Challenges of Adversarial Image Augmentations

Anonymous Author(s)
Affiliation
Address
email

Abstract

Image augmentations applied during training are crucial for the generalization performance of image classifiers. Therefore, a large body of research has focused on finding the optimal augmentation policy for a given task. Yet, RandAugment [2], a simple random augmentation policy, has recently been shown to outperform existing sophisticated policies. Only Adversarial AutoAugment (AdvAA) [11], an approach based on the idea of adversarial training, has shown to be better than RandAugment. In this paper, we show that random augmentations are still competitive compared to an optimal adversarial approach, as well as to simple curricula, and conjecture that the success of AdvAA is due to the stochasticity of the policy controller network, which introduces a mild form of curriculum.

1 Introduction

The success of a deep vision model can greatly depend on the data augmentations that were used during training [1]. For natural images, these augmentations can be categorized as geometric (e.g, horizontal flips, translations), color (e.g., histogram equalization), and distortion (e.g., random cropping, or random greypainting of parts of the images, commonly referred to as Cutout [3]).

Finding the right augmentation policy, which determines the type and magnitude of the augmentation to use at a given training point, has been the subject of a large amount of research. Novel approaches range from using reinforcement learning [1], Bayesian optimization [9], or evolutionary algorithms [5] to only name a few. While these methods managed to surpass the performance of existing hand-crafted augmentation baselines, they were subsequently outperformed by RandAugment [2], a simple random augmentation policy. The only exception is Adversarial AutoAugment [11] (AdvAA), which uses an adversarial approach to guide the search for the right augmentation policy and significantly outperforms RandAugment. Yet, it is unclear to which extent the polices selected by AdvAA are adversarial; hence, the question remains: do adversarial augmentations truly help?

In this work, we shed more light on this open question and show that exactly solving the adversarial objective in AdvAA is detrimental to generalization. We also show that curricula that increase the adversarial degree of the augmentation is only as effective, if not less, than a random policy. The comparison between the policies selected by AdvAA and those selected by an exhaustive search suggests that the success of [11] might lie in the implicit randomness of the solution to the optimization of the adversarial objective, as we further explain in the Section 5. We conclude our work with a glimpse of hope suggesting a new direction that has interesting preliminary results and could be a founding principle for new policies.

[1] For example, when training a WideResNet-28-10 on CIFAR10, we observed a test set accuracy difference of 6.8% between not using and using augmentations.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

2 Background

AutoAugment [1] is a natural starting point for an overview of recent works on automating the search for the ideal augmentation policy, as it spans the policy search space that many subsequent works adopt for their own policy search. The basis of AutoAugment's search space consists of 16 operations (ShearX/Y, TranslateX/Y, Rotate, AutoContrast, Invert, Equalize, Solarize, Posterize, Contrast, Color, Brightness, Sharpness, Cutout [3], Sample Pairing [6]). The search space for augmentation policies is then spanned by pairs of these operations, where each operation has a probability and magnitude along a uniform spacing of 11 probabilities, within the range $[0, 1]$, and 10 magnitudes, within the range $[0, \mathrm{maxmagnitude}]$. At each training step, 5 of such policies are sampled with equal probability leading to a search space of $(16 * 11 * 10)^{2\times 5} \approx 10^{32}$ elements. In AutoAugment, the authors aim to find the optimal set of augmentations by using a reinforcement learning approach, where the validation set accuracy is used as reward. The augmentation policies they find improve test set accuracy by 0.4% and 1.3% on CIFAR10 and CIFAR100 [8] when using a Wide-ResNet-28-10 (see Tab. 2 in the Appendix). Lim et al. [9] drastically shorten the duration of the policy search by using a Bayesian optimization approach while maintaining a similar performance. Hataya et al. [4] shorten the duration even further by transforming the search into an online search (that is, during training of the final model). The authors propose to minimize the Wasserstein distance between the augmented and the original training data empirical distribution, again while maintaining a comparable performance in terms of test set accuracy. Ho et al. [5] also search for the optimal augmentation policy in an online fashion using an evolutionary algorithm (population-based training [7]), reducing the search time and even outperforming the original AutoAugment on CIFAR100 when using a Wide-ResNet-28-10.

However, all of these sophisticated approaches were outperformed by RandAugment, a simple random augmentation policy introduced in [2] which uniformly samples the components of operation pairs to be applied and uses either a constant or a random magnitude (to the same effect, as shown in Appendix A.1.1 of [2]). This simple RandAugment policy requires no search and yet achieves test set accuracies, on the aforementioned data sets and model, that are on-par or better than the more complex techniques previously discussed. To the best of our knowledge, only AdvAA [11] has significantly outperformed RandAugment.

