Universal Adaptive Data Augmentation

Xiaogang Xu[1] , **Hengshuang Zhao**[2]
[1]Zhejiang Lab
[2]The University of Hong Kong
xgxu@zhejianglab.com, hszhao@cs.hku.hk

Abstract

Existing automatic data augmentation (DA) methods either ignore updating DA's parameters according to the target model's state during training or adopt update strategies that are not effective enough. In this work, we design a novel data augmentation strategy called "Universal Adaptive Data Augmentation" (UADA). Different from existing methods, UADA would adaptively update DA's parameters according to the target model's gradient information during training: given a pre-defined set of DA operations, we randomly decide types and magnitudes of DA operations for every data batch during training, and adaptively update DA's parameters along the gradient direction of the loss concerning DA's parameters. In this way, UADA can increase the training loss of the target networks, and the target networks would learn features from harder samples to improve the generalization. Moreover, UADA is very general and can be utilized in numerous tasks, e.g., image classification, semantic segmentation and object detection. Extensive experiments with various models are conducted on CIFAR-10, CIFAR-100, ImageNet, tiny-ImageNet, Cityscapes, and VOC07+12 to prove the significant performance improvements brought by UADA.

1 Introduction

The performance of deep neural networks (DNNs) would be improved significantly when more data is available. However, the collection of datasets is expensive. To this, a massive amount of data augmentation (DA) methods have been proposed to remedy the deficiency of supervised data. The generalization of DNNs could be prominently improved if an effective DA method is adapted. The current DA methods can be divided into two categories. The first category is the human-designed DA, e.g., random crop and Cutout [DeVries and Taylor, 2017]. Recently, the human-designed DA methods have been gradually replaced by automatic DA approaches [Zhang *et al.*, 2020; Wu *et al.*, 2020].

Existing automatic DA approaches can be divided into two main categories. The first one would set a search phase before training to obtain DA strategies, while such methods ignore updating DA's parameters according to the target model's state during training. The second one would optimize DA strategies during training. However, they all adopt a common strategy which is not effective enough: randomly sampling DA's parameters and deciding the optimized parameters according to the loss value [Zhang *et al.*, 2020; Wu *et al.*, 2020; Lin *et al.*, 2019]. On the contrary, we claim that DA's parameters should be optimized according to the target model's state, especially the gradient information. We can compute the target model's gradient with respect to DA's parameters, and update DA's parameters along with the direction of such gradient. This strategy can effectively help to explore the hard sample, improving the target model's generalization.

Task	Model	Baseline(%)	Ours(%)
Cls. on CIFAR10	ResNet18	95.22	97.08(+1.86)
	WideResNet-28-10	96.14	97.83(+1.69)
	Shake-Shake (26 2x32d)	96.35	97.68(+1.33)
	Shake-Shake (26 2x96d)	96.96	98.27(+1.31)
Cls. on CIFAR100	ResNet18	77.01	79.88(+2.87)
	WideResNet-28-10	81.03	83.17(+2.14)
	Shake-Shake (26 2x32d)	79.02	82.28(+3.26)
	Shake-Shake (26 2x96d)	82.08	85.88(+3.80)
Cls. on tiny-ImageNet	ResNet18	63.13	67.41(+4.28)
	ResNet50	65.52	70.81(+5.29)
Seg. on Cityscapes	PSPNet50	76.11	77.80(+1.69)
Det. on VOC07+12	RFBNet	80.10	81.00(+0.90)

Table 1: This table summarizes results of UADA on different models, datasets and tasks. For image classification (Cls.), we report the top-1 accuracy, for semantic segmentation (Seg.), we report the mIoU, for object detection (Det.), we report the mAP. "baseline" is the most common DA strategy on the corresponding datasets.

In this paper, we propose a novel automatic augmentation method, called "Universal Adaptive Data Augmentation" (UADA). Given a pre-defined set of DA operations, UADA would determine types and magnitudes of DA operations via random sampling during training, and adaptively update DA's parameters according to the target model's gradient at each training step. With UADA, the augmented data can lead to a larger loss value, avoiding the overfit on the training data and improving the generalizability on validation data.

UADA for adaptively optimizing DA's parameters consists of four stages (as shown in Fig. 1): 1) randomly acquire a set of DA's parameters, load a batch of data, augment the data for training, and compute the loss value; 2) obtain the gradients of the loss with respect to the DA's parameters; 3) update

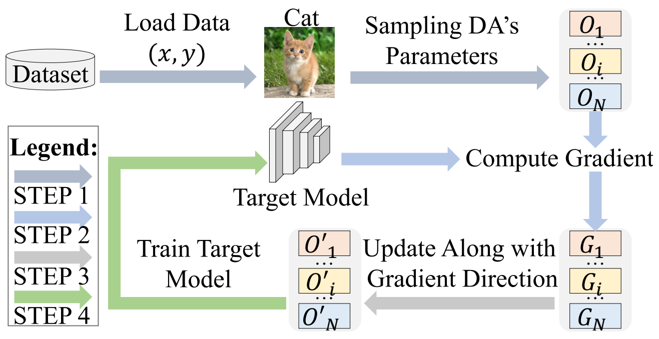

Figure 1: The summary of our method's pipeline, utilizing target model's gradient to adaptively update DA's parameters during training. Details can be viewed in Fig. 2.

