Meta Approach to Data Augmentation Optimization

Ryuichiro Hataya[1,2], Jan Zdenek[1], Kazuki Yoshizoe[2], Hideki Nakayama[1]
[1] The University of Tokyo, [2] RIKEN AIP

Abstract

Data augmentation policies drastically improve the performance of image recognition tasks, especially when the policies are optimized for the target data and tasks. In this paper, we propose to optimize image recognition models and data augmentation policies simultaneously to improve the performance using gradient descent. Unlike prior methods, our approach avoids using proxy tasks or reducing search space, and can directly improve the validation performance. Our method achieves efficient and scalable training by approximating the gradient of policies by implicit gradient with Neumann series approximation. We demonstrate that our approach can improve the performance of various image classification tasks, including fine-grained image recognition, without using dataset-specific hyperparameter tuning.

1. Introduction

Data augmentation is an effective way to improve the performance of CNN models for image recognition tasks, particularly when its policy is optimized for the target model and dataset. Conventional data augmentation for images consists of image transformation operations, such as random cropping and flipping, and color enhancing including modification of color intensities [28, 19]. However, designing good data augmentation strategies requires profound understanding of the target data and operations. For example, CutOut [11] randomly erases a patch region of each image and is known to improve performance on the CIFAR-10 dataset, but is also reported to degrade the performance on other datasets, e.g., ImageNet [33].

Therefore, automatically designing effective augmentation strategies according to target data and tasks is desirable to improve the performance of image recognition models. One approach to augment existing data is to generate new data samples by interpolating several training images [18, 10, 55] or by using conditional generative models [1, 2]. However, this approach requires a large amount of labeled data [46] and sometimes fails to improve the performance [47], even if powerful conditional generative models are used. On the other hand, some methods improve the performance by efficiently selecting effective combinations of image transformation operations from exponentially large candidate pools [46, 8, 56]. In particular, AutoAugment [8] and its family [21, 31, 17, 9, 6, 32, 29] optimize combinations of operations to improve validation performance and achieved state-of-the-art results.

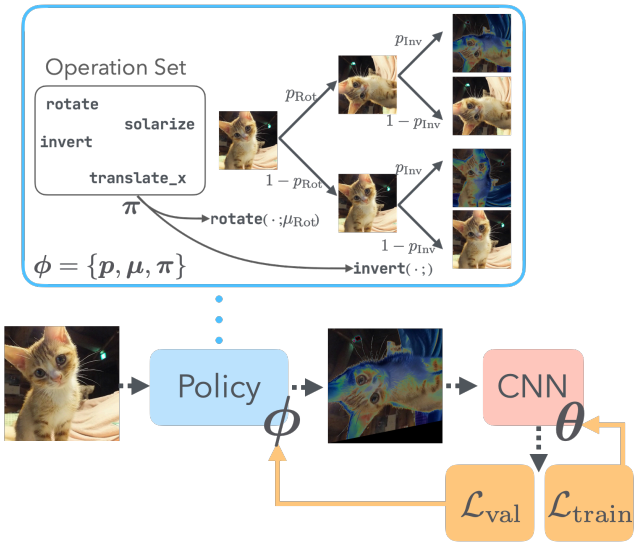

Figure 1. In AutoAugment family, a `policy` consists of data augmentation operations. Each operation, e.g., `Rotate` or `Invert`, augments an image with a probability of p and a magnitude of μ. As can been seen, performing multiple operations virtually increases the diversity of images. The operations are selected from given operation sets according to a selection parameter $\boldsymbol{\pi}$. Our proposed method, MADAO, can optimize a CNN and its data augmentation policy *simultaneously* by gradient descent in an online manner. Namely, the parameters of the CNN $\boldsymbol{\theta}$ is updated to minimize the training loss $\mathcal{L}_{\mathrm{train}}$ (also written as f), and the parameters of the policy $\boldsymbol{\phi}=\{\boldsymbol{p},\boldsymbol{\mu},\boldsymbol{\pi}\}$ is updated to minimize the validation loss $\mathcal{L}_{\mathrm{val}}$ (also written as g).

These methods involve a bi-level optimization: the inner process optimizes parameters of a CNN on training data using a given combination of operations, and the outer process optimizes the combination of operations to maximize the validation performance. Particularly, the inner loop, i.e., training of a CNN model, is usually expensive. Therefore,

```
cnn.initialize() # parameterized by θ
policy.initialize()  # parameterized by ϕ
for epoch in range(num_epochs):
    for train_data, val_data in data_loader:
        for i in range(num_inner_iters):
            input = policy(train_data[i])  # augment data by policy
            criterion = cnn.train(input) # referred to f in the text
            cnn.update(criterion)
        vcriterion = cnn.val(val_data) # referred to g in the text
        policy.update(vcriterion)
```

Algorithm 1. Our proposed method MADAO in a Python-like pseudo code. MADAO optimizes `policy` as well as `cnn` by gradient descent with *slight modifications* (`highlighted`) and *little overhead* to the standard optimization protocol.

prior works use proxy tasks that adopt small subsets of training datasets or small models [8, 21, 31, 17, 29] or reduce the search space [9, 6, 32] to keep the entire training feasible. Such approaches may result in sub-optimal solutions. We will further review and formalize this problem in Section 2.

In this paper, we tackle the original bi-level optimization problem directly without using proxy tasks or reducing the search space. We propose Meta Approach to Data Augmentation Optimization (MADAO), which optimizes CNNs and augmentation policies simultaneously by using gradient-based optimization. Here, policies are updated so that they directly increase CNNs' validation performance. Naïvely applying gradient-based optimization to this bi-level optimization requires differentiation through the inner optimization process [15] or computation of the inverse Hessian matrix [3], both of which suffer from large space complexity. These problems are fatal because data augmentation optimization needs to handle large networks, e.g., CNNs for ImageNet. We bypass these issues by using the implicit gradient method with Neumann series approximation. Thanks to these approximations, MADAO is simple with little overhead as shown in Algorithm 1. Notably, this simplicity allows MADAO to scale to problems of ImageNet size, which has been nearly impossible for existing bi-level optimization methods [23].

