Supplementary Material for "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

In this supplementary, we first give a short overview of prior work's key aspects (Section A), then we provide additional details about the experimental protocol that was used to obtain our results (Section B) and analyse in more details the results reported in the paper and the impact of our approach on the stability of the search process (Section C).

Contents

A. Summary of prior art

Overall, methods for augmentation search mostly differ in key design choices that are highlighted in Table 1.

Method	Optimization	Prior-free*	Full set	Search time
AA [1]	Reinfor. learning	✗	✗	5000
PBA [5]	Population-based	✗	✗	5
Fast AA [7]	Bayesian optim.	✗	✗	3.5
Faster AA [4]	RELAX	✗	✗	0.23
DADA [6]	RELAX	✗	✗	0.1
RA [2]	Exhaustive search	✗	✓	25
TrivialA [8]	No optimization	✗	✓	NA
DeepAA [9]	Greedy algorithm	✓**	✓***	9
SLACK (Ours)	REINFORCE	✓	✓	4

* Prior-free refers to methods that do not use any default transformations.
** DeepAA does not use default transformations during search but during pretraining.
*** DeepAA pretrains on a reduced dataset.

Table 1. **Key aspects of automatic data augmentation methods.** Our approach, SLACK, tackles the corresponding bilevel optimization problem using the REINFORCE gradient estimator. It is prior-free (*i.e.* does not rely on default transformations) and can use the full training set while maintaining a reasonable search time (indicated in GPU hours, for search on CIFAR with WRN-40-2).

B. Experimental setup

In this section, we first describe the practical implementation of our gradient estimation, then we compare our magnitude ranges to those of other methods, and finally we describe in more detail our search and evaluation protocols.

B.1. Gradient estimation

In this section, we describe the practical implementation of the gradient estimates derived in our method section.

To optimize the augmentation policies, we minimize an approximation to the upper-level objective $\mathcal{F}(\phi) := \mathcal{L}_{\text{val}}(\theta^\star(\phi))$ defined as $\hat{\mathcal{F}}(\phi) := \mathcal{L}_{\text{val}}(\hat{\theta}(\phi))$, where we replaced the intractable lower-level solution $\theta^\star(\phi)$ by an approximate solution $\hat{\theta}(\phi)$. Such approximate solution is obtained by performing one gradient step to optimize the lower-level objective starting from the current parameter θ, i.e. $\hat{\theta}(\phi) := \theta - \eta \nabla_\theta \mathcal{L}_{\text{train}}(\theta, \phi)$. The gradient $\nabla_\phi \mathcal{F}$ is then naturally approximated by $\nabla_\phi \hat{\mathcal{F}}$ which is computed by applying the chain rule:

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

The Jacobian $\nabla_\phi \hat{\mathcal{F}}$ can be computed explicitly using the Score method which yields:

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

In practice, expectations over the data and augmentation policies are estimated with batches. At a given iteration, we sample B_{aug} augmentations from p_ϕ and then apply each of

them to a batch of training data B_{train} from $\mathcal{D}_{\text{train}}$ to approximate $\nabla_\phi \hat{\theta}(\phi)$. Finally, we use a batch B_{val} of data from $\mathcal{D}_{\text{val}}$ to estimate the validation loss. Denoting N_a, N_t, N_v the size of the augmentation, training and validation batches respectively, and

$$\hat{l}_{\text{val}}(\theta) := \frac{1}{N_v} \sum_{(x,y)\in B_{\text{val}}} \left[\ell(y, f_\theta(x))\right],$$

$$\hat{l}_{\text{train}}(\theta, \tau) := \frac{1}{N_t} \sum_{(x,y)\in B_{\text{train}}} \left[\ell(y, f_\theta(\tau(x)))\right],$$

our gradient estimate can be expressed as

$$\begin{aligned}\nabla_\phi \mathcal{F} &\approx -\frac{\eta}{N_a} \nabla_{\hat{\theta}} \hat{l}_{\text{val}}(\theta) \left(\sum_{\tau \in B_{\text{aug}}} \nabla_\theta \hat{l}_{\text{train}}(\theta, \tau) \nabla_\phi \log p_\phi(\tau)^T \right) \\ &= -\frac{\eta}{N_a} \sum_{\tau \in B_{\text{aug}}} \left(\nabla_\theta \hat{l}_{\text{val}}(\theta)^T \nabla_\theta \hat{l}_{\text{train}}(\theta, \tau) \right) \nabla_\phi \log p_\phi(\tau)\end{aligned}$$

In other words, the upper-level gradient is a weighted sum of the scores $\nabla_\phi \log p_\phi(\tau)$, with the weights representing the alignment between the gradients of i) the loss on the training data transformed with τ (evaluated at θ), and ii) the loss on the validation data (evaluated at $\hat{\theta}$, *i.e.* one step ahead).

