SLACK: Stable Learning of Augmentations with Cold-start and KL regularization

Juliette Marrie[1,2] Michael Arbel[1] Diane Larlus[2] Julien Mairal[1]

[1] Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK [2] NAVER LABS Europe

Abstract

Data augmentation is known to improve the generalization capabilities of neural networks, provided that the set of transformations is chosen with care, a selection often performed manually. Automatic data augmentation aims at automating this process. However, most recent approaches still rely on some prior information; they start from a small pool of manually-selected default transformations that are either used to pretrain the network or forced to be part of the policy learned by the automatic data augmentation algorithm. In this paper, we propose to directly learn the augmentation policy without leveraging such prior knowledge. The resulting bilevel optimization problem becomes more challenging due to the larger search space and the inherent instability of bilevel optimization algorithms. To mitigate these issues (i) we follow a successive cold-start strategy with a Kullback-Leibler regularization, and (ii) we parameterize magnitudes as continuous distributions. Our approach leads to competitive results on standard benchmarks despite a more challenging setting, and generalizes beyond natural images.[1]

1. Introduction

Data augmentation, which encourages predictions to be stable with respect to particular image transformations, has become an essential component in visual recognition systems. While the data augmentation process is conceptually simple, choosing the optimal set of image transformations for a given task or dataset is challenging. For instance, designing a good set for ImageNet [4] or even CIFAR-10/100 [13] has been the result of a long-standing research effort. Whereas data augmentation strategies that have been chosen by hand for ImageNet have been used successfully for many recognition tasks involving natural images, they may fail to generalize to other domains such as medical imaging, remote sensing or hyperspectral imaging.

This has motivated automating the design of data augmentation strategies [10,12,14–16,19,27,31]. Those are often represented as a stochastic *policy* that randomly draws a combination of transformations along with their magnitudes from a large predefined set, each time an image is sampled. The goal becomes to learn strategies that effectively compose multiple transformations, which is a challenging task given the large search space of augmentations.

A natural framework for learning the parameters of this policy is that of bilevel optimization. Intuitively, one looks for the best possible policy such that a neural network trained with this policy on a training set (inner problem) generalizes well on a distinct validation set (outer problem). Optimizing the resulting formulation is challenging as the outer problem depends on the solution of the inner problem. Classical techniques for solving this bilevel problem, such as unrolled optimization, can become highly unstable as the network weights become progressively suboptimal for the current policy during the learning process.

Moreover, augmentations are often non-differentiable in the parameters of the policy, thus requiring techniques other than direct differentiation, such as Bayesian optimization [15], gradient approximations (*e.g.* RELAX [7]), or the score method / REINFORCE [28] algorithm. While these techniques bypass the differentiability issues, they can suffer from large bias or variance. As a result, learning augmentation policies is a difficult problem whose challenges are exacerbated by the inherent instability of the optimization techniques developed to solve bilevel problems, such as unrolled optimization [1].

A standard way to improve stability and make the automatic data augmentation problem simpler is to reduce the search space. This is often achieved by learning the policy on top of “default” transformations such as Cutout [5], random cropping and resizing, or color jittering, all known to be well-suited to natural images which compose standard benchmarks such as CIFAR or ImageNet, or by discarding transformations known to be harmful such as Invert. Fixing some of the transformations and removing others mitigate the challenges inherent to learning a compo-

[1] Project page: https://europe.naverlabs.com/slack

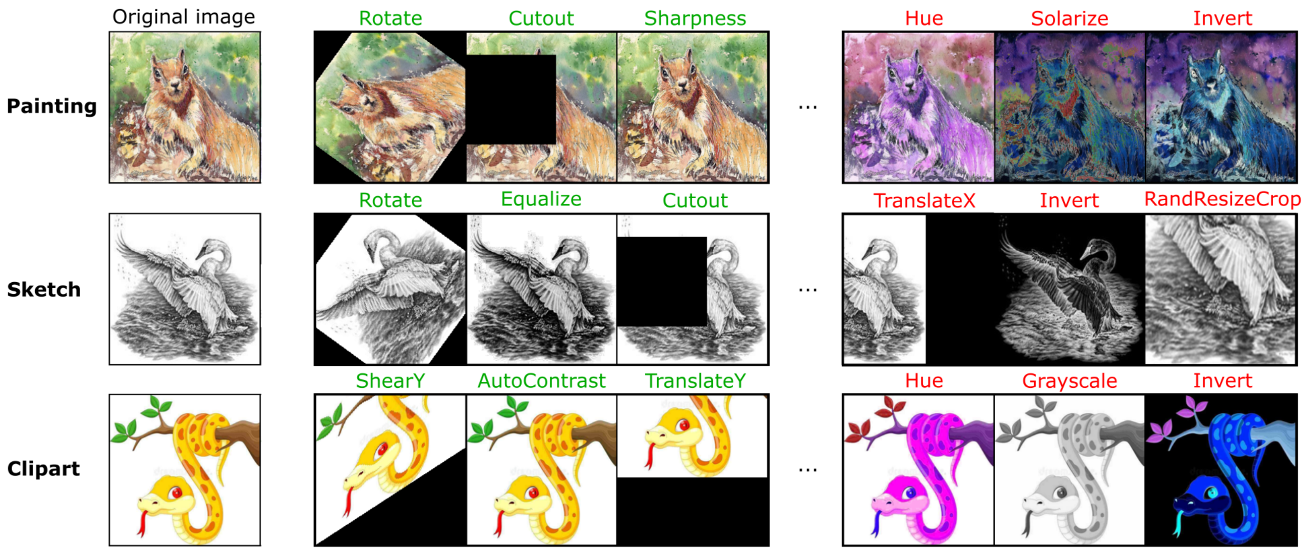


Figure 1. For different domains of the DomainNet dataset [21] (one per line), we show an image from that domain (left) and that image transformed using the three most likely (middle) and the three least likely (right) augmentations for that domain, as estimated by SLACK.

sition of transformations. TrivialAugment [19] also shows that state-of-the-art results can be achieved on these previous benchmarks simply by directly applying the policy classically used for initializing auto-augmentation models, up to minor modifications. Moreover, all methods rely on carefully chosen ranges that constraint the transformation's magnitudes. Despite its effectiveness, manually selecting default transformations and magnitude ranges restricts the applicability of such policies to natural images and prevents generalisation to other domains.