We empirically demonstrate that MADAO learns effective data augmentation policies and achieves performance comparable or even superior to existing methods on benchmark datasets for image classification: CIFAR-10, CIFAR-100, SVHN, and ImageNet, as well as fine-grained datasets. All of the reported results have been achieved *without using dataset-specific configurations*.

The contributions of this paper are summarized as follows:

- We propose MADAO, which directly solves the bi-level problem of data augmentation optimization without using proxy tasks or reduced search space. In other words, MADAO optimizes a CNN model and its data augmentation policy simultaneously.
- To efficiently solve this bi-level problem, MADAO uses differentiable data augmentation and implicit gradient methods with the Neumann series approximation. We empirically demonstrate that MADAO is applicable to large-scale problems, including ImageNet classification, and especially improves the performance on fine-grained classification tasks.

2. Generalizing Data Augmentation Optimization

In Section 2.1, we describe the preliminaries of data augmentation optimization of AutoAugment family, and then review prior methods by generalizing the problem in Section 2.2.

2.1. Designing Data Augmentation Space

Let us define a set of input images $\mathcal{X}$ and a set of operations $\mathcal{S}$ consisting of data augmentation operations such as rotation and color inversion. In the AutoAugment family, each image $\boldsymbol{x} \in \mathcal{X} \subset [0,1]^D$ is augmented by an operation $O : \mathcal{X} \to \mathcal{X}$ with a probability of $p_O \in [0,1]$ and a magnitude of $\mu_O \in [0,1]$ as illustrated in Figure 1. The magnitude parameter can correspond for example to the degree of rotation, while some operations, such as inversion, have no magnitude parameter. By applying K consecutive augmentation operations, each image results in 2^K possible images, that is, the number of images virtually increases. This formulation makes the size of the search space $(|\mathcal{S}| \times [0,1] \times [0,1])^K$, where $|\mathcal{S}|$ is the size of the operation set.

Operations and accompanied parameters need to be selected so that they minimize the validation criterion, such as the error rate. Usually, this selection is performed heuristically [28, 19]. However, Cubuk et al. showed that data-driven optimization surpasses handcrafted selection [8].

2.2. Generalizing AutoAugment Family

Let $\boldsymbol{\theta} \in \mathbb{R}^M$ denote parameters of a CNN model and $\boldsymbol{\phi} \in \mathbb{R}^N$ denote parameters of a policy for augmentation, i.e., selection parameters of operations from the operation set $\boldsymbol{\pi}$ and their accompanied parameters $\{(\mu_O, p_O); O \in \mathcal{S}\}$. Let the empirical risk be $f(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathcal{D}_T)$ and the validation criterion be $g(\boldsymbol{\theta}; \mathcal{D}_V)$. $\mathcal{D}_T$ and $\mathcal{D}_V$ are training and validation datasets. In the case of classification task, $\mathcal{D} \coloneqq \{(\boldsymbol{x}_i, y_i)\} \in \mathcal{X} \times \{1, 2, \ldots, C\}$, where C is the number of classes.

Optimization of data augmentation policy in AutoAugment family methods can be generalized as

$$\operatorname*{argmin}_{\boldsymbol{\phi}} g(\operatorname*{argmin}_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathcal{D}_T); \mathcal{D}_V), \tag{1}$$

that is, optimizing CNNs on training data with policies that minimize validation criteria on validation data.

Naïvely solving this bi-level minimization problem takes a long time because CNN training $\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \boldsymbol{\phi})$ is costly and the number of possible combinations of augmentation operations and their parameters is infeasibly large. Therefore, prior works tried to alleviate this problem in several ways. One direction is to reduce the search space over augmentation policies, i.e., dimension and range of the parameter $\boldsymbol{\phi}$ [9, 6, 32]. For example, RandAugment [9] randomly samples operations from the operation set $\mathcal{S}$ and shares μ_O among all operations. This reduction changes the outer problem in Equation (1) from $\operatorname{argmin}_{\boldsymbol{\phi}} g$ to $\operatorname{argmin}_{\mu} g$, which makes it possible to use a simple searching process, such as grid search. OHL-AutoAug [32] enables online searching using policy gradient [53] by restricting the search space only to a limited range.

On the other hand, some methods use proxy tasks that approximate Equation (1) to obtain (sub-) optimal policy $\boldsymbol{\phi}'$ to reduce the searching time of the inner optimization. The obtained policy $\boldsymbol{\phi}'$ is then used to train a CNN as $\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \boldsymbol{\phi}')$ [8, 21, 31, 17] in an "offline" manner. For instance, AutoAugment employs a proxy task f' that approximates the original inner problem f as

$$\boldsymbol{\phi}' = \operatorname*{argmin}_{\boldsymbol{\phi}} g(\operatorname*{argmin}_{\boldsymbol{\theta}'} f'(\boldsymbol{\theta}', \boldsymbol{\phi}'; \mathcal{D}'); \mathcal{D}_V), \tag{2}$$

with a smaller dataset $|\mathcal{D}'| \ll |\mathcal{D}_T|$ and a smaller network $\dim \boldsymbol{\theta}' \ll \dim \boldsymbol{\theta}$ for efficiency. The outer problem g is optimized by black-box optimization techniques, such as reinforcement learning [8]. Fast AutoAugment [31] and Faster AutoAugment [17] approximate Equation (1) as minimizing distance of distributions between augmented images and original images without directly minimizing f, which allows faster searching.

