论文笔记：Online Hyper-parameter Learning for Auto-Augmentation Strategy

背景及论文简介

数据增强 是对抗模型过拟合的一个较为有效的方法，但增强策略的设计非常依赖专家的经验，而且需要极大的工作量。2018年，自动数据增强的方法被提出（AutoAugment），使用强化学习等方法可搜索出针对模型和数据集的有效策略。然而，即使AutoAugment可以有效对抗过拟合，提升模型精度，但该方法需要极大的计算量，效率较低。对此，作者提出了自己的方法：OHL-Auto-Aug，并总结其三大贡献：

1. 作者将自动数据增强视为超参数优化的问题，将增强策略表述为概率分布，该分布中的参数被视为超参数，

2. 为了使超参数的优化和模型的训练一同进行，达到提高效率的目的，作者引入了一种双层训练框架，

3. 作者提出的方法非常有效，在达到与AutoAugment相近的模型精度的前提下，效率比AutoAugment在CIFAR-10上快60倍，在ImageNet上快24倍。

方法详解

符号及对应公式

增强操作的概率分布：p_θ，该分布的参数为θ

K个候选增强操作：$\{O_k(\cdot)\}_{k=1:K}$

每个增强操作被选中的概率：$p_\theta\left(O_k\right)$

模型$\mathcal{F}(\cdot, w)$及其参数w

训练集：$\mathcal{X}_{tr}=\{(x_i, y_i)\}_{i=1}^{N_{tr}}$

验证集：$\mathcal{X}_{val}=\{(\hat{x}_i, \hat{y}_i)\}_{i=1}^{N_{val}}$

综上，对于作者的增强操作的方法，其目的是：找到一个θ，使得验证集精度最大；并找到一个w^*，使得训练的loss最小。

即θ和w^*要同时满足：

$$\max_\theta \mathcal{J}(\theta)=\operatorname{acc}\left(w^*\right)$$

$$w^*=\arg\min_w \frac{1}{N_{tr}} \sum_{(x,y)\in\mathcal{X}_{tr}} \mathbb{E}_{p_\theta(O)}[\mathcal{L}(\mathcal{F}(O(x), w), y)]$$

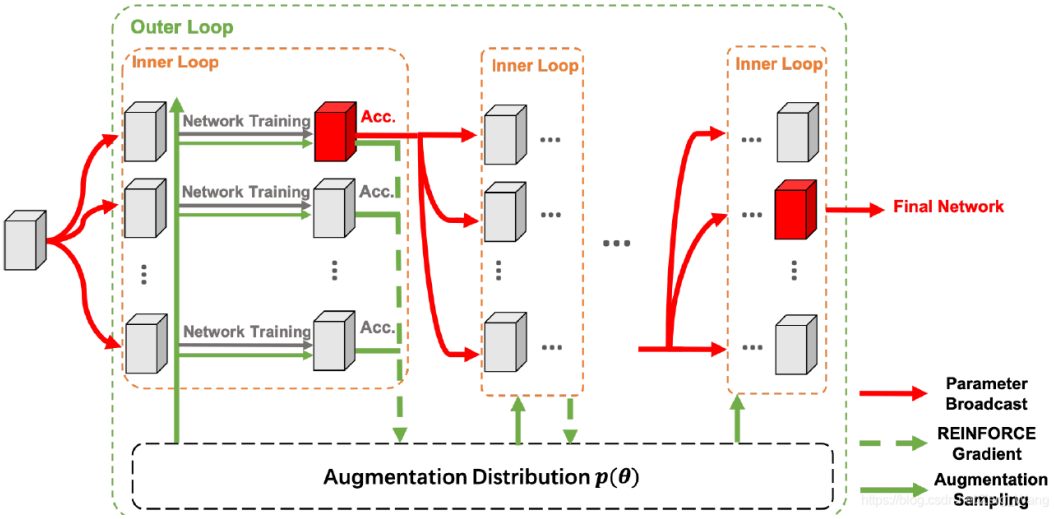
其中，

$$\operatorname{acc}\left(w^*\right)=\frac{1}{N_{val}} \sum_{(\hat{x},\hat{y})\in\mathcal{X}_{val}} \delta\left(\mathcal{F}\left(\hat{x}, w^*\right), \hat{y}\right)$$

值得注意的是，原论文中的这条公式中，等号右边的N_{val}写成了N_{tr}，我感觉这是个错误，所以自作主张在此更正了。

双层训练框架（Bilevel Framework）

以往的自动数据增强方法（AutoAugment）针对所有采样的增强策略，采用子模型（代理模型）对其进行验证，并将精度作为reward喂入RNN Controller，这一过程非常耗时，若再考虑到强化学习的多次迭代，计算量是非常巨大的。对此，作者认为应将模型的训练和增强策略的搜索同时进行，从而避免每次对模型的从头训练，进而提高效率。对此，作者提出了双层的训练框架：