In this paper, our goal is to choose augmentation strategies without relying on default transformations nor on hand-selected magnitude ranges known to suit common benchmarks. To achieve this objective, we first introduce a simple interpretable model for the augmentation policies which allows learning both the frequency by which a given augmentation is selected and the magnitude by which it is applied. Then, we propose a method for learning these augmentation policies by solving a bilevel optimization problem. Our method relies on the REINFORCE technique for computing the gradient of the policy and on unrolled optimization for learning the policy, both of which can result in instabilities and yield high variance estimates.

To address these issues, we introduce an efficient multi-stage algorithm with a cold-start strategy and a Kullback-Leibler (KL) regularization that are designed to improve the stability of the process for learning the data augmentation policy. More precisely, the algorithm first pre-trains a network with a data augmentation policy uniformly sampling over all transformations. Then, each stage uses a “cold-start” strategy by restarting from the pre-trained network and performs incremental updates of the current policy.

This multi-stage approach with cold start prevents the network from becoming progressively suboptimal as the policy is updated using unrolled optimization. The KL regularization defines a trust region for the policy to compensate for the possibly high variance of gradient estimates obtained using the REINFORCE technique and encourages exploration during training, preventing collapse to trivial solutions. This regularization is inspired by proximal point algorithms in convex optimization [22], which have also been successful in reinforcement learning tasks [23].

By combining the regularized multi-stage approach with our interpretable model of the augmentation policies, we obtained the proposed SLACK method, which stands for *Stable Learning of Augmentations with Cold-start and Kullback-Leibler regularization*. SLACK is an efficient data augmentation learning method that is able to address the challenging bilevel optimization problem of learning a stochastic data augmentation policy without relying strongly on prior knowledge. Figure 1 illustrates the transformations found by SLACK to be most important / detrimental on a dataset of different domains including non-natural images.

To summarize, our contribution is threefold. (i) We propose a simple and interpretable model of the policies which allows learning both frequency and magnitudes of the augmentations. (ii) We propose a regularized multi-stage strategy to improve the stability of the bilevel optimization algorithm used for solving the data augmentation learning problem. (iii) We evaluate our method on challenging experimental settings, and show that it finds competitive augmentation strategies on natural images without resorting to prior information and generalizes to other domains.

2. Related Work

The choice of image transformation (also known as data augmentation) has become central in the design of computer vision pipelines. To remove the burden of manual selection, automatic data augmentation strategies have been proposed [20, 26]. AutoAugment [2], one of the earliest methods, uses for instance a recurrent neural network for designing the augmentation policy. Because such an approach requires retraining a prediction model at each iteration, it is prohibitively slow, and more efficient alternatives have been proposed. They aim at reducing the training cost using, *e.g.*, population-based training [12], Bayesian optimization [15], and more recently, gradient-based approaches based on bilevel optimization [9, 10, 14, 15, 18, 27, 30], relying on various gradient estimation techniques such as RELAX [7] or the Score method [28]. While the former is inherently biased, the latter is theoretically exact, but has a high variance when approximated in the context of stochastic optimization. Therefore, these approximations may lead to diverging gradient updates. Our method alleviates this by introducing a KL regularization that defines a trust-region for the policy.

Automatic augmentation using prior knowledge. Most previous works learn augmentations using a small network learned on a subset of the dataset of interest, before retraining the prediction model on a larger network using the full (augmented) data. This choice is appealing to recent gradient-based methods [9, 14] as the search phase for an augmentation policy is often reduced to minutes. Nevertheless, [10, 30] have observed that policies found with such a reduced setup may be suboptimal compared to approaches exploiting full datasets for training both the augmentation policy and the prediction model. This observation was confirmed in [3], which shows that a naive grid search could actually yield state-of-the-art results when directly training on the full-size network and the full data. These results are however obtained at the expense of using strong prior knowledge: augmentation policies are applied on top of default transformations that are manually and independently chosen for each benchmark. Lately, [19] has shown that with a few additional careful choices regarding the augmentation policies, applying a single random transformation on top of the default ones could lead to state-of-the-art results.

To avoid relying on default augmentations, DeepAA [31] has recently proposed a greedy approach that is able to learn these transformations. Yet, learning is performed after a "pre-training" phase leveraging the usual default transformations. Moreover, while such a greedy approach simplifies the search procedure and reduces its stochasticity, the resulting computational cost is high. Instead, our approach improves stability and allows directly learning the joint probability of sampling multiple transformations, reducing the search time twofold compared to DeepAA.

