MetaAugment: Sample-Aware Data Augmentation Policy Learning

Fengwei Zhou*, Jiawei Li*, Chuanlong Xie*, Fei Chen, Lanqing Hong, Rui Sun, Zhenguo Li†
Huawei Noah's Ark Lab
{zhoufengwei, li.jiawei, xie.chuanlong, chen.f, honglanqing, sun.rui3, li.zhenguo}@huawei.com

Abstract

Automated data augmentation has shown superior performance in image recognition. Existing works search for dataset-level augmentation policies without considering individual sample variations, which are likely to be sub-optimal. On the other hand, learning different policies for different samples naively could greatly increase the computing cost. In this paper, we learn a sample-aware data augmentation policy efficiently by formulating it as a sample reweighting problem. Specifically, an augmentation policy network takes a transformation and the corresponding augmented image as inputs, and outputs a weight to adjust the augmented image loss computed by a task network. At training stage, the task network minimizes the weighted losses of augmented training images, while the policy network minimizes the loss of the task network on a validation set via meta-learning. We theoretically prove the convergence of the training procedure and further derive the exact convergence rate. Superior performance is achieved on widely-used benchmarks including CIFAR-10/100, Omniglot, and ImageNet.

Introduction

Data augmentation is widely used to increase the diversity of training data in order to improve model generalization (Krizhevsky, Sutskever, and Hinton 2012; Srivastava, Greff, and Schmidhuber 2015; Han, Kim, and Kim 2017; DeVries and Taylor 2017; Zhang et al. 2017; Yun et al. 2019). Automated data augmentation that searches for data-driven augmentation policies improves the performance of deep models in image recognition compared with the manually designed ones. A data augmentation policy is a distribution of transformations, according to which training samples are augmented. Reinforcement learning (Cubuk et al. 2019a; Zhang et al. 2020), population-based training (Ho et al. 2019), and Bayesian optimization (Lim et al. 2019) have been employed to learn augmentation policies from target datasets. Despite the difference of search algorithms, these approaches search for policies at the dataset level, i.e., all samples in the dataset are augmented with the same policy. For an image recognition task, left translation may be suitable for the image where the target object is on the right, but may not be suitable for the image where the target object is on the left (see Figure 4). According to this observation, dataset-level polices may give rise to various noises such as noisy labels, misalignment, or image distortion, since different samples vary greatly in object scale, position, color, illumination, etc.

To increase data diversity while avoiding noises, it is appealing to learn a sample-aware data augmentation policy, i.e., learning different distributions of transformations for different samples. However, it is time-consuming to evaluate a large number of distributions and non-trivial to determine the relation among the distributions. Augmenting training samples with the corresponding policies, we consider the augmented sample loss as a random variable and train a task network to minimize the expectation of the augmented sample loss. From this perspective, learning a sample-aware policy can be regarded as reweighting the augmented sample losses and the computing cost can be greatly reduced.

In this paper, we propose an efficient method, called MetaAugment, to learn a sample-aware data augmentation policy by formulating it as a sample reweighting problem. An overview of the proposed method is illustrated in Figure 1. Given a transformation and the corresponding augmented image feature, extracted by a task network, an augmentation policy network outputs the weight of the augmented image loss. The task network is optimized by minimizing the weighted training loss, while the goal of the policy network is to improve the performance of the task network on a validation set via adjusting the weights of the losses. This is a bilevel optimization problem (Colson, Marcotte, and Savard 2007) which is hard to be optimized. We leverage the mechanism of meta-learning (Finn, Abbeel, and Levine 2017; Li et al. 2017; Ren et al. 2018; Wu et al. 2018; Liu, Simonyan, and Yang 2019; Shu et al. 2019) to solve this problem. The motivation is based on the ability of meta-learning to extract useful knowledge from related tasks. During training, classification for each batch of samples is treated as a task. The policy network acts as a meta-learner to adapt the task network with the augmented samples such that it can perform well on a batch of validation samples. Instead of learning an initialization for fast adaptation in downstream tasks, the policy network learns to augment while guiding the actual training process of the task network. We also propose a novel transformation sampler that samples transformations accord-

*Equal Contribution
†Corresponding Author

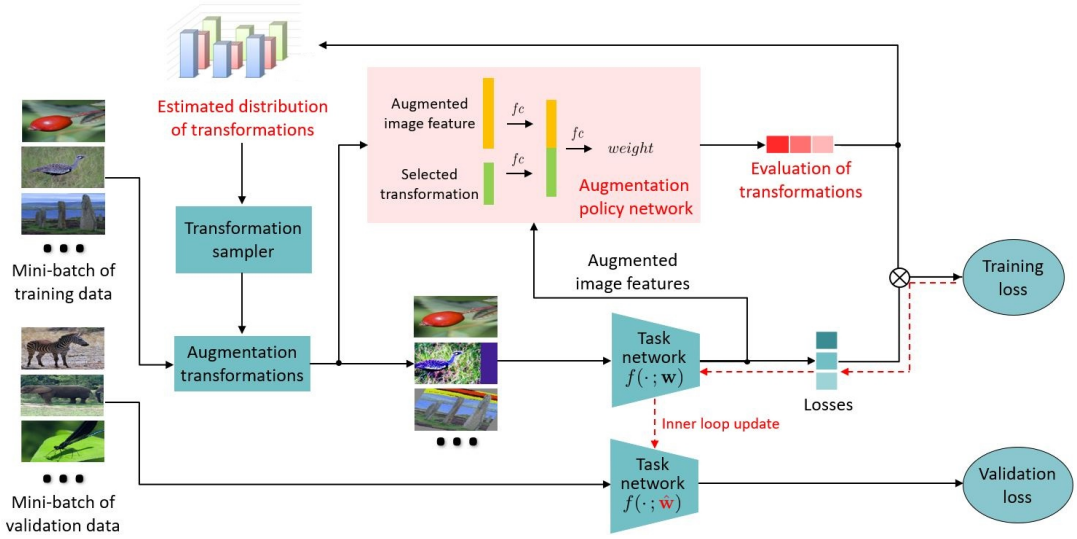


Figure 1: An overview of the proposed MetaAugment. The augmentation policy network outputs the weights of the augmented image losses and is learned to evaluate the effectiveness of different transformations for different training images via meta-learning, while the task network is trained to minimize the weighted training loss alternately with the updating of the policy network. For higher training efficiency, the transformation sampler samples transformations according to a distribution refined with the training process of the policy network.

ing to a distribution estimated by the outputs of the policy network. In principle, the distribution reflects the overall effectiveness of the transformations for the whole dataset and the transformation sampler can avoid invalid ones to improve the training efficiency. Furthermore, we theoretically show the convergence guarantee of our algorithm.