外层的训练：即反复进行内层训练，一共进行T_{max}轮($T = 1, 2, 3...T_{max}$)，每轮都要进行采样以及更新策略的概率分布。

内层的训练：

- 训练开始前，并列的多个模型参数是一致的。
- 训练时，一共迭代I轮（$i = 1, 2, 3...I$，不是I个epoch），batchsize为B，训练使用SGD来更新网络模型的参数（即w）。训练时的每个样本（图像），会被从增强策略的概率分布中采样得到的一个操作增强。也就是说，在一个完整的内层训练中，会采样得到$I \times B$个增强操作。
- 训练完毕后，通过REINFORCE梯度更新策略的概率分布，并将并列训练的多个模型中，精度最高的模型的参数broadcast到下一轮内层训练的所有初始模型中。

网络模型内层迭代时的更新与外层迭代时的传播：

$$w_{T-1}^{(i)} = w_{T-1}^{(i-1)} - \eta_w \nabla_w \left[\mathcal{L}_B \left(\mathcal{T}^{(i)}, w_T^{(i-1)}, x_B, y_B \right) \right]$$
$$w_{T-1}^{(0)} = w_{T-1}, w_T = w_{T-1}^{(I)}$$

其中η_w为模型参数的学习率，采样到的增强操作为$\mathcal{T} = \{O_{k(j)}\}_{j=1:(I \times B)}$

外层训练时，概率分布参数的更新：

$$\begin{aligned} \theta_T &= \theta_{T-1} + \eta_\theta \nabla_\theta \mathcal{J}\left(\theta_{T-1}\right) \\ &= \theta_{T-1} + \eta_\theta \nabla_\theta \mathbb{E}_{w_T}\left[\operatorname{acc}\left(w_T\right)\right] \end{aligned}$$

η_θ为概率分布的学习率。由于外层训练的目标是最大的验证集精度，因此是往梯度上升方向更新，因此公式中为加号。

自此，作者遇到的两个问题：一方面，验证集精度无法微分求导；另一方面，网络模型参数w多且杂，其微积分计算也难以为继。为此，作者使用REINFORCE梯度方法，利用蒙特卡洛采样来得到梯度$\nabla_\theta \mathcal{J}\left(\theta_{T-1}\right)$的近似值。这个采样过程即为内层训练中，并行训练$N$个模型。另外，在内层的训练中，模型一开始都是相同的参数w，但由于随机采样导致的训练过程中，样本应用的增强操作不同，训练结束后每个模型的参数也不同，因此作者认为w仅受到$\mathcal{T}$的影响，因此$p(w_T) = p(\mathcal{T})$（为了保证该等式成立，训练时batchsize应该固定），且有：

$$p(\mathcal{T}) = \prod_{j=1}^{I \times B} p_{\theta_{T-1}}\left(O_{k(j)}\right)$$

至此，外层训练时概率分布参数的更新可以写为：

$$\begin{aligned} \nabla_\theta \mathcal{J}(\theta) = \nabla_\theta \mathbb{E}_{w_T}\left[\operatorname{acc}\left(w_T\right)\right] &\approx \frac{1}{N} \sum_{n=1}^{N} \nabla_\theta \log\left(p\left(w_{T,n}\right)\right) \operatorname{acc}\left(w_{T,n}\right) \\ &= \frac{1}{N} \sum_{n=1}^{N} \nabla_\theta \log\left(p\left(\mathcal{T}_n\right)\right) \operatorname{acc}\left(w_{T,n}\right) \\ &= \frac{1}{N} \sum_{n=1}^{N} \sum_{j=1}^{I \times B} \nabla_\theta \log\left(p_{\theta_{T-1}}\left(O_{k(j),n}\right)\right) \operatorname{acc}\left(w_{T,n}\right) \end{aligned}$$

算法流程和候选增强操作

Algorithm 1 Online Optimization for Auto-Augmentation Strategy

Initialize θ_0, initialize the same w_0 for N models;
while $T \leq T_{max}$ **do**
  for all n such that $1 \leq n \leq N$ **do**
    for all i such that $0 \leq i \leq I$ **do**
      Compute $\{w^i_{T,n}\}$ in Equation 2;
    end for
    return $w_{T,n}$
  end for
  return $\{w_{T,n}\}_{n=1:N}$
  Fix $\{w_{T,n}\}_{n=1:N}$, calulate $\nabla_\theta \mathcal{J}(\theta)$ in Equation 6;
  Update θ_T according to Equation 3;
  Select w_T from $\{w_{T,n}\}_{n=1:N}$ with the best validation accuracy;
  Broadcast w_T to all the N models;
end while
return $w_{T_{max}}, \theta_{T_{max}}$;