3 The Adversarial Approach to Image Augmentation

The core idea of AdvAA consists of finding those augmentations that lead to "hard" samples, as measured by their training loss. The hypothesis is that learning from hard examples should force the model to learn more robust features [11]. The optimization objective becomes a min-max problem:

$$\min_{w} \max_{\tau \in \mathcal{S}} \mathbb{E}_{x,y\sim P(\mathcal{D})}[\mathcal{L}(f_w(\tau(x), y))], \tag{1}$$

where, $\mathcal{D}$ is the training set, x is a sample with label y, w are the model parameters, $f_w(\cdot)$ is the model output, $\mathcal{L}(f_w(\cdot))$ is the loss function, and $\mathcal{S}$ is the set of all available augmentations (as in AutoAugment). In [11] the inner maximization (i.e., find the most adversarial policies) is solved by introducing a controller augmentation policy network that, given the losses produced by the main model, is trained to output probabilities $p_1, \ldots, p_{|S|}$ for each $\tau_1, \ldots, \tau_{|S|}$, such that the maximization is solved in expectation (combining Eqns. (3) and (7) in [11]):

$$\min_{w} \max_{\theta} \mathbb{E}_{x,y\sim P(\mathcal{D})} \mathbb{E}_{\tau\sim P(\cdot,\theta)}[\mathcal{L}(f_w(\tau(x), y))], \tag{2}$$

where $P(\cdot, \theta)$ is determined by the output of the controller policy network with parameters θ.

The approach in [11], summarized by Eq. (2), significantly outperforms the other works on automated image augmentation (Tab. 2 in Appendix). Yet, the training of the controller augmentation policy network introduces a large amount of stochasticity due to: (i) the training dynamics inside the min-max problem and (ii) the REINFORCE algorithm [10] necessary to cope with the non-differentiable (discrete) augmentation operations in the objective. Hence, it is logical to assume that the policy network will act as a random sampling algorithm, at least until it has learnt how to correctly predict which policies are the hardest for the given model. To complicate things, the model being trained is constantly evolving making the optimal policy a moving target. Therefore, it is unclear that the policies selected by the controller network are strictly adversarial.

Table 1: Test set accuracy gains over the baseline with M=1 (our implementation of Cutout [3]) with Wide-ResNet-28-10 trained for 200 epochs. Mean of 5 runs, standard deviations in brackets).

Accuracy %		Baseline	Random	TrueAdv	1-Adv-0Ep	1-Adv-100Ep	Smooth	Cyclic
CIFAR10	M=1	96.8 (0.1)	**+0.5 (0.0)**	-1.1 (0.2)	-	-	+0.3 (0.0)	**+0.5 (0.1)**
	M=2	-	+0.8 (0.1)	-	+0.8 (0.1)	+0.7 (0.0)	+0.8 (0.1)	**+0.9 (0.1)**
CIFAR100	M=1	81.1 (0.2)	**+1.3 (0.1)**	-0.8 (0.3)	-	-	+0.7 (0.1)	+1.1 (0.2)
	M=2	-	**+1.8 (0.3)**	-	+0.9 (0.1)	+0.9 (0.2)	+0.6 (0.3)	+1.2 (0.4)

In Section 4, we focus on determining whether being truly adversarial actually helps the model learning more robust features, or if there are more complex dynamics that should be considered.

4 An Adversarial Approach

All our experiments are obtained following exactly the same training protocol, which is described below and very similar to [11]. For computational reasons we discretize the magnitudes to 5 values rather than 10, and reduced the search space of the operations to 15, as in [5]. We use an initial batch size of 128, we adopt a multiplicity of $M = 1$ and $M = 2$, that is augmenting each batch using one or two operations (the latter approach effectively doubling the batch size as in [11]). We train for 200 epochs and report the best test-accuracy over all training epochs, averaged across five runs. The learning rate is set to 0.1 with decay of 0.2 after the 50th, 100th, and 140th epochs, we use a warm-up of 10 epochs in which no augmentation is used, Nesterov optimizer, batch-norm momentum set to 0.1, and weight decay set to $5e$-4. Given the slightly different setup, results in Tab. 1 are not meant for a direct comparison with [11]. They can instead be compared with each other since we have followed the same training protocol.

4.1 Removing the Stochasticity of the Controller Augmentation Policy Network Training

We note that the optimal solution to Eq. (2) is determined by the optimal solution to Eq. (1).