DA's parameters along with the directions of the gradients; 4) utilize the optimized DA's parameters to augment the data for training, and employ the loss value and backward operation to update the weights of the network. The tremendous challenge is the discrete property of DA's parameters, i.e., it is challenging to compute the accurate gradient of the loss concerning DA's parameters. In contrast to backward the gradient from the network to DA's parameters, we design an effective approach to simulate the required gradients. This gradient simulation module is motivated by the black-box adversarial method [Chen *et al.*, 2017]: negligible perturbations are added to DA's parameters, and we measure the change of loss to decide the value of gradients. In this way, an approximate value for the gradient of the loss concerning DA's parameters can be obtained with two network forwarding operations.

Our UADA is generally applicable to different tasks, including image classification, semantic segmentation and object detection, since UADA only requires knowing what kinds of DA operations will be utilized during the training. Extensive experiments are conducted with existing representative image classification datasets, including CIFAR-10/CIFAR-100 [Krizhevsky *et al.*, 2009], and ImageNet [Deng *et al.*, 2009]/tiny-ImageNet [Le and Yang, 2015]. The results on all datasets with miscellaneous network structures demonstrate the effectiveness of UADA. Moreover, experiments are also conducted in the semantic segmentation and object detection task, and the results on the Cityscapes [Cordts *et al.*, 2016] and VOC07+12 [Everingham *et al.*, 2012] datasets prove the effects of UADA, as summarized in Table 1.

In summary, our contributions in this paper are threefold.

- We propose a novel Universal Adaptive Data Augmentation (UADA) strategy. This strategy can adaptively optimize DA's parameters along with the directions of loss gradient concerning DA's parameters during training, enhancing the generalization of the trained models.
- An effective gradient simulation approach is proposed in UADA to acquire the gradient of the loss with respect to DA's parameters.
- We conduct experiments with various model structures on different datasets and tasks, which manifest the effectiveness and generality of our UADA.

2 Related Work

2.1 Human-designed DA

Existing DA methods can be divided into human-designed DA and automatic DA. They can generate extra samples by some label-preserved transformations to increase the size of datasets and improve the generalization of networks. Generally speaking, human-designed DA policies are specified for some datasets. Therefore, varying human-designed DA methods should be chosen for different datasets. For example, Cutout [DeVries and Taylor, 2017] is widely utilized in the training of CIFAR-10/CIFAR-100 [Krizhevsky *et al.*, 2009], while it is not suitable for ImageNet [Deng *et al.*, 2009] since Cutout would destroy the property of original samples.

2.2 Automatic DA

AutoAugment [Cubuk *et al.*, 2019] is the first automatic augmentation method. The strategy searched by AutoAugment [Cubuk *et al.*, 2019] includes the operation of ShearX/Y, TranslateX/Y, Rotate, AutoContrast, Invert, Equalize, Solarize, Posterize, Contrast, Color, Brightness, Sharpness, Cutout, and Sample Pairing. The range of the magnitude for each operation is also discretized uniformly into ten values. For every dataset, an augmentation policy in AutoAugment is composed of 5 sub-policies, and 5 best policies are concatenated to form a single policy with 25 sub-policies. Each sub-policy contains two operations to be applied orderly, and each operation in AutoAugment is conducted with a chosen magnitude and a direction. However, it has large search space and the expensive search cost [Cubuk *et al.*, 2019]. Two kinds of following works are proposed: reducing the cost of search phase or searching during training to eliminate search phase.

Reduce search space. Recently, more and more researchers focus on reducing search space to acquire automatically-learned DA and speeding up the search process, e.g., the RandAugment [Cubuk *et al.*, 2020], Fast AutoAugment [Lim *et al.*, 2019], Faster AutoAugment [Hataya *et al.*, 2020], DADA [Li *et al.*, 2020], UA [LingChen *et al.*, 2020], and TrivialAugment [Müller and Hutter, 2021]. RandAugment [Cubuk *et al.*, 2020] proposed a simplified search space that vastly reduces the computational expense of automatic augmentation; Fast AutoAugment [Lim *et al.*, 2019] found effective augmentation policies via a more efficient search strategy based on density matching; Faster AutoAugment [Hataya *et al.*, 2020] designed a differentiable policy search pipeline for data augmentation; DADA [Li *et al.*, 2020] also formulated a differentiable automatic data augmentation strategy to dramatically reduces the search cost; TrivialAugment [Müller and Hutter, 2021] proposed a simple strategy which is parameter-free and only applies a single augmentation to each image.