Table 1. List of Candidate Augmentation Elements

Elements Name	range of magnitude
$HorizontalShear$	$\{0.1, 0.2, 0.3\}$
$VerticalShear$	$\{0.1, 0.2, 0.3\}$
$HorizontalTranslate$	$\{0.15, 0.3, 0.45\}$
$VerticalTranslate$	$\{0.15, 0.3, 0.45\}$
$Rotate$	$\{10, 20, 30\}$
$ColorAdjust$	$\{0.3, 0.6, 0.9\}$
$Posterize$	$\{4.4, 5.6, 6.8\}$
$Solarize$	$\{26, 102, 179\}$
$Contrast$	$\{1.3, 1.6, 1.9\}$
$Sharpness$	$\{1.3, 1.6, 1.9\}$
$Brightness$	$\{1.3, 1.6, 1.9\}$
$AutoContrast$	None
$Equalize$	None
$Invert$	None

由上图可得，对于单张样本的一次增强（即应用一个增强元素），可以有36种，作者定义单个增强操作由两种增强元素组成，那也就是说可以有36^2种不同的增强操作（单一操作中允许存在两个一样的元素）。这样一来，所谓的增强策略的概率分布参数，就是一个36×36的矩阵。

实验结果

作者在CIFAR-10和ImageNet上验证了其方法：

CIFAR-10

作者的验证集从CIFAR-10的50000张样本的训练集中随机取得，验证集包括5000张样本。

为了与当时的SOTA保持一致，作者的训练过程为：pre-processing，OHL-Auto-Aug，Cutout。而SOTA的过程除了缺少OHL-Auto-Aug，其他一致。训练所用的详细超参数这里就不赘述了。

注意：这里所述的OHL-Auto-Aug，是指通过将该方法找到的增强策略应用到模型中，而不是进行完整搜索过程。

Table 2. Test error rates (%) on CIFAR-10. The number in brackets refers to the results of our implementation. We compare our OHL-Auto-Aug with standard augmentation (Baseline), standard augmentation with Cutout (Cutout), augmentation strategy discovered by [6] (Auto-Augment). Compared to Baseline, our OHL-Auto-Aug achieves about 30% reduction of error rate.

Model	Baseline	Cutout [8]	Auto-Augment [6]	OHL-Auto-Aug	Error Reduce (Baseline/Cutout)
ResNet-18 [12]	4.66	3.62	3.46	**3.29**	**1.37/0.33**
WideResNet-28-10 [33]	3.87	3.08	2.68	**2.61**	**1.26/0.47**
DualPathNet-92 [4]	4.55	3.71	3.16	**2.75**	**1.8/0.96**
AmoebaNet-B(6, 128) [6] (our impl.)	2.98 (3.4)	2.13 (2.9)	**1.75**	1.89	**1.09/0.24 (1.51/1.01)**

ImageNet

Table 3. Top-1 and Top-5 error rates (%) on ImageNet. We compare our OHL-Auto-Aug with standard augmentation (Baseline), standard augmentation with mixup [34] (mixup), augmentation strategy discovered by [6] (Auto-Augment). For both the ResNet-50 and SE-ResNeXt-101, our OHL-Auto-Aug improves the performance significantly.

Method	ResNet-50 [12]	SE-ResNeXt-101 [13]
Baseline	24.70/7.8	20.70/5.01
mixup $\alpha = 0.2$ [34]	23.3/6.6	–
Auto-Augment [6]	22.37/6.18	20.03/5.23 (our impl.)
OHL-Auto-Aug	**21.07/5.68**	**19.30/4.57**

OHL-Auto-Aug计算量分析

作者通过计算AutoAugment方法和OHL-Auto-Aug的迭代次数来探究两种方法在计算量和效率上的差距。两者统一采用1024的batchsize：

对于AutoAugment，15000个子模型，每个模型训练120epoch，reduced CIFAR-10有4000张样本，则总的迭代次数为：15000×4000×120/1024 = 7.03 × 10^6，同理，对于reduced ImageNet的6000张样本，对应的迭代次数为1.76 × 10^7。

对于OHL-Auto-Aug，并行模型8个，每个300epoch，完整CIFAR-10有50000张样本，则迭代次数为300 × 8 × 50000/1024=1.17 × 10^5，同理，在完整ImageNet的128万张样本上，并行模型4个，每个150epoch，其迭代次数为7.5 × 10^5。

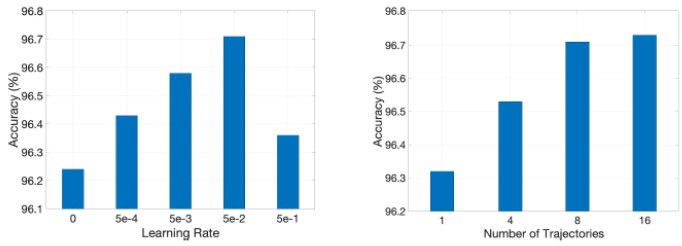
Table 4. Search cost of our method compared with Auto-Augment [6]. For fair comparison, we compute the total training iterations with conversion under a same batch size 1024 and denote as '$\#Iterations$'. Specific computing implementation is detailed in Section 4.2. Our OHL-Auto-Aug achieves 60× faster on CIFAR-10 and 24× faster on ImageNet than Auto-Augment. Our OHL-Auto-Aug also gets rid of the need of retraining from scratch, further saving computation resources.

