Headline

大家好：

这是2018年度第11篇Arxiv Weekly。

本文是 **模型研究** 方向的文章。

Highlight

> **自动数据增强(Auto Augment)算法，利用更高级的数据增强涨点**

Information

Title

AutoAugment: Learning Augmentation Policies from Data

Link

https://arxiv.org/pdf/1805.09501.pdf

Code

https://github.com/tensorflow/models/tree/master/
research/autoaugment

Source

- 谷歌（Google Brain）[Quoc V.Le 组]

Introduction

本文中试图进一步探究数据增强来提升分类网络的性能。文章提出了一种简明的数据增强流程，称为AutoAugment，下文简称AA。AA能够自动地搜索出针对数据集的最佳增强策略，相比于传统的经验主义增强更能提升网络在测试集上的表现。

AA的关键在于合理构建一个数据增强的待搜索空间，并且直接依据数据集的某些特性来给出某个候选增强策略的优劣。

在AA的实际实现中，我们设计了一个包含很多sub-policies的数据增强空间，并对每个mini-batch的数据随机挑选一种sub-policy进行增强。

每个sub-policy包含两个能被连续执行的子操作，所有操作均为某种意义上的图像增强算法（包括平移、旋转、切割、对称）等与一个概率函数的组合。这个函数决定这个图像增强算法以什么概率和什么强度被执行。

本文的方法在CIFAR、SVHN和ImageNet上都进行了实验，均取得了很好的效果。其中在ImageNet上获得了83.54%的t1；在CIFAR10上获得了98.52%的acc。

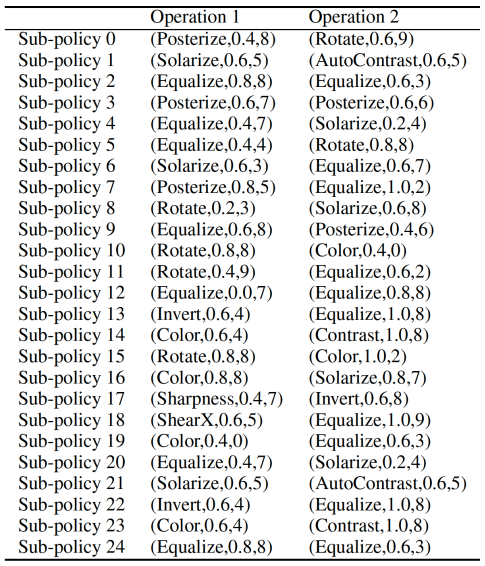
文章也通过实验证明，学习到的数据增强策略本身具备一定的可迁移性，能够不加finetune地应用在别的一些场景下。

Wide-ResNet, Shake-Shake和ShakeDrop backbone附加AA的版本已经开源在Codes中供大家参考查阅。

Keys

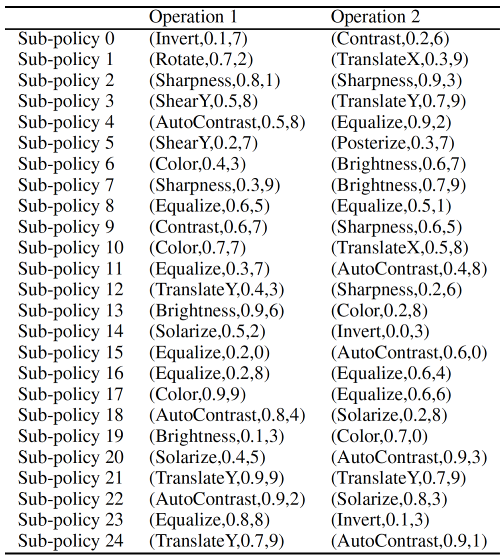
AA的核心非常容易解释，也即在通用的数据增强之余，想办法进一步进行一些能捞到好处的额外增强。

要简明扼要地捞到好处，就需要一个简明扼要的机制。文中最后给出的机制是：对于每个mini-batch，从一个sub-policy集合中随机挑选一种进行额外的增强。ImageNet、CIFAR和SVHN如下：