To summarize, prior methods indirectly solve the bi-level optimization problem in Equation (1), as displayed in Table 1. We instead propose to tackle this problem directly.

Method	Direct Inner Problem	Direct Outer Problem
AutoAugment [8]		✓
Fast AutoAugment [31]		✓
Faster AutoAugment [17]		✓
OHL-AutoAug [32]	✓	
RandAugment [9]	✓	
MADAO (ours)	✓	✓

Table 1. Comparison of previous methods and our proposal, MADAO. **MADAO can efficiently search *full space* for policies on *full datasets* with *full CNN models***, e.g., on ImageNet with ResNet-50.

3. Method

In this paper, we propose to directly optimize the bi-level optimization problem in Equation (1). In other words, we optimize CNNs and augmentation policies *simultaneously*, i.e., in an *online* manner, without reducing the search space or using proxy tasks. With this simultaneous optimization, policies are expected to augment images according to the learning state of CNN models. Algorithm 1 is a simple depiction of our approach, which we call Meta Approach to Data Augmentation Optimization, or in short, MADAO.

3.1. Optimizing Policies by Gradient Descent

MADAO directly optimizes the bi-level problem Equation (1) using gradient descent. To this end, we assume that f and g are differentiable w.r.t. $\boldsymbol{\theta}$. Taking C-category classification as an example, cross entropy $-\mathbb{E}_{\boldsymbol{x}_i, y_i} \log[\boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)]_{y_i}$ can be used as f and g, but error rate $\mathbb{E}_{\boldsymbol{x}_i, y_i} \mathbb{1}(\operatorname{argmax} \boldsymbol{h}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \neq y_i)$ cannot. Here, $\mathbb{1}$ is the indicator function, and $\boldsymbol{h}_{\boldsymbol{\theta}} : \mathbb{R}^D \to \mathbb{R}^C$ is a CNN with a softmax output layer.

Gradient-based optimization of Equation (1) requires $\nabla_{\boldsymbol{\phi}} g$ for iterative updating. Since the data augmentation implicitly affects the validation criterion, in other words, data augmentation is not used for validation, we obtain

$$\frac{\partial g}{\partial \boldsymbol{\phi}} = \frac{\partial g}{\partial \boldsymbol{\theta}} \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\phi}}. \tag{3}$$

Because of the requirement of g, $\nabla_{\boldsymbol{\theta}} g$ can be obtained. On the other hand, the exact computation of $\nabla_{\boldsymbol{\phi}} \boldsymbol{\theta}$ has a large space complexity, as we will describe in Section 3.4. Yet, if this gradient was available, the policies could be optimized by gradient descent.

3.2. Approximating Gradients of Policy and Inverse Hessian

To obtain $\nabla_{\boldsymbol{\phi}} \boldsymbol{\theta}$ without suffering from a large space complexity, we can use the Implicit Function Theorem. If there exists a fixed point $(\boldsymbol{\theta}^\star, \boldsymbol{\phi}^\star)$ that satisfies $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^\star, \boldsymbol{\phi}^\star) = 0$, then there exists a function $\hat{\theta}$ around $\boldsymbol{\phi}^\star$ such that $\hat{\boldsymbol{\theta}}(\boldsymbol{\phi}^\star) = \boldsymbol{\theta}^\star$. If this condition holds, we also obtain

$$\frac{\partial \hat{\boldsymbol{\theta}}}{\partial \boldsymbol{\phi}} = -\left(\frac{\partial^2 f}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top}\right)^{-1} \frac{\partial^2 f}{\partial \boldsymbol{\theta} \partial \boldsymbol{\phi}^\top}\Big|_{\boldsymbol{\theta}^\star, \boldsymbol{\phi}^\star}. \tag{4}$$

We can obtain an approximated gradient using this property. Unfortunately, $M = \dim \boldsymbol{\theta}$ is usually large; therefore, computing the inverse Hessian matrix $(\nabla^2_{\boldsymbol{\theta}} f)^{-1}$ is prohibitively expensive as it usually requires $\mathcal{O}(M^3)$ computations. To avoid computing the inverse Hessian matrix, we use iterative methods based on the Neumann series, which boast better scalability than conjugate gradients in various problems [25, 30, 34].

The Neumann series $\boldsymbol{I} + \boldsymbol{A} + \boldsymbol{A}^2 + \cdots = \sum_i^J \boldsymbol{A}^j \to (\boldsymbol{I} - \boldsymbol{A})^{-1}$ $(J \to \infty)$ holds with a given squared matrix $\boldsymbol{A}$ if $\|\boldsymbol{A}\| < 1$. Using this property, Equation (3) can be approximated with a positive integer J as

$$\frac{\partial g}{\partial \boldsymbol{\phi}} = -\frac{\partial g}{\partial \boldsymbol{\theta}} \left(\frac{\partial^2 f}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top}\right)^{-1} \frac{\partial^2 f}{\partial \boldsymbol{\theta} \partial \boldsymbol{\phi}^\top} \tag{5}$$

$$\approx -\frac{\partial g}{\partial \boldsymbol{\theta}} \sum_{j=0}^{J} \left(\boldsymbol{I} - \frac{\partial^2 f}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top}\right)^{j} \frac{\partial^2 f}{\partial \boldsymbol{\theta} \partial \boldsymbol{\phi}^\top}. \tag{6}$$

We regularize the norm by simply introducing a scalar $\alpha \in \mathbb{R}^+$ as $\boldsymbol{I} - \alpha \nabla^2_{\boldsymbol{\theta}} f$ as [34]. We use $\alpha = 10^{-3}$ with $J = 5$ in the experiments. This Neumann series approximation can also help us avoid explicitly storing the Hessian matrix, whose space complexity is $\mathcal{O}(M^2)$, by using Hessian-vector products derived from the following identity:

$$\boldsymbol{v}^\top \left(\frac{\partial^2 f}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top}\right) = \frac{\partial}{\partial \boldsymbol{\theta}} \left(\boldsymbol{v}^\top \frac{\partial f}{\partial \boldsymbol{\theta}}\right) \qquad (\forall \boldsymbol{v} \in \mathbb{R}^M). \tag{7}$$

This right-hand side can be used instead of storing the Hessian matrix and only has the space complexity of $\mathcal{O}(M)$.