Search with training. There is another kind of approach which learn the augmentation policy online as the training goes, e.g., PBA [Ho *et al.*, 2019], OHL [Lin *et al.*, 2019], Adversarial AutoAugment [Zhang *et al.*, 2020], LTDA [Wu *et al.*, 2020], DDAS [Liu *et al.*, 2021a], DivAug [Liu *et al.*, 2021b] and OnlineAugment [Tang *et al.*, 2020]. PBA [Ho *et al.*, 2019] employed multiple workers that each uses a different policy and are updated in an evolutionary fashion; OHL [Lin *et al.*, 2019] adopted multiple parallel workers with reinforcement

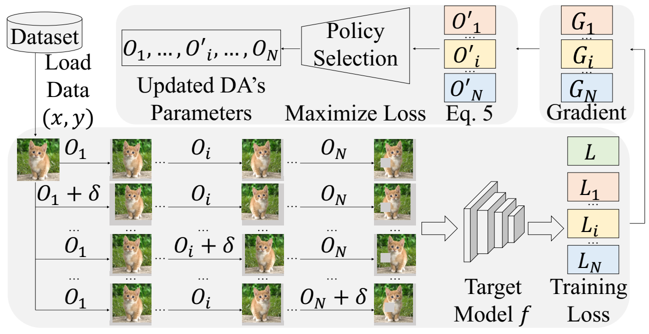

Figure 2: The illustration of our Universal Adaptive Data Augmentation (UADA). Given a set of DA's parameters, we update them along with the direction of loss gradient concerning DA's parameters.

learning, and defined the accuracy on held-out data after a part of training; Adversarial AutoAugment [Zhang *et al.*, 2020] made a reinforcement-learning based update, rewarding the policy yielding the lowest accuracy training accuracy, and causing the policy distribution to shift towards progressively stronger augmentations; LTDA [Wu *et al.*, 2020] proposed strategy to maximize the training loss via multiple sampling for DA's parameters; DDAS [Liu *et al.*, 2021a] exploited meta-learning with one-step gradient update and continuous relaxation to the expected training loss for efficient search; DivAug [Liu *et al.*, 2021b] designed a strategy to maximize the diversity of training data and hence strengthen the regularization effect. OnlineAugment [Tang *et al.*, 2020] employs an augmentation model to perform data augmentation, which is differentiable and can be optimized during training. And the augmentation model can only conduct certain types of DA operations.

Our adaptive strategy. Different from these approaches, we propose a new framework (UADA) to online learn the augmentation policy: employ only one worker for sampling DA' parameters in each batch during training and adaptively update the DA' parameters according to the model state. In this strategy, we randomly determine the DA operations for every data batch, randomly decide types and magnitudes of DA operations during training, and adaptively update DA's parameters according to model state at each step.

Our method is very different from existing differentiable and adversarial DA methods: 1) We directly compute the gradient of the loss concerning DA's parameters which is different from existing differentiable policy [Hataya *et al.*, 2020; Li *et al.*, 2020; Liu *et al.*, 2021a]. And our strategy to obtain the gradient is achieved with adversarial learning that is different from current approaches. 2) We directly update DA's parameters along the gradient direction of loss with respect to DA's parameters, maximizing the training loss, and this is different from Adversarial AutoAugment [Zhang *et al.*, 2020], which training a policy network with reinforcement-learning, and LTDA [Wu *et al.*, 2020], which randomly samples several sets of values for augmentation operations to conduct augmentation and choose the loss with maximum value for backward. UADA can be utilized for various tasks, and is implemented without search strategies, avoiding the expensive search costs.

3 Method

3.1 Motivation

Generally speaking, the target of a DA method is to avoid overfitting. To this, DA methods would normally adopt the policy to increase the training loss of the target networks, promoting the learning from hard samples.

Given a set of pre-defined DA operations, most of the current automatic DA methods [Cubuk *et al.*, 2019; Cubuk *et al.*, 2020] adopt the strategy of random sampling to decide the corresponding parameters. In contrast to prior works, we claim that the DA's parameters should be decided according to the target model's state. Therefore, we propose to adaptively update the DA's parameters along with the direction of loss gradient concerning DA's parameters.

3.2 Notation

Given a set of DA operation $\mathcal{O}$, it consists of several operations $\mathcal{O}_i, i \in [1, N]$, where N is the number of the operation. $\mathcal{O}_i$ is combined with the corresponding parameters O_i, including types and magnitudes. Assume x and y are the notations of data and its corresponding ground truth, respectively. The target model for training is represented as $\mathcal{F}$, and the loss function to train the model is denoted $\mathcal{L}()$. Associate with the chosen DA, for a batch of data x, the loss function can be

$$\mathcal{L}(\mathcal{F}(\mathcal{O}_N(...\mathcal{O}_1(x|O_1)|O_N)), y), \tag{1}$$

where $\mathcal{O}_i(\quad |O_i)$ means applying the i-th DA's operation, $\mathcal{O}_i$, to process the data, with its parameters O_i.

3.3 The Pipeline of UADA

Overview. The overview of our UADA strategy can be viewed in Fig. 2, which consists of four stages. First, we load a batch of data (x, y), acquire a set of DA's parameters $O_i, i \in [1, N]$ via random sampling, apply them to process the data x, and obtain the corresponding loss value $\mathcal{L}$. Next, we add perturbations δ to DA's parameters, augment the data, compute the loss value $\mathcal{L}_i, i \in [1, N]$, and measure the change of loss compared with $\mathcal{L}$. In this way, the gradients of the loss with respect to DA's parameters ($G_i, i \in [1, N]$) can be simulated. Finally, we update DA's parameters with the direction of the computed gradients (maximizing the training loss), apply the updated DA's operations to process the batch of data, use the processed data to update the network's weights. Details will be described in the following.

