Evolutionary Approach for AutoAugment Using the Thermodynamical Genetic Algorithm

Akira Terauchi, Naoki Mori
Osaka Prefecture University, 1-1 Gakuen-cho, Naka-ku, Sakai, Osaka, Japan
terauchi@ss.cs.osakafu-u.ac.jp, mori@cs.osakafu-u.ac.jp

Abstract

Data augmentation is one of the most effective ways to stabilize learning by improving the generalization of machine-learning models. In recent years, automatic data augmentation methods, such as AutoAugment or Fast AutoAugment have been attracting attention; and these methods improved the results of image classification and object detection tasks. However, several problems remain. Most notably, a larger training dataset requires higher computational costs. When searching with a small dataset in an attempt to determine the data augmentation approach, the true data space and sampling data space do not fully correspond with each other, thereby causing the generalization performance to deteriorate. Moreover, in the existing automatic augmentation methods, the search phase is often dominated by an exceptional sub-policy, which results in a loss of diversity of transformations. In this study, we solved these problems by introducing evolutionary computation to previous methods. As mentioned earlier, maintaining diversity of transformations is essential. Therefore, we adopted the thermodynamical genetic algorithm (TDGA), which can control the population diversity with a specific genetic operator, known as the thermodynamical selection rule. To confirm the effectiveness of the proposed method, computational experiments were conducted using two benchmark datasets, CIFAR-10 and SVHN, as examples. The experimental results show that the proposed method can obtain various useful augmentation sub-policies for the problems while reducing the computational cost.

Introduction

Data augmentation methods have long been used to generate additional data and improve the robustness of many computer vision tasks (Shorten and Khoshgoftaar 2019). An appropriate data augmentation method can be used to obtain a more comprehensive set of data points and effectively reduce the distance between the training and validation, as well as the test sets. However, selecting a data augmentation method that efficiently matches the data with tasks requires a high degree of specialized knowledge. It is time consuming and often requires a thorough understanding of the data domains.

The development of automated machine learning (AutoML) in recent years has focused the attention on automatic search methods for augmentation policies of datasets, and has produced state-of-the-art results in various image classification tasks. For example, AutoAugment (Cubuk et al. 2019) was employed to search for an augmentation policy for datasets as a discrete problem using the reinforcement learning (RL) method for the search controller. AutoAugment treats the accuracy of the validation set after learning as a reward. Although AutoAugment achieves higher accuracy than the conventional naive image classification method, the calculation cost is considerably higher in that the method requires thousands of GPU hours and is not realistic to reproduce. In addition, the difference between the search space and application space remains problematic when using a reduced dataset. A policy obtained using a reduced dataset may not fit the original dataset. Population Based Augmentation (PBA) (Ho et al. 2019) accelerates learning using Population Based Training method (Jaderberg et al. 2017) and significantly reduces the computational cost. Fast AutoAugment (Lim et al. 2019) also drastically reduces the computational cost by dividing the dataset, training the child model in parallel, and searching for the augmentation policy without additional model learning. RandAugment (Cubuk et al. 2020) removes the separation of the search and training phases by limiting the hyperparameters to the number and constant magnitude of operations. These methods significantly improve the accuracy of the image classification task (Devries and Taylor 2017; Yamada et al. 2019) using conventional naive methods, and solve the problem of computational cost, which is a major disadvantage of AutoAugment. However, these methods continue to be problematic in that the variety of augmentation transformations obtained by the search phase is not considered, and the search may be dominated by one dominant transformation. In addition, because RandAugment does not include a search phase and samples transformations randomly from the operation set, the design of the operation set cannot be automated and requires human coordination according to the domain and task.

The application of evolutionary computation (EC) to the search controller is expected to optimize the combinatorial problem in AutoAugment. When applying EC to target problems, controlling the diversity of genotypes in the population is essential. However, no research of automatic policy search that considers the diversity of EC directly has been proposed.

	Search method	Search space	CIFAR-10 WRN28-2
AA	RNN	10^{32}	95.9
PBA	PBT	10^{61}	-
Fast AA	Bayes	10^{32}	-
RA	Grid Search	10^{2}	95.8
TDGA AA	TDGA	10^{8}	95.92

Table 1: Summarized comparison of our method with other methods in terms of the search method, search space, and test accuracy of CIFAR-10 on Wide-ResNet-28-2.

In this paper, we use a genetic algorithm (GA) as the EC for the search controller and propose the thermodynamical genetic algorithm based AutoAugment (TDGA AutoAugment), which is designed to solve this lack of variety effectively by considering the diversity in GA. The test accuracy of both Fast AutoAugment and RandAugment was reportedly improved using various operations. Although it has been stated that a GA controller may be used in AutoAugment, because of the problem of the premature convergence of the population in simple GA (SGA) (Goldberg 1989) that has no mechanism to control population diversity, an automatic augmentation method using a GA has not been reported thus far. TDGA (Mori et al. 1995) overcame this problem by applying thermodynamic selection rules to extend the SGA. Using a TDGA controller, various augmentation operations can be selected while maintaining a good fit in the evaluation space.

