Sensors and Materials, Vol. 38, No. 6 (2026) 3255–3270
MYU Tokyo

S & M 4501

Dataset Distillation on Medical Dataset Using Genetic Algorithm

Takumi Sato,[1,2*] Yasumasa Tamura,[1] and Masahito Yamamoto[1]

[1]Faculty of Information Science and Technology, Hokkaido University,
Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido 060-0814, Japan
[2]Kadinche Corporation, F14-23-5F Daikanyama-cho, Shibuya-ku, Tokyo 150-0034, Japan

(Received January 5, 2026; accepted June 11, 2026)

Keywords: dataset distillation, genetic algorithm, MedMNIST

MedMNIST, which is a collection of medical classification datasets, contains medical images obtained with a variety of techniques, such as computed tomography and optical coherence tomography, and images captured using a microscope. Compressing MedMNIST datasets using a dataset distillation method is a challenging task in the images per class (IPC) = 1 setting. Regarding this problem, methods that do not use gradients to update a compressed dataset largely remain unexplored, especially in MedMNIST. We propose a simple dataset distillation method based on the Deep Learning Evolutionary Algorithm (DL-EA) to optimize synthetic images mainly focused on MedMNIST. Our proposed method updates synthetic images and evaluates them using a subset of training data. Since DL-EA is a neural network (NN) training method, it cannot be applied to dataset distillation directly so we extended it by updating synthetic images rather than NN weights; this is our main contribution. To extend our method even further, we made two improvements. (1) We added the Laplace crossover method to optimize synthetic images. (2) To reduce the computational cost of each generation, we used only the subset of the training dataset. Our method is evaluated on MNIST and seven medical datasets included in MedMNIST. These datasets have been evaluated by gradient-based dataset distillation. We mainly focus on IPC = 1. Our method has a higher accuracy for MNIST and seven medical datasets included in MedMNIST than does strong random selection.

1. Introduction

Dataset distillation[(1)] is a method for compressing a dataset into a few synthetic images. Compressing a dataset greatly reduces training time and is considered to be one way to lower privacy concerns. In recent years, dataset distillation has been improved and is able to compress datasets such as CIFAR-100,[(2)] ImageNet,[(3)] medical image,[(4)] and EUROSAT.[(5)] Dataset condensation (DC)[(6)] and matching training trajectory (MTT)[(7)] frameworks have been introduced, which improved dataset distillation. Squeeze, recover, and relabel (SRe^{2}L)[(8)] uses the decoupling approach to recover data from a pretrained model.

*Corresponding author: e-mail: sato.takumi.g0@elms.hokudai.ac.jp
https://doi.org/10.18494/SAM6159

ISSN 0914-4935

However, the dataset distillation of medical images has received less attention than that of natural images. Medical images are obtained using a wide variety of capture devices such as computed tomography (CT), ultrasound, optical coherence tomography OCT, and microscopy devices, which is different from natural images. We are focusing on the MedMNIST dataset since it comprises images obtained by CT, OCT, and microscopy and has been evaluated[9,10] with regard to dataset distillation. Looking at the evaluated[9] results of MedMNIST images, DC and MTT attain comparable accuracies for IPC = 10 and 50 but struggle in IPC = 1 settings. High-order progressive trajectory matching (HoP-TM)[10] has improved MTT for medical images with IPC = 5, 10, 100, and 1000, but was evaluated only on PathMNIST and COVID19-CXR datasets, not other datasets included in MedMNIST.

Applying dataset distillation to MedMNIST will contribute to solving the above problem. As far as we know, there are a few studies[11,12] in which an evolutionary algorithm was used in dataset distillation whereby comparable accuracies were achieved on CIFAR-10 and MNIST, but it has not been examined on datasets such as ImageNet[3] and MedMNIST.[4] Our method uses a genetic algorithm to optimize the synthetic image on the basis of the Deep Learning Evolutionary Algorithm (DL-EA). Our genetic algorithm uses synthetic images as individuals, whereas the original DL-EA uses neural networks (NNs) as individuals. Unlike previous methods,[11,12] our method is based on DL-EA,[13] which is an NN training method. It does not use clustering or multi-objective optimization, which makes our proposed method simple and easy to implement. We extended DL-EA in order to update the synthetic images rather than the NNs, which is our main contribution. We made two further improvements. First, we added Laplace crossover to DL-EA instead of just adding noise. By adding an interaction between selected individuals using a crossover method,[14] there is a higher possibility that the individuals will be updated to better synthetic images. This cannot be accomplished by only adding noise interaction between selected individuals. Second, we used only a subset of the training dataset for calculating the fitness value, which uses most of the computational time in our method. The last evaluation will be performed on test data. Previous methods[11,12] tend to solve high computational time requirement by using a small population or reducing the population during optimization. Our approach reduces the computational cost when using the training dataset for calculating the fitness value in MNIST, because the subset of the training dataset has 12000 images, while the rest of the training dataset has 48000 images, and it will take much time if the batch size is smaller than the subset of the training dataset.

We evaluated our proposed method on the datasets MNIST, PathMNIST, DermaMNIST, OCTMNIST, BloodMNIST, TissueMNIST, OrganAMNIST, OrganCMNIST, and OrganSMNIST, and achieved higher accuracy in about eight of the datasets than when using strong random selection. Strong random selection selects the best synthetic dataset by evaluating a randomly selected dataset. Strong random selection is more optimized than just selecting a dataset randomly. We are focusing on IPC = 1 settings since it presents a more difficult task.

In this paper, we propose a dataset distillation method that uses an evolutionary algorithm to create a synthetic dataset. Our contributions are as follows.

- We extended DL-EA and applied it to the dataset distillation task.
- We reduced the computational time by using a subset of training data, which makes setting a larger population size possible.
- We added the Laplace crossover method to DL-EA to improve accuracy.
- The evolutionary-algorithm-based dataset distillation method was applied to a medical dataset and higher accuracy was achieved than by strong random selection.

