P3-6

17th International Conference on Machine Vision Applications (MVA)
Fully Online, July 25–27, 2021.

Output augmentation works well without any domain knowledge

Shu Eguchi
Fukuoka University
eguchi.math@gmail.com

Ryo Nakamura
Fukuoka University
sd210501@cis.fukuoka-u.ac.jp

Masaru Tanaka
Fukuoka University
sieger.web@gmail.com

Abstract

Data augmentation is a method to compensate a lack of sufficient amount of training data by increasing variations of the training data. It is also used even when there is a huge amount of training data to improve a generalization performance on the test data. In this paper, we propose a new method, Output-Augmentation (OA), which we use to improve the generalization performance without data augmentation. It augments each original output (but not input) and produces an arbitrary number of outputs which average to the original output. Updating the parameters is done by using the gradient over both the original and the augmented outputs. We conclude that the proposed novel method strongly complements the existing ones by showing empirical evaluations where we see improvements of the generalization performance in the task of image classification.

1 Introduction

In recent years, researches using Convolutional Neural Networks (CNN) for image data have succeeded in achieving better performance than ever before in a variety of tasks. It is also known that huge amount of training data is required to achieve high performance using CNN, though it is also true that there are many tasks for which it is not possible to obtain huge amount of training data. For this, a method called data augmentation is commonly used. Based on knowledge of the domain in which the data are lying, data augmentation enables us to make up for a lack of the training data. Furthermore, it allows us to improve the generalization performance on test data by increasing variations of the training data even when the dataset is huge.

We propose a new approach, which we call Output-Augmentation (OA), to improve the generalization performance without data augmentation. Specifically, each original output from the dataset is augmented, and an arbitrary number of outputs are randomly produced in a way that the mean of them equals to the original output. Given the augmented outputs, our learning process consists of two kinds of gradient descents; After updating the parameters by the gradient descent with respect to the original output, the parameters are further updated by that for the augmented outputs (see Section 3 and Table 1). While data augmentation needs to know the domain of the dataset, our method does not need the knowledge and does improve the generalization performance by adjusting only one hyper-parameter.

In addition, there is a closely related work, Sharpness-Aware Minimization (SAM) (Pierre Foret et al. [1]), which performs multiple gradient calculations for a single input to which we address later. In this paper, we demonstrate effectiveness of OA by using image classification for CIFAR-10 and CIFAR-100, which are widely used in the field of computer vision. Specifically, we conducted experiments using models without data augmentation, with data augmentation (AutoAugment (Ekin Dogus Cubuk et al. [2])), with SAM, and with OA. We showed, through rigorous empirical studies, that the proposed method improves the generalization performance in image classification, and obtained the following results.

- The proposed method significantly improves the test accuracy of the model without using data augmentation.
- It is interesting to note that the proposed method is effective for models without data augmentation, but not so much for models with data augmentation.

The structure of this paper is organized as follows. Section 2 gives an overview of related works (data augmentation, SAM). Section 3 is devoted to describe the parameter updating procedure in the OA method. Section 4 presents empirical results of OA. Finally, conclusions and future work are summarized in Section 5.

Table 1. Our approach: The case of K augmentations for each output.

Number of the parameters updating	Input	Output	Parameter updating procedure
1	x	$h^{(L)}(u^{(L)})$	$w_{t+1} = w_t - \eta\nabla_w\mathcal{L}\vert_{w=w_t}$
2	None	$h^{(L)}(u^{(L)}+\varepsilon_1)$	$w_{t+1+1} = w_{t+1} - \eta\nabla_w\mathcal{L}_{\mathrm{aug}_1}\vert_{w=w_{t+1}}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
1+K	None	$h^{(L)}(u^{(L)}+\varepsilon_K)$	$w_{t+1+K} = w_{t+K} - \eta\nabla_w\mathcal{L}_{\mathrm{aug}_K}\vert_{w=w_{t+K}}$

2 Related Work

Data augmentation Data augmentation method uses rotations, reflections and scalings to operate effects of geometric distortions and deformations of images. This method can be used to increase the number of training data (A. Kwasigroch et al. [3]) and to balance the dataset (A. Kwasigroch et al. [4] and Wasowicz et al. [5]). As the result, the method has improved the efficiency of training. The most common applications of this are histogram equalization, contrast or brightness adjustment, white balance, sharpening, and blurring (Galdran et al. [6]). These have been proven to be faster, more reproducible (Galdran et al. [7]). Moreover, there is a very powerful method called AutoAugment (Ekin Dogus Cubuk et al. [2]) that automatically selects a suitable method from among those introduced so far, and performs the method.