Our experiments confirmed that various operations were selected and that our method achieved competitive performance and was faster than PBA and Fast AutoAugment. On the CIFAR-10 dataset using Wide-ResNet-28-2, we achieved 95.92% accuracy, which is comparable with that of AutoAugment. A comparison between our method and other methods is summarized in Table 1.

The remainder of this paper is organized as follows. First, we introduce various methods related to automatic data augmentation. We then introduce the TDGA, which is the core of our method. The TDGA AutoAugment method is then described in detail. Finally, we demonstrate the effectiveness of our method in certain benchmark problems compared with the baseline and previous methods.

Related Work

Thus far, many data augmentation methods for image datasets have been reported. On natural benchmark image datasets, such as CIFAR-10 (Krizhevsky 2009) and ImageNet (Deng et al. 2009), geometric transformations, such as random crop, horizontal flips, rotating and translating, and color manipulation transformations, such as color shifting and whitening are commonly used as baseline augmentation methods (Krizhevsky, Sutskever, and Hinton 2012; Zagoruyko and Komodakis 2016; Han, Kim, and Kim 2017). In addition, Cutout (Devries and Taylor 2017) stabilizes the learning process and improves the accuracy by randomly erasing certain areas of images. Mixup (Zhang et al. 2018) linearly complements both labels and data to create new data, enabling data augmentation without domain knowledge, thus, enhancing the robustness of learning. However, choosing these augmentations requires a considerable amount of expert time and knowledge of the data domain.

Against this background, methods that can automatically acquire data augmentation strategies have been developed. Smart Augmentation (Lemley, Bazrafkan, and Corcoran 2017) generates additional data by training the network that generates a new image and the network that identifies it using the same class of samples as input. Smart Augmentation contributes to suppressing over-fitting during training. High accuracy was achieved by generating additional data using generative adversarial networks (GAN) (Ratner et al. 2017). Bayesian DA (Tran et al. 2017) combined the Bayesian approach with GAN and generated annotated training points based on the distribution learned from the training set. The augmented data are treated as missing variables.

Apart from these methods, the rise of AutoML in recent years (Zoph et al. 2017; Real et al. 2018) has resulted in a focus on the use of augmentation to determine the data augmentation policy rather than generating additional data directly. AutoAugment (Cubuk et al. 2019) uses Recurrent Neural Network (RNN) as a controller to search for operation types, probabilities, and intensities, and generates a policy consisting of multiple sub-policies. The augmentation policy that was obtained greatly improved the accuracy of the conventional methods. In addition, AutoAugment showed that the obtained policy can be used for various datasets. However, because AutoAugment trains a child model from scratch, it is very time-consuming to train. The computational cost of PBA (Ho et al. 2019) was reduced with the PBT algorithm, and the accuracy was reported to have been improved by optimizing the schedule of the augmentation. Fast AutoAugment (Lim et al. 2019) accelerates policy search by Bayesian search. This was accomplished by parallelizing learning and not performing additional training for the child models. In RandAugment (Cubuk et al. 2020), the hyperparameters were limited to the number of operations N and the magnitude M; the accuracy was improved by applying the augmentation policy directly during training. In addition, for RandAugment, it was reported that, even if the magnitude of all operations were to be fixed with the operation magnitude scaled within a certain range, the accuracy would not be significantly affected. The development of TDGA AutoAugment was greatly inspired by these developments; we have used some of these techniques in our paper.

Thermodynamical Genetic Algorithm

A GA is an optimization method based on the evolution process of a system (Goldberg 1989; Holland 1992). It has been used to solve the discrete optimization problem because it does not require prior knowledge of rewards or differential gradients; however, GA is adversely affected by a phenomenon known as premature convergence. In other words, population diversity is often lost in the early stages of searching. TDGA (Mori et al. 1995) solves this premature convergence problem by adding the concept of temperature

and entropy to the selection rule. This was inspired by the method of simulated annealing (SA) (Kirkpatrick, Gelatt, and Vecchi 1983) and the principle of minimum free energy in thermodynamics with the aim of maintaining and controlling the diversity of the population.

Simulated Annealing and the Principle of Minimal Free Energy

Let us consider the following optimization problem:

$$\min_{\boldsymbol{x}} E(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathcal{F}, \tag{1}$$

where the set of feasible solutions $\mathcal{F}$ is assumed to be finite. Hereafter, the objective function E is called the energy function.

SA obtains a new state $\boldsymbol{x}'$ to perturb an old state $\boldsymbol{x}$. If the energy of the new state $E(\boldsymbol{x}')$ is less than the energy of the old state $E(\boldsymbol{x})$, the state translates $\boldsymbol{x}'$ with high probability. If $\boldsymbol{x}'$ is larger than $\boldsymbol{x}$, the transition occurs with a low probability using the parameter T (the temperature). SA uses this approach to find the minimum state, that is, state with the lowest energy.