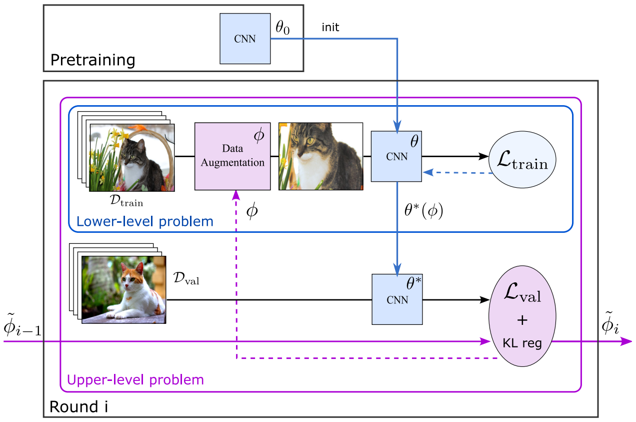


Figure 2. **Overview of our proposed SLACK method.** We learn a data augmentation policy parameterized by ϕ using bilevel optimization. The inner loop finds the optimal network parameter θ^* on images from $\mathcal{D}_{\text{train}}$. The outer loop trains on a disjoint set of images $\mathcal{D}_{\text{val}}$ using this network and finds the optimal transformation parameters ϕ. The method is enhanced with i) a cold-start strategy that structures the learning into rounds which share ϕ but restart the network from the pretrained one, and ii) a KL regularization.

3. Method

Our method, SLACK, defines an *augmentation policy*, which is a probabilistic model for generating data augmentations. The goal is to learn the parameters of this augmentation policy so as to improve the performance of the trained classifier on a held-out dataset. We first describe the augmentation policy (Sec. 3.1) and then formalize the problem of learning data augmentations with bilevel optimization (Sec. 3.2). We then describe our approach for solving such a bilevel optimization problem, which aims at stabilizing the optimization (Sec. 3.3).

3.1. Stochastic data augmentations policy

An augmentation function τ transforms an image x into another *augmented* image $\tau(x)$ of the same dimensions. We consider composite augmentations obtained by combining simpler augmentations selected from a finite set $\mathcal{S} = \{s_1, \ldots, s_N\}$ of N candidate *elementary transformations*, such as rotations, translations, shearing, etc. Each elementary transformation depends on a *magnitude* parameter m that controls the strength of the transformations, for instance, the angle by which an image is rotated. Magnitudes are normalized to be in the unit interval $[0, 1]$.

Augmentation policy. We define the augmentation policy as a probabilistic model p_ϕ that generates composite augmentations given some parameter ϕ to be learned. The model generates an augmentation in three steps: (1) it samples K elementary transformations $t_1, \ldots, t_K$ from $\mathcal{S}$ according to a categorical distribution p_π of parameter π, (2) it samples values for the magnitudes $m_1, \dots, m_K$ for each of the selected elementary transformations t_k according to a smoothed uniform distribution p_μ of parameter μ and (3) it composes the K elementary transformations to obtain the composite augmentation, with each t_k applied using its corresponding magnitude m_k. Therefore, the augmentation policy $p_\phi(\tau)$ takes the form

$$p_\phi(\tau) = \prod_{i=1}^{K} p_\pi(t_i) p_\mu(m_i|t_i), \tag{1}$$

where the parameters $\phi = (\pi, \mu)$ are learned jointly. Next, we describe the sampling of the transformations and of their magnitudes.

Sampling transformations. We sample elementary transformations t_k with replacement from a categorical distribution Cat_{π_k} of dimension N parameterized by a logit vector $\pi_k{:=}(\pi_{k,n})_{1\leq n\leq N}$. The probability $p_\pi(t_1, \dots, t_K)$ of sampling the K transformations is given by:

$$p_\pi(t_1, \dots, t_K) = \Pi_{k=1}^{K} \mathrm{Cat}_{\pi_k}(t_k), \tag{2}$$

where we collect all logits to form a parameter matrix π of size $K \times N$. These parameters are learned.

Sampling magnitudes. The magnitudes of each elementary transformation s_i in S are sampled from a smoothed uniform distribution between $[0, \mu_i]$ whose upper-bound μ_i is learned. More precisely, the distribution's density is defined as

$$p_{\mu_i}(m_i) = \frac{1}{\mu_i} \int_0^{\mu_i} \mathcal{N}(m_i, \sigma)(u) du,$$

where $\mathcal{N}(m_i, \sigma)$ is the Gaussian distribution of mean m_i and deviation σ. The density $p_{\mu_i}(m_i)$ approximates the uniform distribution $\frac{1}{\mu_i}\mathbf{1}_{[0,\mu_i]}$ as the deviation σ approaches 0. In practice, we set $\sigma = 0.1$ as we found it to achieve a good trade-off between smoothing and approximation.

Why a uniform distribution? We ran some ablations on previous methods (see supplementary) which suggest that a uniform sampling works on par with more elaborate sampling strategies, and that the magnitude range has more impact on the results.

3.2. Bilevel formulation for policy search

We consider a prediction task, such as predicting the class y of some natural image x using a model $f_\theta(x)$ with parameter θ. We are interested in finding the best policy parameter ϕ so that the prediction model f_θ, when trained using such policy on a training set $\mathcal{D}$ of input/output pairs (x, y), generalizes well on the test set $\mathcal{D}_{\text{test}}$. The problem naturally decomposes in two phases. During the *search phase*, the optimal augmentation policy p_ϕ is learned on $\mathcal{D}$. During the *evaluation phase*, the model is re-trained on $\mathcal{D}$ using p_ϕ and is then evaluated on $\mathcal{D}_{\text{test}}$. The *evaluation phase* is performed using standard optimization methods. However, the *search phase* requires solving a complex optimization problem that we describe next.

Search phase as a bilevel problem. The *search phase* naturally writes as a bilevel problem involving two interdependent losses: a lower-level loss $\mathcal{L}_{\text{train}}(\theta, \phi)$ for learning an optimal model parameter $\theta^\star(\phi)$ obtained using the augmentation policy p_ϕ and an upper-level loss $\mathcal{F}(\phi)$ for learning the policy parameter ϕ by evaluating the optimal model with parameter $\theta^\star(\phi)$. Each of these objectives is evaluated on two separate splits of the available data $\mathcal{D}$: a training split $\mathcal{D}_{\text{train}}$ for the lower-level loss and a validation split $\mathcal{D}_{\text{val}}$ for the upper-level loss. Below, we describe both losses.