Sharpness-Aware Minimization (SAM) SAM (Pierre Foret et al. [1]) is a new method that improves the generalization of the model by simultaneously minimizing the loss value and the sharpness of the loss by using the following objective function. We denote the models parameters w.

$$\min_w \mathcal{L}_S^{\mathrm{SAM}}(w) + \lambda\|w\|_2^2 = \min_w \max_{\|\varepsilon\|_p \le \rho} \mathcal{L}_S(w+\varepsilon) + \lambda\|w\|_2^2,$$
$$\text{where } \rho \ge 0,\ p \in [1,\infty]\,. \tag{1}$$

They have reported that $p = 2$ of the p-norm and the neighborhood size $\rho = 0.05$ are often satisfactory.

The parameter updating procedure in SAM is performed using two gradient calculations when data is input once. To be more precise, the gradient at a maximum point of the loss function on the neighborhood of the parameter before updating is used in the algorithm of gradient descent, aiming that the loss function is flat around the goal. By using SAM, they updated State of The Art (SoTA) on nine image-classification datasets including ImageNet and CIFAR.

The parameter updating procedure in our approach is similar to SAM, though there are three differences.

- Similarities
  - Given an input, a gradient different to the gradient at the parameter before updating is used.
  - Both performs multiple gradient calculations for parameter updating.
- Differences
  - SAM updates the parameters with using a gradient at a point slightly varied from the parameter before updating, while OA does with using the gradient of augmented output instead of the original one.
  - SAM needs to calculate a maximum point of the loss function on a neighborhood of the parameter before updating, while OA does not; we just need to compute the gradient of the output shifted randomly (augmented output). Hence, SAM needs the knowledge of local behavior of the loss function at each step of the updating, while OA does not.
  - SAM performs two gradient calculations for an input; OA performs by using an arbitrary number of gradient calculations for an input.

3 Approach

In this section, for neural networks, we will look at the differences between the training method proposed in this paper and traditional one.

We consider the following L-layer neural network consisting of an input layer, hidden layers, and an output layer.

Here, the training dataset is $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, the input data is x, the parameters of the l-th layer are $w^{(l)}$, the activation function of the l-th layer is $h^{(l)}$, and the output from the l-th layer is $z^{(l)}$. Then the input to the l-th layer can be written as $u^{(l)} = w^{(l)}z^{(l-1)} = w^{(l)}h^{(l-1)}(u^{(l-1)})$. Since this method can be used for all loss functions, we will use the mean square error in this chapter for simplicity.

3.1 Updating Parameters with Using Original Output

The output $\hat{y}$ of the network

By using the input $u^{(L)}$ to the L-th layer (the output-layer), the output $\hat{y}$ for the input data x is expressed as

$$\begin{aligned}\hat{y} &= h^{(L)}(u^{(L)}) \\ &= h^{(L)}(w^{(L)}h^{(L-1)}(w^{(L-1)}h^{(L-2)} \\ &\quad (\cdots w^{(l+1)}h^{(l)}(w^l h^{(l-1)}(\cdots h^{(1)}(x)))\cdots))).\end{aligned} \tag{2}$$

The loss function $\mathcal{L}$

With using the output $\hat{y}$ of the network and the true class-label y of the input x, we define a loss function $\mathcal{L}$ by

$$\mathcal{L} = \frac{1}{2}\|\hat{y} - y\|_2^2 = \frac{1}{2}\left\|h^{(L)}(u^{(L)})\ - y\right\|_2^2. \tag{3}$$

Updating the parameters

$$w_{t+1} = w_t - \eta\nabla_w\mathcal{L}(w)|_{w=w_t}$$

The parameter w_{t+1} is obtained by updating w_t with using the gradient $\nabla_w\mathcal{L}(w)$ and the learning rate η. Then $\nabla_w\mathcal{L}(w)$ is calculated as

$$\nabla_w\mathcal{L}(w) = \frac{\partial\mathcal{L}}{\partial w^{(l)}} = \frac{\partial\mathcal{L}}{\partial u^{(l)}}\frac{\partial u^{(l)}}{\partial w^{(l)}} = \delta^{(l)}z^{(l-1)}, \tag{4}$$

where $\delta^{(l)} := \frac{\partial \mathcal{L}}{\partial u^{(l)}} = \frac{\partial \mathcal{L}}{\partial u^{(l+1)}} \frac{\partial u^{(l+1)}}{\partial u^{(l)}}$. Then the value of $\delta^{(l)}$ at the j-th unit in the output-layer is

$$\delta_j^{(L)} = \frac{\partial \mathcal{L}}{\partial u_j^{(L)}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_j} \frac{\partial h^{(L)}(u_j^{(L)})}{\partial u_j^{(L)}}. \quad (5)$$

Traditional method of learning, such as Stochastic Gradient Descent (SGD), updates parameters using gradient (equations 4, 5) for each batch of input data.