Random sampling DA's parameters. Like normal DA strategy [Cubuk *et al.*, 2019; Cubuk *et al.*, 2020], we shall first obtain a set of DA's parameters $O_i, i \in [1, N]$ via random sampling. And we can then update the corresponding values according to target model's state. This is different from [Zhang *et al.*, 2020; Wu *et al.*, 2020; Lin *et al.*, 2019] since they would adopt multiple sampling operations to obtain multiple sets of DA's parameters during training.

Compute gradient. It is nearly impossible to obtain the gradient of loss concerning DA's parameters, since DA's parameters are often non-differentiable since their values are discrete, e.g., the coordinates in the Cutout, as well as the policy of the searched AutoAugment [Cubuk *et al.*, 2019]. Even some operations can be differentiable, e.g., random rotation and

Algorithm 1 The training algorithm with our Adaptive Adversarial Data Augmentation († means can be done *parallel*)

```
1: Input: training data $(X, Y)$, the target model $\mathcal{F}$, perturbation value $\delta$, step size $\epsilon$, total training epoch $t_{max}$
2: for $t = 1$ to $t_{max}$ do
3:     Load a batch of data $(x, y)$ from the training data $(X, Y)$.
4:     †Determine parameters of $\mathcal{O}_1, ..., \mathcal{O}_N$, as $O_1, ..., O_N$, via random sampling.
5:     †Apply the augmentation operations $\mathcal{O}_1, ..., \mathcal{O}_N$ for $(x, y)$. Compute the loss $\mathcal{L}(\mathcal{F}(\mathcal{O}_N(...\mathcal{O}_1(x|O_1)|O_N)), y)$.
6:     †For $i = 1 : N$, add the perturbation $\delta$ to DA's parameters and update them with $\epsilon$ with Eq. 5, obtaining $O'_1, ..., O'_N$.
7:     †For $i = 1 : N$, apply the augmentation with parameters as $O_1, ...O'_i, ..., O_N$ for $(x, y)$ to compute the loss.
8:     Choose the loss with the maximal value to update the network $\mathcal{F}$.
9: end for
10: return the trained network $\mathcal{F}$
```

random translation [Riba *et al.*, 2020], they would become non-differentiable if they are combined with the other non-differentiable operations.

Suppose a negligible random perturbation δ is added to the DA's parameter O_i, then we can obtain intermediate DA's parameters $\widehat{O}_i = O_i + \delta$. Meanwhile, we can compute the new loss as $\mathcal{L}(\mathcal{F}(\mathcal{O}_N(...\mathcal{O}_i(...\mathcal{O}_1(x|O_1)|\widehat{O}_i)|O_N)), y)$. Comparing the new loss with the original loss $\mathcal{L}(\mathcal{F}(\mathcal{O}_N(...\mathcal{O}_i(...\mathcal{O}_1(x|O_1)|O_i)|O_N)), y)$, we can acquire the simulated gradient of the loss with respect to the DA's parameter O_i, as

$$\begin{aligned} G_i = [&\mathcal{L}(\mathcal{F}(\mathcal{O}_N(...\mathcal{O}_i(...\mathcal{O}_1(x|O_1)|\widehat{O}_i)|O_N)), y) - \\ &\mathcal{L}(\mathcal{F}(\mathcal{O}_N(...\mathcal{O}_i(...\mathcal{O}_1(x|O_1)|O_i)|O_N)), y)]/\delta. \end{aligned} \tag{2}$$

δ is small enough (its value can be positive or negative).

Update DA's parameters along with gradient direction. The adversarial learning methods [Goodfellow *et al.*, 2015; Madry *et al.*, 2018; Chen *et al.*, 2017; Guo *et al.*, 2019] would compute the gradient of the loss with respect to the input data x. The computed gradient is employed to create the perturbation and obtain the perturbed input data x'. A representative process, which is called FGSM [Goodfellow *et al.*, 2015], can be written as

$$x' = x + \alpha \cdot sign(\bigtriangledown_x(\mathcal{L}(\mathcal{F}(x), y))), \tag{3}$$

where $\bigtriangledown_x$ means computing the gradient concerning the input data x, $sign()$ is the operation to return the sign of the input data, and α is the step size, i.e., the magnitude of the perturbation. The perturbed data x' can cause an increase of loss for the target model $\mathcal{F}$. Training with both clean samples x and the perturbed data x' can enhance the robustness of the target model $\mathcal{F}$. However, such training will also sacrifice the generalization of the target model, leading to a decrease in performance on clean samples in the validation set.

