自动化测试作业报告

181250077 李玥

Abstract

本次作业使用了auto machine learning领域内近年来新出现的一些成果来实现自动化生成增广测试数据，主要参考了AutoAugment(CVPR2019)[1], RandAugment(CVPR2020)[2], CutOut(CVPR2018)[3] 。不同于常规的人为设计和选用图像增广方法（如镜像变化，旋转，缩放，剪裁，平移，亮度修改，变换颜色etc），我仿照上述几篇paper中提出的在一系列图像增广子策略的搜索空间中通过搜索算法找到适应特定数据集的图像增广方案。如对于CIFAR-10数据集，搜索算法从15个增广方法中获得了25个子策略组合，每个子策略包含三种变换，针对每一个图像都随机的挑选出一个子策略组合，并以一定的概率来决定是否执行子策略中的每种变换。

Methods

增广方法

operation	description	range
ShearX(Y)	沿水平（垂直）轴以速率剪切图像	[-0.3,0.3]
TranslateX(Y)	按像素的数量级在水平（垂直）方向上平移图像。	[-150,150]
Rotate	旋转图像若干度	[-30,30]
AutoContrast	通过使最暗的像素最大化图像对比度黑色，最浅的像素为白色。	
Invert	反转图像的像素	
Equalize	均衡图像直方图	
Solarize	反转所有超过幅度阈值的像素	[0,256]
Posterize	将每个像素的位数减少到幅度位数	[4,8]
Contrast	控制图像的对比度。幅度= 0给出灰色 图像，而幅值= 1则给出原始图像。	[0.1,1.9]
Color	调整图像的色彩平衡，方法类似于 彩色电视机上的控件。幅度= 0表示黑色＆ 白色图像，而幅值= 1则给出原始图像。	[0.1,1.9]
Brightness	调整图像的亮度。幅度= 0表示黑色 图像，而幅值= 1则给出原始图像。	[0.1,1.9]
Sharpness	调整图像的清晰度。幅度= 0给出一个 模糊图像，而幅值= 1则给出原始图像	[0.1,1.9]
Cutout[3]	将边长大小像素的随机正方形补丁设置为 灰色。	[0,60]
Sample Pairing	将图像与权重大小的另一幅图像（从同一小批量中随机选择）线性地相加，而无需更改标签（分类）。	[0, 0.4]

出于时间和GPU因素（搜索空间大小为（10*11*16)^2）考虑，本次作业没有完整复现auto augment[1]中的搜索算法部分，原文中作者使用了PPO，本质上是policy gradient，通过训练一个rnn controller，policy gradient的reward为child model在验证集上的accuracy。作者在原文的附录中给出了CIFAR-10,CIFAR-100,ImageNet等数据集上通过强化学习得到的子策略集合以及其概率，本次作业中直接选用了CIFAR数据集相关的参数。

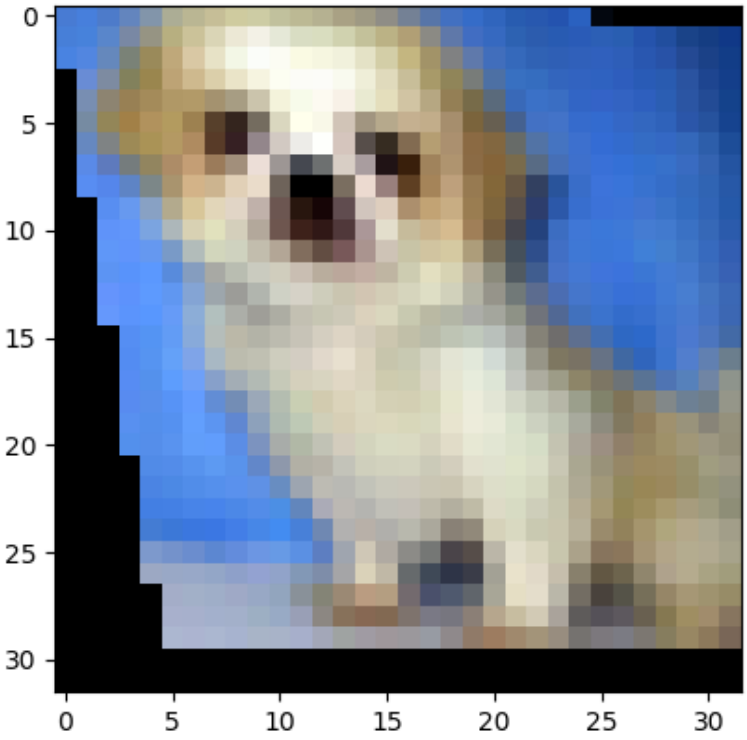
CIFAR-10搜索后得到的子策略集合(见dataset.py)

