Supplementary Material for FreeAugment: Data Augmentation Search Across All Degrees of Freedom

Tom Bekor*, Niv Nayman*, and Lihi Zelnik-Manor

Technion - Israel Institute of Technology, Haifa, Israel
Corresponding author: tom.bekor@campus.technion.ac.il

A Policy Search Cost

Tab. 1 summarizes the policy search cost for different DAS methods. While achieving better or comparable performance with all of those, FreeAugment finds augmentation policies with a relatively small cost, especially considering that the cost is reported for a single search, while some other methods have to repeatedly apply it for several policy depth values. Some other methods are limited to searching merely two consecutive transformations due to the exponential growth of the cost with the policy depth. FreeAugment is the only one to eliminate that entirely, requiring a single search to obtain all degrees of freedom, including policy depth, without resulting in iterative solutions.

Table 1: Policy search cost in GPU hours on CIFAR10/100 with WRN40-2. Each method is reported with its corresponding hardware and time. -: missing information. *Reported in DDAS. **Exponential complexity in policy depth and hence limited to 2 transformations only. ***Requires additional search on policy depth.

Method	GPU hours	Hardware	Architecture
AA [1] **	5000	Tesla P100	WRN-40-2
PBA [8] **	5	Titan XP	WRN-40-2
FastAA [10] **	3.5	Tesla V100	WRN-40-2
DADA [9] **	0.1/0.2	Titan XP	WRN-28-10
RA [2] * **	33	RTX 2080Ti	WRN-28-10
FasterAA [5] **	0.23	Tesla V100	WRN-40-2
DeepAA [16]	9	-	WRN-28-10
DRA [15] ***	0.4	Tesla P100	WRN-28-10
DDAS [11] ***	0.15	RTX 2080Ti	WRN-28-10
SLACK [12] ***	4	-	WRN-40-2
MADAO [6] ***	1.7	-	WRN-28-2
FreeAugment	1.2	RTX A6000	WRN-40-2

* Equal contribution.

B Found Augmentation Policies

Fig. 1 visualizes the data augmentation policies found by FreeAugment.