2. Related Work

2.1 Dataset distillation

Knowledge distillation,[15] where knowledge from the teacher network to the student network is compressed, has been introduced. Dataset distillation[1] has also been introduced, where knowledge to a synthetic dataset is compressed. This reduces training time and is one way by which privacy may be protected. DC[6] was introduced to match the gradients of the model trained on real data and the model trained on a synthetic dataset. This decreases the computational time compared with that of the previous method. MTT[7] has been proposed as a trajectory-based method, and automatic training trajectory (ATT)[16] is an improved version. Random truncated backpropagation through time (RaT-BPTT)[17] is an MTT-based method that has achieved high accuracy, but requires a huge memory. SRe^{2}L[8] uses a decoupling approach to recover data from the model and utilizes soft labels, achieving comparable accuracy on ImageNet-1K. Realistic, diverse, and efficient dataset distillation (RDED)[18] improved on it by using V-information[19] and reduced the computational cost. Recently, a method that uses a diffusion model[20] and a method [21] that does not need to pretrain the diffusion model have been introduced. Gaining improvement from full labels at near-zero cost (GIFT)[22] uses soft labels to achieve higher accuracy. Poster dataset distillation (PoDD)[23] is a poster-like method and has achieved IPC lower than 1. These methods have achieved high accuracy with natural images, but most of them have not been evaluated on medical datasets.

There are some methods in which dataset distillation is applied to medical datasets. Li *et al.*[24] applied dataset distillation to medical datasets. There is a paper[9] on the evaluation of dataset distillation using DC and MTT on PathMNIST, DermaMNIST, OCTMNIST, BloodMNIST, TissueMNIST, OrganAMNIST, OrganCMNIST, and OrganSMNIST, which are included in the MedMNIST dataset, where an accuracy reduction of about 30% on some datasets compared with the accuracy when trained on a full dataset was reported. HoP-TM,[10] which is based on MTT, has achieved high accuracy compared with DC and MTT on the PathMNIST dataset, but it has not been evaluated on other datasets included in MedMNIST.

There have been some approaches of using evolutionary algorithms to generate synthetic images. Barbiero *et al.*[25] proposed a coreset selection method of using an evolutionary algorithm. A method that uses a genetic algorithm[11] achieved accuracy comparable to dataset distillation[1] on MNIST by using 100 images with IPC higher than 1. The

population size was 30. Dataset distillation evolution strategy (DEvS)[12] achieved comparable accuracy on CIFAR-10. Furthermore, by using clustering to reduce population size, it can be run in a few hours. Neither of them have been evaluated on medical datasets. In our work, we applied our method to medical datasets and used subsets of the train dataset to reduce computational load.

We used a slightly modified version of random selection (RS) as a baseline. Repeated sampling of random subsets (RS2),[26] in which a subset of the training dataset is selected every epoch, has achieved high accuracy on ImageNet.

2.2 Genetic algorithm

Training an NN by using a genetic algorithm has been proposed.[27] A batch of training samples is used to reduce the fitness calculation time. DL-EA[13] proved that EA can be used to train ResNet-18. We believe that if DL-EA can train ResNet-18, which has more parameters than a synthetic dataset, it should able to optimize synthetic datasets.

There has been some research on the use of real values in genetic algorithms. Some crossover methods, such as simulated binary crossover (SBX),[28] simplex crossover (SPX),[29] parent-centric recombination operator (PCX),[30] Laplace crossover (LX),[14] and direction-based exponential crossover (DEX)[31] have been proposed. Kobayashi[32] stated that mutation may not be necessary.

3. Data, Materials, and Methods

3.1 Methods

Our method is based on DL-EA[13] and is described in Fig. 1. Our method uses DL-EA with Laplace crossover that uses two parents to create one child.

First, we create some individuals containing synthetic datasets. The number of synthetic images per individual is the same as the number of classes in the dataset since we are mainly

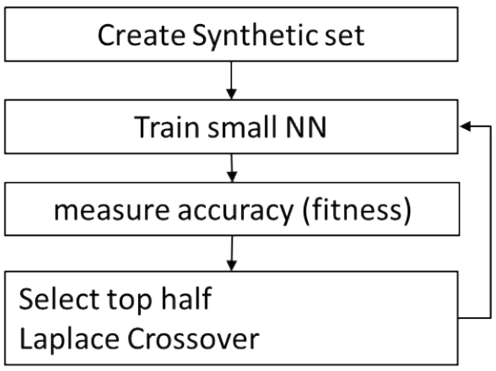

Fig. 1. Proposed method.

focusing on IPC = 1. The individual is resized to a one-dimensional vector and fed into the crossover method. We divide the vector and reconstruct the synthetic image by resizing on the evaluation phase and linking it to the corresponding one-hot label. For example, when using MNIST, the individual contains 10 synthetic images corresponding to each class since MNIST has 10 class datasets. The label of each synthetic image is one-hot vector and is not optimized in our method. The population is created to run the genetic algorithm. The population size is 300.

Secondly, we initialize the synthetic dataset by selecting a random image from the training dataset other than the subset of the training dataset. We select one image from each class for each individual and the same image may be used for different individuals. When IPC = 1, one image is selected randomly for each class. Normalization is applied to the image, but only to initial individuals. We do not use the subset of the training dataset for synthetic dataset initialization.

Next, we evaluate the synthetic dataset by training the target network using only the synthetic data. The accuracy of the subset of the training dataset is used as an optimization parameter. We only evaluate the accuracy of individuals per generation once to reduce computational load. We use conv-128[33] as the network and train for 300 epochs. Adam is used for the optimizer and cross–entropy loss for loss calculation. Only normalization is used for data preprocessing; random rotation, random clip, and ZCA whitening are not used. The model is recreated every time and trained from scratch for evaluation.

Then, we keep the top 50% of the population among the total population and create a new individual by the crossover method. We do not keep or use the other 50% in any way. For example, when the population size is 300, we keep the top 150 individuals and create 150 new individuals by the crossover method. This means that if one individual's accuracy is always in the top 50% of the whole population, it will exist until the last generation. Every remaining individual is used as a parent in the crossover method. If the crossover method requires multiple parents, we select random parents. For example, in Laplace crossover, we need two parents so we select two parents randomly; the same parent may be called multiple times. The selection after keeping the top 50% of individuals is random and accuracy does not have any effect on parent selection. The crossover method will generate one child from a pair of parents. If each value of the child vector is out of the 0–1 range, the value will be clipped. This means that the initial population is not clipped, but every child individual will be in the 0–1 range. We apply the DL-EA method. Equation (1) describes the update method of DL-EA. ρ is the Gaussian with mean = 0 and variance = 1. g is the generation index.

