AutoAugment: Learning Augmentation Strategies from Data

Ekin D. Cubuk *, Barret Zoph*, Dandelion Mané, Vijay Vasudevan, Quoc V. Le
Google Brain

【强化学习 108】AutoAugment

张楚珩
清华大学 交叉信息院博士在读

这里介绍 AutoML 领域另外一个研究方向，数据增广。根据我们组周璟师妹的组会报告整理而成。

原文传送门

AutoAugment（CVPR 2019） Cubuk, Ekin D., et al. "Autoaugment: Learning augmentation strategies from data." Proceedings of the IEEE conference on computer vision and pattern recognition. 2019.

PBA（ICML 2019） Ho, Daniel, et al. "Population based augmentation: Efficient learning of augmentation policy schedules." arXiv preprint arXiv:1905.05393 (2019).

Fast AutoAugment（NIPS 2019）Lim, Sungbin, et al. "Fast autoaugment." Advances in Neural Information Processing Systems. 2019.

Adversarial AutoAugment（ICLR 2020）Zhang, Xinyu, et al. "Adversarial AutoAugment." arXiv preprint arXiv:1912.11188 (2019).

RandAugment（Arxiv 2019.10） Cubuk, Ekin D., et al. "Randaugment: Practical automated data augmentation with a reduced search space." arXiv preprint arXiv:1909.13719 (2019).

特色

这也是强化学习在 AutoML 中的一个重要应用。ps. 边开组会边做的笔记，可能有点乱。

过程

1. 数据增广

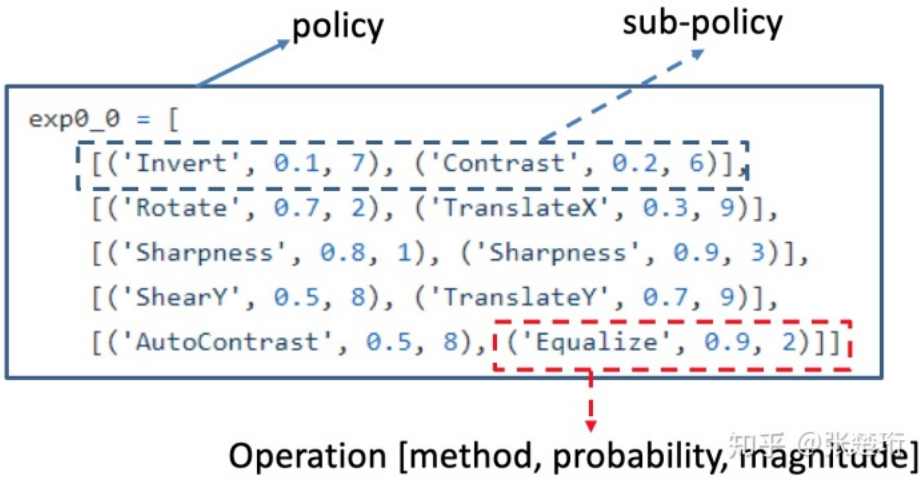
为了使得算法结果更稳定，可以考虑对图片数据做一些变换，用变换之后的图片来做训练。比如一些常见的变换和相应的变化范围如下图所示。