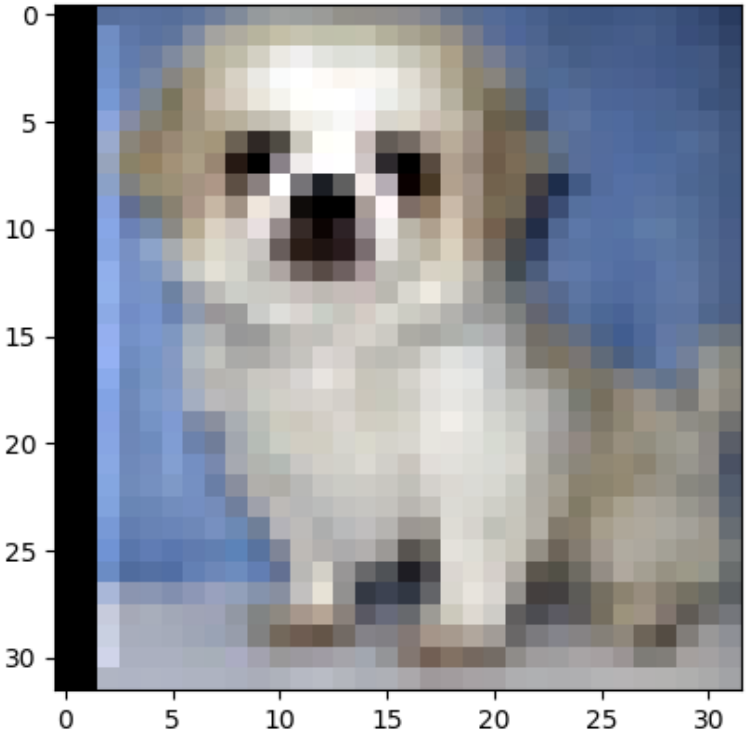
在paper中作者也表明CIFAR和ImageNet这类数据集上更吻合的增广策略是色彩方面变换