In this paper, we apply the idea of adversarial learning to the update of DA's parameters, resulting in the enhancement of the target model's generalization. Given a batch of data x, we compute the loss as $\mathcal{L}(\mathcal{F}(\mathcal{O}_N(...\mathcal{O}_1(x|O_1)|O_N)), y)$. Suppose we can compute the gradient of the loss with respect to the DA's parameters, the parameters of each DA's operation can be updated as

$$O'_i = O_i + \epsilon \cdot sign(\bigtriangledown_{O_i}(\mathcal{L}(\mathcal{F}(\mathcal{O}_N(...\mathcal{O}_1(x|O_1)|O_N)), y))), \tag{4}$$

where $\bigtriangledown_{O_i}(\mathcal{L}(\mathcal{F}(\mathcal{O}_N(...\mathcal{O}_1(x|O_1)|O_N)), y))$ is indeed G_i in Eq. 2. The training with DA's parameters O'_i can also lead to the increase of loss compared with the training with O_i.

Thus, the updating of the parameter O_i can be written as

$$\begin{aligned} \widehat{O}_i &= O_i + \delta \\ G_i &= [\mathcal{L}(\mathcal{F}(\mathcal{O}_N(...\mathcal{O}_i(...\mathcal{O}_1(x|O_1)|\widehat{O}_i)|O_N)), y) \\ &\quad - \mathcal{L}(\mathcal{F}(\mathcal{O}_N(...\mathcal{O}_i(...\mathcal{O}_1(x|O_1)|O_i)|O_N)), y)]/\delta, \\ O'_i &= O_i + \epsilon \cdot sign(G_i), \end{aligned} \tag{5}$$

where ϵ is the positive hyper-parameter for control, and it should be small enough for the accuracy of simulated gradients. During the training, there are usually multiple DA's operations. We select the operation that has the most decisive influence on the change of loss value, i.e., its gradient is crucial enough. The details of the training algorithm by using UADA can be viewed in Alg. 1.

4 Experiments

4.1 Datasets

For the classification task, the datasets include CIFAR-10/CIFAR-100 [Krizhevsky *et al.*, 2009], and ImageNet [Deng *et al.*, 2009]/tiny-ImageNet [Le and Yang, 2015]. For experiments of image classification, we report the accuracy value (%). For the semantic segmentation and object detection, experiments are conducted on Cityscapes [Cordts *et al.*, 2016] and VOC07+12 [Everingham *et al.*, 2012].

4.2 Implementation Details

We use PyTorch [Paszke *et al.*, 2017] to implement ResNet-18 [He *et al.*, 2016b], WideResNet-28-10 [Zagoruyko and Komodakis, 2016], and Shake-Shake [Gastaldi, 2017] for CIFAR-10/CIFAR-100; ResNet-18 and ResNet-50 for ImageNet [Deng *et al.*, 2009]/tiny-ImageNet [Le and Yang, 2015]; PSPNet [Zhao *et al.*, 2017] for the Cityscapes [Cordts *et al.*, 2016]; SSD [Liu *et al.*, 2016] and RFBNet [Liu *et al.*, 2018] for VOC07+12 [Everingham *et al.*, 2012].

δ is set as 1, and ϵ in Eq. 5 is also set as 1 in the classification experiments (except the ablation study). And the values of (δ, ϵ) could be decided for different datasets adaptively according to the augmentation types. To ensure the accuracy of the estimated gradient, the value of (δ, ϵ) should not be large. Thus, the search space for the hyper-parameter (δ, ϵ) is not large for different datasets. And to decide the optimal value of (δ, ϵ) for a target dataset, the value range of (δ, ϵ) can be quantized into several points. The search can be completed on these points without much tuning. Especially, as proved by our ablation study, the performance of UADA is not sensitive to the value of (δ, ϵ).

CIFAR-10	Baseline	Cutout	AA	RA	DADA	Fast AA	Faster AA	PBA
ResNet18	95.22	96.17	96.49	96.33	-	-	-	-
WideResNet-28-10	96.14	96.96	97.34	97.47	97.30	97.30	97.40	97.40
Shake-Shake (26 2x32d)	96.35	96.90	97.53	97.38	97.30	97.30	97.30	97.46
Shake-Shake (26 2x96d)	96.96	97.44	97.98	98.03	98.00	98.00	98.00	97.97
CIFAR-10	O.A.	TA	UA	LTDA	DDAS	DivAug	Adv. AA	**Ours**
ResNet18	-	-	-	-	-	-	-	**97.08**
WideResNet-28-10	97.60	97.46	97.33	97.89	97.30	**98.10**	97.50	97.83
Shake-Shake (26 2x32d)	-	-	-	-	-	-	97.64	**97.68**
Shake-Shake (26 2x96d)	-	98.21	98.10	98.22	97.90	98.10	98.15	**98.27**
CIFAR-100	Baseline	Cutout	AA	RA	DADA	Fast AA	Faster AA	PBA
ResNet18	77.01	77.50	79.05	78.75	-	-	-	-
Shake-Shake (26 2x32d)	79.02	80.55	82.20	82.03	-	-	-	-
Shake-Shake (26 2x96d)	82.08	83.08	85.61	85.12	84.70	85.10	84.40	84.69
CIFAR-100	O.A.	TA	UA	LTDA	DDAS	DivAug	Adv. AA	**Ours**
ResNet18	-	-	-	-	-	-	-	**79.88**
Shake-Shake (26 2x32d)	-	-	-	-	-	-	-	**82.28**
Shake-Shake (26 2x96d)	-	85.16	85.00	-	84.90	85.30	**85.90**	85.88

Table 2: The comparison with SOTA DA methods on CIFAR.