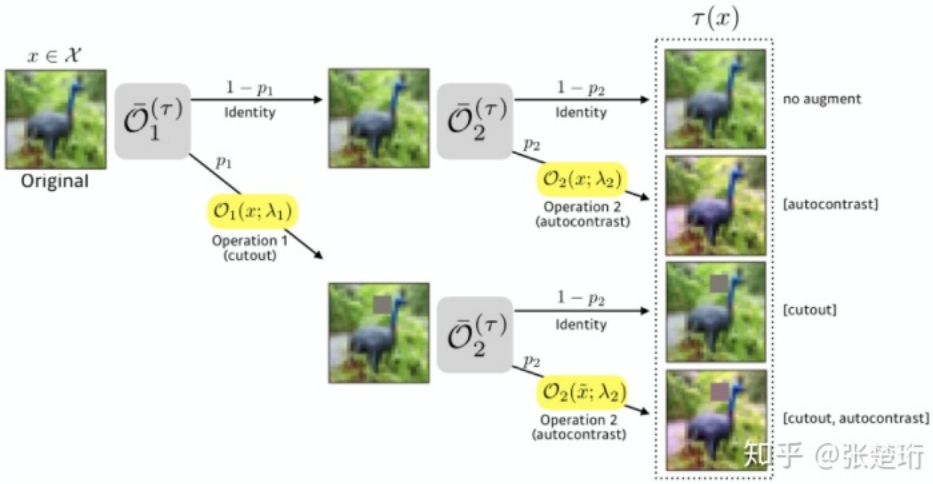
Operation Name	Description	Range of magnitudes
ShearX(Y)	Shear the image along the horizontal (vertical) axis with rate *magnitude*.	[-0.3,0.3]
TranslateX(Y)	Translate the image in the horizontal (vertical) direction by *magnitude* number of pixels.	[-150,150]
Rotate	Rotate the image *magnitude* degrees.	[-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 [12, 69]	Set a random square patch of side-length *magnitude* pixels to gray.	[0,60]
Sample Pairing [24, 68]	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]

2. AutoAugment

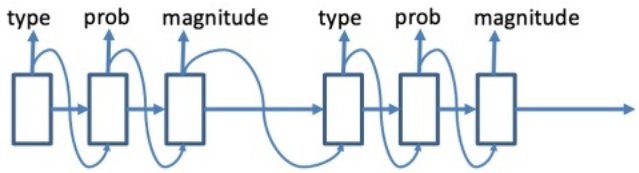
这种方法和我们前面介绍的 NAS 用到的 RL 方法差不多，也是用一个 RNN 来顺序输出一个数据增广策略，把每一步的输出看做一个 action，最后每个轨迹有一个总的 sparse reward，然后使用强化学习方法来训练。

RNN 输出一个 policy 包含五个 sub-policy，每个 sub-policy 包含两个变换，每个变换由三个元素组成，分别是“变换方式”、“变换概率“（可以选择进行该变换或者不进行该变换）和”变换幅度“。对于一幅图片来说，先从各个 sub-policy 中等概率选择一个，然后再顺序进行该 sub-policy 中的两个操作。


policy
sub-policy
exp0_0 = [
[('Invert', 0.1, 7), ('Contrast', 0.2, 6)],
[('Rotate', 0.7, 2), ('TranslateX', 0.3, 9)],
[('Sharpness', 0.8, 1), ('Sharpness', 0.9, 3)],
[('ShearY', 0.5, 8), ('TranslateY', 0.7, 9)],
[('AutoContrast', 0.5, 8), ('Equalize', 0.9, 2)]]
Operation [method, probability, magnitude]


$\tau(x)$
$x \in \mathcal{X}$
Original
$\bar{\mathcal{O}}_1^{(\tau)}$
$1-p_1$
Identity
p_1
$\mathcal{O}_1(x;\lambda_1)$
Operation 1
(cutout)
$\bar{\mathcal{O}}_2^{(\tau)}$
$1-p_2$
Identity
p_2
$\mathcal{O}_2(x;\lambda_2)$
Operation 2
(autocontrast)
$\mathcal{O}_2(\tilde{x};\lambda_2)$
no augment
[autocontrast]
[cutout]
[cutout, autocontrast]

- Controller: RNN
  - 30 softmax predictions:
  - 5 sub-policies*2 operations*(operation type, magnitude, probability)



- Reward: child model's validation accuracy
- Proximal Policy Optimization(PPO)
- Each dataset, sample about 15000 policies

该方法的主要优点是效果好，主要缺点是训练费时间，因为对于每个数据集需要需要采样非常多的数据增广策略。因此这篇文章只能在小数据集上做：reduced CIFAR-10。同时一个增广策略也不能串特别多的增广变换操作；文章搜索的时候只能选择了五个 sub-policy 串在一块；找到最优的策略之后，文章选了五组比较好的使用（相当于 25 个 sub-policy）。