In practice, the lower-level learning rate decreases with a cosine schedule. As we do not want our upper-level gradient updates to shrink, we set η to the *initial* value of the lower-level learning rate instead of its current value.

B.2. Magnitude ranges.

The ranges used for mapping the magnitudes to [0,1] vary across methods; we indicate this mapping for each method in Table 2. For transformations with respect to which the datasets naturally exhibit symmetries (Shear, Translate, Rotate, Enhance), once we have sampled a magnitude, we randomly select a direction. Note that SLACK's ranges are larger than the usual ones (*i.e.* those of TA (RA)), which gives more flexibility during the optimization of our magnitude upper-bounds μ. The latter is initialized at 0.75. Experimentally, we noted that this initialization should be high enough to favour exploration and avoid over-fitting during pre-training. We observed that any initialization in the $[0.75, 0.9]$ range consistently works well across datasets.

B.3. Image pre-processing

Table 3 indicates the image pre-processing choices on ImageNet-100 and DomainNet for TrivialAugment [8], DomainBed [3] and SLACK.

ImageNet-100 and DomainNet images have variable original sizes. In the literature, training images are commonly resized with RandomResizeCrop. For testing, TrivialAugment uses Resize(256)+CenterCrop((224,224)), preserving the aspect ratio, while DomainBed directly applies Resize((224,224)), degrading the aspect ratio but preserving the image content. For each method, we stick to the authors' choices, as we experimentally noted that they yield the best results (*e.g.* using TrivialAugment's pre-processing for DomainBed degrades the performance, and vice-versa).

For SLACK, which does not apply RandomResizeCrop by default, we preprocess the training data and validation/testing data in the same way. For training, random cropping is applied instead of center cropping to fully exploit the data. For ImageNet-100, we use TrivialAugment's pre-processing. For DomainNet, we select the pre-processing strategy by cross-validation after pre-training.

B.4. Policy search

Hyperparameters used for policy search are indicated in Table 4. They are chosen to satisfy two criteria that we found to be useful for obtaining a successful policy search: i) the validation loss after re-training should be similar (experimentally, slightly lower) to the one obtained after pre-training, and ii) the probability distributions should vary at the same speed for all datasets. Our learning rate is 4 times larger for re-training on CIFAR10 than on CIFAR100. We observed that gradients on CIFAR10 are 4 times smaller in norm than those on CIFAR100, and that re-scaling the updates allows satisfying i) and ii) empirically. For DomainNet, we adapt the number of re-training steps to the dataset size. A fixed lower-level learning rate for all datasets experimentally satisfies i). We observed that the lower-level gradients differ in scale for each dataset. Satisfying ii) requires re-scaling the KL regularisation and accordingly changing the upper-level learning rate (so that KL weight $\times$ upper-level lr is constant).

The upper-level learning rate indicated in the Tables is the one used for updating π. We divide it by 40 for the optimization of μ to ensure slower updates for the magnitude parameter which we found to be sensitive to variations (or by 10 for ablations removing the KL regularization).

B.5. Policy evaluation

The hyperparameters used for the evaluation phase are indicated in Table 5. For CIFAR10 and CIFAR100, we use the same hyperparameters as prior work.

C. Extended analysis

In this section, we further analyse the results of our search algorithm. We first illustrate the policies learned for all datasets. We then study in more detail the impact of our multi-stage approach with KL regularization, comparing it with single-stage (unrolled) and unregularized approaches and illustrating their instability.

Application	Transformation	Method			
		TA (RA)	TA (Wide)	DomainBed	Ours
Sampled	ShearX/Y	$[0, 0.3]$	$[0, 0.99]$	-	$[0, 1]$
	Translate X/Y	$[0, 0.45]^*$	$[0, 32\text{px}]^{**}$	-	$[0, 0.75]$
	Rotate	$[0, 30]$	$[0, 135]$	-	$[0, 90]$
	Posterize	$[4, 8]$	$[2, 8]$	-	$[2, 8]$
	Solarize	$[0, 255]$	$[0, 255]$	-	$[0, 255]$
	Enhance***	$[0, 0.9]$	$[0, 0.99]$	-	$[0, 0.99]$
	Cutout	$[0, 0.2]$	$[0, 0.6]$	-	$[0, 1]$
	RandCrop	-	-	-	$[0, 0.5]$
	RandResizeCrop	-	-	-	$[0.05, 1]$
Default ImageNet/DomainNet	ColorJitter****	$[0, 0.4]$	$[0, 0.4]$	$[0, 0.3]$	-
	RandResizeCrop	$[0.08, 1]$	$[0.08, 1]$	$[0.7, 1]$	-
Default CIFAR	Cutout	0.5	0.5	NA	-
	RandCrop	0.125	0.125	NA	-

Table 2. Our magnitude ranges compared to those used by other methods.
* TrivialAugment [8] uses $[-0.31, 0.31]$, ** TrivialAugment [8] sets the upper-bounds in pixels, not in proportion
*** Color, Contrast, Brightness, Sharpness, **** Color, Contrast, Brightness