Our main contributions can be summarized as follows:

1) We propose MetaAugment to learn a sample-aware augmentation policy network that captures the variability of training samples and evaluates the effectiveness of transformations for different samples.

2) We systematically investigate the convergence properties under two cases: (i) the policy network has its own feature extractor; (ii) the policy network depends on the parameters of the task network. We also point out the exact convergence rate and the optimization bias of our algorithm.

3) Extensive experimental results show that our method consistently improves the performance of various deep networks and outperforms previous automated data augmentation methods on CIFAR-10/100, Omniglot, and ImageNet.

Related Work

Automated Data Augmentation. There are rich studies on data augmentation in the past few decades, while automated data augmentation is a relatively new topic. Inspired by neural architecture search, AutoAugment (Cubuk et al. 2019a) adopts reinforcement learning to train a controller to generate augmentation policies such that a task network trained along with the policies may have the highest validation accuracy. Adversarial AutoAugment (Zhang et al. 2020) trains a controller to generate adversarial augmentation policies that increase the training loss of a task network. Inspired by hyper-parameter optimization, PBA (Ho et al. 2019) learns an epoch-aware augmentation schedule instead of a fixed policy for all training epochs. Following Bayesian optimization, FAA (Lim et al. 2019) searches for policies that match the distribution of augmented data with that of unaugmented data. DADA (Li et al. 2020) proposes to relax the discrete selection of augmentation policies to be differentiable and uses gradient-based optimization to do policy search. These methods overlook the variability of training samples and adopt the same policy for all samples. RandAugment (Cubuk et al. 2019b) shows that hyper-parameters in such policies do not affect the results a lot. Our method learns a sample-aware policy network that associates different pairs of transformations and augmented samples with different weights.

Sample Reweighting. There are many studies on sample reweighting for specific issues, e.g., class imbalance (Johnson and Khoshgoftaar 2019) and label noise (Zhang and Sabuncu 2018). Among them, there are mainly two types of weighting functions. The first one, suitable for class imbalance, is to increase the weights of hard samples (Freund and Schapire 1995; Johnson and Khoshgoftaar 2019; Malisiewicz, Gupta, and Efros 2011; Lin et al. 2017), while the second one, suitable for noise label, is to increase the weights of easy samples (Kumar, Packer, and Koller 2010; Jiang et al. 2014a,b; Zhang and Sabuncu 2018). Instead of manually designing the weight functions, Ren et al. (2018) propose an online reweighting method that learns sample weights directly from data via meta-learning. Meta-Weight-Net (Shu et al. 2019) adopts a neural network to learn the mapping from sample loss to sample weight, which stabilize the weighting behavior. Wang et al. (2019) train a scorer network to up-weight training data that have similar loss gradients with validation data via reinforcement learning. Different from these works, our policy network aims to evaluate different transformations for different samples and assign weights to augmented samples.

Methodology

Sample-Aware Data Augmentation

Consider an image recognition task with the training set $\mathcal{D}^{tr} = \{(x_i, y_i)\}_{i=1}^{N^{tr}}$, where y_i is the label of the image x_i, and N^{tr} is the sample size. Training samples are augmented by various transformations. Each transformation consists of two image processing functions, such as rotation, translation, coloring, etc., to be applied in sequence. Each function is associated with a magnitude that is rescaled to and sampled uniformly from $[0, 10]$. Given K image processing functions in order, let $\mathcal{T}_{j,k}^{m_1,m_2}(x_i)$ be a transformation applied on an image x_i with j-th and k-th functions in order and the magnitudes are m_1 and m_2, respectively.

Intuitively, not all of the augmented samples may help to improve the performance of a task network, and thus, an augmentation policy network is proposed to learn the effectiveness of different transformations for different training samples. Let $f(x_i; \mathbf{w})$ be the task network with parameters $\mathbf{w}$. By abuse of notation, the deep feature of x_i extracted by the task network is also denoted by $f(x_i; \mathbf{w})$. For each pair of augmented sample feature $f(\mathcal{T}_{j,k}^{m_1,m_2}(x_i); \mathbf{w})$ and the embedding of the applied transformation $e(\mathcal{T}_{j,k}^{m_1,m_2})$, the policy network $P(\cdot, \cdot; \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$ takes the pair as input and outputs a weight that is imposed on the augmented sample loss $L_{i,j,k}(m_1, m_2; \mathbf{w}) = \ell(f(\mathcal{T}_{j,k}^{m_1,m_2}(x_i); \mathbf{w}), y_i)$. The task network is trained to minimize the following weighted training loss:

$$\mathcal{L}^{tr}(\mathbf{w}, \boldsymbol{\theta}) = \frac{1}{N^{tr}} \sum_{i=1}^{N^{tr}} \frac{1}{K^2} \sum_{j,k=1}^{K} \mathbb{E}_{m_1,m_2 \sim U(0,10)} \Big[P_{i,j,k}(m_1, m_2; \mathbf{w}, \boldsymbol{\theta}) L_{i,j,k}(m_1, m_2; \mathbf{w}) \Big],$$

