TrivialAugment: Tuning-free Yet State-of-the-Art Data Augmentation

Samuel G. Müller
University of Freiburg
muellesa@cs.uni-freiburg.de

Frank Hutter
University of Freiburg &
Bosch Center for Artificial Intelligence, Germany
fh@cs.uni-freiburg.de

Abstract

Automatic augmentation methods have recently become a crucial pillar for strong model performance in vision tasks. While existing automatic augmentation methods need to trade off simplicity, cost and performance, we present a most simple baseline, TrivialAugment, that outperforms previous methods for almost free. TrivialAugment is parameter-free and only applies a single augmentation to each image. Thus, TrivialAugment's effectiveness is very unexpected to us and we performed very thorough experiments to study its performance. First, we compare TrivialAugment to previous state-of-the-art methods in a variety of image classification scenarios. Then, we perform multiple ablation studies with different augmentation spaces, augmentation methods and setups to understand the crucial requirements for its performance. Additionally, we provide a simple interface to facilitate the widespread adoption of automatic augmentation methods, as well as our full code base for reproducibility[1]. Since our work reveals a stagnation in many parts of automatic augmentation research, we end with a short proposal of best practices for sustained future progress in automatic augmentation methods.
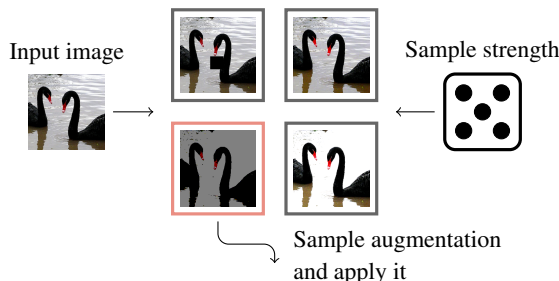
Figure 1: A visualization of TA. For each image, TA (uniformly) samples an augmentation strength and an augmentation. This augmentation is then applied to the image with the sampled strength.

Method	Search Overhead	CIFAR-10 ShakeShake	CIFAR-100 WRN	SVHN WRN	ImageNet ResNet
AA	40 - 800×	98.0	82.9	98.9	77.6
RA	4 - 80×	98.0	83.3	**99.0**	77.6
Fast AA	1×	98.0	82.7	98.8	77.6
TA (ours)	**0×**	**98.2**	**84.3**	98.9	**78.1**

Table 1: **TrivialAugment compares very favourably to previous augmentation methods.** In this table we summarize some results from Table 2 and present augmentation search overhead estimates.

1. Introduction

Data Augmentation is a very popular approach to increase generalization of machine learning models by generating additional data. It is applied in many areas, such as machine translation [4], object detection [6] or semi-supervised learning [20]. In this work, we focus on the application of data augmentation to image classification [3, 12].

Image augmentations for image classification generate novel images based on images in a dataset, which are likely to still belong to the same classification category. This way the dataset can grow based on the biases that come with the augmentations. While data augmentations can yield considerable performance improvements, they do require domain knowledge. An example of an augmentation, with a likely class-preserving behaviour, is the rotation of an image by some small number of degrees. The image's class is still recognized by humans and so this allows the model to generalize in a way humans expect it to generalize.

Automatic augmentation methods are a set of methods that design augmentation policies automatically. They have been shown to improve model performance significantly across tasks [2, 23, 20].

Automatic augmentation methods have flourished especially for image classification in recent years [2, 13, 14, 9] with many different approaches that learn policies over aug-

---

1. https://github.com/automl/trivialaugment

mentation combinations. The promise of this field is to learn custom augmentation policies that are strong for a particular model and dataset. While the application of an augmentation policy found automatically is cheap, the search for it can be much more expensive than the training itself.

In this work, we challenge the belief that the resulting augmentation policies of current automatic augmentation methods are actually particularly well fit to the model and dataset. We do this by introducing a trivial baseline method that performs comparably to more expensive augmentation methods without learning a specific augmentation policy per task. Our method does not even combine augmentations in any way. We fittingly call it TrivialAugment (TA).

The contributions of this paper are threefold:

- We analyze the minimal requirements for well-performing automatic augmentation methods and propose TrivialAugment (TA), a trivial augmentation baseline that poses state-of-the-art performance in most setups. At the same time, TA is the most practical automatic augmentation method to date.

- We comprehensively analyze the performance of TA and multiple other automatic augmentation methods in many setups, using a unified open-source codebase to compare apples to apples.

- We make recommendations on the practical usage of automatic augmentation methods and collect best practices for automatic augmentation research. Additionally, we provide our code for easy application and future research.

