



Differentiable Augmentation for Data-Efficient GAN Training

彭杨

北京邮电大学

计算机学院



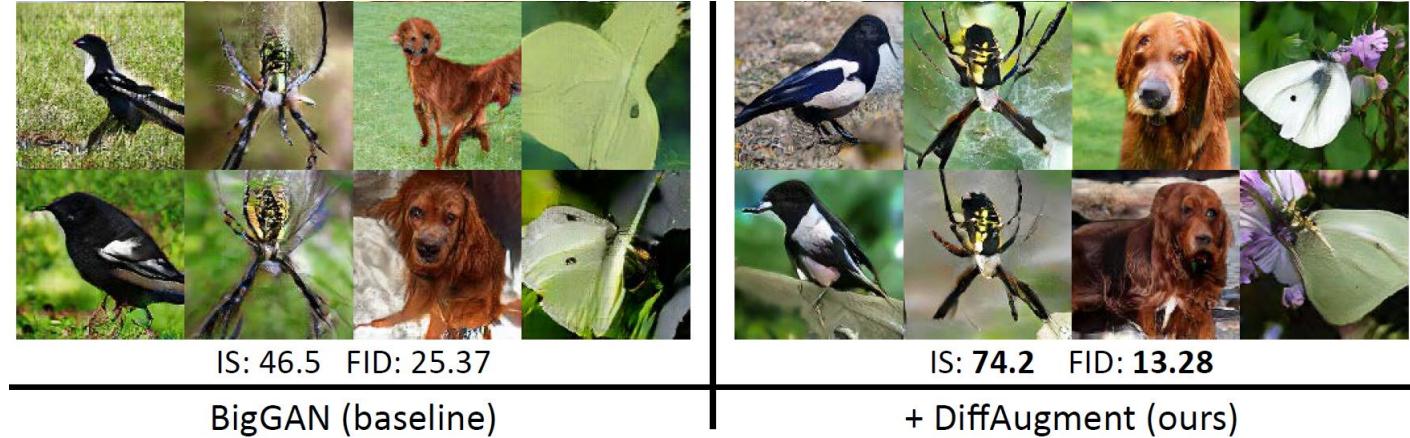
北京郵電
博物館

Part 01 解决问题

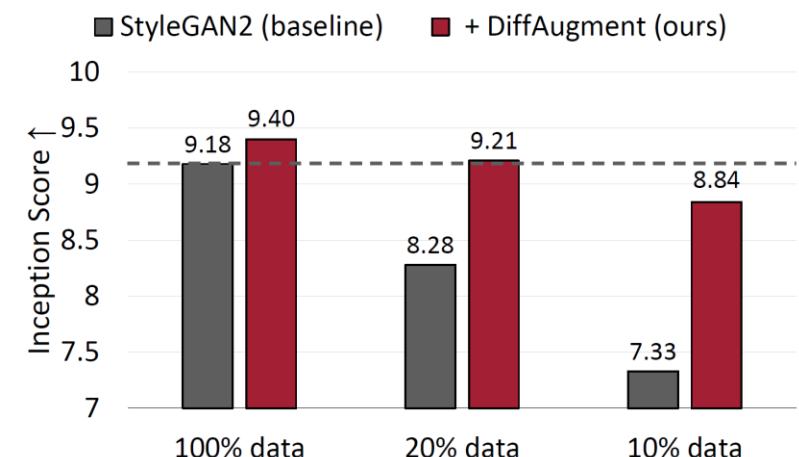
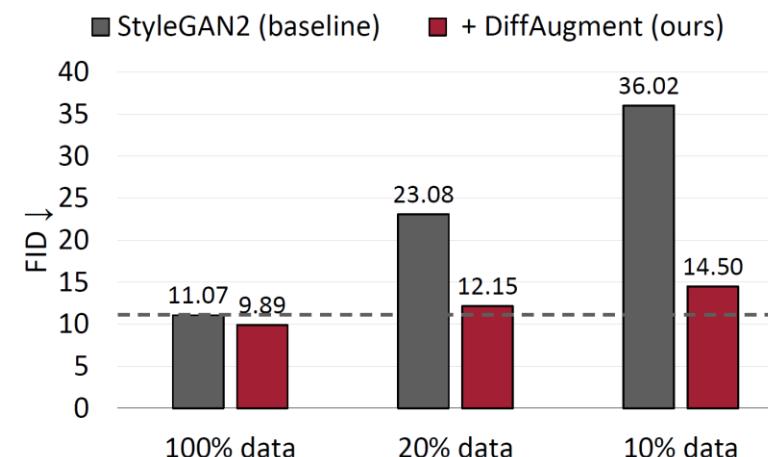
1.1 用差分增强方法提高GAN训练过程中的数据效率

