments on a machine with 3GHz CPU, 32GB RAM, and NVIDIA GeForce GTX TITAN X GPU.

For comparison, we tested the five following methods:

1) Single transformation (Proposal 1). This architecture is $I \rightarrow F \rightarrow P \rightarrow C$. In the filter layer, we adopt the CGP-based method with 40 filtering nodes and 8 output nodes for evolving the filtering network, i.e., the filtering layer generates 8 feature maps. The average pooling layer performs an average operation with 2×2 window.
2) Double transformation (Proposal 2). This architecture is $F_{\rm best} \rightarrow P \rightarrow T \rightarrow P \rightarrow C$. The filtering layer $F_{\rm best}$ is the one obtained by the method of 1) single transformation. The transformation layer uses $3 \times 3 \times 12$ patches as the input and generates 16 feature maps as the outputs. The number 12 indicates that the transformation layer receives 8 feature maps obtained by the filtering layer and down-sampled 4 original images. The number of the function nodes is 50. The average pooling layer performs the average operation with 2×2 window.
3) A GP method that transforms feature maps generated by random-valued filter bank into the final feature maps using an evolved program (SST) [22]. This architecture is $I \rightarrow F_{\rm rand} \rightarrow P \rightarrow T \rightarrow P \rightarrow C$. The random filter bank $F_{\rm rand}$ is composed of 50 filters with 3×3 receptive fields. We generated 10 filters from each of $U(-1.0, 1.0)$, $U(-5.0, 5.0)$, $D(1, 5)$, $N(1.0)$, $N(5.0)$, where $U(a, b)$ denotes the uniform sampling of real valued numbers within the $[a, b]$ interval, $D(a, b)$ denotes uniform sampling of integer numbers $[a, b]$ interval, and $N(a)$ denotes sampling from a normal distribution with zero mean and the standard deviation a. As a result, $F_{\rm rand}$ generates 50 feature maps. The transformation layer uses $3 \times 3 \times 50$ pixels as the input and generates 16 outputs, i.e., the transformation layer generates 16 feature maps. The architecture of the transformation layer is identical to that in the proposed method. The number of the function nodes is 200.
4) A method that transforms an original image using the filter bank and then extracts the statistical moments from the transformed images [21] (we call this method GPIP in this paper) . The filter bank consists of 14 well-known image processing filters, such as the laplacian filter, and the statistical moments consist of simple 17 statistical values, such as standard deviation and mean value, see [21] for details. The parameter setting for this method is shown in Table IV. We set the number of generations to 500 since the fitness transition was converged ade-

[1]http://cvcl.mit.edu/database.htm, Urban and Natural Scene Categories.

[2]https://developer.nvidia.com/cuda-zone, CUDA Zone

TABLE V
COMPARISON OF CLASSIFICATION ACCURACY.

	Best	Average
Proposal 1	0.786	0.759
Proposal 2	0.831	0.810
SST [22]	0.681	0.667
GPIP [21]	0.771	0.718
Gabor bank [8]	0.764	0.764

TABLE VI
CONFUSION MATRIX OF THE PROPOSED DOUBLE TRANSFORMATION METHOD ON THE MIT URBAN AND NATURE SCENE DATASET.

	(c)	(f)	(h)	(cc)	(m)	(o)	(s)	(f)
(c) Coast	177	1	9	0	10	13	0	0
(f) Forest	0	154	0	0	9	14	0	1
(h) Highway	7	0	73	13	6	11	0	0
(cc)City center	0	2	1	152	1	2	0	0
(m) Mountain	15	6	7	3	156	37	0	0
(o) Open country	13	17	7	2	23	197	1	0
(s) Street	0	0	1	0	0	0	125	16
(f) Tall building	0	0	0	0	1	0	2	203

quately. In this method, we used a gaussian kernel with parameter $\gamma = 0.6$ as the kernel function of SVM.

5) Gabor filtering with 8 orientation at 4 different scales, and the output of each filter is averaged on a 4×4 grid to form a feature vector [8].

In order to compare the classification errors by the each method on the test set, we employ 10-fold cross-validation. The averaged 10 SVM test-fold accuracies are computed for the comparison of each method.

C. Results

Table V shows the classification accuracies of each method in five evolutionary runs. The results suggest that the proposed method, i.e., double transformation (Proposal 2), outperforms the rest of the methods. Comparing the double transformation (Proposal 2) and the SST, we observe that the classification accuracy of the double transformation is significantly better than that of the SST (from 0.681 to 0.831). This means that incorporating the evolved filter bank can be beneficial for improving the classification performance as compared to the random-value filters. Fig. 5 shows examples of a tall building image and its transformed images by the filtering layer of the proposed method. From these transformed images, the distinctive features such as vertical edges seen in the tall building are extracted by the filtering layer, which seems to be beneficial for the classification. The double transformation method shows better classification accuracy than that of the single transformation method and the GPIP, which indicates that two-stage transformation can contribute to constructing the discriminative descriptors. In addition, the descriptor constructed by the proposed method outperforms the hand-crafted features (the Gabor bank).