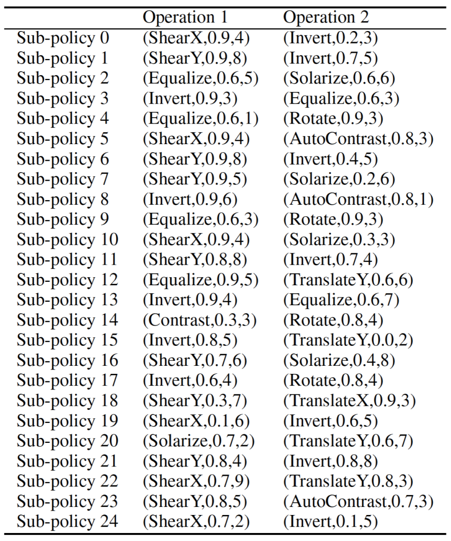
	Operation 1	Operation 2
Sub-policy 0	(Posterize,0.4,8)	(Rotate,0.6,9)
Sub-policy 1	(Solarize,0.6,5)	(AutoContrast,0.6,5)
Sub-policy 2	(Equalize,0.8,8)	(Equalize,0.6,3)
Sub-policy 3	(Posterize,0.6,7)	(Posterize,0.6,6)
Sub-policy 4	(Equalize,0.4,7)	(Solarize,0.2,4)
Sub-policy 5	(Equalize,0.4,4)	(Rotate,0.8,8)
Sub-policy 6	(Solarize,0.6,3)	(Equalize,0.6,7)
Sub-policy 7	(Posterize,0.8,5)	(Equalize,1.0,2)
Sub-policy 8	(Rotate,0.2,3)	(Solarize,0.6,8)
Sub-policy 9	(Equalize,0.6,8)	(Posterize,0.4,6)
Sub-policy 10	(Rotate,0.8,8)	(Color,0.4,0)
Sub-policy 11	(Rotate,0.4,9)	(Equalize,0.6,2)
Sub-policy 12	(Equalize,0.0,7)	(Equalize,0.8,8)
Sub-policy 13	(Invert,0.6,4)	(Equalize,1.0,8)
Sub-policy 14	(Color,0.6,4)	(Contrast,1.0,8)
Sub-policy 15	(Rotate,0.8,8)	(Color,1.0,2)
Sub-policy 16	(Color,0.8,8)	(Solarize,0.8,7)
Sub-policy 17	(Sharpness,0.4,7)	(Invert,0.6,8)
Sub-policy 18	(ShearX,0.6,5)	(Equalize,1.0,9)
Sub-policy 19	(Color,0.4,0)	(Equalize,0.6,3)
Sub-policy 20	(Equalize,0.4,7)	(Solarize,0.2,4)
Sub-policy 21	(Solarize,0.6,5)	(AutoContrast,0.6,5)
Sub-policy 22	(Invert,0.6,4)	(Equalize,1.0,8)
Sub-policy 23	(Color,0.6,4)	(Contrast,1.0,8)
Sub-policy 24	(Equalize,0.8,8)	(Equalize,0.6,3)

Table 10: AutoAugment policy found on reduced ImageNet.



	Operation 1	Operation 2
Sub-policy 0	(Invert,0.1,7)	(Contrast,0.2,6)
Sub-policy 1	(Rotate,0.7,2)	(TranslateX,0.3,9)
Sub-policy 2	(Sharpness,0.8,1)	(Sharpness,0.9,3)
Sub-policy 3	(ShearY,0.5,8)	(TranslateY,0.7,9)
Sub-policy 4	(AutoContrast,0.5,8)	(Equalize,0.9,2)
Sub-policy 5	(ShearY,0.2,7)	(Posterize,0.3,7)
Sub-policy 6	(Color,0.4,3)	(Brightness,0.6,7)
Sub-policy 7	(Sharpness,0.3,9)	(Brightness,0.7,9)
Sub-policy 8	(Equalize,0.6,5)	(Equalize,0.5,1)
Sub-policy 9	(Contrast,0.6,7)	(Sharpness,0.6,5)
Sub-policy 10	(Color,0.7,7)	(TranslateX,0.5,8)
Sub-policy 11	(Equalize,0.3,7)	(AutoContrast,0.4,8)
Sub-policy 12	(TranslateY,0.4,3)	(Sharpness,0.2,6)
Sub-policy 13	(Brightness,0.9,6)	(Color,0.2,8)
Sub-policy 14	(Solarize,0.5,2)	(Invert,0.0,3)
Sub-policy 15	(Equalize,0.2,0)	(AutoContrast,0.6,0)
Sub-policy 16	(Equalize,0.2,8)	(Equalize,0.6,4)
Sub-policy 17	(Color,0.9,9)	(Equalize,0.6,6)
Sub-policy 18	(AutoContrast,0.8,4)	(Solarize,0.2,8)
Sub-policy 19	(Brightness,0.1,3)	(Color,0.7,0)
Sub-policy 20	(Solarize,0.4,5)	(AutoContrast,0.9,3)
Sub-policy 21	(TranslateY,0.9,9)	(TranslateY,0.7,9)
Sub-policy 22	(AutoContrast,0.9,2)	(Solarize,0.8,3)
Sub-policy 23	(Equalize,0.8,8)	(Invert,0.1,3)
Sub-policy 24	(TranslateY,0.7,9)	(AutoContrast,0.9,1)

