


论文笔记：RandAugment

 Machine Learning 专栏收录该内容

0 订阅 13 篇文章

梗概

自动 **数据增强** 方法（AutoAugment、Fast AutoAugment、PBA等）针对网络和数据集，自动搜索出最为适合的数据增强方式。然而在本篇论文中，作者对以上这些方法表示了怀疑。一方面，以上方法使用“代理任务”或“子实验”，即使用子模型在部分数据集上搜索策略的做法，作者认为是不够“全面”的，认为这种搜索方法只能得到次优的结果，且无法根据模型或数据集大小调整正则化强度；另一方面，以上任务计算量过大、运行时间过长，效率较低。对此，作者提出了新的自动增强方法“RandAugment”，应用该方法，作者在图像分类和目标检测任务中都取得了较好的成绩。

	search space	CIFAR-10 PyramidNet	SVHN WRN	ImageNet ResNet	ImageNet E. Net-B7
Baseline	0	97.3	98.5	76.3	84.0
AA	10^{32}	98.5	98.9	77.6	84.4
Fast AA	10^{32}	98.3	98.8	77.6	-
PBA	10^{61}	98.5	98.9	-	-
RA (ours)	10^2	98.5	99.0	77.6	85.0

RandAugment方法

自动数据增强方法的随机性

首先讲讲自动数据增强方法中的“随机性”。以AutoAugment为例，通过搜索得出的最佳增强策略中，有多个（假设有K个）子策略，然后在训练时，对于某一批次的数据，从这K个中随机选择一个进行应用（这是随机性的第一层体现）。每个子策略由两种增强操作构成，每个操作有概率和幅度两个参数，也就是说，即使选中了这个子策略，这一批次的数据有没有被增强，或者经过这两个操作中哪个的增强，都由该子策略中的这两个概率参数控制（这是随机性的第二层体现）。即使应用了某一操作，比如旋转，虽然幅度参数指定了增强的幅度（比如说旋转45度），但具体是逆时针还是顺时针同样也是随机的（这是随机性的第三层体现）。

RandAugment方法

对此，作者舍弃掉这一系列的随机性（按我的理解是，这些反复的随机性，其实是削弱了AutoAugment中概率和幅度参数的作用。因此作者认为舍弃掉概率参数，或者说，将这些参数设为统一的值，从而免去搜索这些值的时间，换来效率的提高是值得的），直接将每种操作的应用概率设置为一样。因此，RandAugment方法为：

1. 设定一个操作集，例如作者的操作集由14种操作构成：Identity、AutoContrast、Equalize、Rotate、Solarize、Color、Posterize、Contrast、Brightness、Sharpness、ShearX、ShearY、TranslateX、TranslateY。
2. RandAugment只有两个参数：N和M。其中N为在每次增强时使用N次操作（使用的这N个操作，都是从操作集中等概率抽取的，例如操作集中有14种操作，则每种操作被选中的概率为1/14，每张图像的N次增强中，选到的操作可能是一样的），M为正整数，表示所有操作在应用时，幅度都为M。
3. 使用网格搜索，或者更为高端的方法（如反向传播等）在完整数据集、完整网络上实验，找到最适合的N和M。这样一来，假如说N的搜索空间为1和2，M为1至10，则搜索空间仅为 10^2 ，远小于之前的自动增强方法。

通过更改N、M的值，便能控制训练时的正则化强度。N、M越大，正则化强度则越高。

```

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]

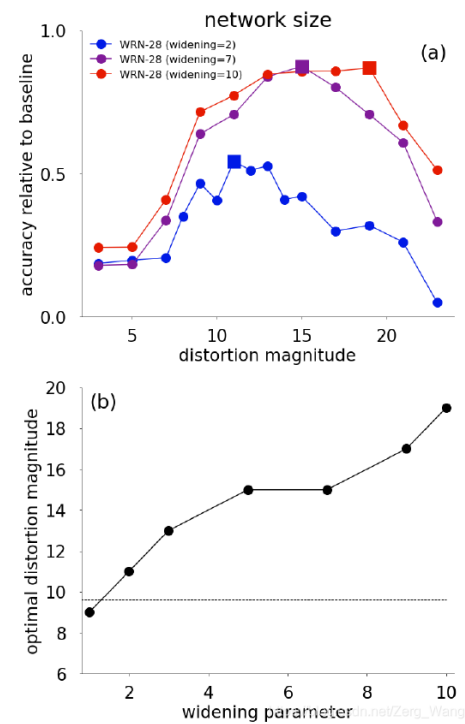
```