3.2 Updating Parameters with OA

The output $\hat{y}_{\text{aug}}$ of the network

By using the input $u^{(L)}$ to the L-th layer (the output-layer), the output $\hat{y}_{\text{aug}}$ is obtained from the original output $\hat{y}$. The output $\hat{y}_{\text{aug}}$ for the input x is defined as

$$\begin{aligned}\hat{y}_{\text{aug}} &= h^{(L)}(u^{(L)} + \varepsilon),\ |\varepsilon| \ll 1 \\ &= h^{(L)}(w^{(L)} h^{(L-1)}(w^{(L-1)} h^{(L-2)} \\ &(\cdots w^{(l+1)} h^{(l)}(w^{l} h^{(l-1)}(\cdots h^{(1)}(x))) \cdots)) + \varepsilon). \quad (6)\end{aligned}$$

In this study, we use the normal and uniform distributions, which are simple method to add the noise ε.

The loss function $\mathcal{L}_{\text{aug}}$

By using the true class-label y of the input x, we define the loss function $\mathcal{L}_{\text{aug}}$ as

$$\mathcal{L}_{\text{aug}} = \frac{1}{2} \left\| \hat{y}_{\text{aug}} - y \right\|_2^2 = \frac{1}{2} \left\| h^{(L)}(u^{(L)} + \varepsilon) - y \right\|_2^2. \quad (7)$$

Updating the parameters

$w_{t+1} = w_t - \eta \nabla_w \mathcal{L}_{\text{aug}}(w)|_{w=w_t}$

The parameter w_{t+1} is obtained by updating w_t with using the gradient $\nabla_w \mathcal{L}_{\text{aug}}(w)$ and the learning rate η. Then $\nabla_w \mathcal{L}_{\text{aug}}(w)$ is calculated as follows:

$$\nabla_w \mathcal{L}_{\text{aug}}(w) = \frac{\partial \mathcal{L}_{\text{aug}}}{\partial w^{(l)}} = \frac{\partial \mathcal{L}_{\text{aug}}}{\partial u^{(l)}} \frac{\partial u^{(l)}}{\partial w^{(l)}} = \delta_{\text{aug}}^{(l)} z^{(l-1)}, \quad (8)$$

where $\delta_{\text{aug}}^{(l)} := \frac{\partial \mathcal{L}_{\text{aug}}}{\partial u^{(l)}} = \frac{\partial \mathcal{L}_{\text{aug}}}{\partial u^{(l+1)}} \frac{\partial u^{(l+1)}}{\partial u^{(l)}}$. Then the value of $\delta_{\text{aug},j}^{(L)}$ at the j-th unit in the output-layer is

$$\delta_{\text{aug},j}^{(L)} = \frac{\partial \mathcal{L}_{\text{aug}}}{\partial u_j^{(L)}} = \frac{\partial \mathcal{L}_{\text{aug}}}{\partial \hat{y}_j} \frac{\partial h^{(L)}(u_j^{(L)} + \varepsilon_j)}{\partial u_j^{(L)}}. \quad (9)$$

The proposed training method for neural networks uses both traditional parameter updating (subsection 3.1) and parameter updating using output augmentation (subsection 3.2). Specifically, when a batch size of data is input, parameters are updated according to the algorithm described in subsection 3.1, and then they are further updated along the procedure subsection 3.2 with using an arbitrary number of augmented outputs without further data input. This procedures continue until the parameters converge.

The algorithm of the traditional method of updating parameters (Algorithm 1) and the proposed method of updating parameters (Algorithm 2) can be summarized in the following figures.

```
Input: Training dataset $\mathcal{D} = \{(x_i, y_i)\}$, Loss function
    $\mathcal{L}$, Batch size $\mathcal{B}$, Learning rate $\eta > 0$.
Output: Model trained
 1: Initialize weights $w_0, t = 0$;
 2: for $t = 1$ to last-epoch do
 3:     Sample batch $\mathcal{B} = \{(x_1, y_1), ...(x_b, y_b)\}$;
 4:     Compute a gradient $\nabla_w \mathcal{L}_{\mathcal{B}}(w)$;
 5:     Update weights: $w_{t+1} = w_t - \eta \nabla_w \mathcal{L}_{\mathcal{B}}(w)|_{w=w_t}$;
 6: end for
```