3.3. Differentiable Data Augmentation

To differentiate through $\boldsymbol{\phi}$, MADAO adopts the differentiable data augmentation pipeline following [17]. As described in Section 2.1, each image $\boldsymbol{x}$ is augmented with an operation O with a magnitude of μ_O and a probability of p_O, which can be written as

$$\boldsymbol{x} \mapsto \begin{cases} O(\boldsymbol{x}; \mu_O) & \text{with probability of } p_O \\ \boldsymbol{x} & \text{with probability of } 1 - p_O. \end{cases} \tag{8}$$

This right-hand side can be written as $bO(\boldsymbol{x}; \mu_O) + (1 - b)\boldsymbol{x}$ with a binary stochastic variable $b \sim \text{Bern}(b; p_O)$. Although sampling from the original Bernoulli distribution $\text{Bern}(b; p_O)$ is not differentiable w.r.t. p_O, Gumbel trick [24] relaxes this restriction to enable backpropagation to update p_O. Similarly, some color-enhancing operations are non-differentiable w.r.t. the magnitude μ_O because of discretization, so the straight-through estimator [4] is used for such operations. We clamp μ_O and p_O by a sigmoid function to limit their range to [0, 1]. We denote the operation in eq. (8) as $O(\boldsymbol{x}; \mu_O, p_O)$ for simplicity.

MADAO uses a different method to select operations compared to Faster AutoAugment in order to accelerate training. MADAO selects operations using categorical distribution parameterized by a weight parameter $\boldsymbol{\pi} \in [0, 1]^{|\mathcal{S}|}$, where $\sum_i \pi_i = 1$. Since the original categorical distribution is non-differentiable as Bernoulli distribution, we use Gumbel-softmax with a temperature of $\tau \in \mathbb{R}^+$. This distribution, referred to as $\text{RelaxCat}(\boldsymbol{\pi}; \tau)$, samples one-hot-like vectors as $\tau \longrightarrow 0$. Using this distribution, an operation is selected and applied as

$$\begin{aligned} \boldsymbol{x} &\mapsto \frac{u_i}{\text{SG}(u_i)} O_i(\boldsymbol{x}; \mu_{O_i}, p_{O_i}), \\ i &= \operatorname{argmax} \boldsymbol{u}, \\ \boldsymbol{u} &\sim \text{RelaxCat}(\boldsymbol{\pi}; \tau). \end{aligned} \tag{9}$$

Here, SG is the stop gradient operation, and thus, $\frac{u_i}{\text{SG}(u_i)} = 1$ so that the transformation Equation (9) keeps the range in $(0, 1)$. O_i is the ith operation that is used with probability p_{O_i}. Different from this approach, Faster AutoAugment applies all operations and takes the weighted sum of the outputs to approximate this selection.

3.4. Connection to Gradient-based Hyperparameter Optimization

As can been seen, Equation (1) is a hyperparameter optimization (HO) problem to neural networks. Traditional HO methods, such as grid search, random search [5] and Bayesian optimization [50], have poor scalability to increasing dimensionality of hyperparameters [23]. For this reason, gradient-based HO attracts attention. From HO viewpoint, policy parameters $\boldsymbol{\phi}$ are hyperparameters.

As shown in Equation (3), we need to obtain $\nabla_{\boldsymbol{\phi}} \boldsymbol{\theta}$ to optimize the outer problem g. The inner optimization process (Algorithm 1 Line 5-8) can be rewritten as

$$\begin{aligned} \boldsymbol{\theta}_T =& \boldsymbol{\theta}_{T-1} - \eta \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}_{T-1}, \boldsymbol{\phi}) \\ =& \cdots \\ =& \boldsymbol{\theta}_0 - \eta \nabla_{\boldsymbol{\theta}} \sum_{t=0}^{T-1} f(\boldsymbol{\theta}_t, \boldsymbol{\phi}) =: \boldsymbol{\theta}_T(\boldsymbol{\theta}_0, \boldsymbol{\phi}) \end{aligned} \tag{10}$$

after T SGD steps with learning rate of η. One approach to obtain $\nabla_{\boldsymbol{\phi}} \boldsymbol{\theta}$ is to unroll the steps in Equation (10) as [36, 15, 16, 49]:

$$\nabla_{\boldsymbol{\phi}} \boldsymbol{\theta} = \nabla_{\boldsymbol{\phi}} \boldsymbol{\theta}_T(\boldsymbol{\theta}_0, \boldsymbol{\phi}). \tag{11}$$

Practically, this unrolling requires to cache $\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_{T-1}$, and thus, the space complexity becomes $\mathcal{O}(TM)$, which might be prohibitive for large neural networks, while MADAO only requires $\mathcal{O}(M)$. [38] uses eq. (11) for data augmentation optimization, they only considered the $T = 1$ case empirically. This unrolling can be approximated in first-order manners [15, 40], but these approximations succeed when tasks are simple [45].