where

$$P_{i,j,k}(m_1, m_2; \mathbf{w}, \boldsymbol{\theta}) = P(f(\mathcal{T}_{j,k}^{m_1,m_2}(x_i); \mathbf{w}), e(\mathcal{T}_{j,k}^{m_1,m_2}); \boldsymbol{\theta})$$

and $U(0, 10)$ denotes the uniform distribution over $[0, 10]$. The objective of the policy network is to output the accurate sample weights such that the task network has the best performance on a validation set $\mathcal{D}^{val} = \{(x_{i'}^{val}, y_{i'}^{val})\}_{i'=1}^{N^{val}}$ via minimizing $\mathcal{L}^{tr}(\mathbf{w}, \boldsymbol{\theta})$. Mathematically, we formulate the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & \mathcal{L}^{val}(\mathbf{w}^*(\boldsymbol{\theta})) = \frac{1}{N^{val}} \sum_{i'=1}^{N^{val}} L_{i'}^{val}(\mathbf{w}^*(\boldsymbol{\theta})) \\ \text{subject to} \quad & \mathbf{w}^*(\boldsymbol{\theta}) = \arg\min_{\mathbf{w}} \mathcal{L}^{tr}(\mathbf{w}, \boldsymbol{\theta}), \end{aligned} \tag{1}$$

where $L_{i'}^{val}(\mathbf{w}^*(\boldsymbol{\theta})) = \ell(f(x_{i'}^{val}; \mathbf{w}^*(\boldsymbol{\theta})), y_{i'}^{val})$. This is a bilevel optimization problem (Colson, Marcotte, and Savard 2007), which is hard to solve since as the updating of $\boldsymbol{\theta}$, the parameters of the task network are required to be optimized accordingly. Recent works (Ren et al. 2018; Wu et al. 2018; Liu, Simonyan, and Yang 2019; Shu et al. 2019) use meta-learning techniques to get approximate optimal solutions for such bilevel optimization problems. We also leverage meta-learning and employ the updating rules proposed in (Shu et al. 2019; Li et al. 2017; Antoniou, Edwards, and Storkey 2019) to solve problem (1).

Proposed MetaAugment Algorithm

The policy and task networks are trained alternately. For each iteration, a mini-batch of training data $\mathcal{D}_{mi}^{tr} = \{(x_i, y_i)\}_{i=1}^{n^{tr}}$ with batch size n^{tr} is sampled and for each x_i, a transformation $\mathcal{T}_{j_i,k_i}^{m_1,m_2}$ is sampled to augment x_i. For notation simplicity, let $P_i(\mathbf{w}, \boldsymbol{\theta}) = P(f(\mathcal{T}_{j_i,k_i}^{m_1,m_2}(x_i); \mathbf{w}), e(\mathcal{T}_{j_i,k_i}^{m_1,m_2}); \boldsymbol{\theta})$ and $L_i(\mathbf{w}) = \ell(f(\mathcal{T}_{j_i,k_i}^{m_1,m_2}(x_i); \mathbf{w}), y_i)$. Then the inner loop update of $\mathbf{w}$ in iteration $t+1$ is

$$\hat{\mathbf{w}}^{(t)}(\boldsymbol{\theta}, \alpha) = \mathbf{w}^{(t)} - \alpha \frac{1}{n^{tr}} \sum_{i=1}^{n^{tr}} P_i(\mathbf{w}^{(t)}, \boldsymbol{\theta}) \nabla_{\mathbf{w}} L_i(\mathbf{w}^{(t)}), \tag{2}$$

where α is a learnable learning rate (Li et al. 2017; Antoniou, Edwards, and Storkey 2019) and $\nabla_{\mathbf{w}} L_i(\mathbf{w}^{(t)}) = \nabla_{\mathbf{w}} L_i(\mathbf{w})\big|_{\mathbf{w}^{(t)}}$. We adopt a learnable α because it is unclear how to set the learning rate schedule manually for this inner loop update and proper schedules may vary for different training datasets. We regard $P_i(\mathbf{w}, \boldsymbol{\theta})$ as a function of $\boldsymbol{\theta}$ and do not take derivative of $P_i(\mathbf{w}, \boldsymbol{\theta})$ with respect to $\mathbf{w}$ in Eq. (2). This is because $P_i(\mathbf{w}, \boldsymbol{\theta})$ shall be fixed when updating $\mathbf{w}$ and the weighted training loss shall not be minimized via minimizing $P_i(\mathbf{w}, \boldsymbol{\theta})$. It can also avoid a second-order derivative when updating the policy network, which otherwise will substantially increase the computational complexity.

The formulation $\hat{\mathbf{w}}^{(t)}(\boldsymbol{\theta}, \alpha)$ is regarded as a function of $\boldsymbol{\theta}$ and α, and then $\boldsymbol{\theta}$ and α can be updated via the validation loss computed by $\hat{\mathbf{w}}^{(t)}(\boldsymbol{\theta}, \alpha)$ on a mini-batch of validation samples $\mathcal{D}_{mi}^{val} = \{(x_{i'}^{val}, y_{i'}^{val})\}_{i'=1}^{n^{val}}$ with batch size n^{val}. The outer loop updates of $\boldsymbol{\theta}$ and α are formulated by

$$\begin{aligned} (\boldsymbol{\theta}^{(t+1)}, \alpha^{(t+1)}) = (\boldsymbol{\theta}^{(t)}, \alpha^{(t)}) \\ - \beta \frac{1}{n^{val}} \sum_{i'=1}^{n^{val}} \nabla_{(\boldsymbol{\theta},\alpha)} L_{i'}^{val}(\hat{\mathbf{w}}^{(t)}(\boldsymbol{\theta}^{(t)}, \alpha^{(t)})), \end{aligned} \tag{3}$$

where β is a learning rate and $\nabla_{(\boldsymbol{\theta},\alpha)} L_{i'}^{val}(\hat{\mathbf{w}}^{(t)}(\boldsymbol{\theta}^{(t)}, \alpha^{(t)}))$ $= \nabla_{(\boldsymbol{\theta},\alpha)} L_{i'}^{val}(\hat{\mathbf{w}}^{(t)}(\boldsymbol{\theta}, \alpha))\big|_{(\boldsymbol{\theta}^{(t)}, \alpha^{(t)})}$. The third step in iteration $t+1$ is the outer loop update of $\mathbf{w}^{(t)}$ with the updated $\boldsymbol{\theta}^{(t+1)}$:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \gamma \frac{1}{n^{tr}} \sum_{i=1}^{n^{tr}} P_i(\mathbf{w}^{(t)}, \boldsymbol{\theta}^{(t+1)}) \nabla_{\mathbf{w}} L_i(\mathbf{w}^{(t)}), \tag{4}$$

where γ is a learning rate. With these updating rules, the two networks can be trained efficiently.