Dataset	Model	Train	Test
ImageNet-100	TrivialAugment	RandResizeCrop((224,224))	Resize(256)+CenterCrop((224,224))
	SLACK	Resize(256)+RandomCro((224,224))	Resize(256)+CenterCrop((224,224))
DomainNet	TrivialAugment ImageNet	RandResizeCrop(224,224)	Resize(256)+CenterCrop(224,224)
	TrivialAugment CIFAR	Resize(256)+RandomCrop((224,224),padding=28)	Resize(256)+CenterCrop((224,224))
	DomainBed	RandResizeCrop((224,224))	Resize((224,224))
	SLACK (Clipart, Sketch, Quickdraw)	Resize((224,224))	Resize((224,224))
	SLACK (Painting, Infograph, Real)	Resize(256)+RandomCrop((224,224))	Resize(256)+CenterCrop((224,224))

Table 3. Image pre-processing on ImageNet-100 and DomainNet.

Dataset	Network	Re-train iter	Unrolled iter	Batch size	Lower lr	Upper lr	KL weight $\times$ Upper lr
CIFAR10/100	WRN-40-2/WRN-28-10	1000	400	8×128	0.4 / 0.1	1	0.02
ImageNet-100	ResNet-18	2000	800	8×256	0.1	0.5	0.005
DomainNet	ResNet-18	800 - 1200	400	8×128	0.1	0.625 - 1.25	0.01

Table 4. Hyperparameters used for search on CIFAR, ImageNet-100 and DomainNet.

Dataset	Network	Epochs	Batch size	Learning rate	Weight decay
CIFAR10/100	WRN-40-2, WRN-28x10	200	128	0.1	0.0005*
ImageNet-100	ResNet-18	270	256	0.1	0.001
DomainNet	ResNet-18	200	128	0.1	0.001

Table 5. Hyperparameters used for training on CIFAR, ImageNet-100 and DomainNet. *: 0.0002 for CIFAR10 on WRN-40-2

C.1. Uniform distribution: ablations on prior work

In this section, we motivate our choice of a uniform magnitude distribution, showing that it globally outperforms optimized magnitude models in prior work. To this end, we directly evaluate the policies provided by the authors without re-running their search procedure. We compare their learned magnitude model with a simpler one that consists in sampling the magnitudes uniformly on their $[0, 1]$-mapped ranges, We study three baselines: DADA [6], FastAA [7] and DeepAA [9]. Note that their policy results from a search on CIFAR10, that they also use when evaluating on CIFAR100. Results are reported in Table 6.

Parametrization. DADA and FastAA directly optimize a probability distribution over the set of all possible composite transformations (*sub-policies*) and lean a single magnitude value for each transformation in a sub-policy. They keep the top-k sub-policies for evaluation. DeepAA learns to compose transformations in a greedy manner and discretizes the magnitude ranges, learning a probability for

Model	Magnitude model	CIFAR10		CIFAR100	
		WRN-40-2	WRN-28-10	WRN-40-2	WRN-28-10
FastAA/DADA initialization	Theirs	96.22	97.08	78.26	82.17
	Uniform	96.37	97.25	79.10	82.80
FastAA, reported	Original	96.4	97.3	79.3	82.7
FastAA, reproduced (evaluation only)	Original	96.4	97.22	79.11	82.82
	Uniform	96.37	97.30	79.15	82.84
DADA, reported	Original	96.4	97.3	79.1	82.5
DADA, reproduced (evaluation only)	Original	96.33	97.19	79.07	82.05
	Uniform	96.37	97.35	78.97	82.57
DeepAA, reported	Original	-	97.56	-	84.02
DeepAA, reproduced (evaluation only)	Original	96.46	97.48	79.62	83.85
	Uniform	96.55	97.47	78.89	83.62

Table 6. **Why learning the magnitude range?** CIFAR10/100 accuracies for DADA, FastAA and DeepAA, when using their original magnitude model, or a simpler one which samples in a uniform manner in their ranges.

each magnitude. We compare these learned magnitude values (FastAA, DADA) or learned probabilities (DeepAA) to our approach based on a uniform sampling.

DADA/FastAA. Our approach compares favorably to DADA's and FastAA's optimized models. We also compare both approaches on their initial policy (equal probabilities for all sub-policies, magnitudes set at mid-range). With uniform magnitude sampling, their initial policy (sampling among all possible sub-policies) performs similarly if not better than their optimized one (sampling among their top-k sub-policies).

DeepAA. Results on the policy provided by DeepAA are more nuanced: using uniform sampling improves results on CIFAR10 (on which their search was conduced) and degrades them on CIFAR100.