实验结果

Dataset	Model	Baseline	Cutout [12]	AutoAugment	
CIFAR-10	Wide-ResNet-28-10 [67]	3.9	3.1	2.6±0.1	
	Shake-Shake (26 2x32d) [17]	3.6	3.0	2.5±0.1	
	Shake-Shake (26 2x96d) [17]	2.9	2.6	2.0±0.1	**5000 GPU hours**
	Shake-Shake (26 2x112d) [17]	2.8	2.6	1.9±0.1	
	AmoebaNet-B (6,128) [48]	3.0	2.1	1.8±0.1	
	PyramidNet+ShakeDrop [65]	2.7	2.3	**1.5 ± 0.1**	
Reduced CIFAR-10	Wide-ResNet-28-10 [67]	18.8	16.5	14.1±0.3	4,000 randomly
	Shake-Shake (26 2x96d) [17]	17.1	13.4	**10.0 ± 0.2**	chosen examples
CIFAR-100	Wide-ResNet-28-10 [67]	18.8	18.4	17.1±0.3	
	Shake-Shake (26 2x96d) [17]	17.1	16.0	14.3±0.2	
	PyramidNet+ShakeDrop [65]	14.0	12.2	**10.7 ± 0.2**	
SVHN	Wide-ResNet-28-10 [67]	1.5	1.3	1.1	**1000 GPU hours**
	Shake-Shake (26 2x96d) [17]	1.4	1.2	**1.0**	
Reduced SVHN	Wide-ResNet-28-10 [67]	13.2	32.5	8.2	1,000 randomly
	Shake-Shake (26 2x96d) [17]	12.3	24.2	**5.9**	chosen examples

ImageNet

Model	Inception Pre-processing [59]	AutoAugment ours
ResNet-50	76.3 / 93.1	77.6 / 93.8
ResNet-200	78.5 / 94.2	80.0 / 95.0
AmoebaNet-B (6,190)	82.2 / 96.0	82.8 / 96.2
AmoebaNet-C (6,228)	83.1 / 96.1	**83.5 / 96.5**

Table 3. Validation set Top-1 / Top-5 accuracy (%) on ImageNet. Higher is better. ResNet-50 with baseline augmentation result is taken from [20]. AmoebaNet-B,C results with Inception-style pre-processing are replicated in our experiments and match the previously reported result by [48]. There exists a better result of 85.4% Top-1 error rate [37] but their method makes use of a large amount of weakly labeled extra data. Ref. [68] reports an improvement of 1.5% for a ResNet-50 model.

NVIDIA Tesla P100
15000 GPU hours

同时，文章还做了实验说明一个数据集上学习到的增广方式可以迁移到其他数据集上

Apply the same policy that is learned on ImageNet on five FGVC datasets with image size similar to ImageNet

Dataset	Train Size	Classes	Baseline	AutoAugment-transfer
Oxford 102 Flowers [43]	2,040	102	6.7	**4.6**
Caltech-101 [15]	3,060	102	19.4	**13.1**
Oxford-IIIT Pets [14]	3,680	37	13.5	**11.0**
FGVC Aircraft [38]	6,667	100	9.1	**7.3**
Stanford Cars [27]	8,144	196	6.4	**5.2**

Table 4. Test set Top-1 error rates (%) on FGVC datasets for Inception v4 models trained from scratch with and without AutoAugment-transfer. Lower rates are better. AutoAugment-transfer results use the policy found on ImageNet. Baseline models used Inception pre-processing.

3. PBA

全称是 population based augmentation。这篇文章针对之前每次为了确定一个增广方式好不好需要完整地把网络训练一边，这样非常耗时。本文则一边训练一边观察各种不同增广方式的效果。本文主要基于population based training（PBT）（ps. DeepMind StarCraft 文章中也用到了该方法）。该方法同时训练好几份网络，并且在训练的过程中改变数据增广的方式，来观察效果好不好，实时替换掉效果不太好的，并且在效果较好的个体上做扰动。

Step: In each iteration we run an epoch of gradient descent.

Eval: We evaluate a trial on a validation set not used for PBT training and disjoint from the final test set.

Ready: A trial is ready to go through the exploit-and-explore process once 3 steps/epochs have elapsed.

Exploit: We use Truncation Selection (Jaderberg et al., 2017), where a trial in the bottom 25% of the population clones the weights and hyperparameters of a model in the top 25%.