Lower-level loss. We first introduce the training loss $\ell_{\text{train}}(\theta, \tau)$ when only a fixed augmentation τ is used:

$$\ell_{\text{train}}(\theta, \tau) := \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{train}}} \left[\ell(y, f_\theta(\tau(x)))\right],$$

where (x, y) is an (image,label) pair drawn from $\mathcal{D}_{\text{train}}$ and ℓ is a pointwise prediction loss (*e.g.* cross-entropy). We then define the training loss $\mathcal{L}_{\text{train}}(\theta, \phi)$ for an augmentation policy p_ϕ by taking the expectation of $\ell_{\text{train}}(\theta, \tau)$ over augmentations τ sampled according to the policy p_ϕ:

$$\mathcal{L}_{\text{train}}(\theta, \phi) := \mathbb{E}_{\tau\sim p_\phi} \left[\ell_{\text{train}}(\theta, \tau)\right].$$

Hence, for a given policy p_ϕ, the goal is to learn the optimal model parameter $\theta^\star(\phi)$ by minimizing $\mathcal{L}_{\text{train}}(\theta, \phi)$ over θ.

Upper-level loss. We first denote by $\mathcal{L}_{\text{val}}(\theta)$ the validation loss for a given model of parameter θ:

$$\mathcal{L}_{\text{val}}(\theta) := \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{val}}} \left[\ell(y, f_\theta(x))\right].$$

The validation loss $\mathcal{L}_{\text{val}}(\theta)$ is computed over the validation set $\mathcal{D}_{\text{val}}$ *without* applying any augmentation and thus provides a proxy for the performance on the test dataset. We then define the upper-level loss to be the validation loss of an optimal model $\theta^\star(\phi)$ learned using a policy p_ϕ:

$$\mathcal{F}(\phi) := \mathcal{L}_{\text{val}}(\theta^\star(\phi)). \tag{3}$$

While optimizing the lower-level loss is relatively standard, minimizing the upper-level loss $\mathcal{F}$ is more challenging due to the complex dependence of the optimal model parameter $\theta^\star(\phi)$ on the policy. Next, we describe our proposed algorithm for solving the bilevel problem.

3.3. SLACK algorithm

We propose Algorithm 1 for learning the optimal policy during the search phase. SLACK first pre-trains the prediction model using the objective $\mathcal{L}_{\text{train}}(\theta, \phi_{\text{uniform}})$ for an initial policy parametrized by ϕ_{uniform} which samples uniformly among all elementary transformations. It then performs n_{rounds} rounds to update the parameters θ and ϕ jointly using a bilevel optimization algorithm. This approach is reminiscent of the one of AutoAugment (AA) [3] that fully re-trains the network for each policy update. Yet, it is several orders of magnitude faster than AA, as it benefits from pre-training and from our bilevel optimization.

SLACK relies on two strategies to ensure the stability of the parameter updates during each round: a *cold-start strategy* for the prediction model and an *anchoring strategy* for the policy. The *cold-start* initializes the prediction model at the beginning of each round using a pre-trained model θ_0. *Anchoring* is achieved by encouraging the current policy to remain close to some *anchor policy* $p_{\tilde{\phi}}$. We set $\tilde{\phi}$ to the current policy parameter ϕ at the beginning of each round. During the first n_{retrain} steps of each round, the algorithm only updates the model parameter using a stochastic estimate $\hat{g}_\theta$ of $\nabla_\theta \mathcal{L}_{\text{train}}(\theta, \phi)$ while maintaining the policy fixed. Then for the last $n_{\text{total}} - n_{\text{retrain}}$ steps, the algorithm alternates between model updates and policy updates. The policy updates aim to minimize the sum of the upper-level objective $\mathcal{F}$ and an anchoring $d(p_\phi, p_{\tilde{\phi}}) := \mathrm{KL}(p_\pi, p_{\tilde{\pi}})$ encouraging the policy p_ϕ to remain close to the *anchor policy* $p_{\tilde{\phi}}$. These updates are obtained using a stochastic estimate $\hat{G}_\phi$ along with the exact gradient of the KL regularization which admits a closed-form expression. Next we explain how we estimate the gradients $\hat{g}_\theta$ and $\hat{G}_\phi$ and discuss the effect of cold-start and KL-regularization.

Algorithm 1 SLACK

```
**Initialize policy parameter** $\phi \leftarrow \phi_{\text{uniform}}$.
**Pre-training:** $\theta_0 \leftarrow$ optimize $(\mathcal{L}_{\text{train}}(\theta, \phi))$.
**for** $i \in \{1, ..., n_{\text{rounds}}\}$ **do**
    **Cold-start:** $\theta \leftarrow \theta_0$.
    **Update anchor policy:** $\tilde{\phi} \leftarrow \phi$.
    **for** $j \in \{1, ..., n_{\text{total}}\}$ **do**
        Compute stochastic gradient $\hat{g}_\theta \approx \nabla_\theta \mathcal{L}_{\text{train}}(\theta, \phi)$
        Update $\theta$: $\theta \leftarrow \theta - \eta \hat{g}_\theta$.
        **if** $j > n_{\text{retrain}}$ **then**
            Compute stochastic gradient $\hat{G}_\phi \approx \nabla_\phi \mathcal{F}(\phi)$
            Update $\phi$: $\phi \leftarrow \phi - \alpha(\hat{G}_\phi + \lambda \nabla_\phi d(p_\phi, p_{\tilde{\phi}}))$.
        **end if**
    **end for**
**end for**
```