C.2. Visualization of the learned policies

CIFAR. The evolution of probability distributions for CIFAR10 and CIFAR100 and pie charts of the final policies are illustrated in Fig. 1. It can be noted that Invert and Solarize, known to be detrimental, are systematically discarded. The policies learned are quite diverse, with different leading transformations for each distribution but global predominance of some transformations such as Cutout or Rotate. It can also be noted that magnitudes upper-bounds are in average higher for the larger WideResNet-28x10 networks: a larger learning capacity benefits more from harder transformations.

ImageNet-100. The best policy found for ImageNet-100 is illustrated in Fig. 2. Interestingly, RandomResizeCrop is ranked quite low, yet our policy yields results comparable to TrivialAugment's (with a 86.18 average accuracy on this split), suggesting that other geometrical transformations such as Cutout, Rotate and ShearY are equivalently beneficial for training on ImageNet-100. We can note rather high magnitudes upper-bounds for the color jittering transformation TrivialAugment also applies by default (Color, Contrast, Brightness), which is consistent with the higher performance of TrivialAugment's (Wide) version compared to (RA).

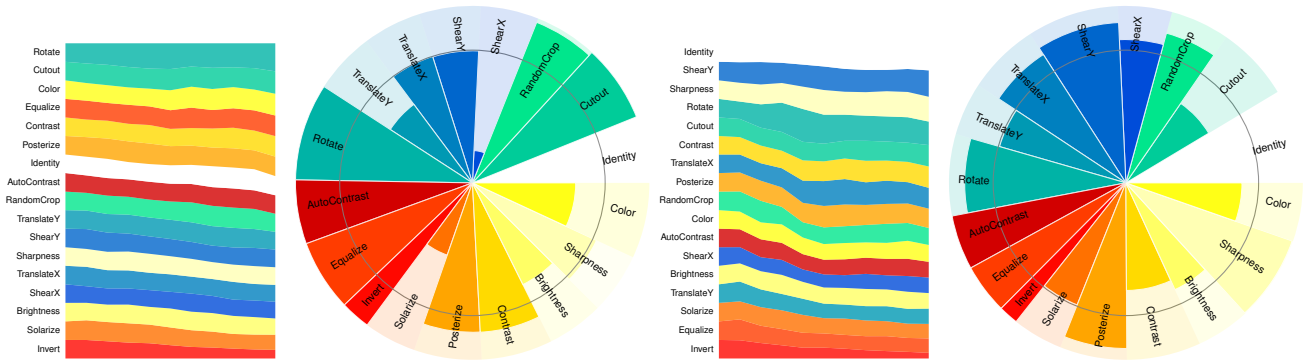
DomainNet. The policies found by SLACK on DomainNet are illustrated in the pie charts Fig. 3 for all domains. Some similarities with policies found on CIFAR and ImageNet can be noted. In particular, Invert and Solarize (that only inverts part of the pixels) are systematically discarded for all domains except Quickdraw. Invert is manually removed from TrivialAugment's baseline as it is known to be detrimental, and this seems to generalize to other domains. Also, Rotate and Cutout are globally favoured, similarly to the policies found on CIFAR and ImageNet-100.

However some differences mark specificities to each domain: i) on the strength of the transformations: for example, geometrical transformations are given high magnitudes on Clipart and lower ones on Real, ii) on their probabilities: color jittering transformations used for real images are globally assigned a high probability for Real, Painting and Infograph domains, and a much lower one for Clipart, which suggests that changes in color, contrast or brightness are less meaningful for this domain.

C.3. Avoiding instabilities with SLACK

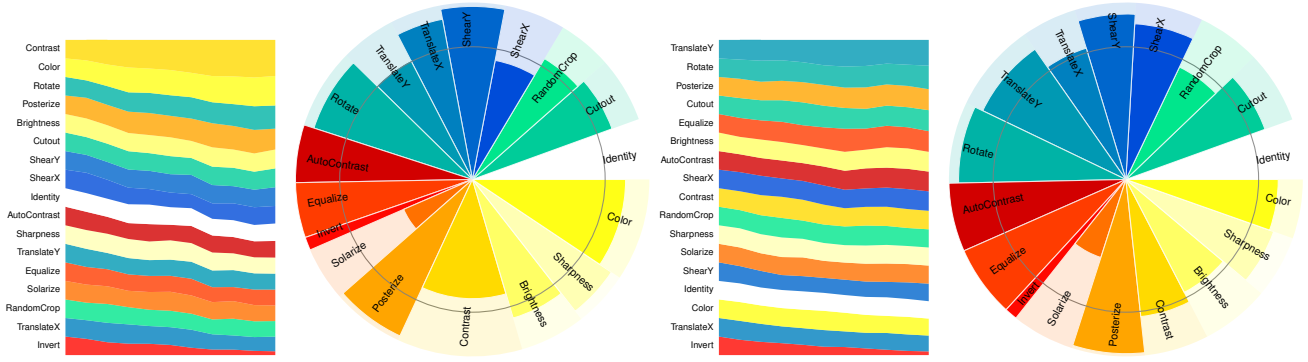
In this section, we study how our augmentation policies evolve when removing the KL regularization or when using a single optimization stage instead of multiple ones, which corresponds to standard unrolled optimization. Evaluations in both settings are reported in Table 7.