Dataset	Auto-Augment [6]		OHL-Auto-Aug	
	$\#Iterations$	Usage of Dataset (%)	$\#Iterations$	Usage of Dataset (%)
CIFAR-10	7.03×10^6	8%	1.17×10^5	100%
ImageNet	1.76×10^7	0.5%	7.5×10^5	100%
No Need to Retrain	×		✓	

可见，不仅在效率上OHL-Auto-Aug要显著优于AutoAugment，由于在完整数据集上进行训练，还避免了受到输入数据偏差的影响。另外，由于OHL-Auto-Aug的策略搜索与模型训练一同进行，后续省下了模型再训练的时间，效率可进一步提高。

实验分析

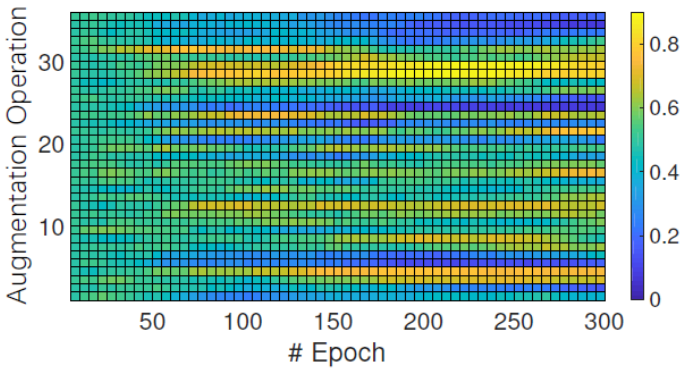
作者探究了不同的策略的概率分布参数的学习率对精度的影响，以及内层训练中并行训练的模型数量对精度的影响：



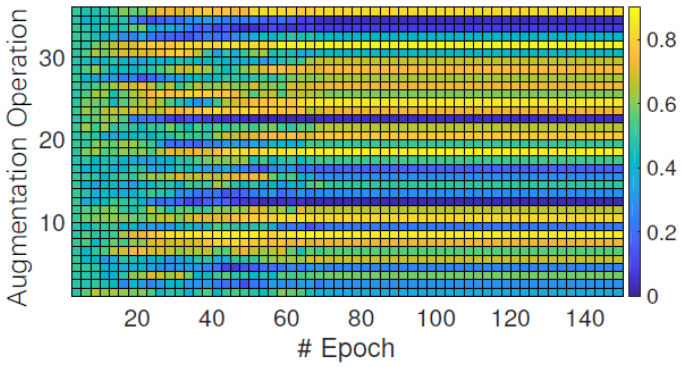
(a) (b)

Figure 3. (a) Comparisons with different augmentation distribution learning rates. All the experiments are conducted with ResNet-18 on CIFAR-10 following the same setting as described in Section 4.1 except η_θ. As can be observed, too large η_θ will make the distribution parameters difficult to converge, while too small η_θ will slow down the convergency process, both harming the final performance. The chosen of $\eta_\theta = 5 \times 10^{-2}$ is a trade-off between convergence speed and performance. (b) Comparisons with different number of trajectory samples. All the experiments use ResNet-18 on CIFAR-10 following the same setting as described in Section 4.1 except N. As can be obseved, increasing the number of trajectories from 1 to 8 steadily improves the performance of the network. When the number comes to ≥ 8, the accuracy improvement is minor. We select $N = 8$ on CIFAR-10 as a trade-off between computation cost and performance.

此外，作者还探究在CIFAR-10和ImageNet数据集上，随着训练进行，增强策略概率分布的变化情况。作者将36×36的参数矩阵按行求和并将数据标准化，得到下图：



(a) Visualization of augmentation distribution parameters of ResNet-18 training on CIFAR-10.



(b) Visualization of augmentation distribution parameters of ResNet-50 training on ImageNet.

可见，随着训练的进行，不同数据集上的概率分布也逐渐不同，说明作者的OHL-Auto-Aug方法是可以根据数据集特点进行有效的增强操作的搜索与选择。

参考资料

https://arxiv.org/pdf/1905.07373.pdf

https://zhuanlan.zhihu.com/p/147577672

https://blog.csdn.net/sinat_34686158/article/details/105196664