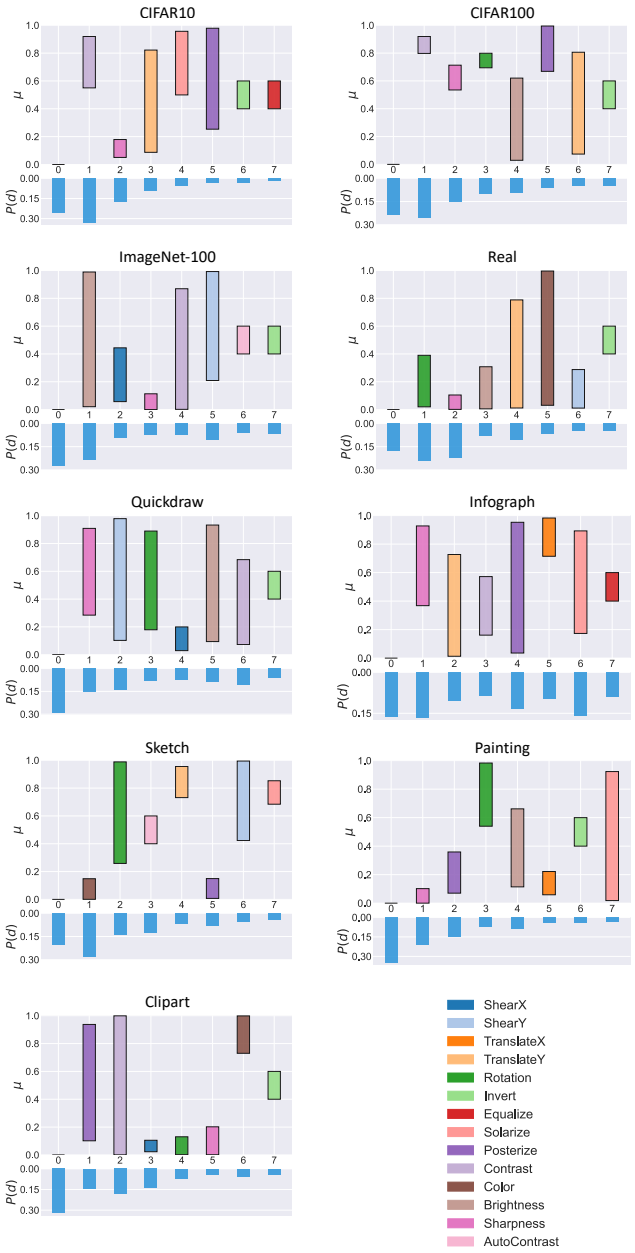


Fig. 1: Found augmentation policies on CIFAR10/100, ImageNet-100, and DomainNet. Transformations that are not associated with a learnable magnitude are depicted with a magnitude range of $[0.4, 0.6]$

C Hyper-Parameters and Training Configurations

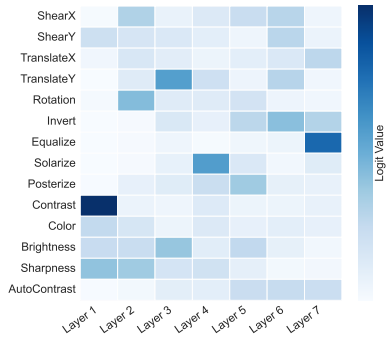
The configurations for policy search and evaluation of each dataset can be found in Tab. 2 and Tab. 3, respectively. Following previous work, both in search and evaluation, the classifier's learning rate is decayed according to a cosine annealing scheduler. Following TA [13], for CIFAR10/100 images we apply pad-and-crop to a resolution of 32×32 with a reflection mode and a pad length of 4, random horizontal flipping with a probability of 0.5, then our generated policy is applied, followed by cutout [4] with a length of 16. For ImageNet-100 and DomainNet we use random resized crops and scales between 0.08 and 1.0 to a resolution of 224×244 using bicubic interpolation, random horizontal flipping with a probability of 0.5, then our generated policy is applied, followed by cutout with a length of 75, which is about 1/3 of the image size, as done in [14].

Table 2: Policy search hyper-parameters. Classifier hyper-parameters are identical to those in the policy evaluation.

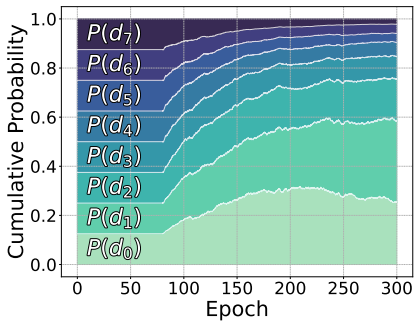
	CIFAR10/100	ImageNet-100	DomainNet
Architecture	WRN-40-2, WRN-28-10	ResNet-18	ResNet-18
Epochs	300	300	300
Batch size	128	64	64
Classifier Hyper-Parameters		– Same as in Tab. 3 –	
Policy optimizer	Adam	Adam	Adam
μ learning rate	0.02	0.02	0.02
Π learning rate	0.01	0.01	0.01
δ learning rate	1.0	1.0	1.0

Table 3: Policy evaluation hyper-parameters.

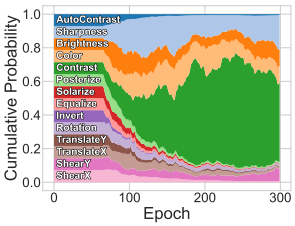
	CIFAR10/100	ImageNet-100	DomainNet
Architecture	WRN-40-2, WRN-28-10	ResNet-18	ResNet-18
Epochs	200	270	200
Batch size	128	256	128
Optimizer	SGD	SGD	SGD
Learning rate	0.1	0.1	0.1
Weight decay	0.0005	0.0001	0.0001
Momentum parameter	0.9	0.9	0.9
Nesterov Momentum	True	True	True