We first show that unrolled optimization is globally unstable and easily collapses, justifying the need for a regular-



(a) CIFAR10, WRN-40x2

(b) CIFAR10, WRN-28x19



(c) CIFAR100, WRN-40x2

(d) CIFAR100, WRN-28x10

Figure 1. Illustrations of best policies found for CIFAR10 and CIFAR100 on WideResNet-40x2 and WideResNet-28x10 architectures. For each dataset and architecture, we show the evolution of the probability distribution π as training progresses (left) and the final learned policy as a pie chart (right), where slice widths represent π and and slice radii represent μ.

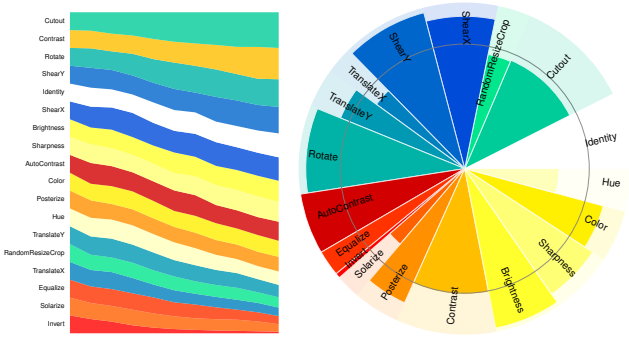
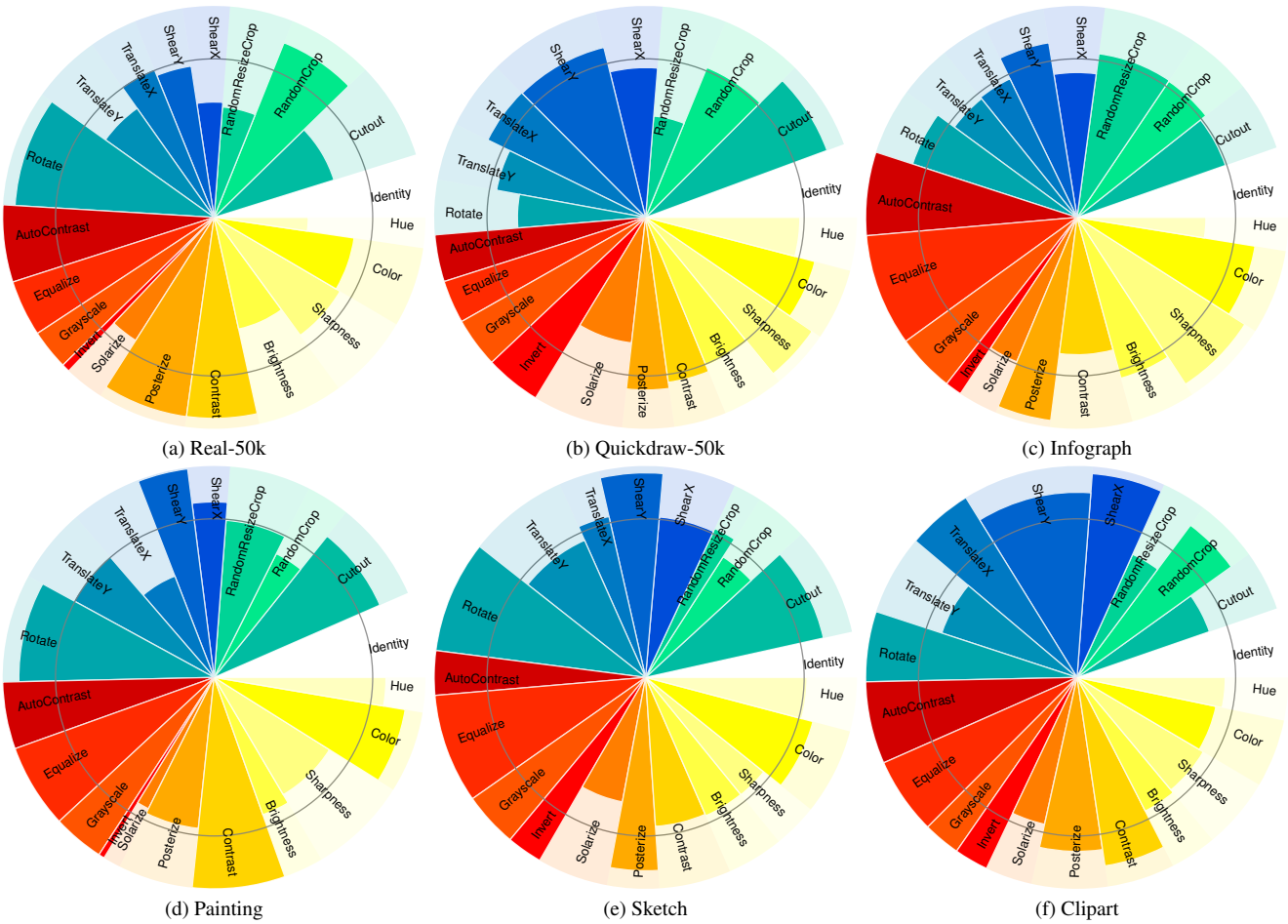