Although the policy network outputs the weights that evaluate the importance of the augmented samples, sampling invalid transformations constantly may lead to poor training efficiency. We propose a novel transformation sampler that sample transformations according to a probability distribution estimated by the outputs of the policy network and refined with the training process of the policy network. Specifically, let $\{P(f(\mathcal{T}_{j_i,k_i}^{m_1,m_2}(x_i); \mathbf{w}), e(\mathcal{T}_{j_i,k_i}^{m_1,m_2}); \boldsymbol{\theta})\}_{i=1}^{r \cdot n^{tr}}$ denote the collection of the policy network outputs in the last r iterations. Then the average value of the outputs corresponding to the transformation with j-th and k-th functions in order (without magnitude) is estimated by

$$v_{j,k} = \frac{1}{c_{j,k}} \sum_{i=1}^{r \cdot n^{tr}} \sum_{j_i=j,k_i=k} P(f(\mathcal{T}_{j_i,k_i}^{m_1,m_2}(x_i); \mathbf{w}), e(\mathcal{T}_{j_i,k_i}^{m_1,m_2}); \boldsymbol{\theta}),$$

Algorithm 1 MetaAugment: Sample-Aware Data Augmentation Policy Learning

Input: Training data $\mathcal{D}^{tr}$, validation data $\mathcal{D}^{val}$, K image processing functions, batch sizes n^{tr}, n^{val}, learning rate β, γ, sampler hyper-parameters r, s, ϵ, iteration number T
Output: $\mathbf{w}^{(T)}$, $\boldsymbol{\theta}^{(T)}$, $\{p_{j,k}\}_{j,k=1}^{K}$

Initialize $\mathbf{w}^{(0)}$, $\boldsymbol{\theta}^{(0)}$, $\alpha^{(0)}$, $\{p_{j,k}=\frac{1}{K^2}\}_{j,k=1}^{K}$;
for $0 \le t \le T-1$ **do**
  Sample a mini-batch of training samples $\mathcal{D}_{mi}^{tr}$ with batch size n^{tr};
  For each sample in the mini-batch, sample a transformation according to $p_{j,k}$ and the corresponding magnitudes uniformly from $[0, 10]$;
  Augment the batch data with the sampled transformations;
  Sample a mini-batch of validation samples $\mathcal{D}_{mi}^{val}$ with batch size n^{val};
  Compute $\hat{\mathbf{w}}^{(t)}(\boldsymbol{\theta}, \alpha)$ according to Eq. (2);
  Update $(\boldsymbol{\theta}^{(t+1)}, \alpha^{(t+1)})$ according to Eq. (3);
  Update $\mathbf{w}^{(t+1)}$ according to Eq. (4);
  if $(t+1) \bmod s = 0$ **then**
    Update $p_{j,k}$ according to Eq. (5) with the policy network outputs in the last $\min(t+1, r)$ iterations;
  end if
end for

where $c_{j,k}$ is the number of terms in the summation. In our implementation, the output of the policy network is with the Sigmoid function to ensure the output is positive. To balance exploration and exploitation, and to avoid the biases caused by underfitting of the policy network, the sampler samples each transformation according to the following distribution:

$$p_{j,k} = (1-\epsilon) \cdot \frac{v_{j,k}}{\sum_{l,m=1}^{K} v_{l,m}} + \epsilon \cdot \frac{1}{K^2}, \tag{5}$$

where ϵ is a hyper-parameter, and the corresponding magnitudes are sampled uniformly from $[0, 10]$. The probability $p_{j,k}$ is updated every s iterations. This estimated distribution reflects the overall effectiveness of the transformations for the whole dataset and evolves synergistically with the policy network. Dataset-level and sample-level augmentation policies are combined together by these two modules. The MetaAugment algorithm is summarized in Algorithm 1.

In each iteration, MetaAugment requires three forward and backward passes of the task network, which makes it take $3\times$ training time than a standard training scheme. However, once trained, the policy network, together with the task network and the estimated distribution $\{p_{j,k}\}_{j,k=1}^{K}$ can be transferred to train different networks on the same dataset efficiently. More details are provided in Appendix.

Convergence Analysis

Motivated by Meta-Weight-Net (Shu et al. 2019), we analyze the convergence of the proposed algorithm. In technical details, we release the assumptions of Meta-Weight-Net, e.g. $\sum_{t=1}^{\infty} \beta_t \le \infty$ and $\sum_{t=1}^{\infty} \beta_t^2 \le \infty$, which are invalid in many cases. We find a proper trade-off between the training and validation convergence and exactly point out the convergence rate and the optimization bias. Furthermore, we systematically investigate two situations: (i) the policy network has its own feature extractor; (ii) the policy network depends on the feature extractor of the task network. For the case (i), the convergence is guaranteed on both validation and training data, while for the case (ii), the conclusion on the validation data still holds, but the convergence is not ensured on the training data. However, if the policy network is also a deep network, it will take nearly $4.5\times$ training time than a standard training scheme. Also, with limited validation data, it may overfit and thus make the task network overfit the validation data. Hence, we choose the latter case in our algorithm. We assume α is fixed during training and postpone the proof into Appendix.