Table 8: AutoAugment policy found on reduced CIFAR-10.



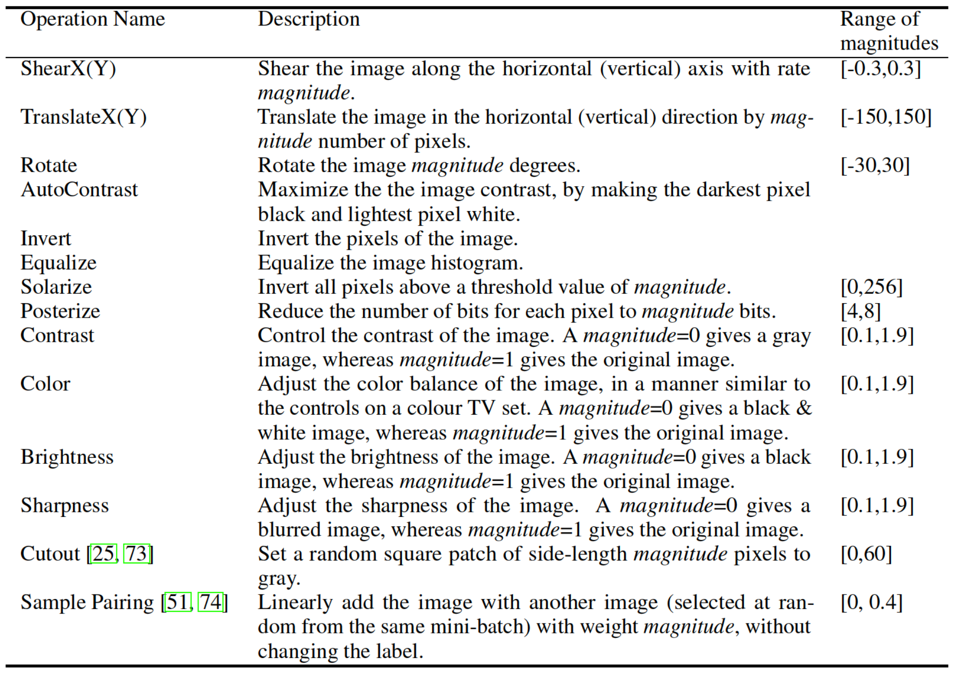
	Operation 1	Operation 2
Sub-policy 0	(ShearX,0.9,4)	(Invert,0.2,3)
Sub-policy 1	(ShearY,0.9,8)	(Invert,0.7,5)
Sub-policy 2	(Equalize,0.6,5)	(Solarize,0.6,6)
Sub-policy 3	(Invert,0.9,3)	(Equalize,0.6,3)
Sub-policy 4	(Equalize,0.6,1)	(Rotate,0.9,3)
Sub-policy 5	(ShearX,0.9,4)	(AutoContrast,0.8,3)
Sub-policy 6	(ShearY,0.9,8)	(Invert,0.4,5)
Sub-policy 7	(ShearY,0.9,5)	(Solarize,0.2,6)
Sub-policy 8	(Invert,0.9,6)	(AutoContrast,0.8,1)
Sub-policy 9	(Equalize,0.6,3)	(Rotate,0.9,3)
Sub-policy 10	(ShearX,0.9,4)	(Solarize,0.3,3)
Sub-policy 11	(ShearY,0.8,8)	(Invert,0.7,4)
Sub-policy 12	(Equalize,0.9,5)	(TranslateY,0.6,6)
Sub-policy 13	(Invert,0.9,4)	(Equalize,0.6,7)
Sub-policy 14	(Contrast,0.3,3)	(Rotate,0.8,4)
Sub-policy 15	(Invert,0.8,5)	(TranslateY,0.0,2)
Sub-policy 16	(ShearY,0.7,6)	(Solarize,0.4,8)
Sub-policy 17	(Invert,0.6,4)	(Rotate,0.8,4)
Sub-policy 18	(ShearY,0.3,7)	(TranslateX,0.9,3)
Sub-policy 19	(ShearX,0.1,6)	(Invert,0.6,5)
Sub-policy 20	(Solarize,0.7,2)	(TranslateY,0.6,7)
Sub-policy 21	(ShearY,0.8,4)	(Invert,0.8,8)
Sub-policy 22	(ShearX,0.7,9)	(TranslateY,0.8,3)
Sub-policy 23	(ShearY,0.8,5)	(AutoContrast,0.7,3)
Sub-policy 24	(ShearX,0.7,2)	(Invert,0.1,5)