Figure 2. ImageNet-100 policy on the best search split

ization. We illustrate how entropy regularization prevents collapse and yields competitive results, but at the cost of high 'local' instability. These instabilities make the final performance highly dependent on the choice of some hyperparameters, such as the learning rate. The necessity to overcome these instabilities motivates our multi-stage procedure with an adaptive anchoring for the regularization. Lastly, we show that unregularized multi-stage optimization, while more stable than unregularized unrolled optimization, does not yield competitive results, confirming again the benefits of our KL regularization.

Unregularized unrolled optimization. Unrolled optimization is subject to two sources of instability: first, the approximation $\theta^\star(\phi) = \hat{\theta}(\phi)$ with a single gradient step inherently leads to wrong gradient updates; second, the REINFORCE gradient estimation is theoretically exact but has a high variance in practice when approximated in the context of stochastic optimization. Fig. 4 illustrates these instabilities: blindly following wrong gradient directions exacerbated by an oversampling of the dominant transformation leads to a progressive collapse of the policy.

Unrolled optimization with entropy regularization. In the case of a single-stage unrolled optimization, the KL regularization uses a uniform distribution as an anchor, which corresponds to an entropy regularization. By maximizing the entropy, the algorithm encourages exploration of the augmentation policies and prevents the divergence phenomenon observed above. While this regularization leads to competitive results as reported in Table 7, it does not mitigate the inherent instability of the gradient updates. On the other hand, the multi-stage algorithm we proposed in SLACK yields more stable gradient updates.



(a) Real-50k (b) Quickdraw-50k (c) Infograph

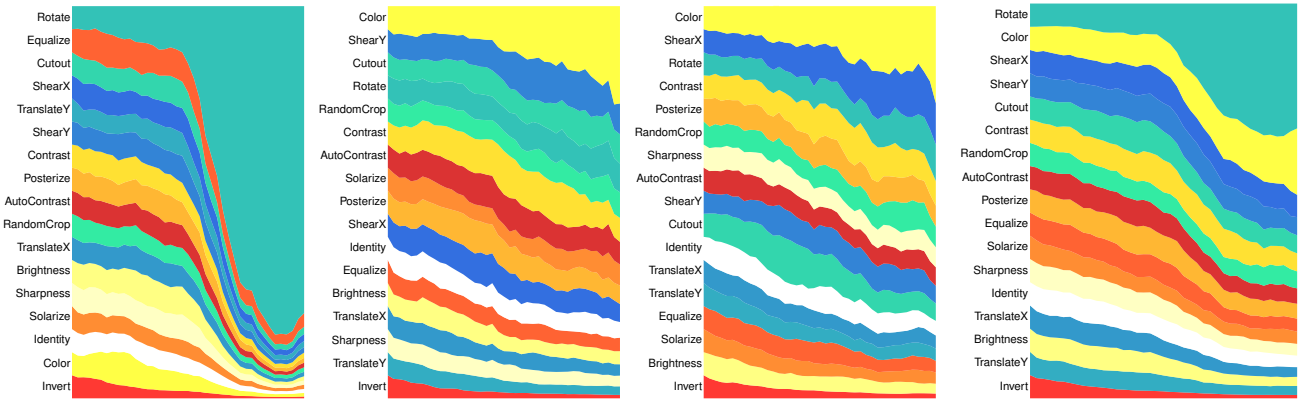
(d) Painting (e) Sketch (f) Clipart

Figure 3. Policies found on DomainNet. The three distributions from π forming the composite transformation are averaged.

				CIFAR10		CIFAR100	
SLACK variant	Upper-level iterations	Upper-level lr	KL weight	WRN-40-2	WRN-28-10	WRN-40-2	WRN-28-10
Unrolled w/ KL (Fig. 5)	10000	0.25	0.005	96.30 ± .08	97.43 ± .04	79.54 ± .20	84.11 ± .13
SLACK w/o KL (Fig. 6)	10 × 400	0.25	0	96.27 ± .05	97.06 ± .11	79.61 ± .13	83.79 ± .19
SLACK (Fig. 1)	10 × 400	1	0.02	96.29 ± .08	97.46 ± .06	79.87 ± .11	84.08 ± .16