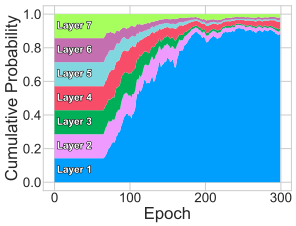
(a) Found policy distribution over transformation types and order



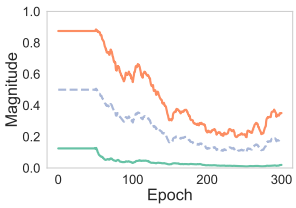
(b) Policy depth dynamics



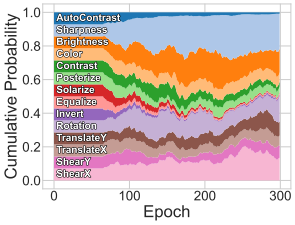
(c) Layer 1



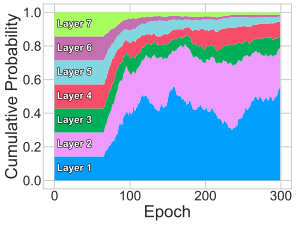
(d) Contrast across layers



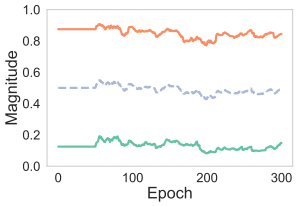
(e) Contrast magnitude



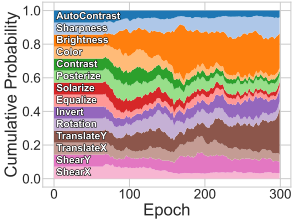
(f) Layer 2



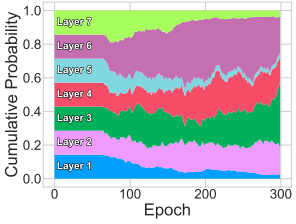
(g) Sharpness across layers



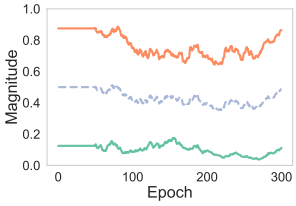
(h) Sharpness magnitude



(i) Layer 3

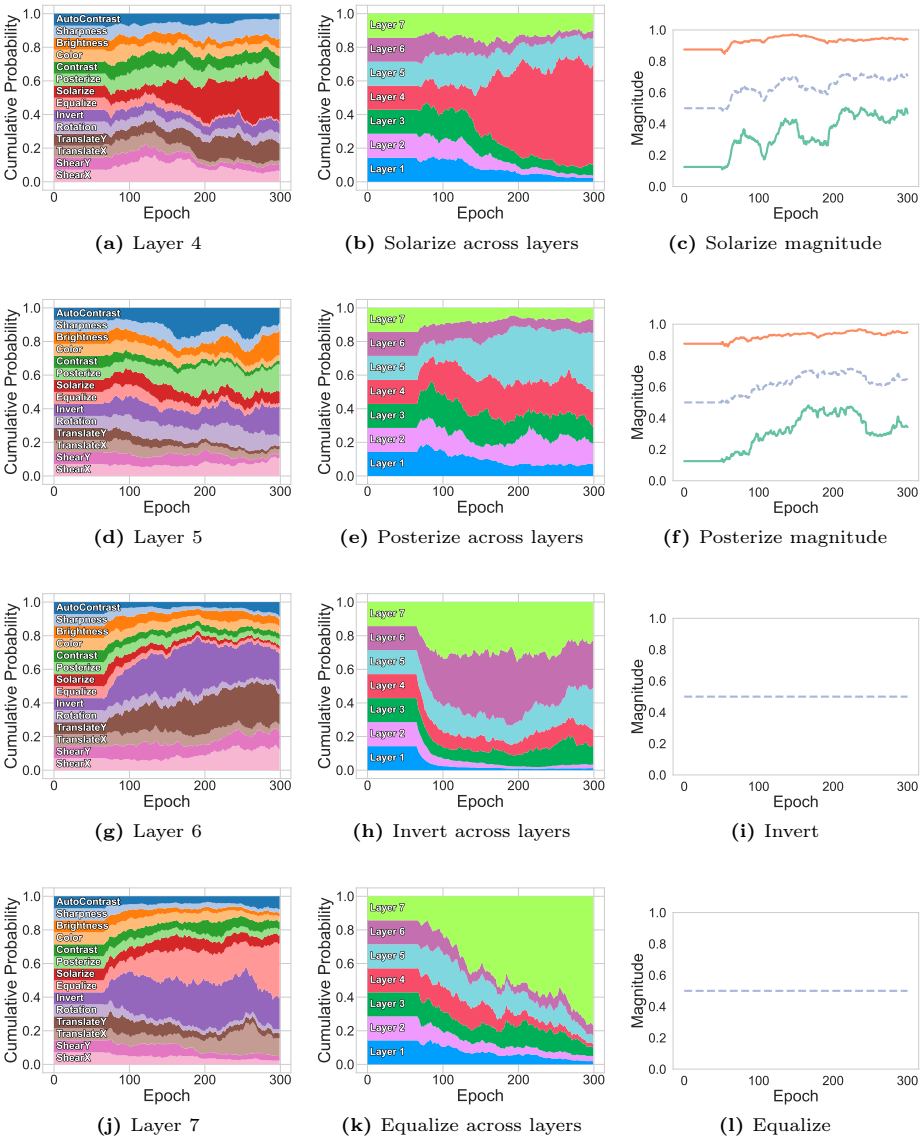


(j) TranslateY across layers



(k) TranslateY magnitude

Fig. 2: Policy dynamics during the search phase of CIFAR10 (Part 1/2)



(a) Layer 4 **(b)** Solarize across layers **(c)** Solarize magnitude

(d) Layer 5 **(e)** Posterize across layers **(f)** Posterize magnitude

(g) Layer 6 **(h)** Invert across layers **(i)** Invert

(j) Layer 7 **(k)** Equalize across layers **(l)** Equalize

Fig. 3: Policy dynamics during the search phase of CIFAR10 (Part 2/2)

D Search Space Details

Tab. 4 lists the transformations in the search space of FreeAugment with their corresponding magnitude ranges. All magnitude parameters correspond to Kornia's image transformations[1].

Table 4: List of transformations in the search space and their corresponding magnitude range. Note that AutoContrast, Invert, and Equalize do not have magnitude parameters.