$$w_i = w_i + \rho \times \delta,\ \delta = 1/g \tag{1}$$

Laplace crossover,[14] which is used in our method, is defined by Eq. (4). Equations (2)–(6) are defined by Kusum and Thakur[14]. u is a distributed random number [0,1]. a (lx_scale) = 0.5 and b (lx_location) = 0 are used in our experiment. Equation (2) describes the density function of the Laplace distribution. Equation (3) describes the distribution function of the Laplace distribution. β is calculated by Eq. (4). Offspring $x^{(1)} = \left(x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \ldots, x_n^{(1)}\right)$ and

$x^{(2)} = \left(x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, \ldots, x_n^{(2)}\right)$ areparents. UsingEqs. (4), (5), and (6), $y^{(1)} = \left(y_1^{(1)}, y_2^{(1)}, y_3^{(1)}, \ldots, y_n^{(1)}\right)$ and $y^{(2)} = \left(y_1^{(2)}, y_2^{(2)}, y_3^{(2)}, \ldots, y_n^{(2)}\right)$ are calculated. Since we only need one child, we use only $y^{(1)}$ in our experiment. The update method of DL-EA is applied after the crossover, followed by clipping. Normalization is not applied to a child individual.

$$f(x) = \frac{1}{2b}\exp\left(-\frac{|x-a|}{b}\right), -\infty < x < \infty \tag{2}$$

$$F(x) = \begin{cases} \frac{1}{2}\exp\left(\frac{|x-a|}{b}\right), x \le a \\ 1 - \frac{1}{2}\exp\left(\frac{|x-a|}{b}\right), a > a \end{cases} \tag{3}$$

$$\beta = \begin{cases} a - b\log_e(u), u \le \frac{1}{2} \\ a + b\log_e(u), u > \frac{1}{2} \end{cases} \tag{4}$$

$$y_i^{(1)} = x_i^{(1)} + \beta\left|x_i^{(1)} - x_i^{(2)}\right| \tag{5}$$

$$y_i^{(2)} = x_i^{(2)} + \beta\left|x_i^{(1)} - x_i^{(2)}\right| \tag{6}$$

Finally, we evaluate all individuals and repeat the process. We recalculate all the individuals and do not reuse the fitness value of the previous generation to check that it will retain the accuracy under a different initial network weight. We also do not use fixed seeds to generate the same initial network weight on every individual or generation.

We will perform this calculation for 1000 generations. We used Laplace crossover[14] for the crossover method. In the previous method,[9] the image resolution was resized to 32 × 32, but in this study, we change the zero padding from 1 to 3 of the first convolution layer to match the output feature. We also change the input channel of the first convolution layer depending on the channel size of the dataset. We select the best synthetic data throughout the entire training process on the basis of the accuracy of the subset of the training dataset, and using the test data, we evaluate the recreated model trained using the best synthetic data.

Our hyperparameters are as follows: survival ratio = 50%, population size = 300, child size = half the population (150), parent selection = random selection from surviving

individuals, number of generations = 1000, crossover method = Laplace crossover, crossover rate = 1.0, lx_scale = 0.5, lx_location = 0, and child clipping = 0–1.

3.2 Dataset

We used MNIST[34] and MedMNIST[4] for evaluation. MNIST is a dataset of handwritten digits with a resolution of 28 × 28 and 10 classes. It has 10000 test images and 60000 training images. We split the training dataset to 8:2 so 48000 images are used for synthetic data initialization and 12000 images for the subset of the training dataset. We used fixed split. For normalization in dataset preprocessing, we used mean = 0.1307 and std = 0.3081.

MedMNIST is a collection of datasets that includes images obtained using colon pathology (PathMNIST), dermatoscopy (DermaMNIST), retinal OCT (OCTMNIST), blood cell microscopy (BloodMNIST), kidney cortex microscopy (TissueMNIST), and abdominal CT (OrganAMNIST, OrganCMNIST, OrganSMNIST). PathMNIST, OCTMNIST, BloodMNIST, TissueMNIST, OrganAMNIST, OrganCMNIST, and OrganSMNIST are licensed under Creative Commons Attribution 4.0 International. DermaMNIST is licensed under Creative Commons Attribution-NonCommercial 4.0 International. The synthetic dataset was generated using training images as initial images and later modified by our method. All datasets were resized to a resolution of 28 × 28. On MedMNIST, since the dataset is split into train/val/test, we used the training dataset only for synthetic data initialization and treated val data as the subset of the training dataset and used it as the fitness value. We did not split the training dataset. The test data was used in the final evaluation. Input image size, the number of classes, train/val/test sample sizes of datasets included in MedMNIST are described on Table 1. For normalization in dataset preprocessing, we used mean = 0.5 and std = 0.5 on all datasets included in MedMNIST.

3.3 Experiment settings

We used population = 300 and number of generations = 1000 for the genetic algorithm. Conv-128[33] is frequently used for evaluation in dataset distillation. It was trained for 300 epochs, as in our previous research. RTX 4080x2, RTX5090, and DGX Spark were used to run the

Table 1
Input image size, data modality, number of classes, train/val/test sample size of dataset included in MedMNIST.

Dataset	Input image size	Data Modality	Number of classes	Training samples	Val samples	Test samples
PathMNIST	28 × 28 × 3	Colon Pathology	9	89996	10004	7180
DermaMNIST	28 × 28 × 3	Dermascope	7	7007	1003	2005
OCTMNIST	28 × 28 × 1	Retinal OCT	4	97477	10832	1000
BloodMNIST	28 × 28 × 3	Blood Cell Microscope	8	11959	1712	3421
TissueMNIST	28 × 28 × 1	Kidney Cortex Microscope	8	165466	23640	47280
OrganAMNIST	28 × 28 × 1	Abdominal CT	11	34561	6491	17778
OrganCMNIST	28 × 28 × 1	Abdominal CT	11	12975	2392	8216
OrganSMNIST	28 × 28 × 1	Abdominal CT	11	13932	2452	8827

program. Inspyred[35] was used to implement Laplace crossover, and Pytorch[36] was used to train and test NN. lx_scale = 0.5, lx_location = 0, and crossover_rate = 1.0 were used as parameters in Laplace crossover. Although we used multiple GPUs, our method was run on a single GPU. The computational time in our algorithm was mostly evaluation time, so in order to reduce the computational time, only the validation data was used for the calculation of the fitness function. The train dataset was only used as an initial value of synthetic images. We executed our proposed method using a random seed once to generate the synthetic dataset and trained and evaluated an NN 10 times using different seeds. Adam with learning rate=0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, ε=1e-08, and weight decay=0 was used in the experiment. The batch size was the same as the number of total synthetic images. We evaluated random selection since the previous method used ResNet-18 to evaluate random selection. We also wanted to see to what degree our method optimized the synthetic image. We used strong random selection, which generates 300 individuals, chose the individual that achieved the highest accuracy in the subset of the training dataset, and evaluated an NN 10 times using different seeds on test data. This strong random selection was evaluated five times despite the fact that our proposed method was only evaluated once. The same model, optimizer, and epoch settings were used to make it as fair a comparison as possible with our proposed method.