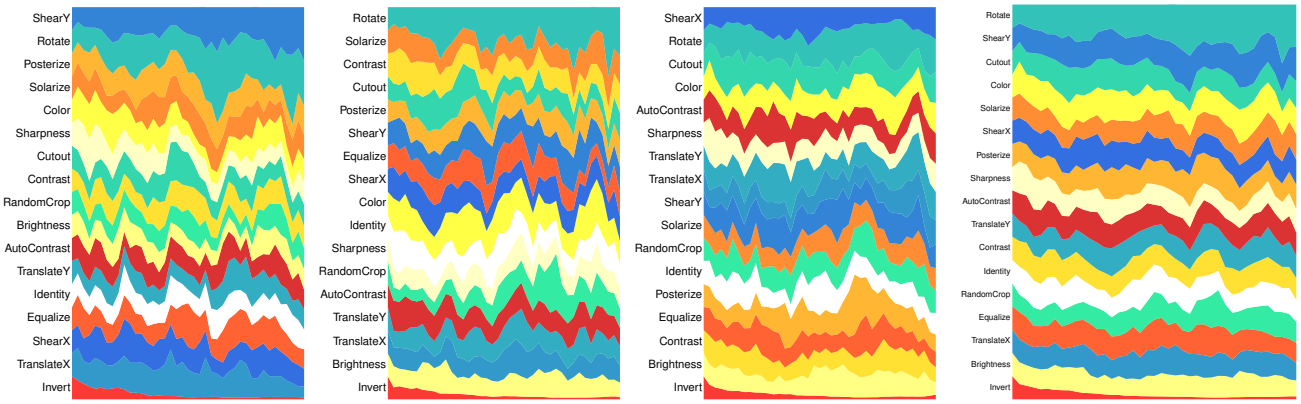
Table 7. CIFAR10/100 accuracy with unregularized and single-stage approaches.

Multi-stage optimization without KL regularization. In our multi-stage approach, $\theta^\star(\tilde{\phi})$ is well-approximated at the beginning of each stage, as the model is re-trained with the current policy $\tilde{\phi}$. Gradient updates close to this policy are 'trusted' since our current θ after re-training stays close to $\theta^\star(\tilde{\phi})$, meaning that we strongly mitigate the approximation inherent to unrolled optimization. Our KL regularization encourages the policy to stay in this *trust region* and without it, the stochasticity of the optimization combined with the high variance from REINFORCE may drive the policy away. Fig. 6 shows the evolution of our probability distributions under an unregularized multi-stage search with two different learning rates, one twice learger than the other. This evolution is smoother than with single-stage unrolled optimization and also quite stable when using a small learning rate, but this slows down convergence, yielding a sub-optimal policy (see Table 7). The larger one leads again to a progressive divergence: the policy is driven too far and the $\theta^\star(\tilde{\phi})$ obtained after re-training becomes sub-optimal for the current ϕ. In other words, the KL regularization allows making large updates in the parameter space, while remaining close to a reference/anchor policy.



(a) The three distributions over transformations forming the composition (b) Average of the 3 distributions

Figure 4. Evolution of the probability distributions π for CIFAR100 with unregularized unrolled optimization in a case of collapse.



(a) The three distributions over transformations forming the composition (b) Average of the 3 distributions

Figure 5. Evolution of the distributions π for CIFAR100 with entropy-regularized unrolled optimization, on one of the search splits.

C.4. An ensembling approach

In this section, we investigate the effect of an ensembling strategy for SLACK to reduce the variance of gradient updates in the search phase. More precisely, the strategy consists in independently training multiple models on the lower-level loss while averaging their contributions to the upper-level gradient. Each model is initialized (and subsequently re-initialized at each stage) based on a pre-training with a different seed. This ensembling strategy was implemented using multiple GPUs, where each GPU trains one copy of the model and only the upper-levels gradients are communicated and averaged across GPUs.

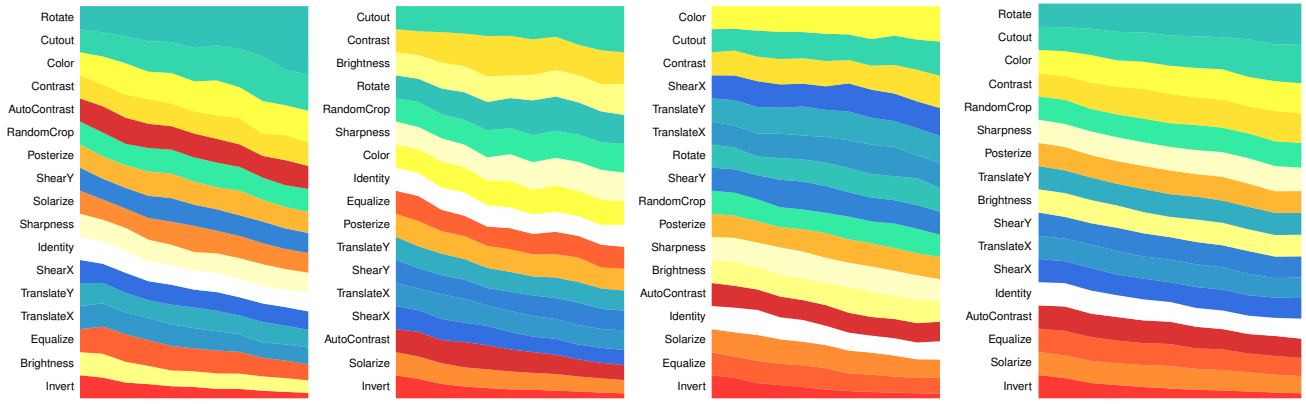
Results on CIFAR10/100 are reported in Table 8. While there is a small improvement in most cases, the method still has a strong computational overhead. Yet it might be a relevant line of research for datasets for which the training procedure has a higher variance, *e.g.* smaller datasets where the additional cost of ensembling is not a significant overhead.