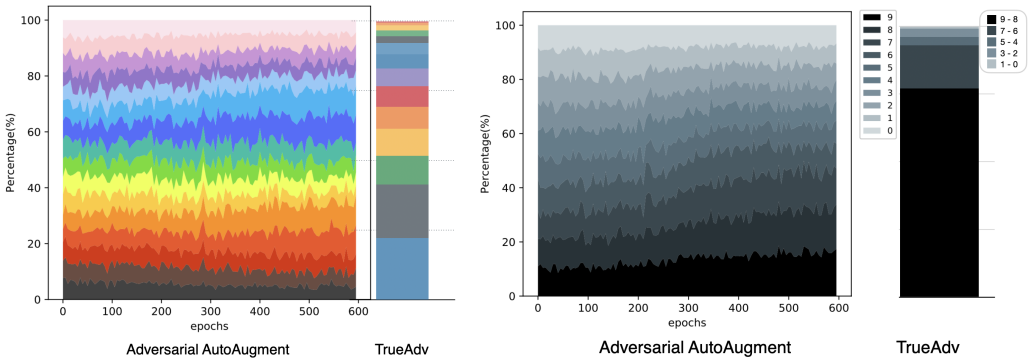
Corollary 1. Let $\tau^* = \tau_j$ be the optimal solution to the inner maximization in Eq. (1) for some $j \leq |\mathcal{S}|$ (for simplicity we assume only one optimum). Then any optimal solution θ^* to the inner maximization in Eq. (2) has to fulfill $p_j(\cdot, \theta^*) = 1$.

Corollary 1 can be proven by using Fubini-Tonelli to interchange the expectations in Eq. (2). The extension to multiple optima would simply share the probability mass among those. We run experiments on CIFAR10 and CIFAR100 with a Wide-ResNet-28-10 in which we directly solve for τ^* in Eq. (1) during training to remove the stochasticity from the learning dynamics of the controller augmentation policy network.

TrueAdv refers to an experiment where in order to find the *true* τ^*, at the end of each epoch the empirical loss is evaluated on the entire training set and all the polices. Hence for all training points N and for each policy $\tau \in \mathcal{S}$ we evaluate: $\frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(f_w(\tau(x_i), y_i) \approx \mathbb{E}_{x,y\sim P(\mathcal{D})}[\mathcal{L}(f_w(\tau(x), y)]$. This algorithm is clearly not scalable but it is used to gain more insights on the matter.

As shown in Tab.1, *TrueAdv* is actually harmful to generalization, as it lowers test set accuracy for all data sets and multiplicities analyzed. Choosing random augmentations is not only computationally more efficient, but also better in terms of generalization. This leaves the question "why do the learned policies in [11] not exhibit the same behavior, despite aiming to solve an objective whose exact solution degrades generalization performance"? We hypothesize that this is due to the stochasticity of the policy network training. Fig. 1 shows the operations and magnitudes used by AdvAA and *TrueAdv* (for details about the *TrueAdv* breakdown, refer to Fig. 2 in the Appendix). The differences in the choice of the policies are evident: (i) *TrueAdv* relies only on 3 operations (Rotation, Inversion, Brightness) for more than 50% of the time, while for AdvAA the probability of sampling a specific operation remains fairly uniform (some milder differences can be appreciated after epoch 300); (ii) *TrueAdv* uses the highest magnitudes (8-9) 77% of the time compared to the 30% of AdvAA.

These observations suggest that AdvAA is not maximizing the inner objective of Eq. 2, rather it samples reasonably uniformly. A form of curriculum could also be playing a role, however, the probabilities of sampling hard policies remain fairly low compared to *TrueAdv*.



(a) Operations (b) Magnitudes

Figure 1: Comparison between AdvAA (as reported in [11]) and *TrueAdv*. We show the percentage of times each operation or magnitude were used during training. For *TrueAdv* we report the average over all epochs, and magnitudes are binned by increasing strength (i.e., bin 8-9 is the highest magnitude). Notice how the operations (a) in AdvAA maintains an approximately uniform sampling across epochs, while *TrueAdv* relies more often on fewer operations. Also, notice in (b) how the top two highest magnitudes (8-9) are selected less than 30% of the time by AdvAA but 77% by *TrueAdv*. If we analyze only the first 200 epochs (the stopping point for *TrueAdv*) this difference would be amplified.

Leveraging this observation, we experiment with 4 different curricula. To make the approaches computationally tractable (unlike *TruAdv*), we evaluate the losses at the end of each epoch on a randomly sampled batch of size 128 and on a random subset of 500 policies. *1-Adv-0Ep* uses multiplicity $M = 2$, one hard augmentation (i.e. τ^*, the augmentation with the highest average loss), and one augmentation that is sampled from the 250 with the lowest average loss (i.e. easiest). *1-Adv-100Ep* begins by using the easiest augmentations until epoch 100, and then switches to *1-Adv-0Ep*. *Smooth* gradually increases the adversarial approach starting with the easiest augmentations, the first hard augmentation (i.e. τ^*) is introduced in epoch 75 (when $M = 2$) or epoch 125 (when $M = 1$). For $M = 2$, the augmentation with the second highest loss is added at epoch 150. As shown in Tab.1, all three policies improve over the baseline, but cannot exceed the simple random policy.