Table 9: AutoAugment policy found on reduced SVHN.

实际上，这是本文最有实际价值的部分。实用主义地讲，文章的讲解已经结束了。

但是我们总会关心这个结果是怎么得到的，答案就是RL暴力搜索，并不很令人激动。

实际上，作者考虑如下的16种图像增强算法作为操作全空间,并且考虑10种不同的执行强度和11个不同的执行概率。



Operation Name	Description	Range of magnitudes
ShearX(Y)	Shear the image along the horizontal (vertical) axis with rate *magnitude*.	[-0.3,0.3]
TranslateX(Y)	Translate the image in the horizontal (vertical) direction by *magnitude* number of pixels.	[-150,150]
Rotate	Rotate the image *magnitude* degrees.	[-30,30]
AutoContrast	Maximize the the image contrast, by making the darkest pixel black and lightest pixel white.	
Invert	Invert the pixels of the image.	
Equalize	Equalize the image histogram.	
Solarize	Invert all pixels above a threshold value of *magnitude*.	[0,256]
Posterize	Reduce the number of bits for each pixel to *magnitude* bits.	[4,8]
Contrast	Control the contrast of the image. A *magnitude*=0 gives a gray image, whereas *magnitude*=1 gives the original image.	[0.1,1.9]
Color	Adjust the color balance of the image, in a manner similar to the controls on a colour TV set. A *magnitude*=0 gives a black & white image, whereas *magnitude*=1 gives the original image.	[0.1,1.9]
Brightness	Adjust the brightness of the image. A *magnitude*=0 gives a black image, whereas *magnitude*=1 gives the original image.	[0.1,1.9]
Sharpness	Adjust the sharpness of the image. A *magnitude*=0 gives a blurred image, whereas *magnitude*=1 gives the original image.	[0.1,1.9]
Cutout [25, 73]	Set a random square patch of side-length *magnitude* pixels to gray.	[0,60]
Sample Pairing [51, 74]	Linearly add the image with another image (selected at random from the same mini-batch) with weight *magnitude*, without changing the label.	[0, 0.4]

Table 7: List of all image transformations that the controller could choose from during the search. Additionally, the values of magnitude that can be predicted by the controller during the search for each operation at shown in the third column (for image size 331x331). Some transformations do not use the magnitude information (e.g. Invert and Equalize).

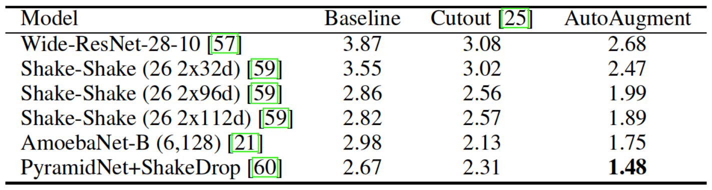
假如每次都从所有策略中找出top10 best构成一个集合来随机选择，我们的问题就具体到了如何在$(16*10*11)^{10}=2.9*10^{32}$的集合中搜索出最好的10个。

然后很自然的，从大的数据集中采样一个子集、给一个小的toy model就能利用AA进行训练，得到reward，最终给出搜索的10个结果。

实际上我们能从RL得到一系列的结果，每个结果是5个sub-policies构成的sub-policies set（包含10个操作）。为了提供一定的鲁棒性，作者最终把reward最高的5个set concatenate起来，就得到了上面的AA操作空间。