Transformation	Magnitude Range
ShearX	[-0.6, 0.6]
ShearY	[-0.6, 0.6]
TranslateX	[-0.5, 0.5]
TranslateY	[-0.5, 0.5]
Rotate	[-30, 30]
Solarize	[0.6, 1]
Posterize	[2, 8]
Contrast	[0.4, 2.0]
Color	[0.0, 1.0]
Brightness	[-0.4, 0.4]
Sharpness	[0.0, 2.0]
AutoContrast	-
Invert	-
Equalize	-

E Additional Ablations

E.1 Sample single augmentation per-batch vs. per-image

FreeAugment samples distinct augmentation sequences for every image in the batch rather than sampling once for the entire batch. Fig. 4 outlines a simple and efficient PyTorch per-image sampling implementation of Algorithm 1 in the main text. Tab. 5 compares the performance of FreeAugment when it samples the same augmentation policy for the entire batch versus per-image. As can be inferred from the table, the per-image sampling helps to reduce the variance of the gradient estimates with respect to ϕ during the search and thus leads to better results.

E.2 No Warm-up For ϕ

Tab. 6 compares the results of a search phase that begins with a warm-up for the learnable policy parameters ϕ and a search phase that starts learning those

[1] https://kornia.readthedocs.io

Table 5: Top-1 test accuracy (%) on CIFAR100 for WRN-40-2 with different variants of FreeAugment's, one that samples different augmentations per-image in the batch, and one that samples the same augmentations for the whole batch. The results are averaged over 5 random seeds, with the 95% confidence interval denoted by ±.