2. Related Work

Many automatic augmentation methods have been proposed in recent years with multiple different setups. Still, all automatic augmentation methods we consider share one property: They work on augmentation spaces that consist of i) a set of prespecified augmentations $\mathcal{A}$ and ii) a set of possible strength settings with which augmentations in $\mathcal{A}$ can be called (in this work $\{0, \ldots, 30\}$). One member of $\mathcal{A}$ might, for example, be the aforementioned rotation operation, where the strength would correspond to the number of degrees. Automatic augmentation methods now learn how to use these augmentations together on training data to yield a well-performing final classifier.

In this section, we provide a thorough overview of relevant previous methods. As the compute requirements for automatic augmentation methods can dominate the training costs, we order this recount by the total cost of each method.

We begin with the first automatic augmentation method, AutoAugment (AA) [1], which also happens to be the most expensive, spending over half a GPU-year of compute to yield a classifier on CIFAR-10. AA uses a recurrent neural network (RNN), which is trained with reinforcement learning methods, to predict a parameterization of augmentation policies. Reward is given for the validation accuracy of a particular model trained on a particular dataset with the predicted policy. AA makes use of multiple sub-policies each consisting of multiple augmentations, which in turn are applied sequentially to an input image. Additionally, augmentations are left out with a specified probability. This allows one sub-policy to represent multiple combinations of augmentations. Since AA is costly it uses not the task at hand for augmentation search, but a reduced dataset and a smaller model variant.

The second most expensive method is Augmentation-wise Sharing for AutoAugment (AWS) [19]. It builds on the same optimization procedure as AA, but uses a simpler search space. The search space consists of a distribution over pairs of augmentations that are applied together. Different from AA, AWS learns the augmentation policy for the last few epochs of training only. It does this on the full dataset with a small model.

A very different approach, called Population-based Augmentation (PBA) [9], is to learn the augmentation policy online as the training goes. PBA does so by using multiple workers that each use a different policy and are updated in an evolutionary fashion. It uses yet another policy parameterization: a vector of augmentations where each augmentation has an attached strength and leave-out probability. From this vector augmentations are sampled uniformly at random and applied with the given strength or left out, depending on the leave-out probability.

Another method based on multiple parallel workers is Online Hyper-Parameter Learning for Auto-Augmentation (OHL) [14]. Here, the policy is defined like for AWS and its parameters are trained using reinforcement learning. The major difference with AWS is that its reward is the accuracy on held-out data after a part of training like for PBA, rather than final accuracy. As an additional way of tuning the neural network weights in the parallel run, the weights of the worker with maximal accuracy are used to initialize all workers in the next part of training.

Adversarial AutoAugment (Adv. AA) [22] is another slightly cheaper method that uses multiple workers and learns the augmentation policy online. It trains only a single model, though. Here, a single batch is copied to eight different workers and each worker applies its own policies to it, similar to the work by Hoffer *et al.* [10]. The worker policies are sampled at the beginning of each epoch from a policy distribution. The policy distribution has a similar form to that of AA. After each epoch, Adv. AA makes a reinforcement-learning based update and rewards the policy yielding the *lowest* accuracy training accuracy, causing the policy distribution to shift towards progressively stronger augmentations over the course of training.

Recently, Cubuk *et al.* proposed RandAugment (RA) [2]. It is much simpler, but only slightly cheaper, compared to the previous methods. RA only tunes two scalar parameters for each task: (i) a single augmentation strength $m \in \{0, \dots, 30\}$ which is applied to all augmentations and (ii) the number of augmentations to combine for each image $n \in \{1, 2, 3\}$. RA therefore reduces the number of hyperparameters from all the weights of an RNN (for AA) or a distribution over more than a thousand augmentation combinations (for AWS and OHL) to just two. This radical simplification, contrary to expectations, does not hurt accuracy scores compared to many other methods. The authors give indication that the strong performance might be due to the fact that n and m are tuned for the exact task at hand and not for a pruned dataset, as is done, for example, in AA. The big downside of RA is that it ends up performing an exhaustive search over a set of options for n and m incurring up to $80\times$ overhead over a single training[2].

Fast AutoAugment (Fast AA) [13] is the cheapest of the learned methods. It is based on AA, but does not directly search for policies with strong validation performance. Rather, it searches for augmentation policies by finding well-performing inference augmentation policies for networks trained on a split of raw, non-augmented, images. All inference augmentations found on different splits are then joined to build a training time augmentation policy. The intuition behind this can be summarized as follows: If a neural network trained on real data generalizes to examples augmented with some policy then this policy produces images that lie in the domain of the class, as approximated by the neural network. The augmentations therefore are class-preserving and useful. This objective stands in contrasts to the approach followed by Adv. AA. Fast AA tries to find augmentations that yield high accuracy when applied to validation data, while Adv. AA tries to find augmentations that yield low accuracy when applied to training data.