Gradient estimation. Algorithm 1 requires estimating the gradient of $\mathcal{F}(\phi)$, which is challenging given the complex dependence of the upper-level loss on the policy p_ϕ through the optimal model parameter $\theta^\star(\phi)$ learned using such a policy. In line with previous works [9, 14, 18, 27], we approximate the optimal model parameter $\theta^*(\phi)$ with a simpler function $\hat{\theta}(\phi)$ that is easier to compute:

$$\hat{\theta}(\phi) := \theta - \eta \nabla_\theta \mathcal{L}_{\text{train}}(\theta, \phi). \tag{4}$$

Eq. (4) corresponds to one gradient step to optimize the lower-level loss starting from the current parameter θ and ϕ and using step-size $\eta > 0$. By keeping track of the dependence in ϕ and exploiting the fact that the augmentation policy p_ϕ has a score $\nabla_\phi \log p_\phi(\tau)$ that can be computed explicitly using Eq. (1), we can use the REINFORCE/Score method [6] to derive a closed-form expression for $\nabla_\phi \hat{\theta}(\phi)$ which will serve for approximating the gradient of $\mathcal{F}$:

$$\nabla_\phi \hat{\theta}(\phi) = -\eta \mathbb{E}_{\tau \sim p_\phi} \left[\nabla_\theta \ell_{\text{train}}(\theta, \tau) \nabla_\phi \log p_\phi(\tau)^\top \right].$$

Then, we approximate the upper-level loss $\mathcal{F}(\phi)$ with a simpler function $\hat{\mathcal{F}}(\phi) := \mathcal{L}_{\text{val}}(\hat{\theta}(\phi))$ and the gradient $\nabla_\phi \mathcal{F}(\phi)$ with $\nabla_\phi \hat{\mathcal{F}}(\phi)$ which is obtained using the chain rule:

$$\nabla_\phi \mathcal{F}(\phi) \approx \nabla_\phi \hat{\mathcal{F}}(\phi) = \nabla_\theta \mathcal{L}_{\text{val}}(\hat{\theta}(\phi))^\top \nabla_\phi \hat{\theta}(\phi). \tag{5}$$

The above expression requires only first-order derivatives and matrix-vector products, which is amenable to efficient implementation using automatic differentiation softwares.

Stochastic gradient estimates. In practice, we replace all expectations by estimates on a batch of data and sampled augmentations. More precisely, to compute the approximation $\hat{g}_\theta$ to $\nabla_\theta \mathcal{L}_{\text{train}}(\theta, \phi)$, we sample B_{aug} augmentations from p_ϕ and then apply each of them to a batch of training data B_{train} from $\mathcal{D}_{\text{train}}$. Using the same batch of data and augmentation, we approximate $\hat{\theta}(\phi)$ and $\nabla_\phi \hat{\theta}(\phi)$ appearing in Eq. (5). Finally, we use a batch B_{val} of data from $\mathcal{D}_{\text{val}}$ to estimate $\nabla_\theta \mathcal{L}_{\text{val}}(\hat{\theta}(\phi))$ and compute $\hat{G}_\phi$, which is a stochastic estimate of $\nabla_\phi \hat{\mathcal{F}}(\phi)$ in Eq. (5).

Cold-start. The cold-start strategy allows to re-train the model at each round with the current augmentation policy starting from the pre-trained model. This approach is closer to the original bilevel formulation which implies finding an optimal prediction model for each policy. Initializing with a pre-trained model yields computational gain as fewer iterations are needed to optimize the model. We could instead use a *warm-start* strategy which initializes the model at each round with the learned model at the previous round. Yet we experimentally observe that such approach progressively leads to overfitting and degrades the quality of the learned policies (see supplementary).

Anchoring using KL regularization. We experimentally found that adding an anchoring $d(p_\phi, p_{\tilde{\phi}}) := \mathrm{KL}(p_\pi, p_{\tilde{\pi}})$ with strength parameter λ when updating the policy prevents the algorithm from collapsing towards trivial policies. The anchoring affects only the categorical distribution p_π. For the magnitudes p_μ, we did not use anchoring as it is ill-defined for a uniform distribution. Instead, we simply used smaller step-sizes. Our approach takes inspiration from Proximal Policy Optimization [23] used in the context of reinforcement learning which is known to improve policy search.

4. Experiments

In this section, we first briefly describe our experimental setup (Sec 4.1). Then we evaluate our approach on several standard benchmarks composed of natural images (Sec 4.2) as well as on a benchmark with other domains (Sec 4.3). We finally report some ablation studies (Sec 4.4).

4.1. Experimental setup

Benchmarks. We first evaluate our model on three standard benchmarks, CIFAR10 [13], CIFAR100 [13] and ImageNet-100 [25], all composed of natural images. To study how well our method generalizes beyond natural images, we also evaluate on the DomainNet dataset [21], which contains 345 classes for 6 different domains. To ensure our protocol uses a similar number of training images for each domain, we use a reduced set of 50,000 training images for the two largest domains (real, quickdraw) and leave the remaining images for testing. For the other domains, we isolate 20% of the data for testing.

Architectures. CIFAR10/100 are evaluated with two architectures that are standard for automatic data augmentation: WideResNet-40x2 and WideResNet-28x10 [29]. Unlike previous works whose search phase is only conducted with the smaller WideResNet-40x2, we search and evaluate with the same architecture, as we found it to be better (see Sec. 4.4). ImageNet-100 and DomainNet are evaluated with a ResNet-18 [11] architecture.