Theorem 1. *Suppose that the loss function ℓ has ρ_1-bounded gradients with respect to $\mathbf{w}$ under both (augmented) training data and validation data, ℓ is Lipschitz smooth with constant ρ_2, the policy network P is differential with a δ_1-bounded gradient and twice differential with its Hessian bounded by δ_2 with respect to $\boldsymbol{\theta}$, and the absolute values of P and ℓ are bounded above by C_1 and C_2, respectively. Furthermore, for any iteration $0 \le t \le T-1$, the variance of the weighted training loss (validation loss) gradient on a mini-batch of training (validation) samples is bounded above. Let*

$$\alpha = \frac{c \log T}{T}, \quad \beta = \sqrt{\frac{c' \log \log T}{T}}, \quad \gamma = \frac{c'' \log T}{T},$$

for some positive constants c, c' and c''. The number of iterations T is sufficiently large such that $\alpha\beta\rho_1^2(\alpha\delta_1^2\rho_2 + \delta_2) < 1$ and $\gamma C_1 \rho_2 < 1$. If the policy network has its own feature extractor, we have

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\Big[\big\|\nabla_{\boldsymbol{\theta}} \mathcal{L}^{val}(\hat{\mathbf{w}}^{(t)}(\boldsymbol{\theta}^{(t)}))\big\|^2\Big] \le O\Big(\frac{\log T}{\sqrt{T \log \log T}}\Big), \tag{6}$$

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\Big[\big\|\nabla_{\mathbf{w}} \mathcal{L}^{tr}(\mathbf{w}^{(t)}, \boldsymbol{\theta}^{(t+1)})\big\|^2\Big] = 0. \tag{7}$$

If the policy network uses the feature extractor of the task network, the weights in the training loss will change when $\mathbf{w}$ updates. Since we regard P as a fixed weight when updating $\mathbf{w}$, the weighted training loss at the end of the last iteration is different from the weighted training loss at the beginning of the current iteration. The discontinuity leads to a bias term in the convergence of the weighted training loss.

Theorem 2. *Suppose the assumptions of Theorem 1 hold. Further assume that the policy network P depends on $\mathbf{w}$ and is differential with a $\tilde{\delta}_1$-bounded gradient with respect to $\mathbf{w}$. Then we have that (6) still holds and*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\Big[\big\|\nabla_{\mathbf{w}} \mathcal{L}^{tr}(\mathbf{w}^{(t)}, \boldsymbol{\theta}^{(t+1)})\big\|^2\Big] - 2\rho_1\tilde{\delta}_1 C_1 C_2 \le o(1). \tag{8}$$

According to the proof of Theorem 2, one can find that under certain conditions, (7) can still hold even if the policy network depends on the feature extractor of the task network.

Experimental Results

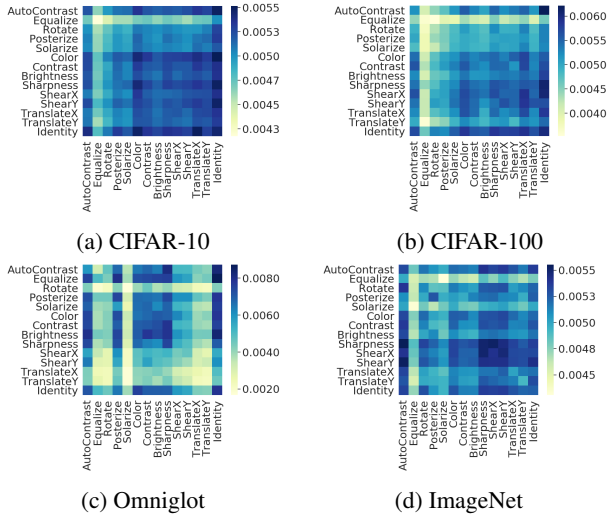
In this section, we evaluate MetaAugment for image recognition tasks on CIFAR-10/100 (Krizhevsky and Hinton 2009), Omniglot (Lake et al. 2011), and ImageNet (Deng et al. 2009).

Dataset	Model	Baseline	AA	FAA	PBA	DADA	RA	AdvAA	MetaAugment
CIFAR-10	WRN-28-10	96.1	97.4	97.3	97.42	97.3	97.3	**98.10**	97.76±0.04
	WRN-40-2	94.7	96.3	96.4	-	96.4	-	-	**96.79±0.06**
	Shake-Shake (26 2x96d)	97.1	98.0	98.0	97.97	98.0	98.0	98.15	**98.29±0.03**
	Shake-Shake (26 2x112d)	97.2	98.1	98.1	97.97	98.0	-	98.22	**98.28±0.01**
	PyramidNet+ShakeDrop	97.3	98.5	98.3	98.54	98.3	98.5	**98.64**	98.57±0.02
CIFAR-100	WRN-28-10	81.2	82.9	82.8	83.27	82.5	83.3	**84.51**	83.79±0.11
	WRN-40-2	74.0	79.3	79.4	-	79.1	-	-	**80.60±0.16**
	Shake-Shake (26 2x96d)	82.9	85.7	85.4	84.69	84.7	-	85.90	**85.97±0.09**
	PyramidNet+ShakeDrop	86.0	89.3	88.3	89.06	88.8	-	**89.58**	89.46±0.11

Table 1: Top-1 test accuracy (%) on CIFAR-10 and CIFAR-100.

Dataset	Model	AdvAA	MetaAugment+MT
CIFAR-10	WRN-28-10	98.10	**98.26±0.02**
CIFAR-100	WRN-28-10	84.51	**85.21±0.09**

Table 2: Top-1 test accuracy (%) on CIFAR using Multiple Transformations (MT) for each sample in a mini-batch.



(a) CIFAR-10 (b) CIFAR-100

(c) Omniglot (d) ImageNet

Figure 2: Estimated distributions of transformations on (a) CIFAR-10, (b) CIFAR-100, (c) Omniglot, and (d) ImageNet.