4. Results

The results are listed in Table 2. We referred to the results of DC and MTT from the literature but used them only as references since some of the terms are not equal. The results of DC and MTT show some differences in the model and preprocessing of the data,[9] so they could not be compared with our method. We changed the zero padding from 1 to 3 of the first convolution layer to match the output feature, but DC and MTT resized the input data to 32 × 32. For the data, we only used normalization, but DC and MTT used ZCA whitening. Our method has higher accuracy on all datasets, except PathMNIST, than does strong random selection. Strong random selection is the main baseline in which the main difference is whether or not a GA that updates 1000 generations is used. The evaluation procedure is the same. There was a 23% increase in accuracy on BloodMNIST and a 14% increase in accuracy on DermaMNIST compared with strong random selection. Our method had higher accuracy on DermaMNIST, OCTMNIST, BloodMNIST, OrganAMNIST, and

Table 2
Evaluation results of our method on MNIST and eight medical datasets. The numbers are reported as *mean* ± *std*.

Dataset	Random (%)	DC (%)	MTT (%)	HoP-TM (%)	Ours (%)
MNIST	68.65 ± 2.02	**91.7 ± 0.5**[6]			91.23 ± 1.26
PathMNIST	39.45 ± 1.16	24.98 ± 2.74[9]	13.40 ± 0.56[9]	**47.71±1.22**[10]	35.65 ± 1.02
DermaMNIST	52.02 ± 2.26	28.22 ± 2.12[9]	25.52 ± 1.75[9]		**66.43 ± 1.09**
OCTMNIST	29.18 ± 1.88	29.92 ± 0.99[9]	25.40 ± 1.57[9]		**35.75 ± 3.29**
BloodMNIST	42.20 ± 2.57	62.46 ± 2.03[9]	60.50 ± 3.01[9]		**65.58 ± 3.39**
TissueMNIST	29.17 ± 1.00	33.83 ± 1.91[9]	13.6 ± 1.01[9]		**36.03 ± 0.22**
OrganAMNIST	42.95 ± 3.03	48.44 ± 0.61[9]	44.04 ± 0.53[9]		**51.82 ± 1.45**
OrganCMNIST	41.34 ± 2.74	50.04 ± 1.54[9]	**67.29 ± 1.10**[9]		59.02 ± 1.47
OrganSMNIST	25.43 ± 1.43	32.89 ± 1.83[9]	31.17 ± 0.68[9]		**38.17 ± 1.27**

OrganSMNIST. We referred also to the result of HoP-TM[10] since it achieved high accuracy on PathMNIST, but only as a reference since some of the terms are not equal.

We tested the difference in validation accuracy when we changed the population as 100, 200, and 300 on DermaMNIST. The results are shown in Fig. 2. We can see that population 300 has the highest validation accuracy compared with population = 100 and 200. Our implementation evaluates each individual one by one, so increasing the population will increase the computational cost of our method.

We compared the accuracy difference when adding Laplace crossover on MNIST. The results are described in Fig. 3. Our proposed method has a higher validation accuracy than the original DL-EA method.

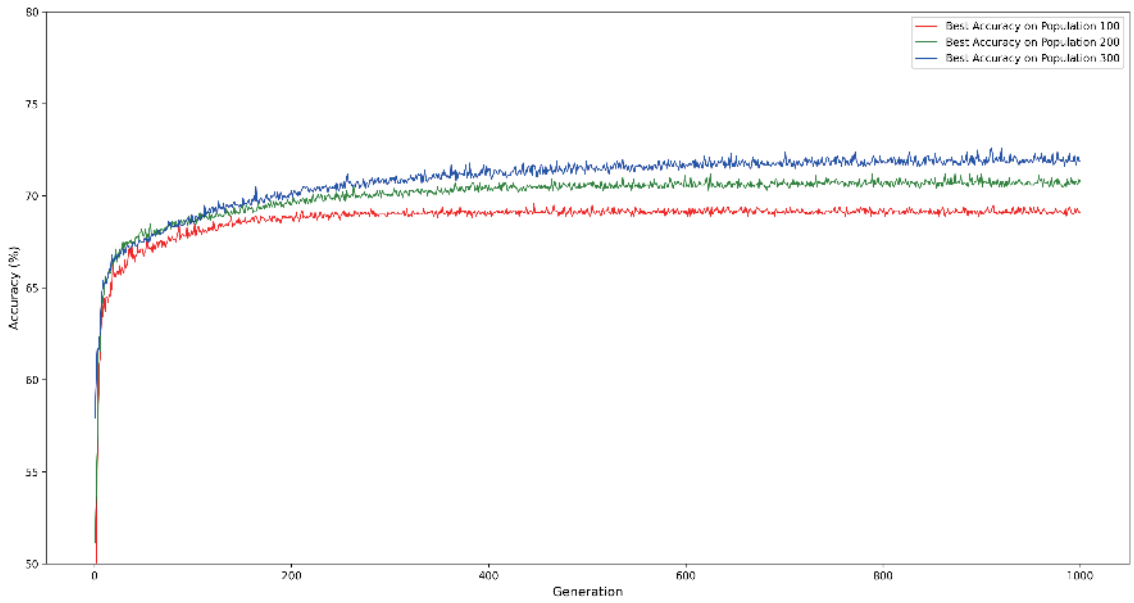

Fig. 2. (Color online) Evolution of highest accuracy for population = 100, 200, and 300.