Alternatively, implicit gradient yields $\nabla_{\boldsymbol{\phi}} \boldsymbol{\theta}$ as explained in Equation (4). This approach needs the inverse Hessian matrix $(\nabla^2_{\boldsymbol{\theta}} f)^{-1}$ [3], but this computation is infeasible for modern neural networks with millions of parameters. Iterative methods, such as conjugate gradient [12, 13, 44, 45] or Neumann series approximation [34], effectively compensate for this issue by approximating this inverse matrix in gradient hyperparameter optimization. Such iterative approximation methods using the Neumann series are also used in approximating influence function [25] and enabling RNNs to handle long sequences [30]. We exploit the knowledge from these prior works and adopt the implicit gradient method with Neumann series approximation to efficiently handle large-scale datasets and CNNs.

4. Experiments and Results

This section describes the empirical results of our proposed method in image classification. We used CIFAR-10, CIFAR-100 [27], SVHN [39] and ImageNet (ILSVRC-2012) [48]. In addition, we also used four fine-grained classification datasets: Oxford 102 Flowers [41], Oxford-IIIT Pets [42], FGVC Aircraft [37], and Stanford Cars [26]. In all experiments except those on ImageNet, we set 10 % of the original training data aside as validation data $\mathcal{D}_V$ and report error rates on test data. For ImageNet, we used 1 % of the training data as validation data. Note that this data split means that we use less training data than previous works and that changes the performance of baseline models.

We used 14 operations for augmentation: `ShearX`, `ShearY`, `TranslateX`, `TranslateY`, `Rotate`, `Invert`, `AutoContrast`, `Equalize`, `Solarize`, `Color`, `Posterize`, `Contrast`, `Brightness`, and `Sharpness` (see Appendix A for more details.) To make these operations differentiable, we implemented them using PyTorch [43], kornia [14], and the implementation of Faster AutoAugment provided by the author [17]. We scaled the magnitude of operations from 0 to 1 so that the magnitude of 0 means no change and the magnitude of 1 corresponds to the strongest level of the particular augmentation operation.

As baseline methods, we selected Faster AutoAugment [17] and RandAugment [9], which are representative methods that use proxy tasks and reduced search spaces, respectively. For a fair comparison, we used the same differentiable operations as Faster AutoAugment. More details about experiment configurations can be found in Appendix B.

MADAO

To optimize the policy of MADAO, we used RMSprop optimizer with learning rate of 10^{-2} following [34]. Equation (4) requires the existence of fixed points that satisfy $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \boldsymbol{\phi}) = 0$. Following [34], we assume that s iterations, which corresponds to `num_inner_iters` in Algorithm 1, makes the parameters hold the condition. To further encourage parameters to satisfy the condition, we also performed a warm up, i.e., training CNNs without policy update for the first w epochs. We set $s = 60$ and $w = 30$ and used this configuration for *all experiments*. Validation criterion g is the same loss function that is used as the training loss function f, cross-entropy, for simplicity.

We initialized the parameters of magnitude and probability with 0.5 and the parameters for operation selection to be equal. This configuration means that MADAO in its initial state is nearly equivalent to RandAugment with the magnitude of 0.5. We set the number of augmentation stages $K = 2$ in all experiments below (see Figure 1).

Faster AutoAugment

We selected Faster AutoAugment [17] as a representative baseline that uses proxy tasks. We set the number of augmentation stages $K = 2$. We modified Faster AutoAugment to use operation sampling of Equation (9) during policy training for a fair comparison.

To train Faster AutoAugment's policies, we used WideResNet-40-2 for CIFAR-10, CIFAR-100, and SVHN, and ResNet-18 for fine-grained datasets and ImageNet.

RandAugment

As a representative baseline of space-reduction methods, we chose RandAugment [9]. Following [9], we discretized the range of the shared magnitude parameters into 30 bins and optimized this parameter by random search. For each configuration, we conducted two runs of random search for a fair comparison with MADAO with respect to computational time as shown in Figure 2[1]. We report the test error rates for the runs with the lowest validation error rates. For a fair comparison, we set the number of augmentation transformations to apply sequentially to 2.

4.1. CIFAR-10, CIFAR-100, and SVHN

Table 2 presents test error rates on CIFAR-10, CIFAR-100, and SVHN with various CNN models: WideResNet-

[1]Precisely, we used the computational time of $s = \infty$ (RandAugment, 1 hour) and $s = 60$ (MADAO, 1.9 hours).

Dataset	Model	Standard	Faster AutoAugment	RandAugment	MADAO (ours)
CIFAR-10	WideResNet 28-2	4.9	4.2	4.4	4.0
	WideResNet 40-2	4.8	3.9	4.1	3.7
	WideResNet 28-10	3.8	2.7	3.1	2.7
	DenseNet 40	5.6	4.6	4.8	4.5
CIFAR-100	WideResNet 28-2	25.4	23.2	24.0	22.9
	WideResNet 40-2	24.1	22.5	23.0	22.4
SVHN	WideResNet 28-2	3.4	2.6	3.0	2.6
	WideResNet 40-2	3.2	2.5	2.8	2.4

Table 2. Test error rates on CIFAR-10, CIFAR-100 and SVHN.

Standard	Faster AutoAugment	RandAugment	MADAO (ours)
23.8/7.0	23.0/6.5	22.9/6.6	22.5/6.5

Table 3. ImageNet test error rates (Top-1/Top-5). MADAO achieves comparable performance to other data augmentation methods *without using ImageNet-specific configurations.*

28-2, WideResNet-40-2, WideResNet-28-10 [54], and DenseNet-BC 40 ($k = 12$) [22]. We trained models for 300 epochs on CIFARs and for 160 epochs on SVHN. We show the average scores of three runs. For comparison, we present the results of Faster AutoAugment, RandAugment, and the default augmentation: random cropping, and random horizontal flipping (except SVHN), following [9], which we refer to as *Standard*. As can been seen, MADAO achieves performance superior or comparable to baseline methods. Specifically, MADAO and Faster AutoAugment consistently achieve better performance than RandAugment, which suggests the importance of policy searching from a large search space in a limited search cost.