Transformation space. Our data augmentation search space is composed of the standard pool of 15 transformations: *Identity, ShearX, ShearY, TranslateX, TranslateY, Rotate, AutoContrast, Equalize, Invert, Solarize, Posterize, Contrast, Brightness, Sharpness, Color*. We add to this pool the transformations that previous methods usually apply by default: Cutout and RandomCrop for CIFAR, RandomResizeCrop for ImageNet, Grayscale for DomainNet. Following standard practice, when RandomResizeCrop is sampled, it is always applied first. We learn the range of its scale parameter. We do not add ColorJitter that is also applied by default in prior work for ImageNet, as it is already a mix of Brightness, Contrast and Color. However we add Hue, which is one component of ColorJitter and never applied by default. Following prior work, the magnitudes are mapped to [0,1]. After mapping, μ is initialized at 0.75 to favour exploration (see details in supplementary). We also uniformly sample magnitudes for Cutout and RandomCrop, whereas their value is hand-picked in prior work. Since the datasets are horizontally symmetric, we follow common practice and apply flip by default.

Policy search. We apply a train/val split of 0.5/0.5, meaning that half of the data is used to train the model parameters while the other half is used to learn the augmentation policy. Pre-training is done in the same setting as the evaluation (see next paragraph), except that we train only with the train data in the train/val split of the search phase. We use SGD with momentum for the optimization of the validation and training losses. For the latter, we use the same weight decay as for the final policy evaluation. We sample 8 different augmentations for computing the expectation that is needed for the stochastic gradient estimate, as detailed in Sec. 3.3.

Policy evaluation. We evaluate our models following the framework of TrivialAugment [19]. The corresponding hyperparameters can be found in the supplementary. We evaluate each policy with 4 independent runs, meaning that our results are averaged over a total of $4 \times 4 = 16$ evaluations. Our Uniform policy (corresponding to SLACK's initialization) and our reported results on TrivialAugment are evaluated with 8 independent runs. We also report a confidence interval which contains the true mean with probability $p = 95\%$, under the assumption of normally distributed accuracies.

4.2. Comparison with the state of the art

We compare our method with a Uniform augmentation policy as well as many previous approaches for data augmentation, including AutoAugment (AA) [2], Fast AutoAugment (FastAA) [15], Differentiable Automatic Data Augmentation (DADA) [14], RandAugment (RA) [3], Teach Augment [24], UniformAugment [20], TrivialAugment (TA) [19], and Deep AutoAugment (DeepAA) [31].

For each method, we indicate the total number of composed transformations, and the number of hard-coded transformations among those (Tables 1 and 2). For SLACK, we evaluate the policies obtained from 4 independent search runs (each with 4 different train/val splits) to assess the robustness of our approach. We follow the same process when reproducing DeepAA on CIFAR10/100. Note that all previous methods use a single run for search, before evaluating the policy with one or multiple runs. We report 95% confidence intervals for those evaluating with multiple runs.

The supplementary provides qualitative results showing the evolution of the probability distributions over the transformations and the final estimated policies for all datasets.

CIFAR. In Table 1, we observe that, despite not hard-coding Cutout and RandomCrop in our policy, our method is competitive on both CIFAR10 and CIFAR100.

We found that, in general, Cutout and Rotate are selected with a high probability, while the Invert transformation is systematically discarded (see supplementary). This is consistent with the choices made in practice by prior work of adding/removing these transformations manually.

We observe a mismatch between DeepAA's reported results [31], and those we obtain when evaluating their approach on multiple search runs, using the author's code and following their recommendations. This is likely due to the stochasticity of the search procedure.

	# Augmentations		CIFAR10		CIFAR100	
	Total	Hard-coded	WRN-40-2	WRN-28-10	WRN-40-2	WRN-28-10
AA [2]	4	2	96.3	97.4	79.3	82.9
FastAA [15]	4	2	96.4	97.3	79.4	82.7
DADA [14]	4	2	96.4	97.3	79.1	82.5
RA [3]	4	2	-	97.3	-	83.3
TeachA [24]	4	2	-	97.5	-	83.2
UniformAugment [17]	4	2	96.25	97.33	79.01	82.82
TA (Wide) [19]	3	2	96.32 ± .05	97.46 ± .06	79.86 ± .19	84.33 ± .17
Uniform policy	3	0	96.12 ± .08	97.26 ± .07	78.79 ± .25	82.82 ± .24
DeepAA [31]	6**	0*	-	97.56 ± .14	-	84.02 ± .18
DeepAA (reproduced)†	6**	0*	96.25 ± .11	97.27 ± .11	79.26 ± .35	83.38 ± .33
SLACK (Ours)	3	0	96.29 ± .08	97.46 ± .06	79.87 ± .11	84.08 ± .16

Table 1. Test accuracies on CIFAR10 and CIFAR100. For SLACK and DeepAA (reproduced) we conduct 4 independent searches, and evaluate each policy with 4 evaluation runs, meaning that we report averages over 16 evaluations. TA and DeepAA are also evaluated with multiple evaluation runs. Results for the remaining methods are reported from the corresponding papers and based on a single run.

*: DeepAA uses hard-coded transformations for pre-training. **: DeepAA learns random flipping unlike other baselines. † We evaluate the policies found from 4 independent search runs as we do for SLACK, using the code from the authors and following their recommendations.

	# Augmentations		ImageNet-100
	Total	Hard-coded	ResNet18
TA (RA) [19]†	5	4	85.87 ± .30
TA (Wide) [19]†	5	4	86.39 ± .18
Uniform policy	3	0	85.78 ± .32
SLACK	3	0	86.06 ± .11

Table 2. Test accuracies on ImageNet-100.

ImageNet-100. Results for ImageNet-100 are reported in Table 2. We compare SLACK to our Uniform policy and to TrivialAugment (RA) and (Wide) variants, the latter using larger magnitude ranges for its random transformation. SLACK's results lie in between both variants and improve over our Uniform policy.

Interestingly, for ImageNet-100, we found that RandomResizeCrop is not favoured during the search phase (see supplementary), suggesting that it is not critical for ImageNet-100. Instead the performance gap between TA (Wide) and TA (RA) suggest that harder transformations are key to a better performance for this dataset.