Explore: See Algorithm 2 for the exploration function. For each hyperparameter, we either uniformly resample from all possible values or perturb the original value.

注意，这篇文章的方法不是一个强化学习方法，最后的结果就是直接选取种群里面最好的增广策略。

PBT 中每次会在当前的增广策略附近做探索，文章中用到了如下一个探索方案：

Algorithm 2 The PBA explore function. Probability parameters have possible values from 0% to 100% in increments of 10%, and magnitdue parameters have values from 0 to 9 inclusive.

Input: Params p, list of augmentation hyperparameters
for $param$ in p **do**
  if random(0, 1) < 0.2 **then**
    Resample $param$ uniformly from domain
  else
    amt = [0,1,2,3] uniformly at random
    if random(0, 1) < 0.5 **then**
      $param = param + amt$
    else
      $param = param - amt$
    end if
    Clip $param$ to stay in domain
  end if
end for

实验结果

可以看到所需要的训练时间大大缩减了，但是一些数据集上的效果会比 AutoAugment 差一些。

Dataset	Model	Baseline	Cutout	AA	AA*	PBA	
CIFAR-10	Wide-ResNet-28-10	3.87	3.08	2.68		2.58 ± 0.062	
	Shake-Shake (26 2x32d)	3.55	3.02	2.47		2.54 ± 0.10	**5 GPU hours**
	Shake-Shake (26 2x96d)	2.86	2.56	1.99		2.03 ± 0.11	
	Shake-Shake (26 2x112d)	2.82	2.57	1.89		2.03 ± 0.080	
	PyramidNet+ShakeDrop	2.67	2.31	1.48		$\mathbf{1.46 \pm 0.077}$	
Reduced CIFAR-10	Wide-ResNet-28-10	18.84	17.05	14.13		12.82 ± 0.26	
	Shake-Shake (26 2x96d)	17.05	13.40	**10.04**		10.64 ± 0.22	
CIFAR-100	Wide-ResNet-28-10	18.8	18.41	17.09		16.73 ± 0.15	
	Shake-Shake (26 2x96d)	17.05	16.00	14.28		15.31 ± 0.28	
	PyramidNet+ShakeDrop	13.99	12.19	**10.67**		10.94 ± 0.094	
SVHN	Wide-ResNet-28-10	1.50	1.40	1.07	1.13 ± 0.024	1.18 ± 0.022	**1 GPU hours**
	Shake-Shake (26 2x96d)	1.40	1.20	**1.02**	1.10 ± 0.032	1.13 ± 0.029	
Reduced SVHN	Wide-ResNet-28-10	13.21	32.5	8.15		7.83 ± 0.22	
	Shake-Shake (26 2x96d)	13.32	24.22	**5.92**		$6.\overline{55} \pm 0.13$	

4. Fast AutoAugment

前面的方法慢是因为要训练太多的子模型，这里希望就训练一次就找到好的增广策略。最关键的想法是：把增广后的数据当做还没遇到的 missing data。具体做法是：把训练集分成两部分，分成训练模型的数据 DM 和训练增广数据的数据 DA。在 DM 上训练模型，并且希望在 DA 上找一个策

略，使得在 DA 上增广的数据在 DM 模型上表现最好（准确率最高）。这个想法就是希望增广之后的数据能和原数据集最"匹配"。个人感觉这个想法挺有意思的，就是先拿掉一部分数据 DM，希望从 DA 增广之后的数据能够恢复 DM，而 DA 的增广和 DM 之间的匹配关系就利用 DM 训练出来的模型在 DA 上的准确率来表示。（ps. 感觉其思路和 RND 也有点像）（ps. 老板 remark 准确率高是数据匹配的一个必要条件，不是充分条件，不过能有一定的指示意义）。

- split D_{train} into D_M and D_A
  - Train model based on D_M, get parameter θ^*
  - Find augmentation policy T on D_A
    - $T_* = argmax_T\ R(\theta^*|T(D_A))$, where R is expected accuracy