When T is a fixed constant, the distribution of the Metropolis method, which is a typical transition rule of SA, becomes a Gibbs distribution. Further, it is also known that this distribution minimizes the free energy F defined by:

$$F = \langle E \rangle - HT, \tag{2}$$

where $\langle E \rangle$ is the mean energy of the system and H is the entropy. This is known as the principle of minimum free energy. TDGA regards the first term of the free energy in Eq.(2) as the energy minimization term; the second term is responsible for maintaining the diversity. These two terms are adjusted by the temperature parameter T.

Calculation of the Entropy and Minimization of the Free Energy in TDGA

At first glance, the entropy is estimated by

$$H^{\mathrm{ALL}} = -\sum_i p_i \log p_i, \tag{3}$$

where p_i is the ratio of genotype i that appears in the population. If the size of the population is sufficiently large, H provides a good estimate of entropy. However, in the GA, the population size is extremely small compared to the number of possible states $|\mathcal{F}|$.

Hence, TDGA alternatively calculates the population entropy H^1 from each allele as follows:

$$H^1 = \sum_{k=1}^{M} H_k^1, \quad H_k^1 = -\sum_{j \in \{0,1\}} P_j^k \log P_j^k, \tag{4}$$

where H_k^1 is the entropy of the locus k, and P_j^k is the existence probability of gene j on locus k. In TDGA, H^1 is regarded as H in the second term of Eq.(2). (Mori et al. 1995) proved that H^{ALL} in Eq.(3) and H^1 in Eq.(4) would be equal under the condition of that the gene of each locus takes an independent value. The basic concept of TDGA is to create a population that minimizes the free energy. However, it is difficult to minimize free energy exactly; TDGA uses an approximate greedy method, which adds the individual making free energy minimum in the temporary generation into the next generation one after another. The pseudocode of the TDGA algorithm is provided in Algorithm 1. TDGA adds a tentative individual to the next population $\mathcal{P}(t+1)$ and calculates the mean energy $\langle E_{\mathcal{P}(t+1,i,h)} \rangle$ and the entropy $H_{\mathcal{P}(t+1,i,h)}$ every time entering the inner loop.

Algorithm 1 TDGA algorithm

Require: (N_{p}: the population size, N_{g}: the number of generations, $\mathcal{T}(t)$: the temperature schedule)
Ensure: The final population $\mathcal{P}(N_{\mathrm{g}})$
  $t \Leftarrow 0$
  Set the initial population $\mathcal{P}(0)$ randomly
  while $t < N_{\mathrm{g}}$ **do**
    $T \Leftarrow \mathcal{T}(t)$
    $e \Leftarrow$ Elite individual having the minimum energy (best solution) from $\mathcal{P}(t)$
    Pair all the individuals and obtain N_{p} offspring $\mathcal{P}_{\mathrm{o}}(t)$
    $\mathcal{P}'(t) \Leftarrow \mathcal{P}(t) \cup \mathcal{P}_{\mathrm{o}}(t)$
    Apply the mutation operator to $\mathcal{P}'(t)$
    $\mathcal{P}'(t) \Leftarrow \mathcal{P}'(t) \cup \{e\}$
    $i \Leftarrow 1$
    $\mathcal{P}(t+1) \Leftarrow \phi$ // next population
    while $i \leq N_{\mathrm{p}}$ **do**
      $\mathcal{P}(t+1, i, h) \Leftarrow \mathcal{P}(t+1) \cup \{h\}$, where h denotes the hth individual in $\mathcal{P}'(t)$.
      // select the individual that minimizes the free energy
      $F_h = \langle E_{\mathcal{P}(t+1,i,h)} \rangle - H_{\mathcal{P}(t+1,i,h)} T$
      $h_{\min} \Leftarrow \arg\min_h F_h$
      $\mathcal{P}(t+1) \Leftarrow \mathcal{P}(t+1) \cup \{h_{\min}\}$
      $i \Leftarrow i + 1$
    end while
    $t \Leftarrow t + 1$
  end while

TDGA AutoAugment

In this section, we introduce the proposed TDGA AutoAugment algorithm, which searches for the augmentation policy based on the evolutionary approach. The main purpose of TDGA AutoAugment is to maintain the diversity of transformations while searching. This approach was inspired by the report that the accuracy of Fast AutoAugment and RandAugment can be expected to be improved by adopting various transformations (Lim et al. 2019; Cubuk et al. 2020). Using TDGA AutoAugment, we extract the appropriate set of operations that fit the problem from the transformation candidates. Figure 1 shows an overview of TDGA AutoAugment.