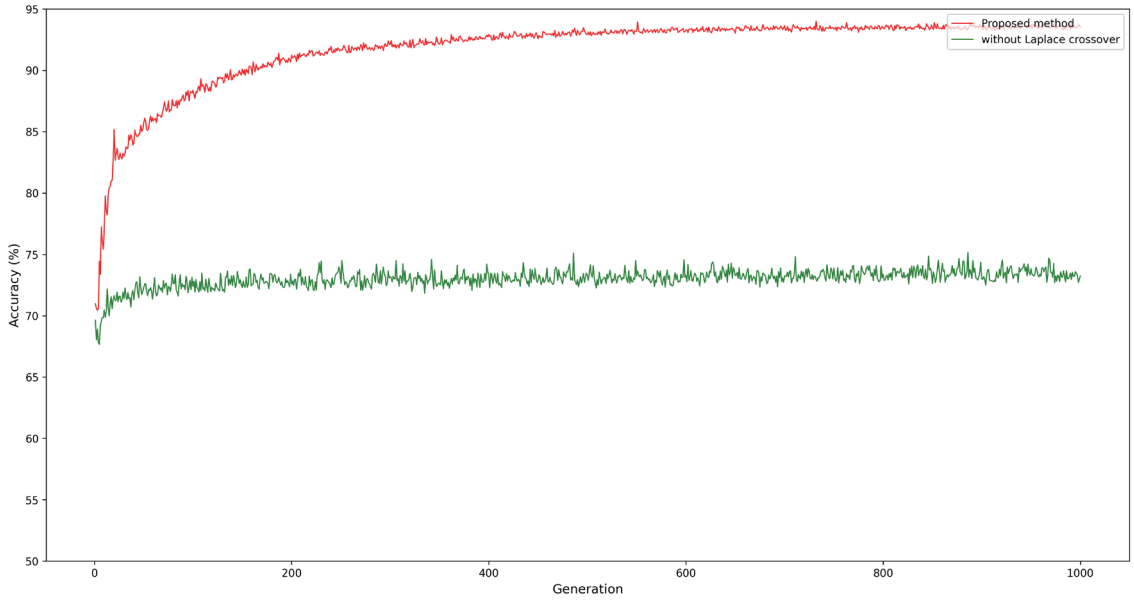

Fig. 3. (Color online) Validation accuracies for DL-EA and proposed method.

The generated synthetic images for MNIST are shown along the bottom row of Fig. 4. The upper row shows images of each class from the training dataset. We can clearly see the numbers in the generated synthetic images for MNIST. Figure 5 shows the generated synthetic images for PathMNIST. It seems that some features of the images were reproduced, but high accuracy was not attained. Figure 6 shows the generated synthetic images for OCTMNIST. Comparing the synthetic images and the training dataset reveals a failure to obtain the features of the training dataset and low accuracy despite the fact that there were four class. We achieved an accuracy of about 29% when we chose the answer randomly. Figure 7 shows the generated synthetic images for BloodMNIST. A comparison of the synthetic images and the training dataset indicates that some features of the training dataset and cells are captured in the synthetic images. In the generated synthetic images for TissueMNIST, OrganAMNIST, OrganCMNIST, and OrganSMNIST shown in Figs. 8–11, respectively, some features of the training dataset have been reproduced.

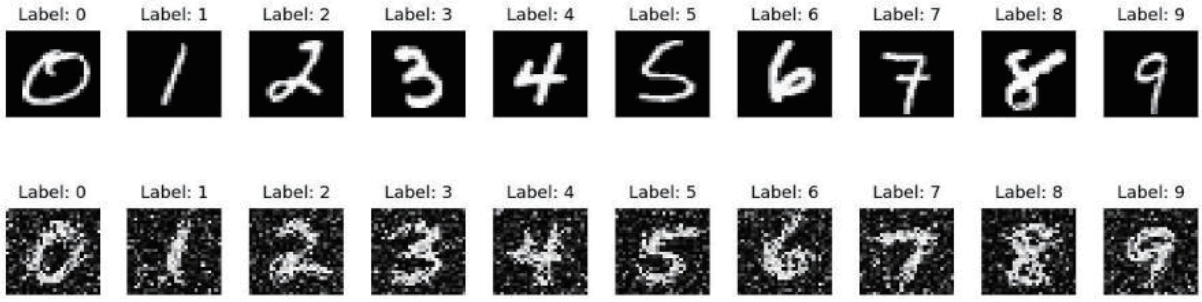

Fig. 4. (Color online) (below) Synthetic images for MNIST. (upper) Images from training dataset for comparison.

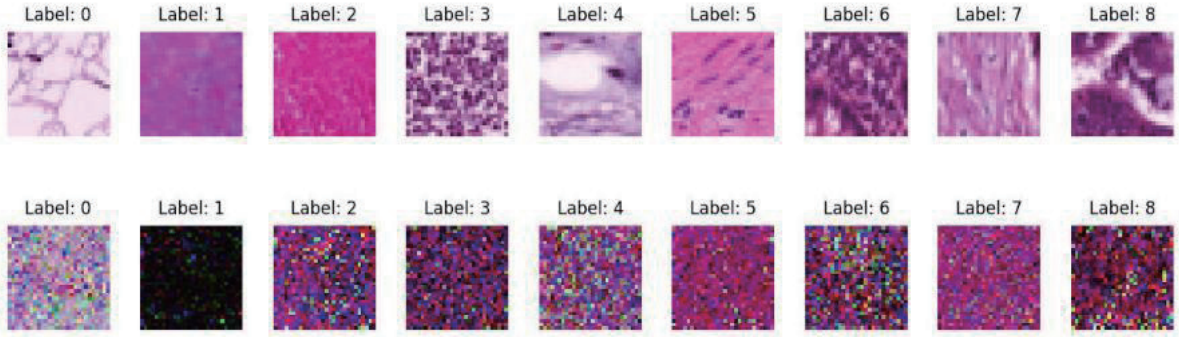

Fig. 5. (below) Synthetic images for PathMNIST. Training images were used as initial images and later modified by our proposed method. (upper) Images from training dataset for comparison. License: CC BY 4.0