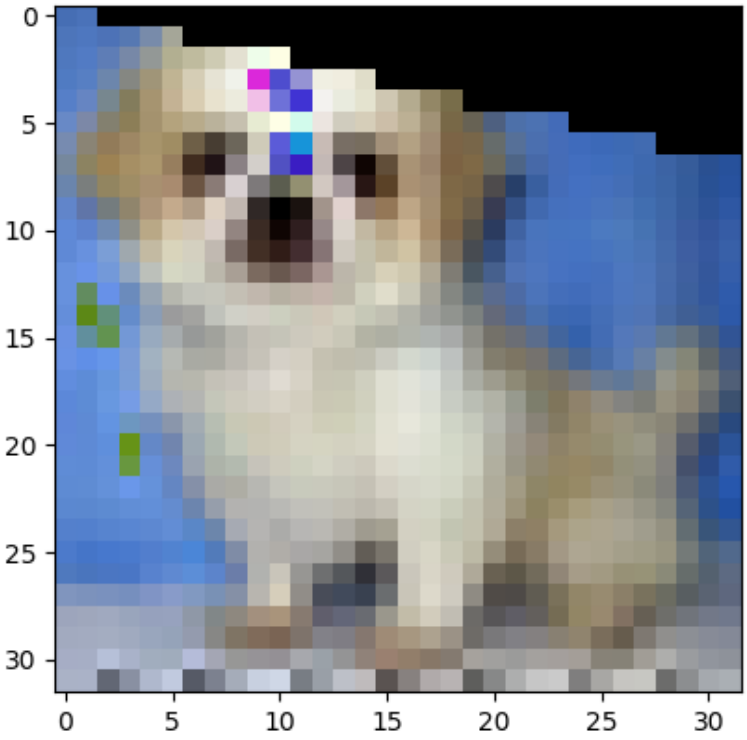
```
self.policies = [
                ['Invert', 0.1, 7, 'Contrast', 0.2, 6],
                ['Rotate', 0.7, 2, 'TranslateX', 0.3, 9],
                ['Sharpness', 0.8, 1, 'Sharpness', 0.9, 3],
                ['ShearY', 0.5, 8, 'TranslateY', 0.7, 9],
                ['AutoContrast', 0.5, 8, 'Equalize', 0.9, 2],
                ['ShearY', 0.2, 7, 'Posterize', 0.3, 7],
                ['Color', 0.4, 3, 'Brightness', 0.6, 7],
                ['Sharpness', 0.3, 9, 'Brightness', 0.7, 9],
                ['Equalize', 0.6, 5, 'Equalize', 0.5, 1],
                ['Contrast', 0.6, 7, 'Sharpness', 0.6, 5],
                ['Color', 0.7, 7, 'TranslateX', 0.5, 8],
                ['Equalize', 0.3, 7, 'AutoContrast', 0.4, 8],
                ['TranslateY', 0.4, 3, 'Sharpness', 0.2, 6],
                ['Brightness', 0.9, 6, 'Color', 0.2, 8],
                ['Solarize', 0.5, 2, 'Invert', 0.0, 3],
                ['Equalize', 0.2, 0, 'AutoContrast', 0.6, 0],
                ['Equalize', 0.2, 8, 'Equalize', 0.6, 4],
                ['Color', 0.9, 9, 'Equalize', 0.6, 6],
                ['AutoContrast', 0.8, 4, 'Solarize', 0.2, 8],
                ['Brightness', 0.1, 3, 'Color', 0.7, 0],
                ['Solarize', 0.4, 5, 'AutoContrast', 0.9, 3],
                ['TranslateY', 0.9, 9, 'TranslateY', 0.7, 9],
                ['AutoContrast', 0.9, 2, 'Solarize', 0.8, 3],
                ['Equalize', 0.8, 8, 'Invert', 0.1, 3],
                ['TranslateY', 0.7, 9, 'AutoContrast', 0.9, 1],
            ]
```