C.5. Warm-start vs cold-start

In this section, we study the model behaviour when searching with warm-start instead of cold-start. By warm-start, we mean that re-training is performed starting from the current network's weights at the beginning of each stage instead of re-initializing it to its pre-trained weights. We experimentally observe that warm-start with the same hyperparameters as for cold-start leads to a progressive overfitting of the network. Increasing the lower-level learning rate mitigates this phenomenon, but still yields sub-optimal results as reported in Table 9. This suggests that re-training from $\theta_0^\star(\phi_0)$ gives a better estimate of $\theta^\star(\phi_i)$ at stage i than re-training from the biased state close to $\theta^\star(\phi_{i-1})$.

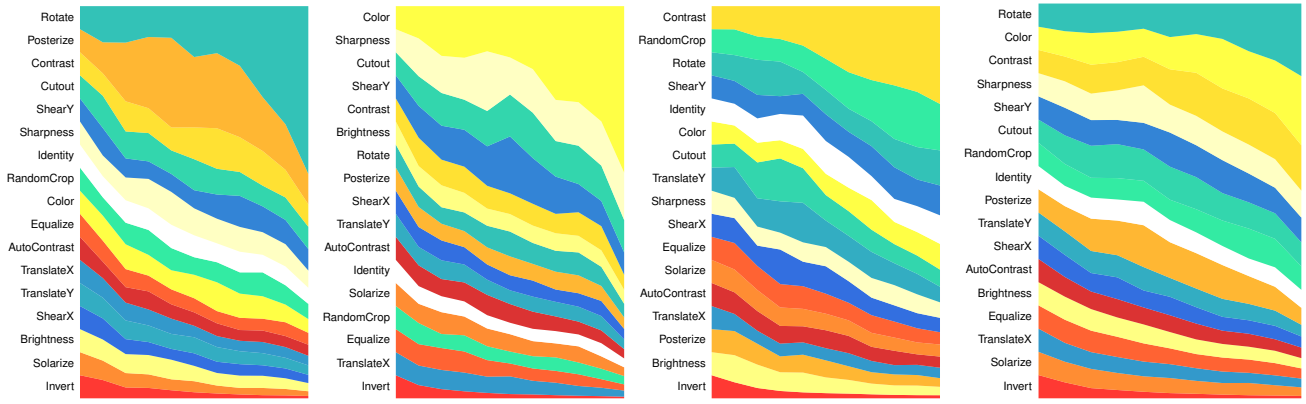
References

[1] 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. 1



(a) 0.25: The three distributions over transformations forming the composition

(b) 0.25: Average of the 3 distributions



(c) 0.5: The three distributions over transformations forming the composition

(d) 0.5: Average of the 3 distributions

Figure 6. Evolution of the probability distributions π for CIFAR100 with unregularized multi-stage optimization using upper-level learning rates of 0.25 and 0.5.

	CIFAR10		CIFAR100	
	WRN-40-2	WRN-28-10	WRN-40-2	WRN-28-10
SLACK	96.29 ± .08	97.46 ± .06	79.87 ± .11	84.08 ± .16
Ensembling of SLACK (4 GPUs)	96.33 ± .08	97.48 ± .06	79.94 ± .13	84.01 ± .14

Table 8. CIFAR10/100 accuracy with ensembling strategy.

	CIFAR10		CIFAR100	
SLACK variant	WRN-40-2	WRN-28-10	WRN-40-2	WRN-28-10
Warm-start	96.27 ± .09	97.05 ± .15	79.70 ± .11	83.90 ± .10
Cold-start (ours)	96.29 ± .08	97.46 ± .06	79.87 ± .11	84.08 ± .16

Table 9. CIFAR10/100 accuracy with cold start and warm start.

[2] Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. RandAugment: Practical automated data augmentation with a reduced search space. In *Proc. NeurIPS*, 2020. 1

[3] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *Proc. ICLR*, 2021. 2

[4] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Faster AutoAugment: Learning augmentation strategies using backpropagation. In *Proc. ECCV*, 2020. 1

[5] 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

[6] 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

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

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

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