Finally, in an unpublished arXiv paper, Lingchen *et al.* [15] very recently suggested UniformAugment (UA), which works almost like RA. Unlike RA, it fixes the number of augmentations to $N = 2$ and drops each augmentation with a fixed probability of 0.5. Furthermore, the strength m is sampled uniformly at random for each applied operation.

In contrast to all above methods methods, we propose TrivialAugment (TA), an augmentation algorithm that is parameter-free like UA, but even simpler. At the same time, TA performs better than any of the comparatively cheap augmentation strategies, making it the most practical automatic augmentation method to date.

Different from all of the work discussed above, which improves final in-distribution test performance with augmentation strategies, AugMix [8] aims to improve model robustness by combining multiple augmentations in application chains, mixing their outputs, and applying a consis-
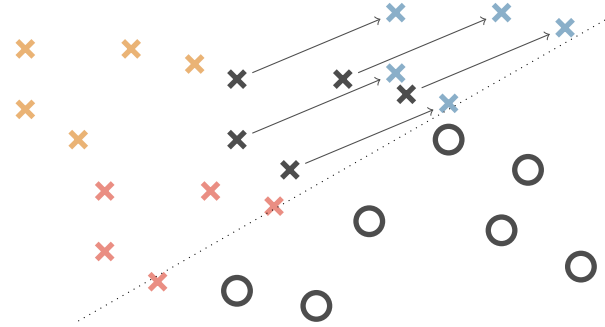

Figure 2: An exemplary visualization of a 2-D dataset with two classes, crosses and circles, separated by a decision boundary, the dotted line. The colored crosses represent deterministic augmentations of the cross class. TA now uniformly samples from all crosses.

tency loss to several augmented images. The only metric we evaluated for which AugMix was evaluated, too, is ResNet-50 performance on the ImageNet test set. Here, TA outperforms AugMix.

3. TrivialAugment

In this section, we present the simplest augmentation algorithm we could come up with that still performs well: *TrivialAugment (TA)*. TA employs the same augmentation style that previous work [2, 15] used: An augmentation is defined as a function a mapping an image x and a discrete strength parameter m to an augmented image. The strength parameter is not used by all augmentations, but most use it to define how strongly to distort the image.

TA works as follows. It takes an image x and a set of augmentations $\mathcal{A}$ as input. It then simply samples an augmentation from $\mathcal{A}$ uniformly at random and applies this augmentation to the given image x with a strength m, sampled uniformly at random from the set of possible strengths $\{0, \dots, 30\}$, and returns the augmented image. We outline this very simple and parameter-free procedure as pseudocode in Algorithm 1 and visualize it in Figure 1. We emphasize that TA is *not* a special case of RandAugment (RA), since RA uses a *fixed* optimized strength for all images while TA samples this strength anew for each image.

While previous methods used multiple subsequent augmentations, TA only applies a single augmentation to each image. This allows viewing the distribution of the TA-augmented dataset as an average of the $|\mathcal{A}|$ data distributions generated by each of the augmentations applied to the full dataset. In Figure 2 we visualize this notion for deterministic augmentations without a strength parameter. Un-

---

2. In the original setups, the authors also used a different choice of n and m for the search on each task. This can be hard to do for new tasks or with less intuition for a task.

Algorithm 1 TrivialAugment Procedure

1: **procedure** TA(x: image)
2:     Sample an augmentation a from $\mathcal{A}$
3:     Sample a strength m from $\{0, \ldots, 30\}$
4:     Return $a(x, m)$
5: **end procedure**
---

like previous work, we do not generate complex distributions out of stochastic combinations of augmentation methods, but simply mean the data distributions of the augmentations applied to the given dataset.

4. Experiments

In this section, we empirically demonstrate TA's surprisingly strong performance, as well as its behaviour across many ablation settings. In all non-ablation experiments we use either the RA augmentation space (RA), i.e., the set of augmentations and their strength parameterization from the RA paper [2], or the wide augmentation space (Wide) for TA. We list the augmentations and their arguments for all augmentation spaces in the appendix in Table 8. We run each experiment ten times, if not stated otherwise. In addition to the average over runs we report a confidence interval, which will contain the true mean with probability $p = 95\%$, under the assumption of normal distributed accuracies. In our code we provide a function to compute this interval. Results that lie within the confidence interval of the best performer for each task are typeset in bold font.