Figure 2. Python code for RandAugment based on numpy. [Wang](#)

猜想与实验

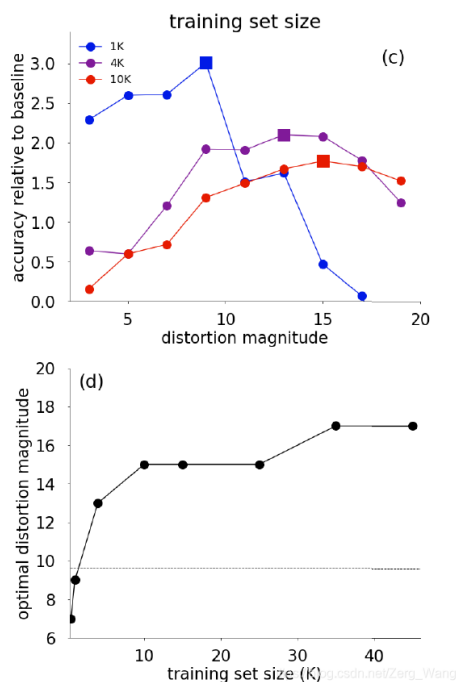
对“代理任务”的质疑

作者在论文开篇就质疑了之前的自动数据增强方法使用“代理任务”，即为了提高效率，在子网络、子数据集上搜索增强策略。作者认为藉由“代理任务”搜索得来的策略不是最适合完整网络和完整数据集的。对此，作者使用RandAugment进行了实验：



在CIFAR10数据集上，设置 $N=1$ ，使用不同宽度的Wide-ResNet网络，得出了：不同的网络经过增强及训练后，达到最佳精度所用的 M 不同。也就是说，如果使用的“代理任务”中，所使用的子网络与最后应用完整增强策略的原网络不同，则通过子网络搜索得到的“最佳策略”，仅仅是对子网络而言的。则也验证了作者通过“代理任务”只能得到次优结果的猜想。

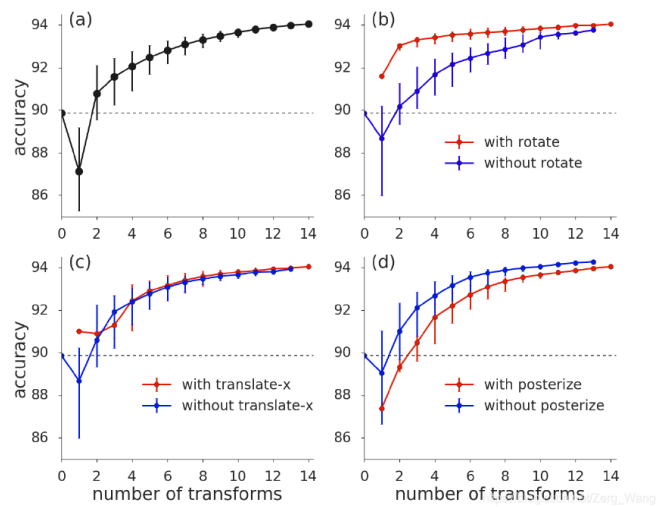
同样的，“代理任务”如果使用子数据集（在AutoAugment和Fast AutoAugment中，是各种Reduced数据集），得到的也很有可能是次优结果。作者的实验如下：



同样的网络，N均设为1，在不同大小的数据集上达到最佳精度时的M并不一致。值得一提的是，实验表明，越大的数据集（CIFAR10）达到最佳精度，所需要的幅度越大，这与以往的经验是相悖的。对于这种现象，其中一种解释是：对于小数据集而言，过于激进的增强操作会带来较低的信噪比，从而导致效果不佳。

对14种增强操作的研究

RandAugment使用默认的14种操作，在不同任务的多个数据集上达到了SOTA精度，一定程度上说明了该方法对于操作的选择是不敏感的。对此，作者进行了更深入的实验，探究每种操作对于整体精度的影响。（以下实验使用Wide-ResNet-28-2以及CIFAR10数据集，N=3，M=4）



左上角图a，作者从14种操作中，分别随机选出1、2、3.....14种操作，来探究操作集数量（不是操作数量）对精度的影响。可以看出，在使用默认的14种操作的情况下，操作集越大，对精度的提升越高。