4.2. ImageNet

Table 3 shows the results on ImageNet with ResNet-50 [19]. We trained models for 180 epochs and used random cropping and random horizontal flipping as the standard augmentation. The performance of MADAO is superior to those of RandAugment and Faster AutoAugment.

Most importantly, the result shows that MADAO can scale to an ImageNet-size problem without using proxy tasks or reducing search space. We believe that class- or instance-conditional data augmentation might be required to further improve performance, which we leave as an open problem.

4.3. Fine-grained Classification

To showcase the ability of MADAO to augment images according to target datasets, we conducted experiments on fine-grained datasets with ResNet-18 [19] trained for 200 epochs. We applied ImageNet's preprocessing to these datasets.

Table 4 shows the average test error rates over three runs. These datasets are nearly ten to twenty times smaller than CIFAR datasets and SVHN, yet MADAO improves the performance *without using dataset-specific hyperparameters*. MADAO outperforms Faster AutoAugment and RandAugment, which emphasizes the importance of searching for good policies based on full target datasets and from the full search space. MADAO can capture the characteristics of each fine-grained dataset and generate tailored policies for each dataset from a large search space.

5. Analysis

5.1. How Inner Steps Affect the Performance

Figure 2 presents the relationship between the number of inner steps s, computational cost, and test error rates when using WideResNet-28-2 on CIFAR-10. CNNs yield the best performance when $s = 60$, where s is the number of inner steps per which the policies are updated. The results indicate that there exists a trade-off between "exploration and exploitation" of obtained policies: a small number of inner steps might not correctly evaluate the current policies, while running a large number of inner steps might fail to explore better strategies. Importantly, unrolled-based implementations would require to store s model caches, which is infeasible for $s = 60$ with modern CNNs. On the other hand, MADAO can efficiently handle a large s.

5.2. How Policies Develop during Training

In Figure 3, we present the development of the selection parameters $\boldsymbol{\pi}$ for each operation during training on fine-grained datasets. As can be seen, each dataset has its specific operations that are selected, which could be thought as reflecting the characteristics of each dataset. Besides, the way of selection changes according to the learning phase. Note that the first 30 epochs are set to warm-up and the augmentation policy parameters are not updated.

Figure 4 shows how the selection parameters develop during training on CIFAR-10 and SVHN. Similar to fine-grained datasets shown in Figure 3, the policies for CIFAR-10 and SVHN also show clear difference to each other. As

Dataset	Model	# Classes	# Training Set	Standard	Faster AutoAugment	RandAugment	MADAO (ours)
Flower	ResNet-18	102	2,040	11.1	9.1	8.8	8.1
Pet	ResNet-18	37	3,680	15.3	12.8	13.7	11.9
Aircraft	ResNet-18	70	6,667	11.1	10.3	10.0	9.8
Car	ResNet-18	196	8,144	14.7	13.6	11.6	11.4

Table 4. Test error rates on fine-grained datasets. *Note that we use the same hyperparemeters as in the experiments in Table 2.*

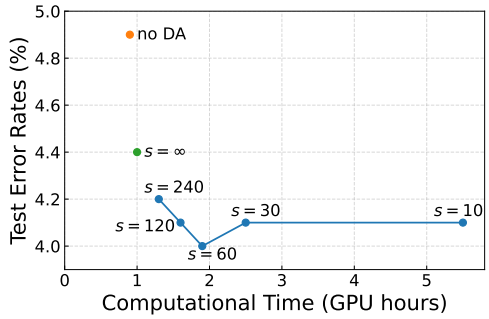

Figure 2. The relationship between the number of inner steps s, computational cost, and test error rates on CIFAR-10 with WideResNet-28-2. $s = \infty$ means the parameters are fixed as RandAugment, and *no DA* shows the cost and error rate without using data augmentation. We used $s = 60$ in the main experiments.

can be observed, the first and second stage for each dataset evolve differently, which indicates that the stages develop complementarily to each other.

Figure 5 presents the development of magnitude parameters $\boldsymbol{\mu}$ and probability parameters $\boldsymbol{p}$ during training of WideResNet-28-2 on SVHN. Both parameters diverge as training proceeds. These results imply that optimal magnitudes and probabilities might be non-global, which disagrees with RandAugment.

5.3. Comparison with Other Gradient Approximations

As discussed in Section 3.4, some alternative approaches can be used to approximate the bi-level problem using (implicit) gradients as Equation (4). In Table 5, we present the test accuracy with WideResNet-28-2 on CIFAR-10 with MADAO variants: the original MADAO using the Neumann series approximation, MADAO-CG using the conjugate gradient algorithm as [15] and MADAO-FO using the first-order approximation as [45]. The conjugate gradient algorithm is an iterative method to solve a linear equation [20], which in our case used to solve $\boldsymbol{x}^\top(\nabla^2_{\boldsymbol{\theta}} f) = \nabla_{\boldsymbol{\theta}} g$ to obtain $\nabla_{\boldsymbol{\theta}} g(\nabla^2_{\boldsymbol{\theta}} f)^{-1}$ in Equation (5).

These results show the effectiveness of MADAO, which uses the Neumann series approximation, compared with MADAO-CG, and MADAO-FO. We did not observe significant difference between MADAO and MADAO-CG in memory consumption and execution time.

MADAO	MADAO-CG	MADAO-FO
4.0	4.4	4.4

Table 5. Test error rates (mean / standard deviation) on CIFAR-10 with WideResNet-28-2 using different MADAO variants. Namely, MADAO-CG uses the conjugate gradient method, and MADAO-FO uses the first-order approximation.