We show the effectiveness of MetaAugment with different task network architectures and visualize the learned augmentation policies to illustrate the necessity of sample-aware data augmentation.

In our implementation, we use $K = 14$ image processing functions: AutoContrast, Equalize, Rotate, Posterize, Solarize, Color, Contrast, Brightness, Sharpness, ShearX/Y, TranslateX/Y, Identity (Cubuk et al. 2019b,a; Ho et al. 2019; Lim et al. 2019; Zhang et al. 2020). The embedding of a particular transformation $\mathcal{T}_{j,k}^{m_1,m_2}$ is a 28-dimensional vector with $m_1 + 1$ in $(2j-1)$-th position, $m_2 + 1$ in $(2k)$-th position, and 0 elsewhere. For AutoContrast, Equalize, and Identity that do not use magnitude, we let 11 be in their positions. The augmentation policy network is an MLP that takes the embedding of the transformation and the corresponding augmented image feature as inputs, each followed by a fully-connected layer of size 100 with ReLU nonlinearities. The two intermediate features are then concatenated together, followed by a fully-connected output layer of size 1. The Sigmoid function is applied to the output. We also normalize the output weights of training samples in each mini-batch, i.e., each weight is divided by the sum of all weights in the mini-batch. More implementation details and the hyper-parameters we used are provided in Appendix. All of the reported results are averaged over five runs with different random seeds.

Results on CIFAR, Omniglot, and ImageNet

CIFAR. CIFAR-10 and CIFAR-100 consist of 50,000 images for training and 10,000 images for testing. For our method, we hold out 1,000 training images as the validation data. We compare MetaAugment with Baseline, AutoAugment (AA) (Cubuk et al. 2019a), FAA (Lim et al. 2019), PBA (Ho et al. 2019), DADA (Li et al. 2020), RandAugment (RA) (Cubuk et al. 2019b), and Adversarial AutoAugment (AdvAA) (Zhang et al. 2020) on Wide-ResNet (WRN) (Zagoruyko and Komodakis 2016), Shake-Shake (Gastaldi 2017), and PyramidNet+ShakeDrop (Han, Kim, and Kim 2017; Yamada et al. 2018). The Baseline adopts the standard data augmentation: horizontal flipping with 50% probability, zero-padding and random cropping. For MetaAugment, the transformation is applied after horizontal flipping, and then Cutout (DeVries and Taylor 2017) with 16×16 pixels is applied.

The mean test accuracy and Standard Deviation (Std Dev) of MetaAugment, together with the results of other competitors, are reported in Table 1. On both of CIFAR-10 and CIFAR-100, our method outperforms AA, FAA, PBA, DADA, and RA on all of the models. Compared with AdvAA, MetaAugment shows slightly worse results on WRN-28-10 and PyramidNet+ShakeDrop, and better results on Shake-Shake. However, AdvAA trains a task network with a large batch consisting of samples augmented by 8 augmentation policies. The Multiple-Transformation-per-sample (MT) trick leads to better performance but $8\times$ more computing cost than the regular training. We also compare MetaAugment with AdvAA in the MT setting. Each training sample in a mini-batch is augmented by 4 transformations and all the augmented samples are used to train the task network. The results are illustrated in Table 2. It can be seen that MetaAugment out-

Model	Baseline	FAA	PBA	RA	MetaAugment
WRN-28-10	87.89	89.24	89.25	87.86	**89.61±0.05**
WRN-40-2	85.86	88.72	88.30	88.10	**89.12±0.10**

Table 3: Top-1 test accuracy (%) on Omniglot.

performs AdvAA in this setting. Moreover, AdvAA assumes all transformations do not change the labels of data, which may not be valid in challenging cases. More details can be found in Figure 3. We visualize the estimated distributions of transformations in Figure 2. The difference in probability values is greater on CIFAR-100 than that on CIFAR-10, which shows the effectiveness of different transformations varies more on CIFAR-100.

We train the policy network to assign proper weights to the augmented samples and use all of them to train the task network instead of rejecting the augmented samples with low weights. We also conduct experiment on the case that the policy network rejects the augmented samples with weights less than the mean of all the weights in a mini-batch. The results on CIFAR-100 with task networks WRN-28-10 and WRN-40-2 are 82.57% and 79.01% respectively, which are worse than the original case. It implies that samples with small weights are still useful. Ideally, the policy network can automatically assign very small weights to augmented samples that hurt the validation accuracy and we need no additional zeroing. Intuitively, rejecting augmented samples using a carefully selected threshold number may be helpful, but it is a bit far from the main idea of this paper.

Omniglot. To investigate the universality of our method, we conduct experiments on Omniglot which contains images of 1,623 characters instead of natural objects. For each character, we select 15, 2, and 3 images as training, validation, and test data. We compare MetaAugment with Baseline, FAA, PBA, and RA on WRN. The Baseline models are trained without data augmentation. For MetaAugment, transformations are applied to training samples directly with no Cutout added. For FAA and PBA, we do experiments with their open-source codes. For RA, we use our own implementation that randomly samples transformations and adopts the same weight for augmented samples. Implementation details are provided in Appendix.