4.3. Beyond natural images

For the DomainNet dataset, we compare SLACK to a Uniform policy, to the augmentations used by DomainBed [8] for domain generalization, and to the TrivialAugment (RA) and (Wide) methods with their ImageNet and CIFAR default settings. Results can be found in Table 3.

DomainBed uses the same default transformations as TA ImageNet together with Grayscale, but with smaller magnitudes and unlike TA, does not add a random transformation. Yet it strongly overfits and performs much lower than TA. This suggests that augmentations well suited for domain generalization do not perform well on the individual tasks. TA (Wide) ImageNet consistently outperforms all other TA flavors. This further justifies the need to learn the magnitude range and to eliminate any manual range selection process.

SLACK is a close second, yet it learns the policy end-to-end. The learned policies are illustrated as pie charts in Fig. 3. The slices represent the probability π over the different transformations while their radius represent the corresponding magnitudes. They differ from a domain to another, suggesting that the gain compared to the initialization (*i.e.* Uniform policy) results from SLACK's ability to learn and adapt to each domain.

4.4. Ablation study

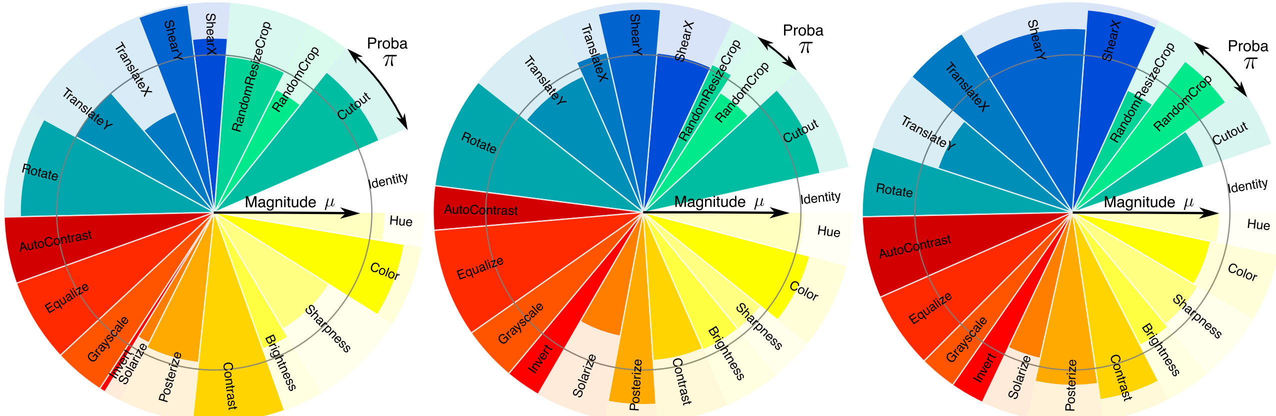
In this section, we evaluate our contributions and main design choices: the network architecture used for search, the KL regularization, and the benefits of learning π and μ. More ablations can be found in the supplementary. Note that hyperparameters are adjusted to each baseline included in the comparison, to make them as competitive as possible.

Network architecture for search. In prior works, the search phase (when there is one) is conducted on the smaller WideResNet-40x2 architecture for CIFAR10 and CIFAR100, and the learned policy is evaluated for both WideResNet-40x2 and WideResNet-28x10. Table 4 shows that for SLACK, searching directly with WideResNet-28x10 gives the best results for that architecture.

KL regularization. We compare SLACK with a flavor that does not apply KL-regularization. For the latter, we reduce the outer learning rate so that the augmentation policies with and without regularization evolve at similar speeds. Results in Table 5 show that our regularization is beneficial.

	# Augmentations		Real-50k	Quickdraw-50k	Inforgraph	Sketch	Painting	Clipart	Average
	Total	Hard-coded							
DomainBed[†]	5	5	62.54 ± .15	66.54 ± .91	26.76 ± .36	59.54 ± .37	58.31 ± .25	66.23 ± .10	57.23 ± .18
TA (RA) ImageNet[†]	5	4	70.85 ± .13	67.85 ± .07	**35.24 ± .19**	65.63 ± .11	64.75 ± .18	70.29 ± .18	62.43 ± .05
TA (Wide) ImageNet[†]	5	4	**71.56 ± .07**	68.60 ± .05	**35.44 ± .33**	**66.21 ± .16**	**65.15 ± .20**	71.19 ± .19	**63.03 ± .07**
TA (RA) CIFAR[†]	3	2	70.28 ± .08	68.35 ± .07	33.85 ± .21	64.13 ± .12	64.73 ± .17	70.33 ± .21	61.94 ± .05
TA (Wide) CIFAR[†]	3	2	71.12 ± .10	**69.29 ± .05**	34.21 ± .29	65.52 ± .25	64.81 ± .14	71.01 ± .21	62.66 ± .07
Uniform policy	3	0	70.37 ± .08	68.27 ± .06	34.11 ± .21	65.22 ± .17	63.97 ± .24	72.26 ± .14	62.37 ± .06
SLACK (ours)	3	0	71.00 ± .13	68.14 ± .11	34.78 ± .18	65.41 ± .16	64.83 ± .12	**72.65 ± .20**	62.80 ± .06

Table 3. Test accuracies on DomainNet.



(a) Painting (b) Sketch (c) Clipart

Figure 3. Policies found on DomainNet for the best search split. Gray circle: initial magnitude upper-bounds. Radius of each pie: learned upper-bounds. Size of each pie: probability of each transformation, averaged over the three composite distributions. Transformations which are parameter-free, *AutoContrast, Equalize, Grayscale, and Invert,* are displayed with maximal magnitude upper-bound.

Search architecture	CIFAR10	CIFAR100
WRN-40-2	97.43 ± .04	83.94 ± .20
WRN-28-10 (ours)	97.46 ± .06	84.08 ± .16

Table 4. CIFAR10/100 accuracy evaluated with WRN-28-10: impact of using a smaller architecture for the search phase.