Search Space

In TDGA AutoAugment, we consider policy search as a discrete combinatorial optimization problem. Each operation is

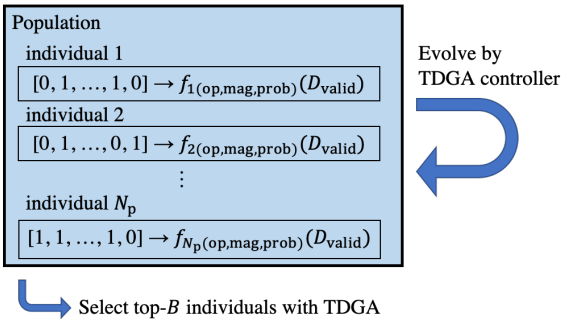

Figure 1: Overview of TDGA AutoAugment. Each individual corresponds to one sub-policy, which is evaluated by the pre-trained model on D_{train}. Its accuracy is considered as the fitness. The TDGA algorithm controls evolution to leave diverse transformations in the next generation.

associated with two possible actions: the operation type is either used or not used. Therefore, the operation set has 2^{16} possibilities. In addition, each operation shares the magnitude M and probability p among all operations. This method of using a constant magnitude was adopted in RandAugment. M takes a value from 0 to 30. All transformations represented by each gene is applied with probability $p = m/L$, where L is chromosome length and m is a positive integer of 1 to 5. Finally, we add a new hyperparameter here in the form of the temperature parameter T, which is used in TDGA AutoAugment. The temperature parameter is a hyperparameter related to the degree to which the diversity of operations is maintained. We used 10 decimal values, which were sampled from [0, 0.1] in our experiments; the value is fixed during the search. This means that an identity function is used as the temperature scheduler $\mathcal{T}(t)$ in the TDGA algorithm. Therefore, the search space of TDGA AutoAugment has approximately $(31 \times 5 \times 10 \times 2^{16}) \approx 1.0 \times 10^8$ possibilities, which is much smaller than 10^{32}, the search space for AutoAugment and Fast AutoAugment.

Implementation

TDGA AutoAugment consists of two simple processes: (1) pre-training the model and (2) searching with the TDGA controller. In TDGA AutoAugment, the training sample is divided into a training dataset D_{train} and an evaluation dataset D_{valid} by stratified shuffling (Shahrokh Esfahani and Dougherty 2013). The accuracy of the transformed D_{valid} for the model trained on D_{train} is regarded as the fitness of the individual and its negative value is regarded as the energy of TDGA free energy. This method is used in Fast AutoAugment to reduce the computational cost by evaluating a sub-policy without additional training. The locus of the individual corresponds to each transform, and each individual represents a sub-policy. When converting from the individual to the sub-policy, the corresponding transformations are applied sequentially with probability p. At the final training phase after evolution, each image is transformed with a randomly chosen sub-policy from B sub-policies. Because TDGA AutoAugment does not use a reduced dataset, the problem of discrepancies between the true data space and sampling space does not arise. Therefore, we can expect more accurate data regularization. The overall procedure is summarized in Algorithm 2.

Algorithm 2 TDGA AutoAugment algorithm. Individuals correspond to sub-policies, and the fitness corresponds to the accuracy of the transformed D_{valid} in the model trained on D_{train}.

Require: ($\theta, D_{\mathrm{train}}, D_{\mathrm{valid}}, N_{\mathrm{p}}, N_{\mathrm{g}}, T$, M: operation magnitude, B: the number of final sub-policies, m: multiplier of probability)
Ensure: The obtained policy $\mathcal{T}_*$
Train θ on D_{train}
$\mathcal{P} \Leftarrow \mathrm{TDGASearch}(\theta, D_{\mathrm{valid}}, N_{\mathrm{p}}, N_{\mathrm{g}}, T, M, m)$
$\mathcal{T}_* \Leftarrow$ TDGA selects top-B sub-policies in $\mathcal{P}$

Experiments and Results

To investigate the performance of TDGA AutoAugment, we applied the method to CIFAR-10 and SVHN (Netzer et al. 2011) datasets and compared the results with those of the baseline and previous methods. Pre-training was conducted for 200 epochs. Referring to the previous methods, we searched over 16 operations (AutoContrast, Equalize, Invert, Rotate, Posterize, Solarize, SolarizeAdd, Color, Contrast, Brightness, Sharpness, ShearX/Y, Cutout, TranslateX/Y). In the GA, the final generation was set to 30 ($N_{\mathrm{g}} = 30$), the mutation rate of each locus was set to 0.06, and uniform crossover was used based on (Mori et al. 1995). In the search process, 90% of the training data were used for D_{train}, and the remaining 10% were used for D_{valid}. The final policy with 16 sub-policies ($B = 16$) out of 64 sub-policies ($N_{\mathrm{p}} = 64$) was used to train the models for each dataset. The magnitude range setting of each operation was from AutoAugment as much as possible to allow a fair comparison to be made.

CIFAR-10 Results

CIFAR-10 is a benchmark dataset for image classification that has been studied for a long time, and includes 50,000 training images and 10,000 test images. Pad-and-crop, RandomHorizontalFlip, and Cutout were used as default augmentations. For the models, we used a learning rate of 0.1, weight decay of 5e-4, batch size of 128, and cosine learning rate decay for 200 epochs. We set $m = 2$ for the experiments. Table 2 lists the test accuracies.