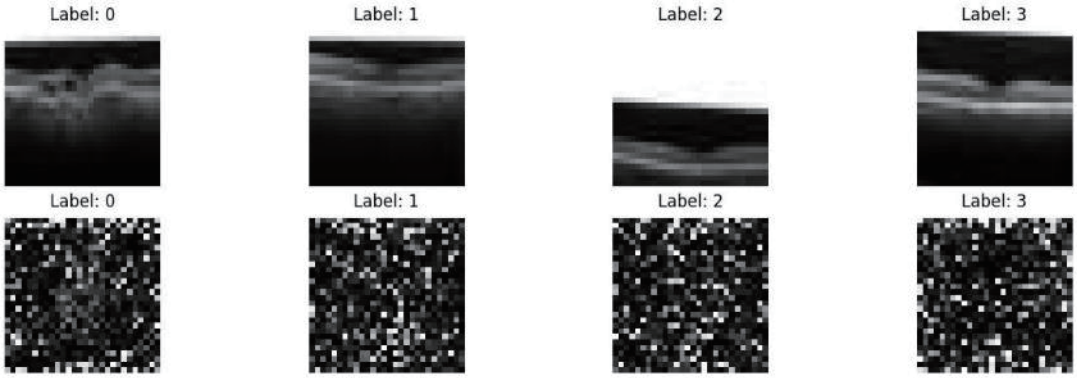

Fig. 6. (below) Synthetic images for OCTMNIST. Training images were used as initial images and later modified by our proposed method. (upper) Images from training dataset for comparison. License: CC BY 4.0

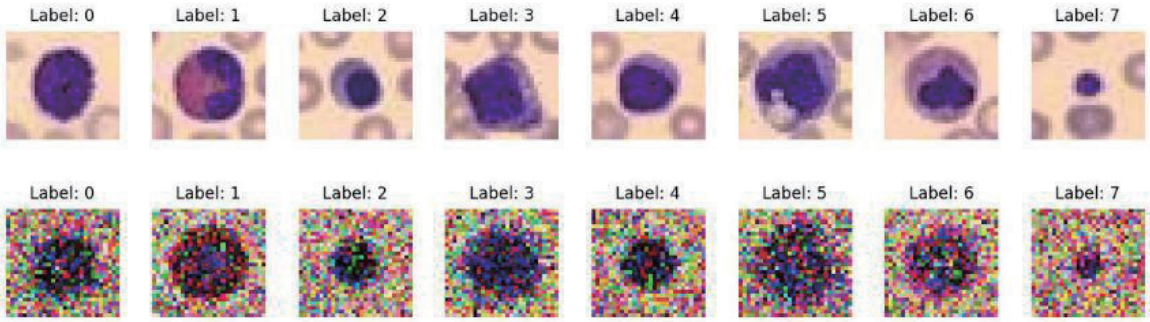

Fig. 7. (Color online) (below) Synthetic images for BloodMNIST. Training images were used as initial images and later modified by our proposed method. (upper) Images from training dataset for comparison. License: CC BY 4.0

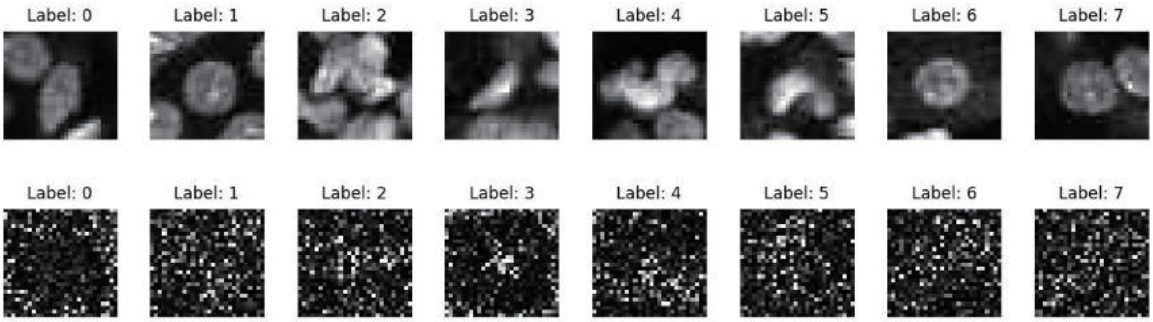

Fig. 8. (below) Synthetic images for TissueMNIST. Training images were used as initial images and later modified by our proposed method. (upper) Images from training dataset for comparison. License: CC BY 4.0

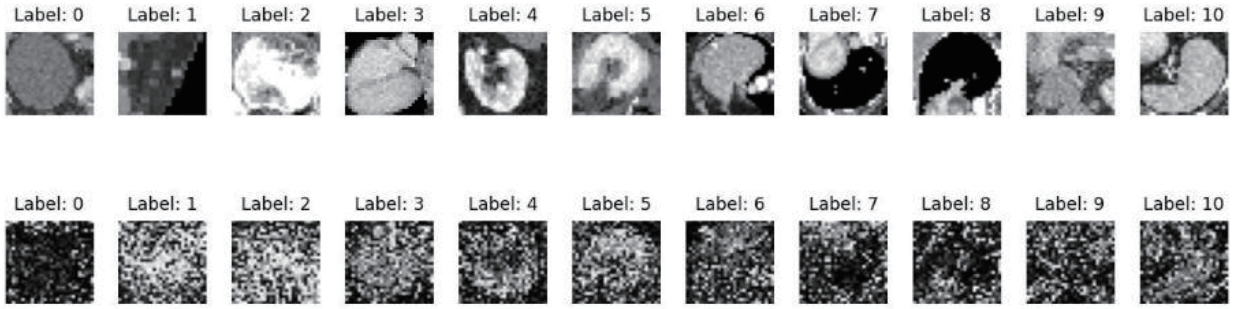

Fig. 9. (below) Synthetic images for OrganAMNIST. Training images were used as initial images and later modified by our proposed method. (upper) Images from training dataset for comparison. License: CC BY 4.0

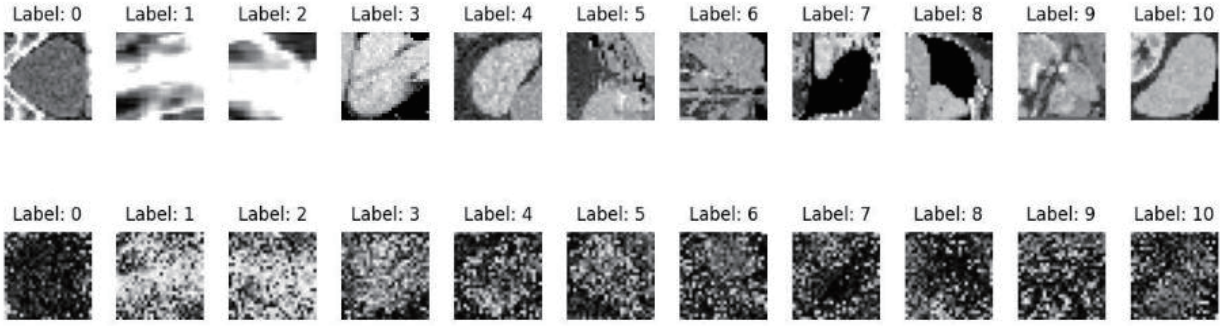

Fig. 10. (below) Synthetic images for OrganCMNIST. Training images were used as initial images and later modified by our proposed method. (upper) Images from training dataset for comparison. License: CC BY 4.0