    $argmin_T\ L(\theta^*|T(D_A))$, where L is expected loss

具体的 argmax 的优化过程使用 Bayesian optimization 的方法 TPE。

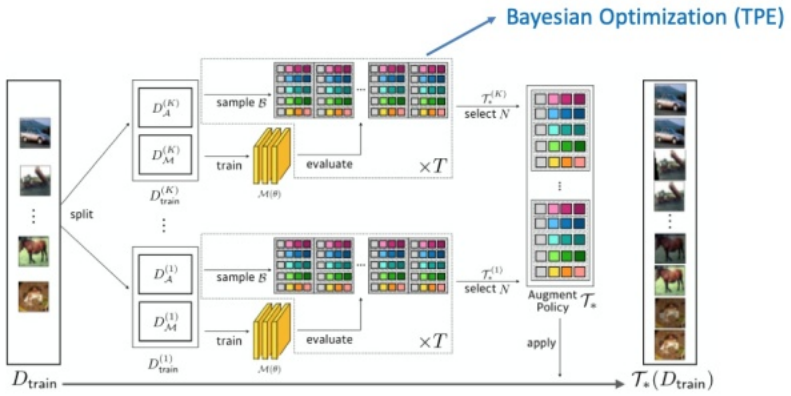


Figure 2: An overall procedure of augmentation search by Fast AutoAugment algorithm. For exploration, the proposed method splits the train dataset D_{train} into K-folds, which consists of two datasets $D_{\mathcal{M}}^{(k)}$ and $D_{\mathcal{A}}^{(k)}$. Then model parameter θ is trained in parallel on each $D_{\mathcal{M}}^{(k)}$. After training θ, the algorithm evaluates B bundles of augmentation policies on $D_{\mathcal{A}}$ without training θ. The top-N policies obtained from each K-fold are appended to an augmentation list $\mathcal{T}_*$.

Algorithm 1: Fast AutoAugment

Input : $(\theta, D_{\text{train}}, K, T, B, N)$

Split D_{train} into K-fold data $D_{\text{train}}^{(k)} = \{(D_{\mathcal{M}}^{(k)}, D_{\mathcal{A}}^{(k)})\}$ // stratified shuffling

for $k \in \{1, \ldots, K\}$ **do**

  $\mathcal{T}_*^{(k)} \leftarrow \emptyset$, $(D_{\mathcal{M}}, D_{\mathcal{A}}) \leftarrow (D_{\mathcal{M}}^{(k)}, D_{\mathcal{A}}^{(k)})$ // initialize

  Train θ on $D_{\mathcal{M}}$

  for $t \in \{0, \ldots, T-1\}$ **do**

    $\mathcal{B} \leftarrow \text{BayesOptim}(\mathcal{T}, \mathcal{L}(\theta|\mathcal{T}(D_{\mathcal{A}})), B)$ // explore-and-exploit

    $\mathcal{T}_t \leftarrow$ Select top-N policies in $\mathcal{B}$

    $\mathcal{T}_*^{(k)} \leftarrow \mathcal{T}_*^{(k)} \cup \mathcal{T}_t$ // merge augmentation policies

return $\mathcal{T}_* = \bigcup_k \mathcal{T}_*^{(k)}$

实验结果

运行效率上看和 PBA 差不多，效果还不错，CIFAR-10 上没有 AutoAugment 好，但是在 ImageNet 上还不错。

Model	Baseline	Cutout [5]	AA [3]	PBA [13]	Fast AA (transfer / direct)
Wide-ResNet-40-2	5.3	4.1	3.7	–	**3.6** / 3.7
Wide-ResNet-28-10	3.9	3.1	**2.6**	**2.6**	2.7 / 2.7
Shake-Shake(26 2×32d)	3.6	3.0	**2.5**	**2.5**	2.7 / **2.5**
Shake-Shake(26 2×96d)	2.9	2.6	**2.0**	**2.0**	**2.0** / **2.0**
Shake-Shake(26 2×112d)	2.8	2.6	**1.9**	2.0	2.0 / **1.9**
PyramidNet+ShakeDrop	2.7	2.3	**1.5**	**1.5**	1.8 / 1.7

3.5 GPU hours

Table 2: Test set error rate (%) on CIFAR-10.