First, we discuss the dependence of TDGA AutoAugment on the temperature parameter T. In Figure 2, we show the valid/test accuracies for various temperature settings. The horizontal axis represents the temperature and the vertical axis represents accuracy. The validation accuracy and test accuracy are almost parallel. For Wide-ResNet-28-2 and Wide-ResNet-28-10, the best settings were $M = 5$, $T = 0.02$ and $M = 7$, $T = 0.01$. The results show that selecting the appropriate temperature T improves the accuracy. We attained a Top-1 accuracy of 95.92%, which is comparable with AutoAugment and 0.12% higher than that of Ran-

Dataset	Model	Baseline	AA	PBA	Fast AA	RA	TDGA AA
CIFAR-10	Wide-ResNet-28-2	94.9	95.9	-	-	95.8	95.92±0.05
	Wide-ResNet-28-10	96.1	97.4	97.4	97.3	97.3	97.25±0.05
SVHN (core)	Wide-ResNet-28-10	96.9	98.1	-	-	98.3	97.95±0.03

Table 2: Test accuracy (%) on CIFAR-10 and SVHN core set. Comparisons across default data augmentation (baseline), AutoAugment (AA), Population Based Augmentation (PBA), Fast AutoAugment (Fast AA), RandAugment (RA), and proposed TDGA AutoAugment (TDGA AA). For accuracy values other than TDGA AutoAugment, the values reported in their respective papers are used. We used the models: Wide-ResNet-28-2 and Wide-ResNet-28-10 (Zagoruyko and Komodakis 2016). Five independent run were performed for Wide-ResNet-28-2, and three independent runs for Wide-ResNet-28-10.

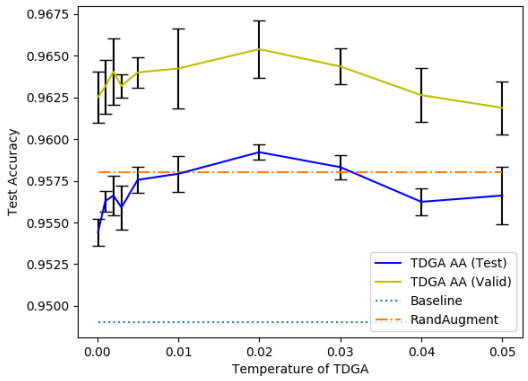

Figure 2: Valid/test accuracy of Wide-ResNet-28-2 trained on CIFAR-10 using various temperatures. We performed five independent runs at each temperature. The bars represent the standard deviation values for each trial.

dAugment. Furthermore, as the temperature increases, more diverse transformations are adopted, and their behavior approaches that of RandAugment. This is because when the temperature rises, there is a tendency to use random transformations regardless of the fitness value. Conversely, as the temperature decreases, the diversity is lost, and the accuracy approaches the baseline of the Cutout model. This indicates that when the temperature is excessively low, the B sub-policies are all similar. Moreover, the dominant sub-policy tends to resemble an identity function that does not change the original image. As shown in Algorithm 1, TDGA always selects the best individual in the current population and places it in the next population regardless of the temperature. This is the paramount reason why the proposed method has a low dependence on temperature.

Figure 3, 4 show the variation in the entropy of each locus during the search process using two temperature settings: T = $\{0.02, 0.002\}$. The x-axis, y-axis, and z-axis represent the generation, the locus, and the entropy value, respectively. The high entropy value means that the diversity of the locus is preserved in that generation. If a transformation is always used because of its effectiveness, the entropy of the locus of that transformation may become low. Besides, if a transformation is not used because of its uselessness, the entropy of the locus also may become low. Conversely, a transformation corresponding to a high entropy locus is difficult to decide its use or not. Therefore, the optimization for such transformations is very important in improving the accuracy. The entropy level shows the essential transformation in RandAugment. Similar to Fast AutoAugment, TDGA AutoAugment uses a method that searches for sub-policies that fit the evaluation data without additional training. This results in a situation in which the default sub-policy dominates the search, and some operations making a big conversion to the image are rarely adopted. Figure 3 shows that the TDGA AutoAugment retains these transformations in a search with appropriate temperature settings whereas the diversity of certain transformations is lost when the temperature T is small (Figure 4). This result shows that the accuracy increases when diverse augmentation operations are used in the final generation.

Figure 5 shows the final entropy values and the numbers of operations when $T = 0.02$. The horizontal axis represents the type of operation. The left vertical axis represents the entropy value of the final generation, and the right vertical axis represents the number of operations. In our experiment, the operation that has a low entropy value tended to be rarely adopted in the final training phase. This indicates that TDGA focused more on maintaining diversity rather than always adopting valid transformations. Here we quote the results shown in Table 5 of the RandAugment paper. TDGA AutoAugment actively adopted operations with positive contributions, such as Rotate, Shear, and Translate. In addition, our method succeeded in efficiently removing the operations with negative contributions, such as Solarize and Posterize.