Method	RA	Adv. AA	LTDA	DivAug	UADA
Training($\times$)	1.0	8.0	6.0	4.5	4.2

Table 3: Comparing the training cost among our method, RA, Adv. AA, LTDA and DivAug on CIFAR-10 relative to RA.

4.3 UADA for Image Classification

Results of CIFAR-10/CIFAR-100. CIFAR-10/CIFAR-100 dataset has total 60,000 images. 50,000 images are set as the training set and 10,000 images are set as the test set. Each image has the size of 32×32 belongs to one of 10 classes. CIFAR-10/CIFAR-100 has been extensively studied with previous data augmentation methods, and we first test our proposed UADA on this data. The results on CIFAR-10/CIFAR-100 are reported in Tables 2. These results have shown that our adaptive strategy can stably outperform most of the baselines with different model structures, where "baseline" is the setting of [DeVries and Taylor, 2017] (randomly crop and random horizontal flip) and "Cutout" is the combination of randomly crop and random horizontal flip and Cutout. UADA can lead to more significant improvement for AutoAugment (AA) [Cubuk *et al.*, 2019] and RandAugment (RA) [Cubuk *et al.*, 2020], compared with the advancement of human-designed DA.

Moreover, the comparisons between UADA and more state-of-the-art (SOTA) approaches with complex model structures, are shown in Tables 2. These top-ranking methods contain Adversarial AutoAugment (Adv. AA) [Zhang *et al.*, 2020], PBA [Ho *et al.*, 2019], DADA [Li *et al.*, 2020], Fast AutoAugment [Lim *et al.*, 2019] (Fast AA), Faster AutoAugment [Hataya *et al.*, 2020] (Faster AA), UA [LingChen *et al.*, 2020], LTDA [Wu *et al.*, 2020], DDAS [Liu *et al.*, 2021a], DivAug [Liu *et al.*, 2021b], O.A. [Tang *et al.*, 2020], and TA [Müller and Hutter, 2021]. As shown in this table, the performance of our UADA is higher than most of these approaches with different network structures on CIFAR-10 and CIFAR-100. The performance of Adversarial AutoAugment, LTDA, DivAug, and UADA on CIFAR is similar while our UADA has fewer training hours as shown in Table 3. The training cost of Adv. AA is cited from [Zhang *et al.*, 2020], the cost of LTDA is cited from [Wu *et al.*, 2020] and the cost of DivAug is cited from [Liu *et al.*, 2021b].

Results of ImageNet/tiny-ImageNet. We shall evaluate the effectiveness of UADA on large-scale datasets, e.g., ImageNet [Deng *et al.*, 2009] which is a significantly challenging dataset in image recognition. We employ two network structures, ResNet18 and ResNet50, for experiments. The batch size is 256, the learning rate is 0.1, the weight decay is 0.0001, the momentum is 0.9. Moreover, we adopt the cosine learning rate for training and train the network for 100 epochs. The results are shown in Table 4 where "Baseline" includes RandomResizedCrop and RandomHorizontalFlip. These results demonstrate that UADA can also lead to performance improvement on ImageNet.

Setting	ResNet18		ResNet50	
	top1	top5	top1	top5
Baseline	69.82	89.32	76.79	93.40
Ours	**70.50**	**89.51**	**77.19**	**93.50**

Table 4: Experiments on ImageNet with UADA.

Setting	ResNet18	ResNet50
Baseline	63.13	65.52
AA	64.94	67.59
RA	66.41	68.64
Ours	**67.41**	**70.81**

Table 5: Experiments on tiny-ImageNet with UADA.

Meanwhile, we also conduct experiments with a sub-set of ImageNet called tiny-ImageNet [Le and Yang, 2015]. The tiny-ImageNet contains 100,000 images of 200 classes for training, and has 10,000 images for testing. The results on tiny-ImageNet are reported in Table 5, where "baseline" is the setting of [Lee *et al.*, 2019] including random crop and random horizontal flip. Clearly, our UADA can also increase the performance of AA and RA.

4.4 Ablation Study

Alternative strategies for updating parameters. In our strategy, we update the DA's parameters to increase the loss value, i.e., the parameters are updated along with the direction of the computed loss gradient concerning DA parameters. And there are several alternative adaptive strategies, including

- The DA's parameters are updated with random strategy, e.g., $O_i' = O_i + \epsilon \cdot sign(R_i)$, compared with Eq. 5 ($R_i$ is the variable sampled from standard normal distribution).
- The DA's parameters are updated with the adversarial strategy that aims to minimize the loss. Different from our strategy, this alternative strategy would update DA's parameters in reverse to the direction of the gradient, e.g., $O_i' = O_i - \epsilon \cdot sign(G_i)$, compared with Eq. 5.