The results are reported in Table 3. It can be seen that MetaAugment outperforms the Baseline and RA by a wide margin and still achieves better results than FAA and PBA. We also visualize the estimated distribution in Figure 2. Different from CIFAR, geometric transformations have low probability values. This is because the geometric structure is the key feature of characters and should not be changed a lot as shown in Figure 3. In contrast, natural images in CIFAR contain rich texture and color information and less depend on geometric structure. The results indicate the robustness of our policy network when dealing with bad transformations. To compare with adversarial strategy in AdvAA, we visualize samples selected by adversarial strategy and our strategy, i.e., samples with high losses but low weights and those with low losses but high weights, in Figure 3. In the first two rows, we observe that geometric transformations with large magnitudes may not preserve the labels and make the characters

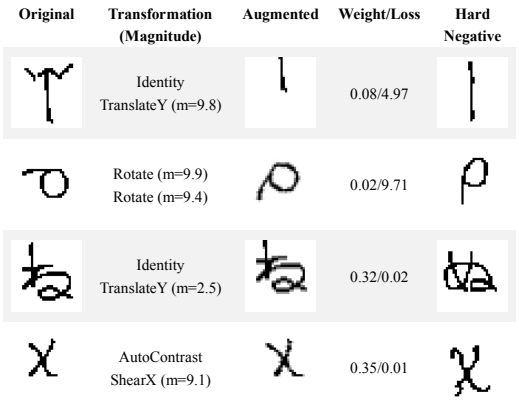


Figure 3: Examples of augmented samples on Omniglot. Here, hard negative means a validation sample w.r.t. similar feature map but different label.

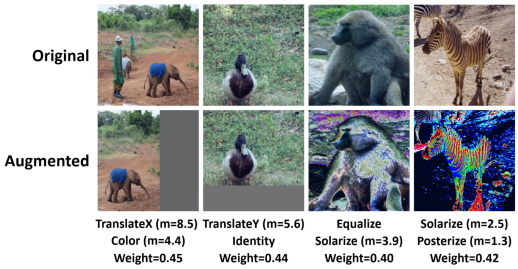
look like samples of different classes (the hard negatives). In this case, AdvAA that prefers the transformations leading to large sample losses may harm the performance. In the last two rows, we observe that our method prefers the transformations that preserve the labels and key features of the augmented samples. Our method is more robust when many bad augmentation transformations are introduced in the search space.

ImageNet. ImageNet consists of colored images in 1,000 classes, with about 1.2 million images for training. For each class, we hold out 2% of training images for validation. We compare MetaAugment with Baseline, AA, FAA, DADA, RA, and AdvAA on ResNet-50 (He et al. 2016a) and ResNet-200 (He et al. 2016b). The Baseline models are trained with the standard Inception-style pre-processing (Szegedy et al. 2015). For MetaAugment, the transformation is applied after random cropping, resizing to 224×224, and horizontal flipping with 50% probability.

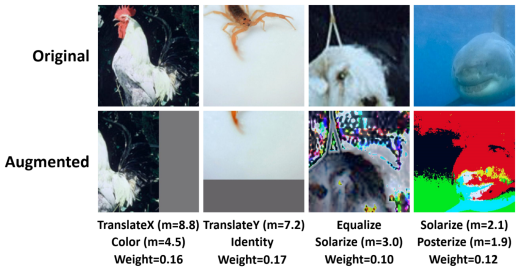
The results are presented in Table 4. MetaAugment outperforms all the other automated data augmentation methods. The model ResNet-50 is trained with Multiple-Transformation-per-sample trick, i.e., each training sample in a mini-batch is augmented by 4 transformations. By assigning proper weights to the augmented samples, MetaAugment achieves superior performance. The estimated distribution of transformations is visualized in Figure 2. Transformations with Sharpness, ShearX, and ShearY have high probability values, while transformations with Equalize, Solarize, and Posterize have low probability values. To illustrate the necessity of sample-aware data augmentation, we display some augmented samples with high and low learned weights in Figure 4. Similar transformations may have very different effects on different images. The policy network imposes high weights on the augmented images with elephant and duck that increase the diversity of training data, and imposes low weights on the augmented images with cock and scorpion that lose semantic information caused by the translation. Even for transformations with Equalize, Solarize, and Posterize that have low priority at the dataset level, the policy net-

Model	Baseline	AA	FAA	DADA	RA	AdvAA	MetaAugment
ResNet-50	76.3 / 93.1	77.6 / 93.8	77.6 / 93.7	77.5 / 93.5	77.6 / 93.8	79.40 / 94.47	**79.74±0.08 / 94.64±0.03**
ResNet-200	78.5 / 94.2	80.0 / 95.0	80.6 / 95.3	-	-	81.32 / 95.30	**81.43±0.08 / 95.52±0.04**

Table 4: Top-1 / Top-5 test accuracy (%) on ImageNet.



(a) Augmented samples with high weights



(b) Augmented samples with low weights

Figure 4: Examples of augmented samples with (a) high and (b) low weights on ImageNet.

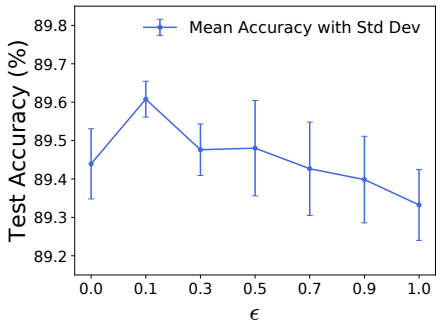


Figure 5: Test accuracy (averaged over five runs) of WRN-28-10 trained on Omniglot with different values of ϵ.

Dataset	o.TE	w.TE	own FE	share FE
CIFAR-10	97.58	**97.76**	97.59	**97.76**
CIFAR-100	83.49	**83.79**	83.68	**83.79**
Omniglot	89.29	**89.61**	89.29	**89.61**

Table 5: Top-1 test accuracy (%) of WRN-28-10 with different designs of the policy network.

work is learned to assign high weights to informative images augmented by such transformations, as shown in Figure 4(a).

Ablation Studies

Transformation Sampler. In the transformation sampler module, the hyper-parameter ϵ in Eq. (5) determines the probability of random sampling transformations. To investigate the influence of ϵ, we conduct experiments on Omniglot with task network WRN-28-10. The mean test accuracy and Std Dev over five runs with different values of ϵ are depicted in Figure 5. As expected, sampling transformations according to the estimated distribution with a certain randomness ($\epsilon = 0.1$) outperforms random sampling ($\epsilon = 1.0$).