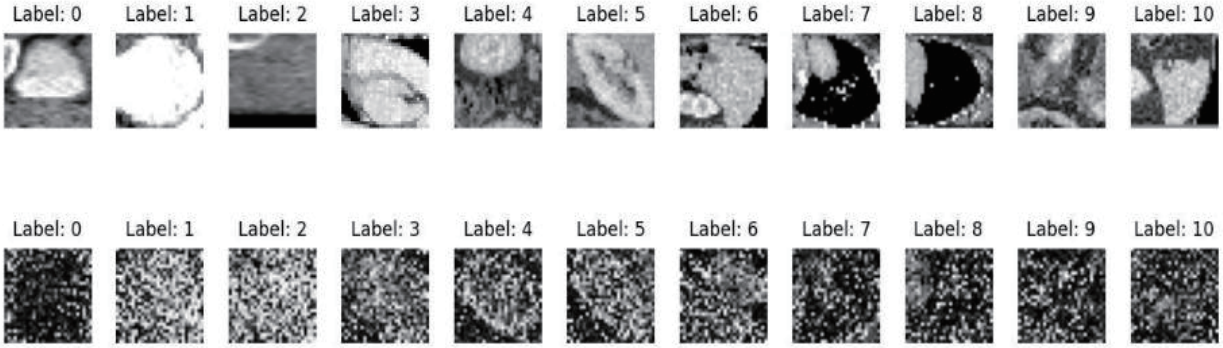

Fig. 11. (below) Synthetic images for OrganSMNIST. Training images were used as initial images and later modified by our proposed method. (upper) Images from training dataset for comparison. License: CC BY 4.0

We compared the validation and test accuracies of each dataset. The results are shown in Table 3. PathMNIST, DermaMNIST, BloodMNIST, and TissueMNIST have an max accuracy decrease of 8%, but OCTMNIST, OrganAMNIST, OrganCMNIST, and OrganSMNIST seem to overfit the validation dataset yet have a higher accuracy than strong random selection.

We also tested how much the accuracy will change when our method is evaluated multiple times. The results are shown in Table 4 . Looking at Table 4, we can see that there is not a huge difference in accuracy, only 1.3%.

5. Discussion

We proposed a dataset distillation method that uses a genetic algorithm and achieved comparable accuracy on MNIST and some datasets included in MedMNIST. There is some possibility that our method is valid on small datasets since it achieved high accuracy on DermaMNIST. Further discussion about this hypothesis requires some additional experiments, which is left as future work.

Our method exhibits some overfit on the validation dataset in OCTMNIST, OrganAMNIST, OrganCMNIST, and OrganSMNIST. We believe that some method to reduce overfitting, such as early stopping, is needed. Early stopping can be easily included by adding validation data to detect overfitting. We could not determine why our method did not attain a higher accuracy than strong random selection on PathMNIST.

Table 3
Validation and test accuracies of MNIST and some datasets included in MedMNIST. The numbers are reported as mean±std.

Dataset	Val accuracy (%)	Test accuracy (%)
MNIST	94.08	91.23 ± 1.26
PathMNIST	37.16	35.65 ± 1.02
DermaMNIST	72.58	66.43 ± 1.09
OCTMNIST	64.00	35.75 ± 3.29
BloodMNIST	73.25	65.58 ± 3.39
TissueMNIST	36.35	36.03 ± 0.22
OrganAMNIST	94.62	51.82 ± 1.45
OrganCMNIST	95.53	59.02 ± 1.47
OrganSMNIST	85.81	38.17 ± 1.27

Table 4
Evaluation results of running our method multiple times on DermaMNIST.
The numbers are reported as *mean* ± *std*.

DermaMNIST	Accuracy (%)
1	66.10 ± 0.83
2	67.41 ± 0.65
3	67.13 ± 0.73
4	66.57 ± 0.92
5	66.12 ± 0.69

The computation time of our method is determined by the dataset size of the subset of the training dataset, so DermaMNIST takes about two days, whereas OCTMNIST takes about four days since it has many more images than DermaMNIST. Since the computation time of our algorithm is mostly for the calculation of the fitness function, further improvement, such as faster convergence and more efficient fitness calculation or method selection of the training dataset subset, is a task in future work.

6. Conclusions

We proposed a dataset distillation method that uses a genetic algorithm and has only a few related methods. Previous methods have not been evaluated on MedMNIST and tend to reduce the population or set a small population. Our method reduces the size of datasets by using a subset of the training dataset. Our method is based on DL-EA and uses Laplace crossover as the crossover method of the genetic algorithm.

We were able to successfully achieve a higher accuracy on MNIST, DermaMNIST, OCTMNIST, BloodMNIST, TissueMNIST, OrganAMNIST, OrganCMNIST, and OrganSMNIST than the strong random selection method, but accuracy was low on PathMNIST. There are some datasets, such as OCTMNIST, OrganAMNIST, OrganCMNIST, and OrganSMNIST, which exhibited overfitting on the validation dataset. Our method takes a few days to generate the synthetic dataset. Future improvement is needed on these points.

Acknowledgments

This research was supported in part by JST CREST Grant Number #JPMJCR22M4, Japan.