Model	Baseline	Cutout [5]	AA [3]	PBA [13]	Fast AA (transfer / direct)
Wide-ResNet-40-2	26.0	25.2	20.7	–	20.7 / **20.6**
Wide-ResNet-28-10	18.8	18.4	17.1	**16.7**	17.3 / 17.3
Shake-Shake(26 2×96d)	17.1	16.0	**14.3**	15.3	14.9 / 14.6
PyramidNet+ShakeDrop	14.0	12.2	**10.7**	10.9	11.9 / 11.7

Table 3: Test set error rate (%) on CIFAR-100.

Model	Baseline	Cutout [5]	AA [3]	PBA [13]	Fast AA
Wide-ResNet-28-10	1.5	1.3	**1.1**	1.2	**1.1**

1.5 GPU hours

Table 4: Test set error rate (%) on SVHN.

Model	Baseline	AutoAugment [3]	Fast AutoAugment
ResNet-50	23.7 / 6.9	**22.4 / 6.2**	**22.4** / 6.3
ResNet-200	21.5 / 5.8	20.00 / 5.0	**19.4 / 4.7**

450 GPU hours

Table 5: Validation set Top-1 / Top-5 error rate (%) on ImageNet.

5. Adversarial AutoAugment

本方法是基于 RL 的。它训练的更快了，之前的都是在小数据集上增广，这里直接对于原数据集做增广。和 PBA 的相似之处在于在训练的过程中，也可能训练过程中改变增广策略。

这篇文章的特色是有两个策略网络：网络 A 希望最大化 loss function，网络 F 希望最小化 loss function。网络 A 的目标是在一定的变换限定范围内生成 adversary example，而 F 的目标就是把变换之后的样本尽可能变回来。最后输出的 data augmentation policy 就是网络 A 和网络 F 的复合。（ps. 有点像 GAN 的思路）

- Augmentation policy network $A(\cdot, \theta)$
  - $\tau(\cdot)$ represents the augmentation policy generated by $A(\cdot, \theta)$

$$\theta^* = \arg\max_{\theta} J(\theta),$$
$$\text{where } J(\theta) = \mathop{\mathbb{E}}_{x \sim \Omega} \mathop{\mathbb{E}}_{\tau \sim \mathcal{A}(\cdot, \theta)} \mathcal{L}[\mathcal{F}(\tau(x), w), y].$$

- Target network $F(\cdot, w)$

$$w^* = \arg\min_{w} \mathop{\mathbb{E}}_{x \sim \Omega} \mathop{\mathbb{E}}_{\tau \sim \mathcal{A}(\cdot, \theta)} \mathcal{L}[\mathcal{F}(\tau(x), w), y],$$

这两个网络和 AutoAugment 类似，也是用 RNN 代表的控制器；不过这里去掉了 probability 的部分，这样避免了采样的过程，运行会更快。强化学习的优化方法使用 REINFORCE。

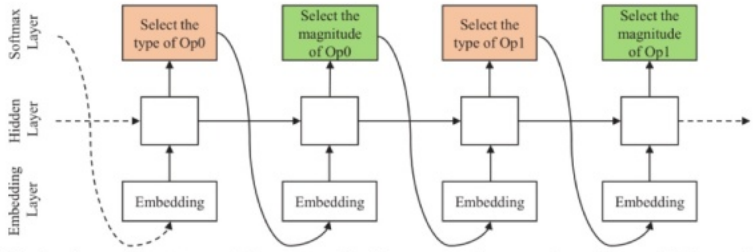


Figure 3: The basic architecture of the controller for generating a sub-policy, which consists of two operations with corresponding parameters, the type and magnitude of each operation. When a policy contains Q sub-policies, the basic architecture will be repeated Q times. Following the setting of AutoAugment (Cubuk et al., 2019), the number of sub-policies Q is set to 5 in this paper.