FreeAugment variant	Top-1 Accuracy
Sample augmentation per-batch	79.43 ± .23
Sample augmentation per-image	**80.04 ± .23**

at the very beginning. At the former, ϕ are not updated until the classifier is sufficiently learned so that the gradients of ϕ backpropagating through it are meaningful. Such strategy aims to avoid noise in the learning process of ϕ caused by a premature classifier, yielding a bad approximation of the lower problem of the bilevel optimization. As can be seen from the results, warm-up improves the performance.

Table 6: Top-1 test accuracy (%) on CIFAR100 for WRN-40-2 where FreeAugmentlearns with and without warm-up. The results are averaged over 5 random seeds, with the 95% confidence interval denoted by ±.

FreeAugment variant	Top-1 Accuracy
Without warm-up	79.53 ± .13
With warm-up	**80.04 ± .23**

E.3 Magnitude Distribution

Tab. 7 compares the results of FreeAugment variant that learns the a uniform distribution over each magnitude m, versus a Gaussian distribution, such that,

$$m = \hat{\sigma} \cdot \epsilon + \hat{\mu} \quad ; \quad \epsilon \sim \mathcal{N}(0, 1)$$

with $\hat{\mu}, \hat{\sigma}$ being the learnable magnitude parameters and $\mathcal{N}(0, 1)$ is the normal distribution.

The uniform distribution is initialized to 75% of the normalized range as $(0.125, 0.875)$, and the normal distribution is initialized with a mean $\hat{\mu}$ of 0.5 and a standard deviation $\hat{\sigma}$ of 0.1875. This results in samples falling in the interval of $(0.125, 0.875)$ with a probability of 0.95. As can be seen from the table, the results using both sampling methods are fairly close, with slightly better results using uniform magnitude sampling.

E.4 Depth Search by Probability of Application

Additional depth-searching strategy treats augmentation application as sampling from a learnable Bernoulli distribution, similar to the methods in [5, 7, 10, 15]

Table 7: Top-1 test accuracy (%) on CIFAR100 for WRN-40-2 with different variants of FreeAugment's, one that learns a uniform magnitude distribution, and another that learns a normal magnitude distribution. The results are averaged over 5 random seeds, with the 95% confidence interval denoted by ±.

FreeAugment variant	Top-1 Accuracy
Gaussian Magnitudes	79.9 ± 0.19
Uniform Magnitudes	**80.04 ± .23**

Tab. 8 compares the results of FreeAugment variants using different depth-searching strategies. Learnable application probabilities for each augmentation were initialized to 0.75. As shown in the table, FreeAugment with a learnable Gumbel-Softmax distribution over policy depths achieves better performance. Compared to FreeAugment with a single representation for each policy depth d out of D transformations, learning the application probability for each transformation effectively yields $\binom{D}{d}$ representations for the same policy depth d. The resulting redundancy makes the DAS operate within a much larger representation of the search space without effectively adding more policies.

Table 8: Comparing strategies of depth search. Top-1 test accuracy (%) on CIFAR100 for WRN-40-2. The results are averaged over 5 random seeds, with the 95% confidence interval denoted by ±.

Variant	Top-1 Accuracy
Application Prob. + Gumbel-Softmax	78.36 ± .51
Application Prob. + Gumbel-Sinkhorn	79.11 ± .30
FreeAugment	**80.04 ± .23**

E.5 Continue Evaluation with End-of-Search Classifier

Tab. 9 compares the results of the evaluation phase where the classifier is being initialized with the end-of-search model weights, with a linear warm-up of 5 epochs for its learning rate, versus training the classifier from scratch. As can be seen from the table, training the classifier from scratch achieves better performance, as it is effectively trained with an optimized augmentation policy from the beginning of its training.

Table 9: Top-1 test accuracy (%) on CIFAR100 for WRN-40-2 where the evaluation phase continues with the same classifier from the end of the search versus a classifier that was trained from scratch. The results are averaged over 5 random seeds, with the 95% confidence interval denoted by ±.