References

1 T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros: arXiv preprint arXiv:1811.10959 (2018).
2 A. Krizhevsky: Learning Multiple Layers of Features from Tiny Images (2009).
3 O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei: Int. J. Comput. Vision **115** (2015) 211.
4 J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, and B. Ni: Scientific Data **10** (2023) 41.
5 P. Helber, B. Bischke, A. R. Dengel, and D. Borth: IEEE J. Selected Topics in Applied Earth Observations and Remote Sensing **12** (2017) 2217.
6 B. Zhao, K. R. Mopuri, and H. Bilen: 9th Int. Conf. Learning Representations 2021 (2021).
7 G. Cazenavette, T. Wang, A. Torralba, A. A Efros, and J. Y Zhu: Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (2022) 4750–4759.
8 Z. Yin, E. Xing, and Z. Shen: Adv. Neural Inf. Proc. Syst. **36** (2023) 73582.
9 M. Li, C. Cui, Q. Liu, R. Deng, T. Yao, M. Lionts, and Y. Huo: Med. Imaging 2025: Image Perception, Observer Performance, and Technology Assessment, SPIE Vol. 13409 (2025) 172–179.
10 L. Dong, J. Bian, J. Hou, J. Hu, Y. Shi, W. Dong, X. X. Zhu, and L. Mou: Int. Conf. Medical Image Computing and Computer-Assisted Intervention (2025) 273–283.
11 R.-M. Ungureanu: 25th Int. Symp. Symbolic and Numeric Algorithms for Scientific Computing (SYNASC) IEEE Computer Society (2023).
12 N. Shvai, A. Llanza, A. Hasnat, and A. Nakib: Proc. Genetic and Evolutionary Computation Conf. Companion (2022) 292–295.
13 T. T. Inan and A. Shehu: Proc. Genetic and Evolutionary Computation Conference Companion (2024) 435–438.
14 K. Deep and Manoj Thakur: Appl. Math. Comp. **188** (2007) 895.

15 G. Hinton, O. Vinyals, and J. Dean: Deep Learning and Representation Learning Workshop in Conjunction with NIPS (2014).
16 D. Liu, J. Gu, H. Cao, C. Trinitis, and M. Schulz: Eur. Conf. Computer Vision (2024) 334–351.
17 Y. Feng, S. R. Vedantam, and J. Kempe: 12th Int. Conf. Learning Representations.
18 P. Sun, B. Shi, D. Yu, and T. Lin: Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (2024) 9390–9399.
19 Y. Xu, S. Zhao, J. Song, R. Stewart, and S. Ermon: Int. Conf. Learning Representations.
20 J. Gu, S. Vahidian, V. Kungurtsev, H. Wang, W. Jiang, Y. You, and Y. Chen: Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (2024) 15793–15803.
21 J. A. Chan-Santiago, P. Tirupattur, G. K. Nayak, G. Liu, and M. Shah: arXiv preprint arXiv:2505.18963 (2025).
22 X. Shang, P. Sun, and T. Lin: 13th Int. Conf. Learning Representations.
23 A. Shul, E. Horwitz, and Y. Hoshen: Trans. Machine Learning Research.
24 G. Li, R. Togo, T. Ogawa, and M. Haseyama: arXiv preprint arXiv:2209.14603 (2022).
25 P. Barbiero, G. Squillero, and A. Tonda: arXiv preprint arXiv:2002.08645 (2020).
26 P. Okanovic, R. Waleffe, V. Mageirakos, K. Nikolakakis, A. Karbasi, D. Kalogerias, N. Merve Gürel, and T. Rekatsinas: Int. Conf. Learning Representations (2024) 24058–24098.
27 G. Morse and K. O. Stanley: Proc. Genetic and Evolutionary Computation Conference (2016) 477–484.
28 K. Deb and R. B. Agrawal: Complex Syst. **9** (1995) 115.
29 T. Higuchi: J. Jpn. Soc. Artif. Intell. **16** (2001) 147.
30 K. Deb, D. Joshi, and A. Anand: Proc. 2002 Congr. Evolutionary Computation. CEC'02 (IEEE, Cat. No. 02TH8600) Vol. 1 (2002) 61–66.
31 A. K. Das and D. K. Pratihar: Recent Advances in Theoretical, Applied, Computational and Experimental Mechanics: Proc. ICTACEM 2017 (2020) 311–323.
32 S. Kobayashi: Trans. Jpn. Soc. Artif. Intell. **24** (2009) 147.
33 S. Gidaris and N. Komodakis: 2018 IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR, IEEE Computer Society) (2018) 4367–4375.
34 Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner: Proc. IEEE **86** (1998).
35 A. Tonda: Genet. Program. Evolvable Mach. **21** (2020) 269.
36 J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. K. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, S. Zhang, M. Suo, P. Tillet, X. Zhao, E. Wang, K. Zhou, R. Zou, X. Wang, A. Mathews, W. Wen, G. Chanan, P. Wu, and S. Chintala: Proc. 29th ACM Int. Conf. Architectural Support for Programming Languages and Operating Systems, Volume 2 (2024) 929–947. https://doi.org/10.1145/3620665.3640366

About the Authors

Takumi Sato received his B.S. and M.S. degrees from Meijo University, Japan, in 2002 and 2005, respectively. In 2021, he started work as an engineer at Kadinche Corporation, Japan. Since 2022, he has been a PhD student at Hokkaido University, Japan. His research interests are in dataset distillation and evolutionary algorithms. (sato.takumi.g0@elms.hokudai.ac.jp)

Yasumasa Tamura received his Ph.D. degree from Hokkaido University, Japan, in 2015. From 2016 to 2017, he was a research fellow of the Japan Society for the Promotion of Science and a visiting researcher in Université Libre de Bruxelles, Belgium. From 2017 to 2023, he was an assistant professor in Tokyo Institute of Technology. From 2024 to 2025, he was an assistant professor in Hokkaido University. Since 2026, he has been an associate professor in Hokkaido University. His research interests include computational intelligence, swarm intelligence, combinatorial optimization, multi-agent systems, swarm robotics, and distributed systems.
(ytamura@ist.hokudai.ac.jp)

Masahito Yamamoto received his PhD degree in the Graduate School of Engineering from Hokkaido University, Japan, in 1996. From 1996 to 1997, he was a research fellow of the Japan Society for the Promotion of Science. He has been an assistant professor (1997–2000) and associate professor (2000–2012) in Hokkaido University. Since 2012, he has been a professor at the autonomous systems engineering laboratory, Hokkaido University, Japan. He has also been a concurrent faculty member of the Center for Human Nature, Artificial Intelligence, and Neuroscience, Hokkaido University, since 2020. His research interests include artificial life and intelligence, swarm intelligence, combinatorial optimization, and game and sports AI programming. (masahito@ist.hokudai.ac.jp)