5.4. Computational Cost

As regards memory consumption when using WideResNet-28-2 on CIFAR-10, training with MADAO requires 3.38 GB of GPU memory, while training without MADAO requires 1.74 GB. This double memory consumption is expected from Equation (7). These values demonstrate that MADAO has not only theoretical but also empirical efficiency.

6. Conclusion

In this paper, we have proposed MADAO, a novel approach to optimize an image recognition model and its data augmentation policy simultaneously. To efficiently achieve this goal, we use differentiable data augmentation and the implicit gradient method with Neumann series approximation. As a result, the overhead of MADAO to the standard CNN training, with respect to time and memory, is marginal, which enables ImageNet-scale training. Empirically, we demonstrate on various tasks that MADAO achieves equal or better performance to prior works without restricting search space or using sub-optimal proxy tasks.

Data augmentation boosts the performance in various visual representation learning settings, such as semi-supervised learning [6, 51], domain generalization [52], and self-supervised learning [7]. We believe that our method can be introduced into these representation learning methods and efficiently enhance their performance by optimizing the policies according to given tasks.

Acknowledgement

This work was supported by Institute of AI and Beyond of the University of Tokyo, the MOE-Microsoft Key Laboratory of the University of Tokyo, JSPS KAKENHI Grant Number JP19H04166 and JP20H04251. We also appreciate anonymous reviewers' comments to improve this work.

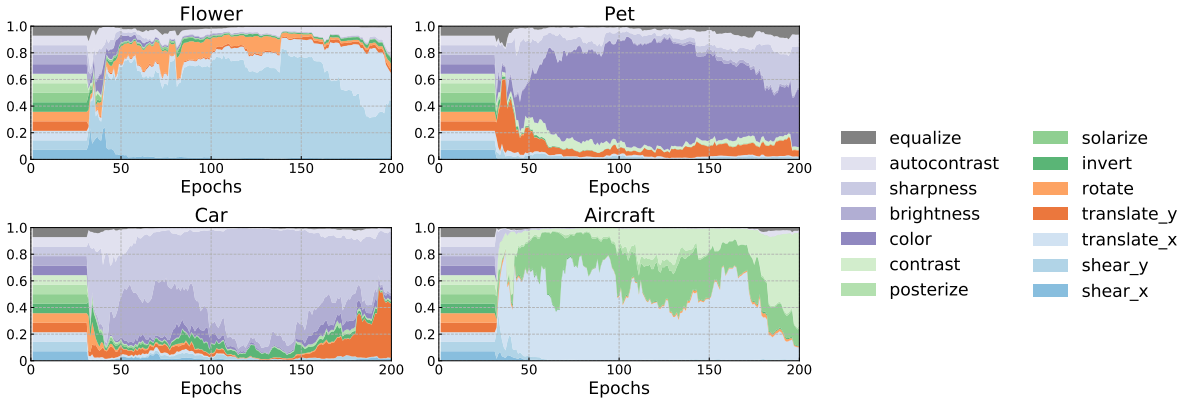

Figure 3. The development of operation selection probabilities π at the first augmentation stage during training on fine-grained datasets. Note that we set the first 30 epochs as a warm-up period during which parameters are not updated.

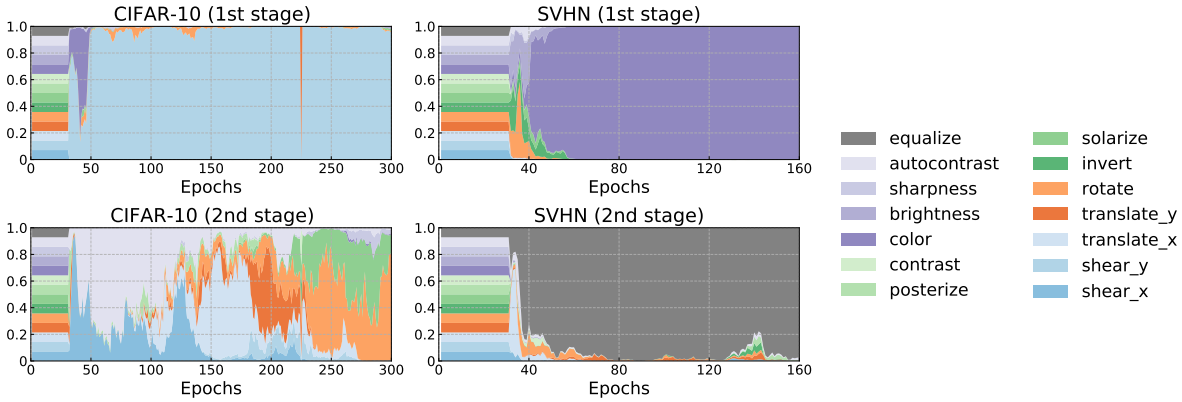

Figure 4. The development of the selection probabilities π for each operation during training on CIFAR-10 and SVHN. Note that we set the first 30 epochs to warm-up period that parameters are not updated.

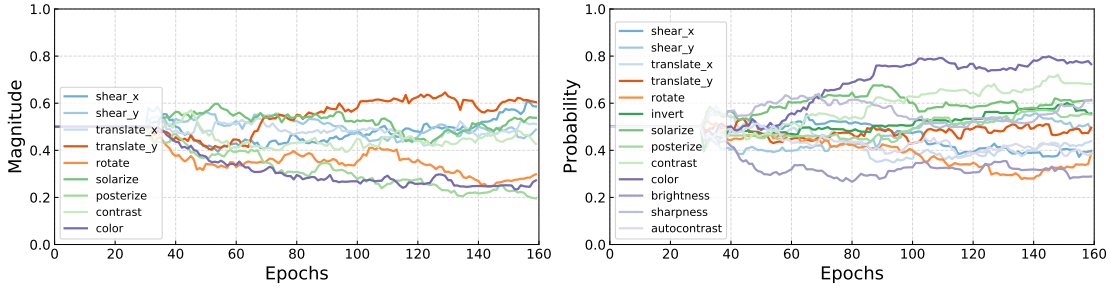

Figure 5. The development of probability parameters p and magnitude parameters μ at the first augmentation stage (corresponds to Figure 4 top right) during training on SVHN with WideResNet-28-2. Note that we set the first 30 epochs to warm-up period that parameters are not updated.