FreeAugment variant	Top-1 Accuracy
End-of-search	79.85 ± .1
Scratch	**80.04 ± .23**

F Additional Experiments

Tab. 10 shows the performance of FreeAugment on the complete ImageNet [3] dataset. While evaluating with ResNet-50, we used the found augmentation policy on ImageNet-100 for ResNet-18. FreeAugment surpasses the baseline by **+1.33%** and achieves competitive performance when compared to more recent methods.

Table 10: Top-1 test accuracy (%) on ImageNet for ResNet-50 using the found augmentation policy on ImageNet-100. Results are obtained following TA training recipe and averaged across 4 random seeds, with the 95% confidence interval denoted by ±.

	Baseline	AA	FastAA	FasterAA	DADA	TA(Wide)	DDAS	DRA	FreeAugment
ResNet-50	76.3	77.6	77.6	76.5	77.5	78.07	77.7	78.19	77.63 ± .15

Tab. 11 and 12 present the same experiments as of Tab. 3 and 2 in the main paper, respectively, with the only change of replacing the more general training scheme proposed by TA [13] by a tailored one for SLACK [12]. FreeAugment shows competitive results even when adopting SLACK's designated training scheme.

Table 11: Top-1 test accuracy (%) on DomainNet for ResNet-18. The left value in each cell is obtained using SLACK's training recipe, while the right value is a reproduced result using TA's training recipe. All results are averaged across 4 random seeds.

	DomainBed	TA (Wide)	SLACK	FreeAugment
Real	62.54, 61.91	71.56, 70.27	71.00, 68.94	71.14, 71.47
Quickdraw	66.54, 64.55	68.60, 67.01	68.14, 66.39	68.58, 68.62
Infograph	26.76, 25.64	35.44, 33.73	34.78, 30.92	34.72, 34.88
Sketch	59.54, 58.57	66.21, 65.38	65.41, 63.75	66.48, 66.74
Painting	58.31, 57.70	65.15, 64.13	64.83, 62.26	64.30, 64.65
Clipart	66.23, 64.07	71.19, 69.77	72.65, 70.92	71.06, 71.26
Average	57.23, 55.40	63.03, 61.71	62.80, 60.53	62.71, 62.93

Table 12: Top-1 test accuracy (%) on ImageNet-100 for ResNet-18. The left value in each cell is obtained using SLACK's training recipe, while the right value is a reproduced result using TA's training recipe. All results are averaged across 4 random seeds.

	Baseline	TA(Wide)	SLACK	FreeAugment
ResNet-18	-, 84.84	86.39, 85.68	86.06, 86.19	86.26, 86.62

G Learning Magnitudes of Non-Differentiable Transformations

It is important to note that some transformations are associated with a learnable magnitude but are not differentiable with respect to it (e.g., posterize and solarize). To get gradients for such transformations' magnitudes, we use the straight-trough estimator as:

$$\frac{\partial X^{k+1}}{\partial M_{ik}} = \frac{\partial \tau_i(X^k)}{\partial M_{ik}} = \mathbf{1} \tag{1}$$

where X^k, is the input image of the k^{th} policy layer, X^{k+1} is the output image of the same layer, τ_i is a non-differentiable elementary transformation, M_{ik} is the sampled magnitude at the k^{th} policy layer for τ_i, and $\mathbf{1}$ is a matrix from the same size as X^k filled with ones. Namely, Eq. (1) means that the gradient of each pixel in the output image X^{k+1} w.r.t. the sampled magnitude X^{k+1} equals 1.