Algorithm 1 Joint Training of Target Network and Augmentation Policy Network

Initialization: target network $\mathcal{F}(\cdot, \boldsymbol{w})$, augmentation policy network $\mathcal{A}(\cdot, \boldsymbol{\theta})$
Input: input examples $\boldsymbol{x}$, corresponding labels $\boldsymbol{y}$

for $1 \leq e \leq epochs$ **do**
  Initialize $\widehat{\mathcal{L}}_m = 0, \forall m \in \{1, 2, \cdots, M\}$;
  Generate M policies with the probabilities $\{p_1, p_2, \cdots, p_M\}$;
  for $1 \leq t \leq T$ **do**
    Augment each batch data with M generated policies, respectively;
    Update $\boldsymbol{w}_{e,t+1}$ according to Equation 4;
    Update $\widehat{\mathcal{L}}_m$ through moving average, $\forall m \in \{1, 2, \cdots, M\}$;
  Collect $\{\widehat{\mathcal{L}}_1, \widehat{\mathcal{L}}_2, \cdots, \widehat{\mathcal{L}}_M\}$;
  Normalize $\widehat{\mathcal{L}}_m$ among M instances as $\widetilde{\mathcal{L}}_m, \forall m \in \{1, 2, \cdots, M\}$;
  Update $\boldsymbol{\theta}_{e+1}$ via Equation 9;
Output $\boldsymbol{w}^*, \boldsymbol{\theta}^*$

实验结果

在各个数据集上都得到了比较好的效果，但是训练时间会比前面两种更长一些。数据较好的原因可能是这个工作是直接在全量数据上做的。

Table 1: Top-1 test error (%) on CIFAR-10. We replicate the results of Baseline, Cutout and AutoAugment methods from Cubuk et al. (2019), and the results of PBA from Ho et al. (2019) in all of our experiments.

Model	Baseline	Cutout	AutoAugment	PBA	Our Method
Wide-ResNet-28-10	3.87	3.08	2.68	2.58	**1.90±0.15**
Shake-Shake (26 2x32d)	3.55	3.02	2.47	2.54	**2.36±0.10**
Shake-Shake (26 2x96d)	2.86	2.56	1.99	2.03	**1.85±0.12**
Shake-Shake (26 2x112d)	2.82	2.57	1.89	2.03	**1.78±0.05**
PyramidNet+ShakeDrop	2.67	2.31	1.48	1.46	**1.56±0.06**

Table 2: Top-1 test error (%) on CIFAR-100.

Model	Baseline	Cutout	AutoAugment	PBA	Our Method
Wide-ResNet-28-10	18.80	18.41	17.09	16.73	**15.49±0.18**
Shake-Shake (26 2x96d)	17.05	16.00	14.28	15.31	**14.10±0.15**
PyramidNet+ShakeDrop	13.99	12.19	10.67	10.94	**10.42±0.20**

ImageNet **1280 GPU hours**

Table 3: Top-1 / Top-5 test error (%) on ImageNet. Note that the result of ResNet-50-D is achieved only through substituting the architecture.

Model	Baseline	AutoAugment	PBA	Our Method
ResNet-50	23.69 / 6.92	22.37 / 6.18	-	**20.60±0.15 / 5.53±0.05**
ResNet-50-D	22.84 / 6.48	-	-	**20.00±0.12 / 5.25±0.03**
ResNet-200	21.52 / 5.85	20.00 / 4.90	-	**18.68±0.18 / 4.77±0.05**

6. RandAugment

本文的作者和第一个 AutoAugment 是同一拨人。他们观察到，随着网络结构和数据集的大小的不同，其对应的最优策略增广策略也不同。具体地，对于较大的数据集或者较大的网络结构，可以容忍的数据增广幅度也相应更大。