References

[1] Anthreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. In *ICLR*, 2018.

[2] Shumeet Baluja and Ian Fischer. Learning to attack: Adversarial transformation networks. In *AAAI*, 2018.

[3] Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural Computation*, 12(8):1889–1900, 2000.

[4] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. arXiv:1308.3432, 2013.

[5] James Bergstra and Yoshua Bengio. Random search for hyperparameter optimization. *JMLR*, 13:281–305, 2012.

[6] David Berthelot, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. ReMixMatch: Semi-Supervised Learning with Distribution Alignment and Augmentation Anchoring. In *ICLR*, 2020.

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. arXiv:2002.05709, 2020.

[8] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. AutoAugment: Learning Augmentation Policies from Data. In *CVPR*, 2018.

[9] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. RandAugment: Practical data augmentation with no separate search. In *CVPRW*, 2020.

[10] Terrance DeVries and Graham W. Taylor. Dataset augmentation in feature space. In *ICLR*, 2017.

[11] Terrance DeVries and Graham W. Taylor. Improved Regularization of Convolutional Neural Networks with Cutout. arXiv:1708.04552, 2017.

[12] Chuong B. Do, Chuan Sheng Foo, and Andrew Y. Ng. Efficient multiple hyperparameter learning for log-linear models. In *NIPS*, 2009.

[13] Justin Domke. Generic methods for optimization-based modeling. *JMLR*, 22:318–326, 2012.

[14] D Ponsa E Rublee E. Riba D. Mishkin and G Bradski. Kornia: an Open Source Differentiable Computer Vision Library for PyTorch. In *WACV*, 2019.

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *ICML*, 2017.

[16] Luca Franceschi, Paolo Frasconi, Saverio Salzo, and Massimilano Pontil. Bilevel Programming for Hyperparameter Optimization and Meta-Learning. In *ICML*, 2018.

[17] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Faster AutoAugment: Learning Augmentation Strategies using Backpropagation. In *ECCV*, 2020.

[18] Søren Hauberg, Oren Freifeld, Anders Boesen Lindbo Larsen, John W. Fisher III, and Lars Kai Hansen. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. In *AISTATS*, 2016.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[20] Magnus R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–435, 1952.

[21] Daniel Ho, Eric Liang, Ion Stoica, Pieter Abbeel, and Xi Chen. Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules. In *ICML*, 2019.

[22] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[23] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automatic Machine Learning: Methods, Systems, Challenges*. Springer, 2019.

[24] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*, 2017.

[25] Pang Wei Koh and Percy Liang. Understanding Black-box Predictions via Influence Functions. In *ICML*, 2017.

[26] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *IEEE Workshop on 3D Representation and Recognition*, 2013.

[27] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

[29] Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy Hospedales, Neil M. Robertson, and Yongxin Yang. Dada: Differentiable automatic data augmentation. In *ECCV*, 2020.

[30] Renjie Liao, Yuwen Xiong, Ethan Fetaya, Lisa Zhang, Ki Jung Yoon, Xaq Pitkow, Raquel Urtasun, and Richard Zemel. Reviving and improving recurrent back-propagation. In *ICML*, 2018.

[31] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast AutoAugment. In *NeurIPS*, 2019.

[32] 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.

[33] Raphael Gontijo Lopes, Dong Yin, Ben Poole, Justin Gilmer, and Ekin D. Cubuk. Improving robustness without sacrificing accuracy with patch gaussian augmentation. arXiv:1906.02611, 2019.

[34] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing Millions of Hyperparameters by Implicit Differentiation. In *AISTATS*, 2020.

[35] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. In *ICLR*, 2016.

[36] Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. Gradient-based Hyperparameter Optimization through Reversible Learning. In *ICML*, 2015.

[37] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, ArXiv (1306.5151), 2013.

[38] Saypraseuth Mounsaveng, Issam Laradji, Ismail Ben Ayed, David Vazquez, and Marco Pedersoli. Learning data augmentation with online bilevel optimization for image classification. In *WACV*, 2021.

[39] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading Digits in Natural Images with Unsupervised Feature LearningNo Title. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[40] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. arXiv:1803.02999, 2018.

[41] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[42] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *ICPR*, 2012.

[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*. 2019.

[44] Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *ICML*, 2016.

[45] Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. Meta-Learning with Implicit Gradients. In *NeurIPS*, 2019.

[46] Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. In *NIPS*, 2017.

[47] Suman Ravuri and Oriol Vinyals. Seeing is not necessarily believing: Limitations of BigGANs for data augmentation. In *ICLR Learning from Limited Labeled Data Workshop*, 2019.

[48] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[49] Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots. Truncated Back-propagation for Bilevel Optimization. In *AISTATS*, 2019.

[50] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *NIPS*. 2012.

[51] Teppei Suzuki and Ikuro Sato. Adversarial transformations for semi-supervised learning. In *AAAI*, 2020.

[52] Riccardo Volpi and Vittorio Murino. Addressing Model Vulnerability to Distributional Shifts over Image Transformation Sets. In *ICCV*, 2019.

[53] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[54] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In *BMVC*, 2016.

[55] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

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