We notice how the *TrueAdv* suffers from over-fitting to the training data, suggesting that truly adversarial augmentations are creating a distribution shift away from *real* data. Therefore, we test a curriculum that reverts back to the easiest augmentations at the end of training, to match the *actual* expected distribution. *Cyclic* starts with the easiest augmentations, for $M = 2$, the first hard augmentation is introduced at epoch 75, the second at epoch 100. At epoch 125 it reverts back to only one hard augmentation, and finally from epoch 150 on-wards it uses only the easiest set. For $M = 1$ it introduces the hard augmentation at epoch 75 before going back to the easy random augmentations at epoch 150. Interestingly, the *Cyclic* schedule achieves strong improvements over the baseline augmentation (Tab.1), even slighlty surpassing random policies on CIFAR10. Random is still better, however, on CIFAR100, especially in the case of $M = 2$.

5 Discussion

We have seen that solving exactly the adversarial objective introduced in [11] goes against its original motivation of finding robust features, and deteriorates the test set accuracy even when compared to the baseline [3]. Thus, we believe that the results achieved in [11] must come from its stochasticity and a mild form of curriculum that emerges from the network policy controller. Our attempts to find an explicit curriculum, however, led to improve the test set accuracy compared to the baseline but not compared to a random augmentation policy. The idea of reverting towards milder augmentations as the model completes the training has proven to be interesting, and in future works we will continue exploring how this strategy can be refined to finally produce results that are better than using random policies, while automatizing as much as possible the hyper parameters of the curriculum.

References

[1] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019.

[2] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

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

[4] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Faster autoaugment: Learning augmentation strategies using backpropagation. In *European Conference on Computer Vision*, pages 1–16. Springer, 2020.

[5] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In *International Conference on Machine Learning*, pages 2731–2741. PMLR, 2019.

[6] Hiroshi Inoue. Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*, 2018.

[7] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

[8] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, 2009.

[9] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. *Advances in Neural Information Processing Systems*, 32:6665–6675, 2019.

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

[11] Xinyu Zhang, Qiang Wang, Jian Zhang, and Zhao Zhong. Adversarial autoaugment. *arXiv preprint arXiv:1912.11188*, 2019.

A Additional Results

Table 2: Test set accuracy gains reported in recent works on automated image augmentation policy search for Wide-ResNet-28-10. Importantly, only AdvAA manages to consistently outperform RandAugment, which ultimately uses random tuples of operations as augmentations during training.

	Data set	
Work	CIFAR10	CIFAR100
Baseline (Cutout [3])	96.9	81.6
AutoAugment [1]	+0.4	+1.3
Fast AutoAugment [9]	+0.4	+1.3
Faster AutoAugment [4]	+0.5	+1.1
PBA [5]	+0.4	+1.7
RandAugment [2]	+0.4	+1.7
AdvAA [11]	+1.2	+2.9

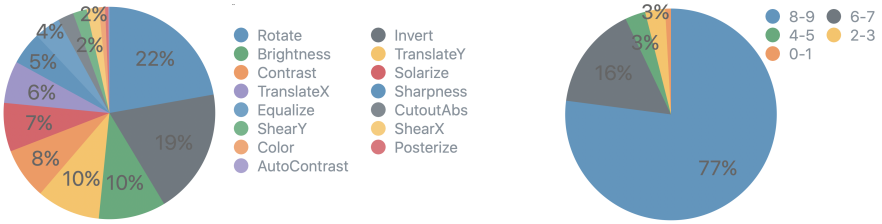


Figure 2: Percentage of times a given operation and magnitude were used during training (*TrueAdv* strategy). Magnitudes have been binned into increasing strength (i.e., bin 8-9 is the highest magnitude, while 0-1 is the lowest magnitude)

.

B More details on training

All our experiments were obtained using an internal GPU cluster equipped with Nvidia GPUs model Tesla V100-SXM2-32GB with Cuda10.1. The code was developed using pytorch1.8.1.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work investigate properties of existing techniques, hence, it is unlikely to have any potential societal negative impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] Our work is empirical.
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We have included all the information needed to reproduce the results but we did not publish the code yet.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] We did not mention the license for CIFAR10 and CIFAR100 but added the citation with all the information.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]