We evaluate our method on five different datasets. i) CIFAR-10 and CIFAR-100 [11] are standard datasets for image classification and each contain 50K training images. We trained Wide-ResNets [21] as well as a ShakeShake model [5]. We follow previous work [1, 2] with our setup. ii) SVHN [17] consists of images of house numbers. It comes with a core set of 73K training images, but offers an additional 531K simpler images as extension of the dataset. We perform experiments with and without the additional images on a Wide-ResNet-28-10. iii) Finally, we perform experiments on ImageNet, a very large image classification corpus with 1000 classes and over 1.2 million images. This experiment is particularly interesting, since it was shown previously that there are augmentations, such as cutout, that do not generalize well to ImageNet. We train a ResNet-50 [7] following the setup of [1]. We use warmup and 32 workers due to cluster limitations, which is less than [1]. We scale the learning rate appropriately. See Appendix A for more details.

4.1. Comparison to State-of-the-Art

It is non-trivial to compare automatic augmentation methods fairly. We therefore compare our method with the previous state-of-the-art in three different setups.

In Section 4.1.1, we follow the majority of previous work [1, 2, 9, 13, 15] and perform a comparison with other methods that use the same model and training pipeline. This setup allows for different search costs of different methods and compares methods with the same inference and training costs.

In Section 4.1.2, we compare in a similar way as above, but against reproductions of other methods in our codebase. This avoids confounding factors, making sure that the methods, and not setup details, explain the differences between results. We reproduced a total of four other methods in our codebase, including the cheapest three previous methods.

In Section 4.1.3, we compare the total cost of each method, both search and model training, with the final accuracy. This comparison has the upside that it can consider work with different pipelines and models more fairly.

4.1.1 Comparison to Published Results

In Table 2, we compare TA to all methods that used the setup of AutoAugment [1] or a very similar setup in terms of hyper-parameters, number of epochs and models.

TA performs as well or better than previous methods in almost all tasks. The SVHN datasets are the only exception, with RA performing somewhat better. This might, however, be due to our training pipeline, since, as we show in Section 4.1.2, we were not able to reproduce RA's performance for SVHN Core with our pipeline and the original training pipeline is not available.

For ImageNet, TA outperformed all other methods in terms of both top-1 accuracy and top-5 accuracy. We used an image width of 244 like RA [2], but even with a lower width of 224 (as was used for AA [1]), TA outperformed the previously best methods (with a $77.97 \pm .21$ top-1 accuracy and $93.98 \pm .07$ top-5 accuracy; not listed in the table).

In this comparison, we cannot compare to all previous methods, since some use different setups. The best-known setup we had to leave out is Adv. AA. Therefore, we perform an extra set of experiments following its setup closely.

Adv. AA uses eight times the compute for its final training compared to other methods and therefore has a significant advantage compared to other methods. Adv. AA is based on batch augmentation [10], where a set of workers in a data parallel setting each compute gradients with respect to the same batch of examples, but apply different augmentations to the images in it. We re-created this setup, including all hyper-parameters and batch augmentation, for TA. In Table 3, we compare TA with Adv. AA with a Wide-ResNet-28-10 and a ShakeShake-26-2x96d for both CIFAR-10 and CIFAR-100. We show that TA's trivial uniform sampling of a single augmentation achieves the same performance as their complex (and unavailable) reinforcement learning pipeline.

	Default	PBA	Fast AA	AA	RA	UA	TA (Wide)
CIFAR-10							
Wide-ResNet-40-2	96.16 ± .08	-	**96.4**	96.3	-	96.25	96.32 ± .05
Wide-ResNet-28-10	97.03 ± .07	**97.4**	97.3	**97.4**	97.3	97.33	**97.46** ± .06
ShakeShake-26-2x96d	97.54 ± .07	98.0	98.0	98.0	98.0	98.10	**98.21** ± .06
PyramidNet	97.95 ± .05	**98.5**	**98.5**	98.3	**98.5**	98.5	**98.58** ± .04
CIFAR-100							
Wide-ResNet-40-2	78.42 ± .31	-	79.4	79.3	-	79.01	**79.86** ± .19
Wide-ResNet-28-10	82.22 ± .25	83.3	82.7	82.9	83.3	82.82	**84.33** ± .17
ShakeShake-26-2x96d	83.28 ± .14	84.7	85.4	85.7	-	85.00	**86.19** ± .15
SVHN Core							
Wide-ResNet-28-10	97.12 ± .05	-	-	98.0	**98.3**	-	98.11 ± .03
SVHN							
Wide-ResNet-28-10	98.67 ± .02	98.9	98.8	98.9	**99.0**	-	98.9 ± .02
ImageNet							
ResNet-50	77.20 ± .32 (93.43 ± .11)	-	77.6 (93.7)	77.6 (**93.8**)	77.6 (**93.8**)	77.63 (-)	**78.07** ± .27 (**93.92** ± .09)

Table 2: The average test accuracies from ten runs, besides for ImageNet, where we used five runs. The 95% confidence interval is noted with ±. The trivial TA is in all benchmarks among the top-performers. The only exception is the comparison to RA's performance on the SVHN benchmarks, but this difference was non-existent in our reimplementation in 4.1.2.