	CIFAR10		CIFAR100	
SLACK variant	WRN-40-2	WRN-28-10	WRN-40-2	WRN-28-10
without KL	96.27 ± .05	97.06 ± .11	79.61 ± .13	83.79 ± .19
with KL (ours)	96.29 ± .08	97.46 ± .06	79.87 ± .11	84.08 ± .16

Table 5. CIFAR10/100 accuracy with/without KL regularization.

Joint learning of π and μ. Lastly, we study how beneficial jointly learning our augmentation parameters is compared to the initial Uniform policy and to a setting where only π or μ is learned. Results can be found in Table 6.

5. Conclusion

In this paper, we address the task of automatic data augmentation. Considering the more challenging bilevel optimization problem that arises when the search space is not reduced with default transformations, our proposed SLACK method tackles the resulting stability issues thanks to a multi-stage approach based on cold-start, coupled with a KL-regularization. Combined, they allow to reduce the variance of the gradient estimate and to better control the optimization process. We have experimentally observed that our method performs on par with recent approaches leveraging prior knowledge. It has also proved versatile enough to select domain-specific transformations when confronted to non-natural images.

	CIFAR10		CIFAR100	
SLACK variant	WRN-40-2	WRN-28-10	WRN-40-2	WRN-28-10
Uniform policy	96.22 ± .10	97.38 ± .05	79.07 ± .24	83.26 ± .17
μ only	96.20 ± .08	97.42 ± .05	79.22 ± .17	83.57 ± .18
π only	96.22 ± .09	97.35 ± .04	79.36 ± .11	83.45 ± .15
π and μ (ours)	96.29 ± .08	97.46 ± .06	79.87 ± .11	84.08 ± .16

Table 6. CIFAR10/100 accuracy when only learning part of the policy parameters

Acknowledgments. This work was supported by ANR 3IA MIAI@Grenoble Alpes (ANR-19-P3IA-0003) and performed using HPC resources from GENCI–IDRIS (Grant 2022-AD011013343).

References

[1] Michael Arbel and Julien Mairal. Non-convex bilevel games with critical point selection maps. *preprint arXiv:2207.04888*, 2022. 1

[2] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018. 3, 6, 7

[3] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. RandAugment: Practical automated data augmentation with a reduced search space. In *Proc. NeurIPS*, 2020. 3, 5, 6, 7

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009. 1

[5] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 1

[6] Michael C Fu. Gradient estimation. *Handbooks in operations research and management science*, 13:575–616, 2006. 5

[7] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *Proc. ICLR*, 2018. 1, 3

[8] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *Proc. ICLR*, 2021. 7

[9] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Faster AutoAugment: Learning augmentation strategies using backpropagation. In *Proc. ECCV*, 2020. 3, 5

[10] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Meta approach to data augmentation optimization. In *Proc. WACV*, 2022. 1, 3

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 6

[12] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In *Proc. ICML*, pages 2731–2741, 2019. 1, 3

[13] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 1, 6

[14] Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy M. Hospedales, Neil Martin Robertson, and Yongxin Yang. DADA: differentiable automatic data augmentation. In *Proc. ECCV*, 2020. 1, 3, 5, 6, 7

[15] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast AutoAugment. In *Proc. NeurIPS*, 2019. 1, 3, 6, 7

[16] C. Lin, M. Guo, C. Li, X. Yuan, W. Wu, J. Yan, D. Lin, and W. Ouyang. Online hyper-parameter learning for auto-augmentation strategy. In *Proc. ICCV*, 2019. 1

[17] Tom Ching LingChen, Ava Khonsari, Amirreza Lashkari, Mina Rafi Nazari, Jaspreet Singh Sambee, and Mario A Nascimento. Uniformaugment: A search-free probabilistic data augmentation approach. *arXiv preprint arXiv:2003.14348*, 2020. 7

[18] Aoming Liu, Zehao Huang, Zhiwu Huang, and Naiyan Wang. Direct differentiable augmentation search. In *Proc. ICCV*, 2021. 3, 5

[19] Samuel G. Müller and Frank Hutter. TrivialAugment: tuning-free yet state-of-the-art data augmentation. In *Proc. ICCV*, 2021. 1, 2, 3, 6, 7

[20] Mattis Paulin, Jérôme Revaud, Zaid Harchaoui, Florent Perronnin, and Cordelia Schmid. Transformation pursuit for image classification. In *Proc. CVPR*, 2014. 3, 6

[21] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proc. ICCV*, pages 1406–1415, 2019. 2, 6

[22] R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976. 2

[23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2, 5

[24] Teppei Suzuki. Teachaugment: Data augmentation optimization using teacher knowledge. In *Proc. CVPR*, 2022. 6, 7

[25] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019. 6

[26] Riccardo Volpi and Vittorio Murino. Addressing model vulnerability to distributional shifts over image transformation sets. In *Proc. ICCV*, pages 7980–7989, 2019. 3

[27] Xiaoxing Wang, Xiangxiang Chu, Junchi Yan, and Xiaokang Yang. DAAS: Differentiable architecture and augmentation policy search. *arXiv preprint arXiv:2109.15273*, 2021. 1, 3, 5

[28] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992. 1, 3

[29] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proc. BMVC*, pages 87.1–87.12. BMVA Press, September 2016. 6

[30] Xinyu Zhang, Qiang Wang, Jian Zhang, and Zhao Zhong. Adversarial AutoAugment. In *Proc. ICLR*, 2020. 3

[31] Yu Zheng, Zhi Zhang, Shen Yan, and Mi Zhang. Deep AutoAugmentation. In *Proc. ICLR*, 2022. 1, 3, 6, 7