Algorithm 1. Traditional algorithm (SGD)

```
Input: Training dataset $\mathcal{D} = \{(x_i, y_i)\}$, Loss function
    $\mathcal{L}$, Batch size $\mathcal{B}$, Learning rate $\eta > 0$.
Output: Model trained with Output Augmentation
 1: Initialize weights $w_0, t = 0$;
 2: for $t = 1$ to last-epoch do
 3:     Sample batch $\mathcal{B} = \{(x_1, y_1), ...(x_b, y_b)\}$;
 4:     Compute a gradient $\nabla_w \mathcal{L}_{\mathcal{B}}(w)$ of the batch ' s
        training loss;
 5:     Update weights: $w_{t+1} = w_t - \eta \nabla_w \mathcal{L}_{\mathcal{B}}(w)|_{w=w_t}$;
 6:     for $k = 1$ to $K$ do
 7:         Compute gradient $\nabla_w \mathcal{L}_{\mathcal{B},\text{aug}_k}(w)$;
 8:         Update weights:
            $w_{(t+1)+k} = w_{t+k} - \eta \nabla_w \mathcal{L}_{\mathcal{B},\text{aug}_k}(w)|_{w=w_{t+k}}$;
 9:     end for
10: end for
```

Algorithm 2. Output-Augmentation: In the case of training by using K augmentations of outputs, the parameters are updated $(1 + K)$ times for a single input.

4 Empirical Evaluation

In this section, we show results of experiments. In the experiments, we use a variety of methods on the model ResNet-18 (Kaiming He et al. [8]) applied to image classification for CIFAR-10 and CIFAR-100.

In order to confirm the effectiveness of the proposed OA, we compare two situations where one is without data augmentation and the other is with that. Specifically, "Basic": ResNet-18 without data augmentation and "AA": ResNet-18 with AutoAugment. The methods to be applied are SGD, SAM and OA. In addition, all experiments are performed under the same number of data and the same learning rate $\eta = 0.01$.

Since the number of parameter-updating for each mini-batch was different for each method, the number of epochs was adjusted so that the number of parameter-updating would be the same. We compare the highest accuracy for the test data up to the last epoch. The following table 2 shows the experimental results on image classification for CIFAR-10 and CIFAR-100.

Table 2. Test accuracy of image classification for CIFAR-10 and CIFAR-100: the exponent parameter p in the p-norm and the neighborhood size ρ are hyper-parameters of SAM (equation 1). We generate K augmentations of each output (Algorithm 2). I is the identity matrix, $\mathbf{e}$ is the unit vector, $\mathcal{N}(\ \cdot\ )$ is the normal distribution, and $\mathcal{U}[\ \cdot\ )$ is the uniform distribution. All training data are input at each epoch. At each epoch, the total numbers of parameter-updating are the same between SGD and SAM, but they don't coincide with that of OA; 'epoch' is not 'iteration' (the number of updating parameters).

Method	CIFAR-10		CIFAR-100	
	Basic	AA	Basic	AA
SGD (300epochs)	0.7765	0.8848	0.4697	0.6360
SAM (300epochs): $p = 2,\ \rho = 0.05$	**0.8509**	**0.9156**	0.5478	**0.6950**
OA (100epochs): $K = 2,\ \varepsilon \sim \mathcal{N}(0, I)$	0.8214	0.8789	0.5287	0.6289
OA (100epochs): $K = 2,\ \varepsilon \sim \mathcal{N}(0, 3I^2)$	0.8411	0.8828	0.5509	0.6330
OA (100epochs): $K = 2,\ \varepsilon \sim \mathcal{U}[-\mathbf{e}, \mathbf{e})$	0.8147	0.8880	0.5328	0.6277
OA (100epochs): $K = 2,\ \varepsilon \sim \mathcal{U}[-3\mathbf{e}, 3\mathbf{e})$	0.8311	0.8922	**0.5799**	0.6391
SGD (600epochs)	0.7765	0.8921	0.4697	0.6490
SAM (600epochs): $p = 2,\ \rho = 0.05$	0.8597	**0.9268**	0.5537	**0.7126**
OA (100epochs): $K = 5,\ \varepsilon \sim \mathcal{N}(0, I^2)$	0.8643	0.8891	0.5990	0.6372
OA (100epochs): $K = 5,\ \varepsilon \sim \mathcal{N}(0, 3I^2)$	**0.8750**	0.8965	**0.6209**	0.6512
OA (100epochs): $K = 5,\ \varepsilon \sim \mathcal{U}[-\mathbf{e}, \mathbf{e})$	0.8604	0.8933	0.5980	0.6391
OA (100epochs): $K = 5,\ \varepsilon \sim \mathcal{U}[-3\mathbf{e}, 3\mathbf{e})$	0.8110	0.8858	0.6013	0.6400
SGD (1100epochs)	0.7765	0.8936	0.4697	0.6501
SAM (1100epochs): $p = 2,\ \rho = 0.05$	**0.8708**	**0.9352**	0.5585	**0.7259**
OA (100epochs): $K = 10,\ \varepsilon \sim \mathcal{N}(0, I^2)$	0.8009	0.8769	0.5416	0.6338
OA (100epochs): $K = 10,\ \varepsilon \sim \mathcal{N}(0, 3I^2)$	0.8621	0.8929	**0.5946**	0.6454
OA (100epochs): $K = 10,\ \varepsilon \sim \mathcal{U}[-\mathbf{e}, \mathbf{e})$	0.8010	0.8694	0.4842	0.6167
OA (100epochs): $K = 10,\ \varepsilon \sim \mathcal{U}[-3\mathbf{e}, 3\mathbf{e})$	0.8110	0.8858	0.5106	0.6266