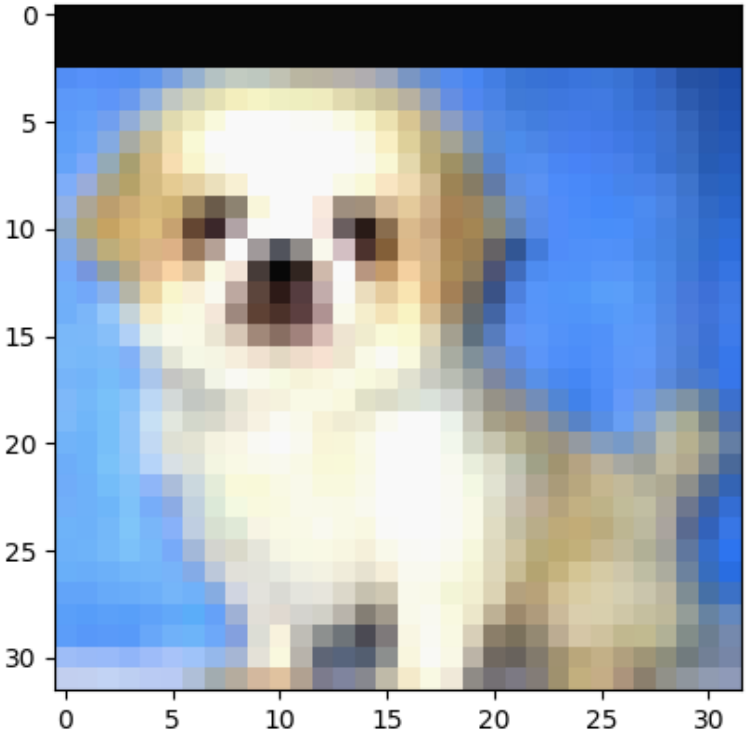
Demonstration

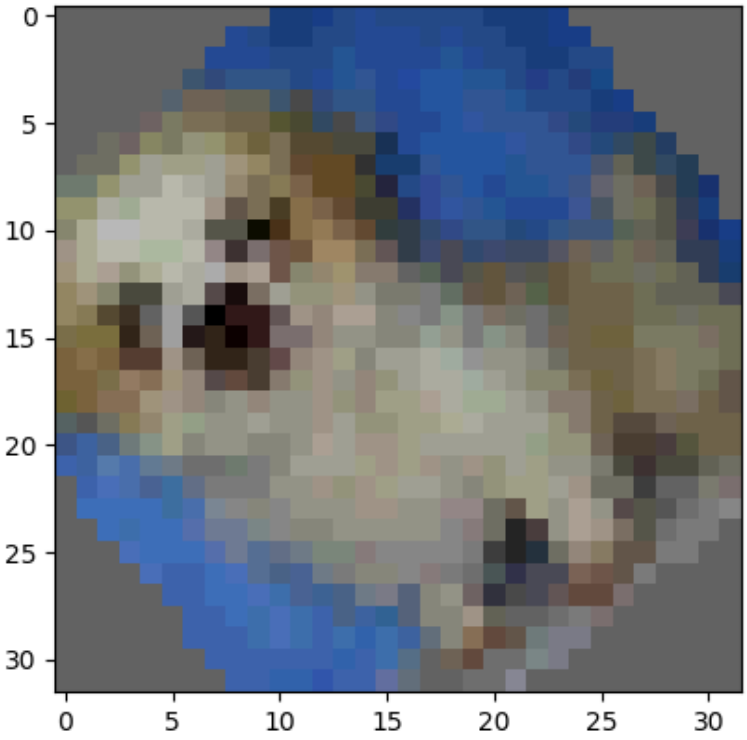
可以使用single.py来选择单张图片测试图像增广效果，样例图片放在example-test-pics文件夹下，修改single.py中line 100可以选择图片，下面多次测试了一张dog的图片增广的效果。


0
5
10
15
20
25
30
0
5
10
15
20
25
30


0
5
10
15
20
25
30
0
5
10
15
20
25
30


0
5
10
15
20
25
30
0
5
10
15
20
25
30


0
5
10
15
20
25
30
0
5
10
15
20
25
30



Process&Code Introduction

train.py为训练（已注释）和在原有/生成验证集测试的主文件

auto_augment.py中复现了paper[1]中的15中增广方法和强化学习策略（未使用）

dataset.py为针对使用的CIFAR系列数据集的数据增广文件，已经直接设定了本应由rnn训练得到的子策略集合参数

utils.py和cosine_annealing.py是命令行处理的工具函数和训练时自动调整学习率的scheduler（由于已经注释了训练部分，暂时未使用）

single.py是通过单张图片展示增广策略效果

测试时的流程：首先根据命令行中是否有auto-augment参数来决定是否对测试集进行自动增广处理，若是，则调用dataset.py中正则化及图像变换策略选取函数，根据选取的策略去auto-augment中调用相应的增广函数。对测试集处理完成后，train.py将load的模型进行验证和准确性、损失计算。

Environment

- Python 3.7
- Keras 2.2.4
- CIFAR-10 CIFAR-100
- Joblib 0.16.0

Testing

默认的测试集为CIFAR-10，可以在train.py中修改line 78将其改为CIFAR-100

修改train.py的line 76,将modelPath改为需要选用的模型路径（已存放在model子文件夹下）

```
    model=load_model(modelPath)
```