We tested a Wilcoxon rank sum test for comparison of Proposal 2 and other methods. We set the significance level at 5%. The p-value for Proposal 2 and Proposal 1 is 0.024, which means that Proposal 2 is statistically better regarding accuracy than Proposal 1. The p-value for Proposal 2 and the SST is 0.0079 and the p-value for Proposal 2 and the GPIP is 0.036, which means that Proposal 2 outperforms the SST and the GPIP.

Original image (Tall building) | Transformed image 1 | Transformed image 2

Fig. 5. Examples of an original image and its transformed images by the filtering layer.

Table VI shows the confusion matrix of the classification result by the proposed double transformation method for the test images. From this confusion matrix, we can see that the evolved descriptor can successfully classify City center, Street and Tall building, whereas it can not classify Highway and Mountain well.

The time costs of Proposal 1 and Proposal 2 are around 11.9 hrs and 10.8 hrs, respectively.

IV. CONCLUSION

In this paper, we have presented a hierarchical feature construction method for image classification. The proposed method has two-stage feature construction: (1) feature construction by a combination of primitive image processing filters, and (2) feature construction by evolved filters. We have evaluated our method on the MIT urban and nature scene dataset, and outperformed several GP methods and hand-crafted descriptor. Experimental results showed that incorporating the filter bank in the evolution and the two-stage evolution can be beneficial to the classification performance.

In our future work, we will evaluate our method with a more challenging dataset.

ACKNOWLEDGMENT

This work was supported by JSPS Grant-in-Aid for Scientific Research (B) Grant Number 26280056.

REFERENCES

[1] O. Chapelle, P. Haffner, and V. N. Vapnik, "Support vector machines for histogram-based image classification," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1055–1064, 1999.

[2] E. Nowak, F. Jurie, and B. Triggs, "Sampling strategies for bag-of-features image classification," in *Proc. European Conf. on Computer Vision (ECCV '06)*, 2006, pp. 490–503.

[3] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '99)*, vol. 2, 1999, pp. 1150–1157.

[4] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[5] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face recognition with local binary patterns," in *Proc. European Conf. on Computer Vision (ECCV '04)*, 2004, pp. 469–481.

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '05)*, vol. 1, 2005, pp. 886–893.

[7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[8] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '10)*, 2010, pp. 3485–3492.

[9] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013.

[10] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[13] J. R. Koza and R. Poli, "Genetic programming," in *Search Methodologies*. New York, USA: Springer, 2005, pp. 127–164.

[14] M. Zhang and W. Smart, "Multiclass object classification using genetic programming," in *Applications of Evolutionary Computing*, 2004, pp. 369–378.

[15] M. Zhang, X. Gao, and W. Lou, "A new crossover operator in genetic programming for object classification," vol. 37, no. 5, 2007, pp. 1332–1343.

[16] T. Kowaliw, W. Banzhaf, N. Kharma, and S. Harding, "Evolving novel image features using genetic programming-based image transforms," in *IEEE Congress on Evolutionary Computation (CEC '09)*, 2009, pp. 2502–2507.

[17] H. Al-Sahaf, A. Song, K. Neshatian, and M. Zhang, "Two-tier genetic programming: towards raw pixel-based image classification," vol. 39, no. 16, 2012, pp. 12 291–12 301.

[18] J. F. Miller and P. Thomson, "Cartesian genetic programming," in *European Conference, EuroGP 2000*, R. Poli, W. Banzhaf, W. B. Langdon, J. Miller, P. Nordin, and T. C. Fogarty, Eds. Springer-Verlag, 2000, pp. 121–132.

[19] S. Harding, "Evolution of image filters on graphics processor units using cartesian genetic programming," in *IEEE Congress on Evolutionary Computation (CEC '08)*, 2008, pp. 1921–1928.

[20] S. Shirakawa, S. Nakayama, and T. Nagao, "Genetic image network for image classification," in *Applications of Evolutionary Computing*, 2009, pp. 395–404.

[21] Y. Hirano and T. Nagao, "Feature transformation using filter array for automatic construction of image classification," in *IEEE 7th International Workshop on Computational Intelligence and Applications (IWCIA '14)*, 2014, pp. 59–64.

[22] A. Agapitos, M. O'Neill, M. Nicolau, D. Fagan, A. Kattan, A. Brabazon, and K. Curran, "Deep evolution of image representations for handwritten digit recognition," in *IEEE Congress on Evolutionary Computation (CEC '15)*, 2015, pp. 2452–2459.

[23] M.-C. Lee, W.-L. Chiang, and C.-J. Lin, "Fast matrix-vector multiplications for large-scale logistic regression on shared-memory systems," in *Proc. IEEE Conf. on Data Mining (ICDM '15)*, 2015, pp. 835–840.

[24] H. Sato, I. Ono, and S. Kobayashi, "A new generation alternation model of genetic algorithms and its assessment," *J. of Japanese Society for Artificial Intelligence*, vol. 12, no. 5, pp. 734–744, 1997 (in Japanese).