We set the experiments to demonstrate the superiority of our adaptive strategy over these alternative strategies. We conduct experiments with the image classification task on CIFAR-10/CIFAR-100 dataset, employing the model structures as ResNet18 and WideResNet28-10. The results are shown in Table 6. Obviously, the performance of these alternative adaptive strategies is weaker than our UADA.

The influence of step size. During the training, a hyper-parameter is the step size ϵ in updating DA's parameters. We will analyze the impact of step size on the final performance. The step size is set as 1 for the experiments in the above sections, and we change its value to 2 and 3 in this experiment.

Dataset	Setting	ResNet18	WideResNet28-10
CIFAR-10	random update strategy	96.31	97.28
	update to minimize loss	89.24	92.23
	our update strategy	**97.08**	**97.83**
CIFAR-100	random update strategy	79.54	82.95
	update to minimize loss	74.22	79.26
	our update strategy	**79.88**	**83.17**

Table 6: The results of the ablation study, evaluating the performance of alternative strategies for updating DA's parameters. We report the accuracy values.

Dataset	Model	ϵ			
		0	1	2	3
CIFAR-10	ResNet18	96.33	97.08	96.95	96.65
	WideResNet28-10	97.47	97.83	97.75	97.71
CIFAR-100	ResNet18	78.75	79.88	80.07	79.25
	WideResNet28-10	82.80	83.17	83.41	83.43

Table 7: The analysis for the influence of step size. "ϵ" is the parameter of the step size in Eq. 5, "0" means the results without adaption, i.e., the results of baselines.

The experimental results are shown in Table 7. Compared to results of other approaches as in Tables 2, the change of step size will influence the final performance of the trained network while the corresponding performance is still higher than most of the baselines.

4.5 UADA for Semantic Segmentation

Compared with the image classification task, the semantic segmentation task aims to give the category prediction of all pixels. Thus, during the training, we employ the sum of all pixels' loss in one image as the loss of one image. To demonstrate that our method can also be applied to the semantic segmentation task, we conduct the experiments on Cityscapes. The Cityscapes dataset is collected for urban scene understanding with 19 categories. It contains high-quality pixel-level annotations with 2,975, 500, and 1,525 images for training, validation, and testing. The network structure is set as PSPNet with different backbones, ResNet50 and ResNet101 [He *et al.*, 2016a]. Since existing automatic DA approaches mainly focus on image classification, there is no automatic DA method for segmentation. Thus, we set the baseline as the DA strategy, which is commonly adopted. The results are displayed in Table 8 and we can observe the improvement brought by our UADA. And visual samples are displayed in Fig. 3.

4.6 UADA for Object Detection

UADA is very general for different tasks, and we apply UADA for the detection task in this section. The experiments are conducted on VOC07+12 [Everingham *et al.*, 2012], and we choose the detection model of SSD [Liu *et al.*, 2016] and RFBNet [Liu *et al.*, 2018] for experiments. SSD is implemented by following the data augmentation strategy in https://github.com/amdegroot/ssd.pytorch, and the same DA operations are applied for the training of RFBNet. Moreover, the training parameters are kept as the default for SSD and RFBNet. As shown in Table 9, our UADA can be applied with these DA operations, and UADA can bring noticeable performance improvement for both SSD and RFBNet.

Model	PSPNet50			PSPNet101		
Setting	mIoU	mAcc	allAcc	mIoU	mAcc	allAcc
Baseline	76.11	83.80	95.77	78.22	85.65	96.12
Ours	**77.80**	**84.87**	**96.02**	**78.83**	**86.08**	**96.16**

Table 8: Experiments conducted on the semantic segmentation task, where "baseline" is the setting containing RandScale, RandRotate, RandomGaussianBlur, RandomHorizontalFlip, and random crop.

Setting	SSD300(VGG16)	RFBNet300(VGG16)
Baseline	76.9	80.1
Ours	**77.8**	**81.0**

Table 9: Experiments conducted on the object detection task, where "baseline" is the original default DA setting for SSD and RFBNet. We report the mAP.

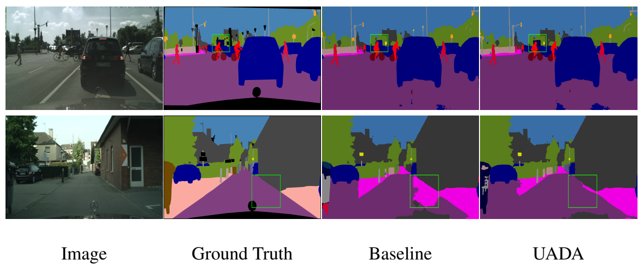

Figure 3: Visual comparisons for different DA methods, which are executed on Cityscapes with PSPNet50 (top two rows) and PSPNet101 (bottom two rows). From the view in the yellow rectangle, we can see more clear differences.

5 Conclusion

In this paper, we propose a universal adaptive data augmentation approach (UADA), where we compute the gradient of the loss with respect to the DA's parameters and use them to update the DA's parameters during training. With such a strategy, we can explore the hard samples during the training and avoid overfitting. Our UADA can be applied to different tasks. Extensive experiments are conducted on the image classification task and the semantic segmentation task with different networks and datasets, proving the effectiveness of our proposed UADA.