Next, we investigated the effect of changing the number of sub-policies B on accuracy. We set $N_{\mathrm{p}} = 64$, $M = 5$, $m = 2$, and $T = 0.02$. Figure 6 shows that the test accuracy improves as the number of sub-policies increases. The horizontal axis represents the number of sub-policies and the vertical axis represents the test accuracy. Because TDGA AutoAugment includes various transformations in one sub-policy, it functions with fewer sub-policies than previous methods (AutoAugment has 25 sub-policies and Fast AutoAugment has 50 sub-policies).

Additionally, we performed comparative experiments between the TDGA and SGA controller. In the SGA, we used tournament selection with a size of 2 and uniform crossover. The probability of the crossover between individuals was set to 0.6. As the TDGA performs fitness evaluations twice as many fitness evaluations as the SGA in one generation, the population size in the SGA is twice as large as that of the

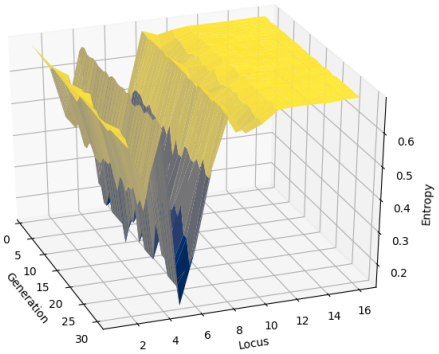

Figure 3: Variation of the entropy of each locus during the search process on CIFAR-10 ($T = 0.02$).

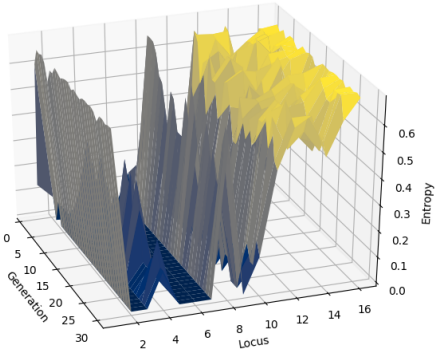

Figure 4: Variation of the entropy of each locus during the search process on CIFAR-10 ($T = 0.002$).

TDGA to compensate for the difference. Other hyperparameters were similar to those of the CIFAR-10 experiment on Wide-ResNet-28-2. With SGA, we obtained a test accuracy of 95.80±0.10%, which is 0.12% lower than that of TDGA AutoAugment. Figure 7 shows the entropy value of the final generation and the number of operations with the SGA controller. The axes are the same as in Figure 5. Figure 7 shows that many operations were completely removed in the final training phase with the SGA controller. Overall, this result enhanced the effectiveness of the TDGA controller in a search space of approximately the same size.

SVHN Results

SVHN is an image dataset that contains numbers and consists of 73,257 training images (the core training set), 531,131 additional training images(the extra training set), and 26,032 test images. In our experiments, we used the core dataset to assess the direct search performance. We used the same default augmentations as for CIFAR-10. For the model, we used a learning rate of 5e-3, weight decay of 5e-3, batch size of 128, and cosine learning rate decay for 200 epochs. We set $m = 3$ for the experiment. The result is presented in Table 2. We find that the best settings were $M = 8$, and $T = 0.02$.

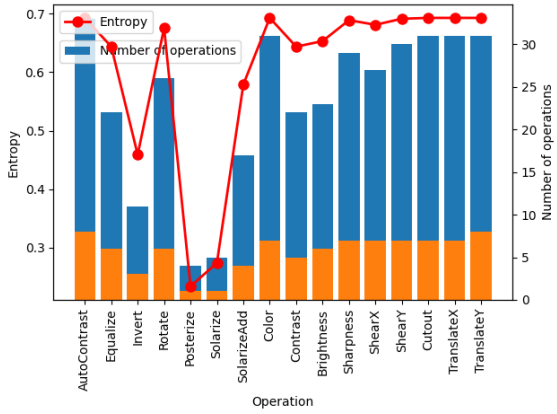

Figure 5: Relationship between the final entropy value and the number of operations. The blue part of the bar graph shows the number of operations in the final generation and the orange part shows the number of operations adopted in the final training phase.

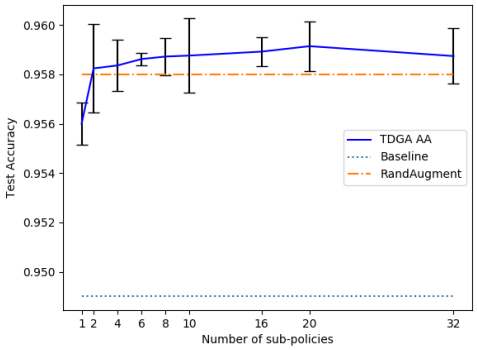

Figure 6: Variation of the test accuracy as the number of sub-policies increases. We performed five independent runs for each setting. The bars represent the standard deviation values for each trial.

In our experiment, we could not obtain results comparable to those of the previous methods for Wide-ResNet-28-10. In the experiments using SVHN, more types and numbers of transformations tended to be adopted than in the experiments using CIFAR-10. According to RandAugment, the number of optimal transformations increases with the size of the dataset and model, and this result itself is natural. On the other hand, if a sub-policy contains a large number of operations, excessive transformations may be applied to the image, and the essence of the data may be lost. This problem is thought to be one of the reasons why the accuracy rate did not improve sufficiently in the experiments using SVHN.