In practice, this trick can be easily implemented via:

$$\hat{X}^{k+1} = X^{k+1} + M_{ik} - \text{StopGrad}(M_{ik}), \tag{2}$$

where $\text{StopGrad}(\cdot)$ is the analog to PyTorch's detach function, which returns the input value without passing its gradient. Note that $\partial X^{k+1}/\partial M_{ik} = \mathbf{0}$, where $\mathbf{0}$ is a matrix with the same size of X^{k+1} filled with zeros. Thus, the use of the trick in Eq. (2) eventually makes Eq. (1) holding.

```python
import torch
from torch.nn.functional import gumbel_softmax
from ops import gumbel_sinkhorn, uniform
from torch.distributions.uniform import

def FreeAugment(x, trans, phi):

    delta, Pi, mu = phi

    N, K = Pi.shape

    assert len(trans) == N
    assert len(delta) == K+1
    assert mu.shape[:-1] == Pi.shape

    bs = x.shape[0]

    delta = delta.repeat(bs, 1) # shape: (bs, K+1)
    Pi = Pi.repeat(bs, 1, 1) # shape: (bs, N, K)
    mu = mu.repeat(bs, 1, 1, 1) # shape: (bs, N, K, 2)

    d_hard = gumbel_softmax(delta) # shape: (bs, K+1)
    P_hard = gumbel_sinkhorn(Pi) # shape: (bs, N, K)
    M = uniform(mu) # shape: (bs, N, K)

    d_hard = d_hard.permute(1,0) # shape: (K+1, bs)
    P_hard = P_hard.permute(2,1,0) # shape: (K, N, bs)
    M = M.permute(2,1,0) # shape: (K, N, bs)

    out = d_hard[0] * x
    for k, (dh, Ph) in enumerate(zip(d_hard[1:], P_hard)):
        for i, (t_oh, t) in enumerate(zip(Ph, trans)):
            out += dh.view(-1, 1, 1, 1) * \
                   t_oh.view(-1, 1, 1, 1) * t(x, M[k,i])

    return out
```

Fig. 4: PyTorch per-image policy sampling code

References

1. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation strategies from data. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 113–123 (2019)
2. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. pp. 702–703 (2020)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
4. DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552 (2017)
5. Hataya, R., Zdenek, J., Yoshizoe, K., Nakayama, H.: Faster autoaugment: Learning augmentation strategies using backpropagation. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16. pp. 1–16. Springer (2020)
6. Hataya, R., Zdenek, J., Yoshizoe, K., Nakayama, H.: Meta approach to data augmentation optimization. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2574–2583 (2022)
7. He, X., Chu, X.: Medpipe: End-to-end joint search of data augmentation and neural architecture for 3d medical image classification. In: 2023 IEEE International Conference on Medical Artificial Intelligence (MedAI). pp. 344–354. IEEE (2023)
8. Ho, D., Liang, E., Chen, X., Stoica, I., Abbeel, P.: Population based augmentation: Efficient learning of augmentation policy schedules. In: International conference on machine learning. pp. 2731–2741. PMLR (2019)
9. Li, Y., Hu, G., Wang, Y., Hospedales, T., Robertson, N.M., Yang, Y.: Differentiable automatic data augmentation. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16. pp. 580–595. Springer (2020)
10. Lim, S., Kim, I., Kim, T., Kim, C., Kim, S.: Fast autoaugment. Advances in Neural Information Processing Systems **32** (2019)
11. Liu, A., Huang, Z., Huang, Z., Wang, N.: Direct differentiable augmentation search. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12219–12228 (2021)
12. Marrie, J., Arbel, M., Larlus, D., Mairal, J.: Slack: Stable learning of augmentations with cold-start and kl regularization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 24306–24314 (2023)
13. Müller, S.G., Hutter, F.: Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 774–782 (2021)
14. Wightman, R., Touvron, H., Jégou, H.: Resnet strikes back: An improved training procedure in timm. CoRR **abs/2110.00476** (2021), `https://arxiv.org/abs/2110.00476`
15. Xiao, A., Shen, B., Tian, J., Hu, Z.: Differentiable randaugment: Learning selecting weights and magnitude distributions of image transformations. IEEE Transactions on Image Processing **32**, 2413–2427 (2023). `https://doi.org/10.1109/TIP.2023.3265266`
16. Zheng, Y., Zhang, Z., Yan, S., Zhang, M.: Deep autoaugmentation. Proc. ICLR **1**(3), 6 (2022)