	Adv. AA	TA (Wide)
CIFAR-10		
Wide-ResNet-28-10	**98.10** ± .15	**98.04** ± .06
ShakeShake-26-2x96d	**98.15** ± .12	**98.12** ± .12
CIFAR-100		
Wide-ResNet-28-10	**84.51** ± .18	**84.62** ± .14
ShakeShake-26-2x96d	**85.90** ± .15	**86.02** ± .13

Table 3: A comparison of TA with Adv. AA in the augmented batch setting on a Wide-ResNet-28-10. We report the average over five runs.

We conclude from this section that, for almost all considered benchmarks across datasets, models and even the way augmentations are applied, TA is among the top-performing methods.

4.1.2 Comparison of Reproduced Results in a Fixed Training Setup

While in the previous section, we tried to mitigate confounding factors by comparing results obtained with very similar setups with each other, in this section, we go one step further. We reproduce the results of four methods and compare our baseline method with these reproductions in order to yield a true apples-to-apples comparison.

As we present a very cheap and simple augmentation method we picked RA, Fast AA and UA as other cheap and simple augmentation methods for our comparison. Additionally, we compare to AA, as an important, common baseline. Moreover, for all of these methods relevant information for reproduction was published[3].

For RA, AA and Fast AA we used the published policies and did not search for an augmentation policy from scratch. We based both our RA and AA implementations on a public codebase[4] by the authors of both RA and AA that implements AA for the CIFAR datasets. Likewise, for Fast AA we based our implementation on a public codebase. No code is published for UA, and there are multiple hyper-parameters missing in the paper; in these cases, we used the hyper-parameters from RA. For our reproduction of UA, we also adopted the same discretization of the augmentation strengths into 31 values used by the other methods. In addition to the original augmentation space of UA we also perform experiments with the RA augmentation space.

We reran experiments for CIFAR-10, CIFAR-100 and SVHN Core, and present the results in Table 4. For each method we ran the benchmarks included in the original work. Generally, we could reproduce most results or even improve upon published results. The only severe exception is RA for which we tried multiple changes to the setup, but were not able to reach their published scores – neither for CIFAR nor for SVHN Core.

In this evaluation, TA (Wide) performed best across all methods for each benchmark with a Wide-Resnet-28-10, and TA (RA) performed best for both Wide-Resnet-40-2 benchmarks.

In addition to the reproductions of published policies, we applied RandAugment to the Wide-ResNet-40-2 on CIFAR-

---

3. See Table 12 in the appendix for an overview of the published materials of different methods

4. https://github.com/tensorflow/models/tree/fd34f711f319d8c6fe85110d9df6e1784cc5a6ca/research/autoaugment

10, which was originally not considered in the RA paper. We therefore had to search for a policy first. Depending on the task, Cubuk *et al.* [2] considered different subsets of the full range of the augmentation strengths $M \subset \{1, \ldots, 30\}$ and the number of consecutive augmentations $N \subset \{1, \ldots, 3\}$. In order to avoid missing the best candidates and to not require human intuition we searched on all 90 resulting combinations of RA's parameters. We split up a validation set of 10000 examples like in the original RandAugment method to evaluate the settings. We then picked the best setting and compared it to TA. Table 5 clearly shows that TA performs better than the costly RA setup, even though the RA setup in total required 91 full trainings, compared to a single training for TA.

Finally, we consider three more evaluations in the appendix: (i) We show that TA performs comparably or better on the same augmentation space with other automatic augmentation methods (see Appendix B), (ii) we show that TA generalizes to more peculiar datasets (see Appendix C) and (iii) we show TA's effectiveness with the EfficientNet Architecture [18] (see Appendix D).

WRN-28-10	CIFAR-10	CIFAR-100
AA	$97.31 \pm .22$ (-.09)	$82.91 \pm .41$ (+.01)
FAA	$\mathbf{97.43} \pm .09$ (+.13)	$83.27 \pm .13$ (+.57)
RA	$97.12 \pm .14$ (-.18)	$83.1 \pm .32$ (-.20)
UA (UA)	$\mathbf{97.46} \pm .14$ (+.13)	$83.08 \pm .27$ (+.26)
UA (RA)	$\mathbf{97.44} \pm .09$	$83.36 \pm .18$
TA (RA)	$\mathbf{97.46} \pm .09$	$83.54 \pm .12$
TA (Wide)	$\mathbf{97.46} \pm .06$	$\mathbf{84.33} \pm .17$