In the future, we would explore more effective gradient simulation methods. Moreover, as a general data augmentation strategy, we would prove the effects of our UADA for more tasks, including middle-level (e.g., depth estimation) and low-level computer vision tasks (e.g., super-resolution and denoising).

Relation with current DA methods. Besides the noticeable differences with existing DA approaches in terms of the augmentation pipeline (as carefully discussed in "Related Work" Section), UADA can also be utilized to improve the results of existing DA methods: given the DA's parameters set by existing DA methods during training, UADA can update these parameters along the direction of loss gradient concerning DA's parameters to improve the generalization.

Ethical Statement

There are no ethical issues.

Acknowledgments

This work is supported by Research Initiation Project of Zhejiang Lab (No. 2022PD0AC02).

References

[Chen *et al.*, 2017] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *ACM workshop on artificial intelligence and security*, 2017.

[Cordts *et al.*, 2016] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[Cubuk *et al.*, 2019] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, and et. al. Autoaugment: Learning augmentation strategies from data. In *CVPR*, 2019.

[Cubuk *et al.*, 2020] Ekin D. Cubuk, Barret Zoph, and et. al. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPRW*, 2020.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

[DeVries and Taylor, 2017] Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv:1708.04552*, 2017.

[Everingham *et al.*, 2012] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 Results, 2012.

[Gastaldi, 2017] Xavier Gastaldi. Shake-shake regularization. *arXiv:1705.07485*, 2017.

[Goodfellow *et al.*, 2015] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[Guo *et al.*, 2019] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In *ICML*, 2019.

[Hataya *et al.*, 2020] Ryuichiro Hataya, Jan Zdenek, and et. al. Faster autoaugment: Learning augmentation strategies using backpropagation. In *ECCV*, 2020.

[He *et al.*, 2016a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[He *et al.*, 2016b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[Ho *et al.*, 2019] Daniel Ho, Eric Liang, Xi Chen, and et. al. Population based augmentation: Efficient learning of augmentation policy schedules. In *ICML*, 2019.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. In *NIPS*, 2009.

[Le and Yang, 2015] Ya Le and Xuan Yang. Tiny ImageNet visual recognition challenge. *CS 231N*, 2015.

[Lee *et al.*, 2019] Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. Rethinking data augmentation: Self-supervision and self-distillation. *arXiv:1910.05872*, 2019.

[Li *et al.*, 2020] Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy Hospedales, and et. al. Differentiable automatic data augmentation. In *ECCV*, 2020.

[Lim *et al.*, 2019] Sungbin Lim, Ildoo Kim, Taesup Kim, and et. al. Fast autoaugment. In *NIPS*, 2019.

[Lin *et al.*, 2019] Chen Lin, Minghao Guo, Chuming Li, Xin Yuan, Wei Wu, Junjie Yan, Dahua Lin, and Wanli Ouyang. Online hyper-parameter learning for auto-augmentation strategy. In *ICCV*, 2019.

[LingChen *et al.*, 2020] Tom Ching LingChen, Ava Khonsari, Amirreza Lashkari, Mina Rafi Nazari, Jaspreet Singh Sambee, and Mario A Nascimento. Uniformaugment: A search-free probabilistic data augmentation approach. *arXiv:2003.14348*, 2020.

[Liu *et al.*, 2016] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, and et. al. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[Liu *et al.*, 2018] Songtao Liu, Di Huang, and et. al. Receptive field block net for accurate and fast object detection. In *ECCV*, 2018.

[Liu *et al.*, 2021a] Aoming Liu, Zehao Huang, Zhiwu Huang, and Naiyan Wang. Direct differentiable augmentation search. In *ICCV*, 2021.

[Liu *et al.*, 2021b] Zirui Liu, Haifeng Jin, Ting-Hsiang Wang, and et. al. Divaug: Plug-in automated data augmentation with explicit diversity maximization. In *ICCV*, 2021.

[Madry *et al.*, 2018] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[Müller and Hutter, 2021] Samuel G Müller and Frank Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *ICCV*, 2021.

[Paszke *et al.*, 2017] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS*, 2017.

[Riba *et al.*, 2020] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, and et. al. Kornia: an open source differentiable computer vision library for PyTorch. In *WACV*, 2020.

[Tang *et al.*, 2020] Zhiqiang Tang, Yunhe Gao, Leonid Karlinsky, Prasanna Sattigeri, Rogerio Feris, and Dimitris Metaxas. Onlineaugment: Online data augmentation with less domain knowledge. In *ECCV*, 2020.

[Wu *et al.*, 2020] Sen Wu, Hongyang Zhang, Gregory Valiant, and Christopher Ré. On the generalization effects of linear transformations in data augmentation. In *ICML*, 2020.

[Zagoruyko and Komodakis, 2016] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *BMVC*, 2016.

[Zhang *et al.*, 2020] Xinyu Zhang, Qiang Wang, Jian Zhang, and Zhao Zhong. Adversarial autoaugment. In *ICLR*, 2020.

[Zhao *et al.*, 2017] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.