For both CIFAR-10 and CIFAR-100 datasets, the proposed method showed a significant improvement in test accuracy without using data augmentation. On the other hand, in the case where data augmentation is also used, the test accuracy is not so bad for the proposed method, but the SAM is even better. It is interesting that OA improves the test accuracy significantly without data augmentation, while it does not much if we perform data augmentation. There is a possibility that the use of AutoAugment for the input data shifted it too far, as the shifted output for the shifted input shifted it further.

It is also worth mentioning that the test accuracy for AA with SGD is close to that of Basic with OA, which suggests us that OA can be used as a substitute for data augmentation if the hyper-parameters ε's of OA are well chosen. As a reason for the success of our method, it is natural to assume that OA corresponds to the shift of input by data augmentation.

5 Conclusions

In this paper, we proposed a novel method Output-Augmentation (OA) to improve the generalization performance on test dataset. Traditional data augmentation for images requires information about the domain where the training data are lying, but not for OA.

In image classification for both CIFAR-10 and CIFAR-100 datasets, we showed that the proposed method significantly improves the test accuracy without data augmentation. On other hand, it is interesting that the our method does not improve the test accuracy much when data augmentation is also used. We expect that OA can be used as a substitute for traditional data augmentation.

References

[1] Pierre Foret and Ariel Kleiner and Hossein Mobahi and Behnam Neyshabur, “Sharpness-aware Minimization for Efficiently Improving Generalization,” International Conference on Learning Representations (ICLR), 2021.

[2] Ekin Dogus Cubuk and Barret Zoph and Dandelion Mané and Vijay Vasudevan and Quoc V. Le, “AutoAugment: Learning Augmentation Strategies from Data,” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[3] A. Kwasigroch, A. Mikołajczyk, and M. Grochowski, “Deep convolutional neural networks as a decision support tool in medical problems-malignant melano ma case study,” In: Mitkowski W., Kacprzyk J., Oprzędkiewicz K., Skruch P. (eds) Trends in Advanced Intelligent Control, Optimization and Automation. KKA 2017. Advances in Intelligent Systems and Computing, vol 577. Springer, Cham, 2017, pp.848-856.

[4] A. Kwasigroch, A. Mikołajczyk and M. Grochowski, “Deep neural networks approach to skin lesions classification A comparative analysis,” 2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, 2017, pp.1069-1074.

[5] Wąsowicz, M., Grochowski, M., Kulka, M. , Mikołajczyk, A., Ficek, M., Karpieńko, K., & Ciśkiewicz, M., “Computed aided system for separation and classification of the abnormal erythrocytes in human blood,” in Bio photonics-Riga 2017, 2017, vol.10592, p.105920A.

[6] Adrian Galdran and Aitor Alvarez-Gila and Maria Inês Meyer and Cristina López Saratxaga and Teresa Araujo and Estíbaliz Garrote and Guilherme Aresta and Pedro Costa and Ana Maria Mendonça and Aurélio J. C. Campilho, “Data-Driven Color Augmentation Techniques for Deep Skin Image Analysis,” ArXiv Prepr. ArXiv170303702, 2017.

[7] J. Wang and L. Perez, “The effectiveness of data augmentation in image classification using deep learning,” Technical report, 2017.

[8] Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun, “Deep Residual Learning for Image Recognition,” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2016.