(a)

WRN-40-2	CIFAR-10	CIFAR-100
AA	$96.38 \pm .10$ (+.08)	$79.66 \pm .17$ (+.36)
FAA	$96.39 \pm .06$ (-.01)	$79.79 \pm .21$ (+.39)
UA (UA)	$96.42 \pm .04$ (+.17)	$79.74 \pm .15$ (+.73)
UA (RA)	$96.45 \pm .06$	$\mathbf{79.95} \pm .20$
TA (RA)	$\mathbf{96.62} \pm .09$	$\mathbf{79.99} \pm .16$
TA (Wide)	$96.32 \pm .05$	$\mathbf{79.86} \pm .19$

(b)

WRN-28-10	SVHN Core
AA	$97.99 \pm .06$ (-.01)
RA	$98.06 \pm .04$ (-.24)
TA (RA)	$98.05 \pm .02$
TA (Wide)	$\mathbf{98.11} \pm .03$

(c)

Table 4: A reproduction of the results of previous work with a Wide-ResNet-28-10 on CIFAR (a) and SVHN Core (c), and with a Wide-ResNet-40-2 on CIFAR (b). We report the relative performance difference to the published results in parentheses.

Method	Acc.
Brute-Force RA	$96.42 \pm .09$
TA (RA)	$\mathbf{96.62} \pm .09$

Table 5: Average over ten runs on CIFAR-10 with a Wide-ResNet-40-2. TA performs better than the over 80-times more expensive exhaustive search over RA's parameters.

4.1.3 Comparison by Total Compute Costs

In the previous sections, we compared different augmentation methods for a fixed training setup. We now consider the other extreme, comparing all methods across models and setups by their compute requirements.

In Figure 3, we plot this comparison for many CIFAR-100 setups across the literature. The question this plot answers is: given some compute budget, what method should we choose for the best final accuracy? For this plot, we used the accuracy numbers published in the literature and estimated the compute costs in RTX 2080 Ti GPU-hours. See Appendix E for a detailed account of the information used to calculate the compute cost approximations for all setups. We had to restrict the set of models we considered to the set of models for which we know from our experiments how expensive they are to run, namely all Wide-ResNet setups and the ShakeShake-26-2x96d. We tried to be as conservative as possible regarding the compute requirements of other methods, to not give TA an unfair advantage.

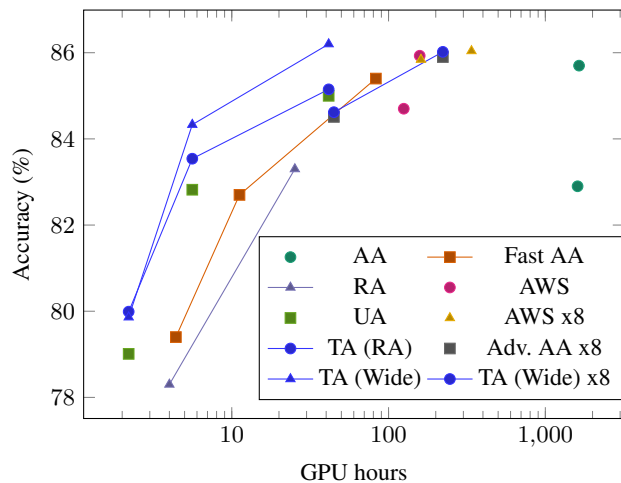In the figure, for all considered budgets, TA and its vari-

Figure 3: Comparison of the final test accuracy on CIFAR-100 in comparison to RTX2080ti GPU-hours compute invested for *augmentation search and final model training* across a set of models. Methods marked with *x8* use batch augmentations[10].

Augmentation space	SVHN Core	CIFAR-10
Full	$97.63 \pm .06$	$97.24 \pm .03$
AA	$98.04 \pm .02$	$97.47 \pm .11$
AA - {Invert}	$97.97 \pm .08$	$\mathbf{97.55} \pm .06$
RA	$98.05 \pm .02$	$97.46 \pm .09$
Wide	$\mathbf{98.11} \pm .03$	$97.46 \pm .06$
UA	$98.06 \pm .04$	$97.42 \pm .07$
OHL	$\mathbf{98.10} \pm .02$	$97.45 \pm .05$

Table 6: Evaluation of TA on SVHN Core and CIFAR-10 with a set of 7 different augmentation spaces. Note that $RA = AA - \{SamplePairing, Invert, Cutout\}$ and $UA = AA - \{SamplePairing\}$.

ant with augmented batch (TA x8) perform among the best methods. TA also has a clear benefit compared to the popular cheap methods Fast AA and RA for all compute budgets; finally, it is dramatically cheaper than AA.