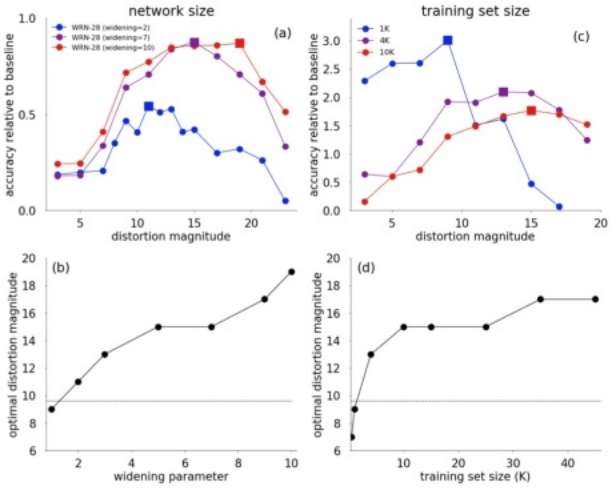


Figure 3. **Optimal magnitude of augmentation depends on the size of the model and the training set.** All results report CIFAR-10 validation accuracy for Wide-ResNet model architectures [53] averaged over 20 random initializations, where $N = 1$. (a) Accuracy of Wide-ResNet-28-2, Wide-ResNet-28-7, and Wide-ResNet-28-10 across varying distortion magnitudes. Models are trained for 200 epochs on 45K training set examples. Squares indicate the distortion magnitude that achieves the maximal accuracy. (b) Optimal distortion magnitude across 7 Wide-ResNet-28 architectures with varying widening parameters (k). (c) Accuracy of Wide-ResNet-28-10 for three training set sizes (1K, 4K, and 10K) across varying distortion magnitudes. Squares indicate the distortion magnitude that achieves the maximal accuracy. (d) Optimal distortion magnitude across 8 training set sizes. Dashed curves show the scaled expectation value of the distortion magnitude in the AutoAugment policy [5].

这篇文章大大减小了搜索空间，这里的搜索空间只包括两个数字：N 是要做几次数据增广；M 是要做多大的增广幅度。并且，等概率地从 K 个不同的增广方式中选择一个。

```
transforms = [
'Identity', 'AutoContrast', 'Equalize',
'Rotate', 'Solarize', 'Color', 'Posterize',
'Contrast', 'Brightness', 'Sharpness',
'ShearX', 'ShearY', 'TranslateX', 'TranslateY']

def randaugment(N, M):
"""Generate a set of distortions.

   Args:
     N: Number of augmentation transformations to
         apply sequentially.
     M: Magnitude for all the transformations.
"""

  sampled_ops = np.random.choice(transforms, N)
  return [(op, M) for op in sampled_ops]
```


实验结果

虽然这个方法很简单，但是实验效果还不错。ps. 所以感觉做了一大圈又回到了原点啊，还是人为设计的这种简单粗暴的方法更有效啊。

	baseline	PBA	Fast AA	AA	RA
CIFAR-10					
Wide-ResNet-28-2	94.9	-	-	**95.9**	95.8
Wide-ResNet-28-10	96.1	**97.4**	97.3	**97.4**	97.3
Shake-Shake	97.1	**98.0**	**98.0**	**98.0**	**98.0**
PyramidNet	97.3	**98.5**	98.3	**98.5**	**98.5**
CIFAR-100					
Wide-ResNet-28-2	75.4	-	-	**78.5**	78.3
Wide-ResNet-28-10	81.2	**83.3**	82.7	82.9	**83.3**
SVHN (core set)					
Wide-ResNet-28-2	96.7	-	-	98.0	**98.3**
Wide-ResNet-28-10	96.9	-	-	98.1	**98.3**
SVHN					
Wide-ResNet-28-2	98.2	-	-	**98.7**	**98.7**
Wide-ResNet-28-10	98.5	98.9	98.8	98.9	**99.0**

ImageNet

	baseline	Fast AA	AA	RA
ResNet-50	76.3 / 93.1	**77.6** / 93.7	**77.6 / 93.8**	**77.6 / 93.8**
EfficientNet-B5	83.2 / 96.7	-	83.3 / 96.7	**83.9 / 96.8**
EfficientNet-B7	84.0 / 96.9	-	84.4 / 97.1	**85.0 / 97.2**

COCO

model	augmentation	mAP	search space
ResNet-101	Baseline	38.8	0
	AutoAugment	**40.4**	10^{34}
	RandAugment	40.1	10^{2}
ResNet-200	Baseline	39.9	0
	AutoAugment	**42.1**	10^{34}
	RandAugment	41.9	10^{2}

编辑于 2020-03-05

强化学习 (Reinforcement Learning) AutoML 深度学习（Deep Learning）

文章被以下专栏收录

强化学习前沿
读呀读paper

进入专栏