Results

1.在CIFAR上的效果

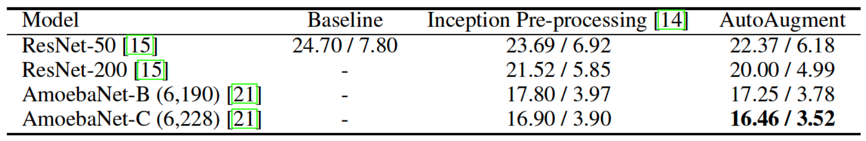


Model	Baseline	Cutout [25]	AutoAugment
Wide-ResNet-28-10 [57]	3.87	3.08	2.68
Shake-Shake (26 2x32d) [59]	3.55	3.02	2.47
Shake-Shake (26 2x96d) [59]	2.86	2.56	1.99
Shake-Shake (26 2x112d) [59]	2.82	2.57	1.89
AmoebaNet-B (6,128) [21]	2.98	2.13	1.75
PyramidNet+ShakeDrop [60]	2.67	2.31	**1.48**

Table 1: Test set error rates (%) on CIFAR-10. Lower is better. All the results of the baseline models, and baseline models with Cutout are replicated in our experiments and match the previously reported results [25, 57, 59, 60]. One exception is Shake-Shake (26 2x112d), which has more filters than the biggest model in [59] – 112 vs 96, and the results were not previously reported. Note that the best policy is found on reduced CIFAR-10. See text for more details.

基于baseline和cutout已经很低的error，添加AA后能够进一步压低之，还是非常舒服的。

2.在ImageNet上的效果

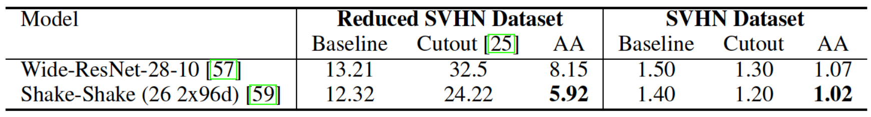


Model	Baseline	Inception Pre-processing [14]	AutoAugment
ResNet-50 [15]	24.70 / 7.80	23.69 / 6.92	22.37 / 6.18
ResNet-200 [15]	-	21.52 / 5.85	20.00 / 4.99
AmoebaNet-B (6,190) [21]	-	17.80 / 3.97	17.25 / 3.78
AmoebaNet-C (6,228) [21]	-	16.90 / 3.90	**16.46 / 3.52**

Table 5: Validation set Top-1 / Top-5 error rates (%) on ImageNet. Lower is better. ResNet-50 with baseline augmentation result is taken from [15]. AmoebaNet-B,C results with Inception-style preprocessing are replicated in our experiments and match the previously reported result by [21]. There exists a better result of 14.6% Top-1 error rate [68] but their method makes use of a large amount of weakly labeled extra data.

可以看到，纯粹通过数据增强提升了Resnet50上的一个点，可以说很开心的。

3.在SVHN上的效果



Model	Reduced SVHN Dataset			SVHN Dataset		
	Baseline	Cutout [25]	AA	Baseline	Cutout	AA
Wide-ResNet-28-10 [57]	13.21	32.5	8.15	1.50	1.30	1.07
Shake-Shake (26 2x96d) [59]	12.32	24.22	**5.92**	1.40	1.20	**1.02**

Table 4: Test set error rates (%). Lower error rates are better. AA refers to AutoAugment. **Reduced SVHN Dataset**: Results on the same training set that we use to find the best policy. Wide-ResNet baseline results on reduced SVHN are replicated in our experiments and roughly match the previously reported results (ours is higher by 0.38%, probably due to hyperparameter tuning) [66]. **SVHN Dataset**: Wide-ResNet results with baseline augmentation [57] and Cutout [25] are replicated in our experiments and match the previously reported results. Shake-Shake results on SVHN have not been reported before. Note that the AutoAugment policy is found on reduced SVHN.

Insights

提出了RL结合数据增强的算法，并给出了ImageNet和CIFAR上的数据增强搜索结果。

如果能够复现出文章中的结果，非常make sense。