4.2. Understanding the Minimal Requirements of TA

While so far, we have demonstrated that in many circumstances TA's approach of only using a single augmentation per image is enough or yields even better performance than more complicated methods, in this section we will dissect other properties of TA.

We first analyse how TA behaves across augmentation spaces from the literature. We then look at its performance after we apply random changes to its augmentation space. Finally, we consider sets of different augmentation strengths from which TA samples.

4.2.1 TA with Different Hand-Picked Augmentation Spaces

For this evaluation, we carefully reimplemented the augmentation spaces of AA, UA and OHL, besides the one of RA. Additionally, we consider a larger augmentation space (Full), which is a super set of AA, and additionally contains a blur, a smooth, a horizontal and a vertical flip. Especially the vertical flip is likely not useful for very many classification tasks. See Table 8 in the appendix for an overview of the augmentation spaces.

Table 6 indeed shows that TA performs worse on the full augmentation space than on all other augmentation spaces for a Wide-ResNet-28-10 on both SVHN Core and CIFAR-10.

We also included another augmentation space not considered in the previous literature: a variant of the AA augmentation space, where we removed the extreme invert operation, which maps each pixel x to $255-x$. We can see that this augmentation space performs very well for CIFAR-10, but not great for SVHN Core. This aligns well with observations made by earlier work, indicating that the invert
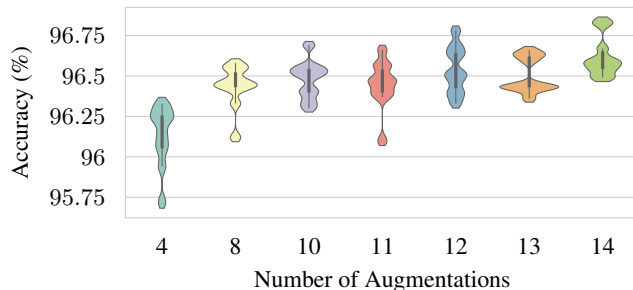

Figure 4: The performance of WRN-40-2 models depending on the size of sampled subsets of the RA augmentations on CIFAR-10. We performed 10 evaluations per subset size.

augmentation fosters generalization on SVHN, but not on the other datasets[1]. A peculiarity of the OHL augmentation space is that it only uses three strengths, unlike all other methods which consider 31 strengths. Interestingly, this is not harmful and OHL yields the best score for SVHN Core.

We can see that the performance of TA is rather stable between augmentation spaces, but still there seems to be room for improvement by a more sophisticated method to choose the augmentation space for TA depending on the task.

4.2.2 TA's Behavior With Randomly Pruned Augmentation Spaces

While we assessed performance with different hand-crafted augmentation spaces above, now we want to analyze how performance is impacted if we only use random subsets of the 14 augmentations in the RA augmentation space (which we used in the other experiments unless otherwise stated).

In Figure 4, we analyze the performance and its variance for multiple augmentation subset sizes for a Wide-ResNet-40-2 on CIFAR-10. We performed 10 evaluations per sample size, where in each evaluation we picked a random sample of augmentations. While performance decreases as fewer and fewer augmentations are considered, we can see that it drops very slowly. We can throw away 4 of 14 augmentations and still obtain performance close to the original performance. Another trend is that with fewer augmentations the variance increases. This is likely due to the randomness of the subset choice per run, which increases for smaller subsets.

4.2.3 The Impact of the Set of Strengths on TA's Performance

Before, we mostly considered the impact of different sets of augmentations; now we consider the other component of the augmentation space: the set of strengths.

Strengths	CIFAR-10	CIFAR-100	SVHN Core
$\{30\}$	**97.45** $\pm$ .05	82.98 $\pm$ .22	**98.16** $\pm$ .03
$\{0, 30\}$	**97.51** $\pm$ .08	**83.46** $\pm$ .10	98.02 $\pm$ .02
$\{0, 15, 30\}$	**97.46** $\pm$ .06	**83.43** $\pm$ .24	98.04 $\pm$ .03
$\{0, \ldots, 30\}$	**97.46** $\pm$ .09	**83.54** $\pm$ .12	98.05 $\pm$ .02

Table 7: A comparison of the performance of TA (RA) on different datasets with a Wide-ResNet-28-10 using different subsets of strengths.

In Table 7, we analyze the performance of TA with a Wide-ResNet-28-10 and different subsets of the original set of possible strengths $\{0, \ldots, 30\}$ on the RA augmentation space. We can see that the CIFAR-10 setup seems to be relatively agnostic to the set of strengths. Performance on CIFAR-100, on the other hand, is very negatively impacted by choosing the subset $\{30\}$. In general, performance improves on CIFAR-100 with larger sets. For SVHN Core, the opposite is the case: performance improves when only considering $\{30\}$. A reason for this could be that the majority of the augmentations are color based and changing the colors of a single-color background and a single-color number drastically, still in most cases yields valid house numbers.