测试模型baseline

```
python train.py
```

测试自动化生成的验证集:

```
python train.py --auto-augment True
```

Results

下表是使用自动生成的测试集在训练完成模型上测试的结果，需要注意的是，这里仅仅更换了验证集，而没有使用生成的增广图像对模型进行再训练

CIFAR-10

Model	Accuracy (%)	Loss
CNN_with_dropout baseline	64.04	1.0475
CNN_with_dropout **on generated testset**	**35.22**	**4.6904**
CNN_without_dropout baseline	71.72	0.8323
CNN_without_dropout **on generated testset**	**43.67**	**3.2337**
lenet5_with_dropout baseline	66.19	0.9698
lenet5_with_dropout **on generated testset**	**23.36**	**31.4828**
lenet5_without_dropout baseline	72.80	0.8092
lenet5_without_dropout **on generated testset**	**39.42**	**5.2990**
ResNet_v1 baseline	68.38	1.1607
ResNet_v1 **on generated testset**	**44.35**	**2.6295**
ResNet_v2 baseline	71.47	1.0964
ResNet_v2 **on generated testset**	**46.64**	**3.1971**
random1_cifar10 baseline	11.72	8.0583
random1_cifar10 **on generated testset**	**9.41**	**7.8863**
random2_cifar10 baseline	11.15	7.9619
random2_cifar10 **on generated testset**	**10.63**	**7.7602**

由此可以看出，各个模型在面对增广生成的验证集均有20%-45%的准确率损失，很大程度上这是由于没有采用的图像增广手段在数据集中产生了更多的特征。而由于没有在增广数据集上训练，而是直接使用增广后的验证集来测试。为了验证这一猜想，我在lenet5_with_dropout的baseline上使用了增广后训练集进行了5个epoch的训练（lenet.h5），其acc率从23.36%上升至60.8%。

CIFAR-100

Model	Accuracy (%)	Loss
CNN_with_dropout baseline	38.25	2.4512
CNN_with_dropout **on generated testset**	12.08	7.9666
CNN_without_dropout baseline	37.33	3.1863
CNN_without_dropout **on generated testset**	14.30	16.4880
lenet5_with_dropout baseline	38.85	2.4238
lenet5_with_dropout **on generated testset**	18.31	8.4810
lenet5_without_dropout baseline	35.60	3.5416
lenet5_without_dropout **on generated testset**	19.43	16.7900
ResNet_v1 baseline	36.17	5.3387
ResNet_v1 **on generated testset**	10.15	17.1823
ResNet_v2 baseline	27.13	3.6930
ResNet_v2 **on generated testset**	6.68	10.7418
random1_cifar100 baseline	0.99	8.2081
random1_cifar100 **on generated testset**	1.09	8.3221
random2_cifar100 baseline	1.30	8.2956
random2_cifar100 **on generated testset**	1.22	8.2420

Evaluation

Model	Accuracy (%)	Loss
lenet5_with_dropout baseline	66.19	0.9698
lenet5_with_dropout **on generated testset**	23.36	31.4828
lenet5_with_dropout **trained on generated trainset (5 epoches)**	60.8	1.3718

纵向对比来看，在使用lenet5模型情况下，没有使用生成的数据集进行训练时，直接使用生成的验证集进行测试得到的效果较差(准确率下降了超过40%)，而在使用同样策略生成训练集进行训练后（5次迭代）再进行测试时accuracy又上升至60.8%。

由于缺少可用GPU，本次作业没有进行resnet这类深层cnn的生成训练集训练后效果测试，但从paper[1]中可以看出，超过200 epoch之后，模型的accuracy已经超出原有baseline 1-2%。

由此可见，现有模型（尤其是lenet这类浅层模型）的鲁棒性和泛化能力较差，再对图像处理后准确性便大幅下滑，而resnet这类模型具有更好的鲁棒性，这从另一方面也反映出了**生成数据集和原数据集有较大的差异性，采用如上的自动生成方法可以有效避免过拟合和缓解训练集过小。**

Reference

[1] arXiv:1805.09501v1 [cs.CV] AutoAugment: Learning Augmentation Policies from Data Ekin D. Cubuk∗†, Barret Zoph† , Dandelion Mané, Vijay Vasudevan, Quoc V. Le

[2] arXiv:1909.13719v2 [cs.CV] RandAugment: Practical automated data augmentation with a reduced search space Ekin D. Cubuk ∗ , Barret Zoph∗ , Jonathon Shlens, Quoc V. Le Google Research, Brain Team {cubuk, barretzoph, shlens, qvl}@google.com

[3] arXiv:1708.04552v2 [cs.CV] Improved Regularization of Convolutional Neural Networks with Cutout Terrance DeVries1 and Graham W. Taylor1,2 1University of Guelph 2Canadian Institute for Advanced Research and Vector Institute