Computational Cost

To compare the computational costs of the methods, we measured the time required to conduct a policy search using reduced CIFAR-10, which consists of 4,000 randomly cho-

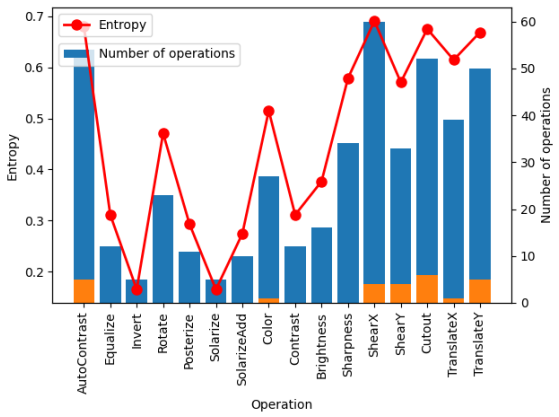

Figure 7: Final entropy value and the number of operations with SGA controller.

	GPU hours	GPU
AA	5000	Tesla P100
PBA	5	Titan XP
Fast AA	3.5	Tesla V100
TDGA AA	2.5	RTX 2080Ti

Table 3: Comparison of the number of GPU hours required by TDGA AutoAugment with those required by other methods.

sen examples. We used the PyramidNet+ShakeDrop model (Yamada, Iwamura, and Kise 2018). We set the number of pre-training epochs to 120 (used in AutoAugment) and used a batch size of 32. The other parameters were similar to those used in the experiment. Table 3 compares the computational costs. TDGA AutoAugment is computationally more efficient than the previous methods (it requires more than a GPU hour for both pre-training and searching). This depends on the small search space of TDGA AutoAugment (see Figure 1); it is expected that additional GPU hours would be required as the problem setting becomes more complex.

Discussion

TDGA AutoAugment solved the problems presented by the huge computational cost and search convergence; however, certain problems should be addressed.

First, in our experiment, the probability of applying the operation was set to m/L and shared among all operations. The role of m in TDGA AutoAugment is similar to that of N in RandAugment. This means that m is responsible for the expected value of the number of operations applied. In RandAugment, it has been reported that the optimal value of N varies depending on the size of the dataset and model. Therefore, we expect the accuracy to improve by searching the application probability of each transformation. As an implementation, the TDGA extension that can handle any non-negative integer as a gene is effective.

Second, TDGA AutoAugment does not consider the order in which operations are applied. The order may affect the accuracy depending on the application operation ($f \circ g$ is not necessarily the same as $g \circ f$). The problem of optimizing the application order of operations can be regarded as a traveling salesman problem (TSP). The graph has operations as nodes and operation connection relationships as edges. The application of the TDGA to solve the TSP is considered effective (Maekawa et al. 1996). In this case, the population will evolve to preserve the diversity of the branch, which is equivalent to the diversity of the application order.

Finally, it was conformed that the accuracy of TDGA AutoAugment can be improved by setting an appropriate temperature. However, the specification of the temperature range to be searched depends on the fitness value of the individual. This means that the policy search is still not completely automatic. Therefore, it is also necessary to adjust the temperature dynamically to maintain the entropy (Mori, Kita, and Nishikawa 1996).

Conclusion

In this paper, we introduced TDGA AutoAugment, which is an automatic data augmentation method for searching a policy including various sub-policies. Our method succeeded in searching for reasonable and diverse augmentation while reducing the calculation time and achieved comparable accuracy to that of previous methods. Besides, we confirmed the effectiveness of our TDGA controller through the comparison experiment using SGA on the same search space.

Future tasks include experiments on other benchmark datasets, such as CIFAR-100 and ImageNet or real-world problems, investigation of the transferability of the obtained augmentation policy to other datasets, and automatic search for the operation sequence, probability, and magnitude.

Acknowledgments

This work was supported by JSPS KAKENHI Grant, Grant-in-Aid for Scientific Research(B), 19H04184.

References

Cubuk, E. D.; Zoph, B.; Mané, D.; Vasudevan, V.; and Le, Q. V. 2019. AutoAugment: Learning Augmentation Strategies From Data. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 113–123.

Cubuk, E. D.; Zoph, B.; Shlens, J.; and Le, Q. 2020. RandAugment: Practical Automated Data Augmentation with a Reduced Search Space. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 18613–18624. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper/2020/file/d85b63ef0ccb114d0a3bb7b7d808028f-Paper.pdf.

Deng, J.; Dong, W.; Socher, R.; Li, L.; Kai Li; and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

Devries, T.; and Taylor, G. W. 2017. Improved Regularization of Convolutional Neural Networks with Cutout. *CoRR* abs/1708.04552. URL http://arxiv.org/abs/1708.04552.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. USA: Addison-Wesley Longman Publishing Co., Inc., 1st edition. ISBN 0201157675.