Another observation we made is that it does not matter so much for any setup whether we reduce to three or just two augmentation strengths, compared to all 31. This seems to point towards the importance of a mixture of strong and weak augmentations. At the same time three different strengths, compared to 31, seem to be enough for these settings.

5. Automatic Augmentation Methods in Practice

While there are many expensive or hard to reproduce automatic augmentation methods, it is important that augmentation methods are practical: the impact of automatic augmentation methods unfolds in the application to new setups and problems. We evaluated many different settings and augmentation methods and we would like to pass on the gained knowledge.

First, we have compiled a short summary of learnings for the application of augmentation methods in Appendix F.

Second, in addition to our full codebase, we provide a simple one-file python library that implements the more practical augmentation methods: RA, UA and TA. It even allows choosing from all augmentation spaces considered in this work. For example, to get an image augmenter for TA and transform a PIL image `img`, one can call

```
aug = TrivialAugment(n,m)
augmented_img = aug(img)
```

6. Best Practices Proposal for Research

We found that it is difficult to reimplement many of the published methods, see Table 12 in the appendix. We also found that many methods performed similarly to the simple TA baseline, when we follow their setup. Here, we compile a short bullet point list of best practices we believe are important for sustainable research in this field.

- Share code as much as possible for easy entry of beginners and to make sure that setups are similar across papers. Otherwise, differences between the actual implementation and its description in the paper can impair reproducibility.

- Compare fairly to other methods and baselines with the same setup, train budget and augmentation space, or reproduce results of previous methods in your setup and mention differences.

- Report confidence intervals to discern "outperforming" from "performing comparably".

7. Limitations

While we could not find settings where TA failed for image classification, we found that TA does not work out-of-the-box for object detection setups and also needs tuning to work for this task. So far, we can only wholeheartedly recommend the use of TA for image classification; its application to other computer vision tasks requires further study.

8. Conclusion

Most of the approaches considered as automatic augmentation methods are complicated. In this work, we presented TA, a very simple augmentation algorithm from which we can learn three main things.

First, TA teaches us about a crucial baseline missing for automatic augmentation methods.

Second, TA teaches us to never overlook the simplest solutions. There are a lot of complicated methods to automatically find augmentation policies, but the simplest method was so-far overlooked, even though it performs comparably or better.

Third, randomness in the chosen strengths appears to be very important for good performance.

Acknowledgements

References

[1] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasude-van, 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 augmenta-tion 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 regular-ization of convolutional neural networks with cutout, 2017.

[4] Marzieh Fadaee, Arianna Bisazza, and Christof Monz. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Associ-ation for Computational Linguistics (Volume 2: Short Pa-pers)*, pages 567–573, 2017.

[5] Xavier Gastaldi. Shake-shake regularization, 2017.

[6] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. `https://github.com/facebookresearch/detectron`, 2018.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceed-ings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

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

[10] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[11] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[12] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet clas-sification with deep convolutional neural networks. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Wein-berger, editors, *Proceedings of the 26th International Con-ference on Advances in Neural Information Processing Sys-tems (NeurIPS'12)*, pages 1097–1105, 2012.

[13] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 6665–6675. Curran Associates, Inc., 2019.

[14] Chen Lin, Minghao Guo, Chuming Li, Xin Yuan, Wei Wu, Junjie Yan, Dahua Lin, and Wanli Ouyang. Online hyper-parameter learning for auto-augmentation strategy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[15] 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, 2020.

[16] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient de-scent with warm restarts. In *Proceedings of the International Conference on Learning Representations (ICLR'17)*, 2017.

[17] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bis-sacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Work-shop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[18] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[19] Keyu Tian, Chen Lin, Ming Sun, Luping Zhou, Junjie Yan, and Wanli Ouyang. Improving auto-augment via augmentation-wise weight sharing. *Advances in Neural In-formation Processing Systems*, 33, 2020.

[20] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6256–6268. Curran Associates, Inc., 2020.

[21] Sergey Zagoruyko and Nikos Komodakis. Wide residual net-works. In *In Proceedings of BMCV'16*, 2016.

[22] Xinyu Zhang, Qiang Wang, Jian Zhang, and Zhao Zhong. Adversarial autoaugment. In *International Conference on Learning Representations*, 2020.

[23] Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmenta-tion strategies for object detection. In *European Conference on Computer Vision*, pages 566–583. Springer, 2020.