对于探究单一操作对整体精度的影响，作者的做法是：在随机采样操作而形成的操作集中，通过对比加入某种操作和不加入该操作，来探究单一操作对整体精度的影响：

transformation	Δ (%)	transformation	Δ (%)
rotate	1.3	shear-x	0.9
shear-y	0.9	translate-y	0.4
translate-x	0.4	autoContrast	0.1
sharpness	0.1	identity	0.1
contrast	0.0	color	0.0
brightness	0.0	equalize	-0.0
solarize	-0.1	posterize	-0.3

通过该图可知，在CIFAR10数据集上，最有效的增强操作为rotate，而posterize效果最差（甚至起到了反效果）。

对概率的考量

将所有操作的选择概率设为相同的值，相比于对概率值进行搜索，两种方法在效果上会有怎样的差距？作者做了以下实验：

首先引入 α_{ij} ，表示第*i*次采取增强操作时，使用第*j*种操作的概率。假设采取*N*次增强，操作集中有*K*种操作，则 α_{ij} 共有参数*K***N*个。

由于大多增强操作是可微分的（如Posterize、Equalize以及AutoContrast），因此可采用反向传播方法。首先将 α_{ij} 初始化为相同的值（即每种操作被选中的概率相同），然后通过评判经过 α_{ij} 增强及训练后的网络精度，来更新 α_{ij} 中的参数。

作者称该方法为“1st-order density matching approximation”，并与其他方法进行对比：

	baseline	AA	RA	+ 1 st
Reduced CIFAR-10				
Wide-ResNet-28-2	82.0	85.6	85.3	85.5
Wide-ResNet-28-10	83.5	87.7	86.8	87.4
CIFAR-10				
Wide-ResNet-28-2	94.9	95.9	95.8	96.1
Wide-ResNet-28-10	96.1	97.4	97.3	97.4

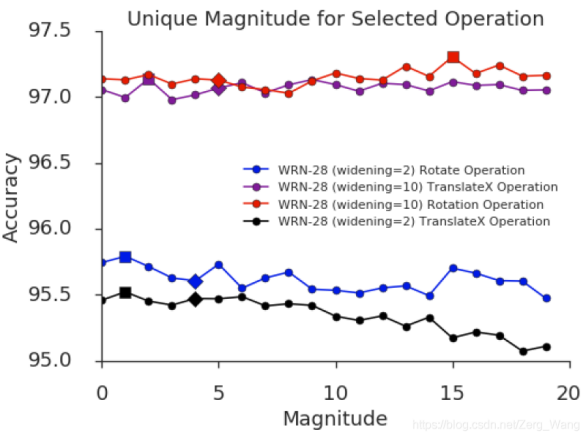
可以看出，提升效果还是很明显的，但作者也说，该方法计算量太大，对于小数据集尚有应用意义，对于大数据集，作者表示“reserve explorations for the future”……

另外，作者在论文附录A.1.提出了Second order term，详细推导这里不再赘述（我看不懂……）

对幅度的考量

将所有操作的幅度设为相同的固定值，效果好不好呢？作者做了以下两个实验：

1. 其他操作幅度一致，但单独改变一个操作的幅度，然后对在该条件训练的网络精度进行测量：



图中折线中，正方形表示幅度取该值的时候精度最大，菱形表示操作的幅度取值与其他操作一致。其中，使用Wide-ResNet-28-2网络的实验，M=4，使用Wide-ResNet-28-10的M=5。从最终实验结果看，在采用全部统一的幅度的条件下得到的精度，和通过实验找到的最佳精度，之间的差距是非常小的。Rotate + Wide-ResNet-28-2下，两者之差为0.19%，Rotate + Wide-ResNet-28-10为0.18%，TranslateX + Wide-ResNet-28-2为0.07%，TranslateX + Wide-ResNet-28-10为0.05%。

2. 训练过程中操作幅度的取值方式，作者设置了四种：在限定范围内随机取值；统一的固定值；线性递增的值；在范围内随机取值，但随着训练进行，范围会越来越大。实验结果如图：（实验条件均为CIFAR10 + Wide-ResNet-28-10）

Magnitude Method	Accuracy
Random Magnitude	97.3
Constant Magnitude	97.2
Linearly Increasing Magnitude	97.2
Random Magnitude with Increasing Upper Bound	97.3

综合以上两个实验，作者认为，采用统一固定的幅度，对精度的影响微乎其微，但却能极大地减少搜索空间，大大提高效率，因此这点精度上的牺牲是值得的。

参考资料

<https://arxiv.org/pdf/1909.13719.pdf>

<https://blog.csdn.net/ch97ckd/article/details/104911938>

开源代码：<https://github.com/tensorflow/tpu/blob/master/models/official/efficientnet/autoaugment.py>