Augmentation Policy Network. To demonstration the contributions of all the components in the policy network, we compare different designs of the policy network. We conduct experiments on the cases that the policy network does not take the transformation embedding as input and the policy network has its own feature extractor. The comparison results of WRN-28-10 trained on CIFAR and Omniglot are shown in Table 5.

First, we observe that the policy network with Transformation Embedding (w.TE) as input achieves 0.3% higher accuracy than that without TE (o.TE) in average. That means TE contains additional information beyond the images. For example, both the augmented sample and the hard negative in the first row of Figure 3 look like vertical lines, but can be generated by different transformations (TranslateY and Identity, respectively) and have different labels. With TE as input, the policy network is learned to impose different weights on them. On the other hand, the dimension of TE (28 in our setting) is much lower than that of the image feature (640 in WRN-28-10), so the TE branch hardly increases the computing cost.

Secondly, we evaluate the performance of the policy network with its own feature extractor (own FE) and that shared a common one with the task network (share FE). The latter one performs consistently better than the former one. Also, the former one takes more training time (1.2× more real running-time) since the feature extraction is repeated twice for the policy network and the task network, respectively.

Conclusions

In this paper, a sample-aware augmentation policy network is proposed to reweight augmented samples. We leverage the mechanism of meta-learning and use gradient-based optimization instead of non-differentiable approaches or reinforcement learning, which can balance the learning efficiency and model performance. As expected, the learned policy network can distinguish informative augmented images from the junks and thus greatly reduce the noises caused by intensive data augmentation. Extensive experiments demonstrate the superiority of the proposed method to the existing methods using dataset-level augmentation policies.

References

Antoniou, A.; Edwards, H.; and Storkey, A. 2019. How to train your MAML. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=HJGven05Y7.

Colson, B.; Marcotte, P.; and Savard, G. 2007. An overview of bilevel optimization. *Annals of operations research* 153(1): 235–256.

Cubuk, E. D.; Zoph, B.; Mane, D.; Vasudevan, V.; and Le, Q. V. 2019a. AutoAugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 113–123.

Cubuk, E. D.; Zoph, B.; Shlens, J.; and Le, Q. V. 2019b. RandAugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719* .

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

DeVries, T.; and Taylor, G. W. 2017. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552* .

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 1126–1135.

Freund, Y.; and Schapire, R. E. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, 23–37. Springer.

Gastaldi, X. 2017. Shake-shake regularization. *arXiv preprint arXiv:1705.07485* .

Han, D.; Kim, J.; and Kim, J. 2017. Deep pyramidal residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5927–5935.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 630–645. Springer.

Ho, D.; Liang, E.; Chen, X.; Stoica, I.; and Abbeel, P. 2019. Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules. In *International Conference on Machine Learning*, 2731–2741.

Jiang, L.; Meng, D.; Mitamura, T.; and Hauptmann, A. G. 2014a. Easy samples first: Self-paced reranking for zero-example multimedia search. In *Proceedings of the 22nd ACM international conference on Multimedia*, 547–556.

Jiang, L.; Meng, D.; Yu, S.-I.; Lan, Z.; Shan, S.; and Hauptmann, A. 2014b. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems*, 2078–2086.

Johnson, J. M.; and Khoshgoftaar, T. M. 2019. Survey on deep learning with class imbalance. *Journal of Big Data* 6(1): 27.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 1097–1105.

Kumar, M. P.; Packer, B.; and Koller, D. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, 1189–1197.

Lake, B. M.; Salakhutdinov, R.; Gross, J.; and Tenenbaum, J. B. 2011. One shot learning of simple visual concepts. In *CogSci*.

Li, Y.; Hu, G.; Wang, Y.; Hospedales, T.; Robertson, N. M.; and Yang, Y. 2020. DADA: Differentiable Automatic Data Augmentation. In *European Conference on Computer Vision*.

Li, Z.; Zhou, F.; Chen, F.; and Li, H. 2017. Meta-SGD: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835* .

Lim, S.; Kim, I.; Kim, T.; Kim, C.; and Kim, S. 2019. Fast AutoAugment. In *Advances in Neural Information Processing Systems*, 6662–6672.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2980–2988.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=S1eYHoC5FX.

Malisiewicz, T.; Gupta, A.; and Efros, A. A. 2011. Ensemble of exemplar-svms for object detection and beyond. In *2011 International conference on computer vision*, 89–96. IEEE.

Ren, M.; Zeng, W.; Yang, B.; and Urtasun, R. 2018. Learning to Reweight Examples for Robust Deep Learning. In *International Conference on Machine Learning*, 4334–4343.

Shu, J.; Xie, Q.; Yi, L.; Zhao, Q.; Zhou, S.; Xu, Z.; and Meng, D. 2019. Meta-Weight-Net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems*, 1917–1928.

Srivastava, R. K.; Greff, K.; and Schmidhuber, J. 2015. Training very deep networks. In *Advances in Neural Information Processing Systems*, 2377–2385.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Wang, X.; Pham, H.; Michel, P.; Anastasopoulos, A.; Neubig, G.; and Carbonell, J. 2019. Optimizing Data Usage via Differentiable Rewards. *arXiv preprint arXiv:1911.10088* .

Wu, L.; Tian, F.; Xia, Y.; Fan, Y.; Qin, T.; Jian-Huang, L.; and Liu, T.-Y. 2018. Learning to teach with dynamic loss functions. In *Advances in Neural Information Processing Systems*, 6466–6477.

Yamada, Y.; Iwamura, M.; Akiba, T.; and Kise, K. 2018. Shakedrop regularization for deep residual learning. *arXiv preprint arXiv:1802.02375* .

Yun, S.; Han, D.; Oh, S. J.; Chun, S.; Choe, J.; and Yoo, Y. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, 6023–6032.

Zagoruyko, S.; and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146* .

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* .

Zhang, X.; Wang, Q.; Zhang, J.; and Zhong, Z. 2020. Adversarial AutoAugment. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=ByxdUySKvS.

Zhang, Z.; and Sabuncu, M. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems*, 8778–8788.