Han, D.; Kim, J.; and Kim, J. 2017. Deep Pyramidal Residual Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6307–6315.

Ho, D.; Liang, E.; Stoica, I.; Abbeel, P.; and Chen, X. 2019. Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules. In *ICML*.

Holland, J. H. 1992. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press. ISBN 0262082136.

Jaderberg, M.; Dalibard, V.; Osindero, S.; Czarnecki, W. M.; Donahue, J.; Razavi, A.; Vinyals, O.; Green, T.; Dunning, I.; Simonyan, K.; Fernando, C.; and Kavukcuoglu, K. 2017. Population Based Training of Neural Networks. *CoRR* abs/1711.09846. URL http://arxiv.org/abs/1711.09846.

Kirkpatrick, S.; Gelatt, C. D.; and Vecchi, M. P. 1983. Optimization by Simulated Annealing. *Science* 220(4598): 671–680. ISSN 0036-8075. doi:10.1126/science.220.4598.671. URL https://science.sciencemag.org/content/220/4598/671.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. *Master's thesis, University of Tront* URL https://ci.nii.ac.jp/naid/20001706980/.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F.; Burges, C. J. C.; Bottou, L.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 25*, 1097–1105. Curran Associates, Inc. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Lemley, J.; Bazrafkan, S.; and Corcoran, P. 2017. Smart Augmentation Learning an Optimal Data Augmentation Strategy. *IEEE Access* 5: 5858–5869.

Lim, S.; Kim, I.; Kim, T.; Kim, C.; and Kim, S. 2019. Fast AutoAugment. *CoRR* abs/1905.00397. URL http://arxiv.org/abs/1905.00397.

Maekawa, K.; Mori, N.; Tamaki, H.; Kita, H.; and Nishikawa, Y. 1996. A genetic solution for the traveling salesman problem by means of a thermodynamical selection rule. In *Proceedings of IEEE International Conference on Evolutionary Computation*, 529–534.

Mori, N.; Kita, H.; and Nishikawa, Y. 1996. Adaptation to a changing environment by means of the thermodynamical genetic algorithm. In Voigt, H.-M.; Ebeling, W.; Rechenberg, I.; and Schwefel, H.-P., eds., *Parallel Problem Solving from Nature — PPSN IV*, 513–522. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-70668-7.

Mori, N.; Yoshida, J.; Tamaki, H.; Kita, H.; and Nishikawa, Y. 1995. A thermodynamical selection rule for the genetic algorithm. In *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, volume 1, 188–. doi:10.1109/ICEC.1995.489142.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Ratner, A. J.; Ehrenberg, H.; Hussain, Z.; Dunnmon, J.; and Ré, C. 2017. Learning to Compose Domain-Specific Transformations for Data Augmentation. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*, 3236–3246. Curran Associates, Inc. URL http://papers.nips.cc/paper/6916-learning-to-compose-domain-specific-transformations-for-data-augmentation.pdf.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2018. Regularized Evolution for Image Classifier Architecture Search. *CoRR* abs/1802.01548. URL http://arxiv.org/abs/1802.01548.

Shahrokh Esfahani, M.; and Dougherty, E. R. 2013. Effect of separate sampling on classification accuracy. *Bioinformatics* 30(2): 242–250. ISSN 1367-4803. doi:10.1093/bioinformatics/btt662. URL https://doi.org/10.1093/bioinformatics/btt662.

Shorten, C.; and Khoshgoftaar, T. M. 2019. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data* 6: 1–48.

Tran, T.; Pham, T.; Carneiro, G.; Palmer, L.; and Reid, I. 2017. A Bayesian Data Augmentation Approach for Learning Deep Models. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*, 2797–2806. Curran Associates, Inc. URL http://papers.nips.cc/paper/6872-a-bayesian-data-augmentation-approach-for-learning-deep-models.pdf.

Yamada, Y.; Iwamura, M.; Akiba, T.; and Kise, K. 2019. Shakedrop Regularization for Deep Residual Learning. *IEEE Access* 7: 186126–186136.

Yamada, Y.; Iwamura, M.; and Kise, K. 2018. ShakeDrop regularization. *CoRR* abs/1802.02375. URL http://arxiv.org/abs/1802.02375.

Zagoruyko, S.; and Komodakis, N. 2016. Wide Residual Networks. In Richard C. Wilson, E. R. H.; and Smith, W. A. P., eds., *Proceedings of the British Machine Vision Conference (BMVC)*, 87.1–87.12. BMVA Press. ISBN 1-901725-59-6. doi:10.5244/C.30.87. URL https://dx.doi.org/10.5244/C.30.87.

Zhang, H.; Cissé, M.; Dauphin, Y.; and Lopez-Paz, D. 2018. mixup: Beyond Empirical Risk Minimization. *ArXiv* abs/1710.09412.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2017. Learning Transferable Architectures for Scalable Image Recognition. *CoRR* abs/1707.07012. URL http://arxiv.org/abs/1707.07012.