解决问题：小数据集下的过拟合问题，达到使用很少的数据就可以训练出效果很好的GAN，提高数据效率

在25%的训练数据下，BigGAN + DiffAugment 生成效果更好 (FID, IS)



在20%的训练数据下，StyleGAN2 + DiffAugment 就能达到100%训练数据下StyleGAN2 的效果



1.2 主要贡献

主要工作

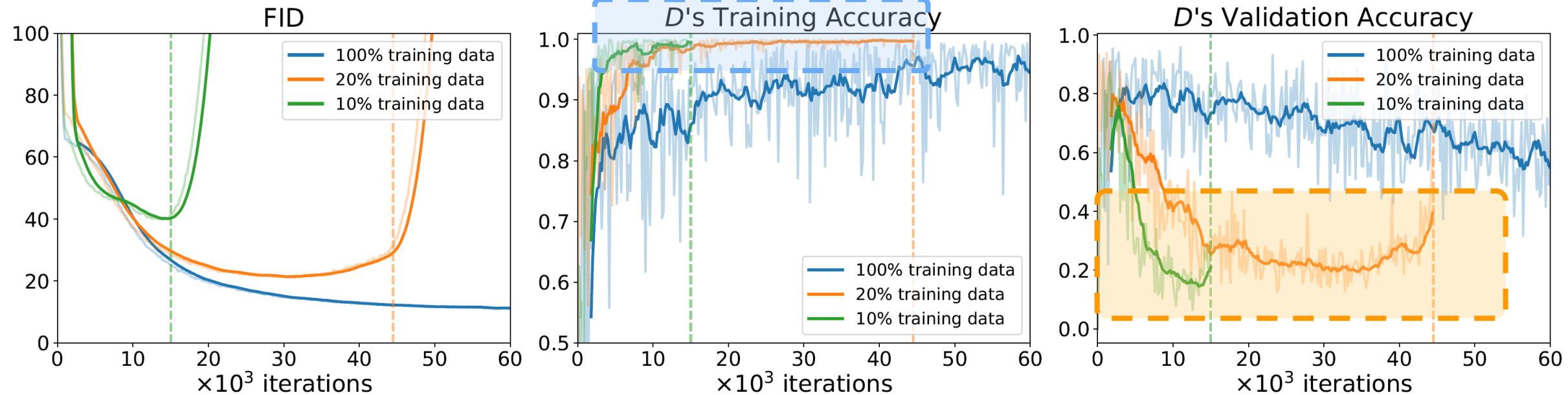
- 1** 尝试了三种数据增强方法，并提出Differentiable Augmentation，使用20%-25%的训练数据达到不使用本文方法情况下100%数据的生成效果
- 2** 尝试L1正则化方法对小数据集训练中预防过拟合的作用
- 3** 尝试减小模型容量对小数据集中预防过拟合的作用

方法概述

- 1** 仅仅做训练集数据增强优化
- 2** 仅仅优化D的输入（包括1）
- 3** 优化D的输入，和G的反向传播

1.3 过拟合

过拟合的表现：①：评价指标（如FID的突然变差）②：D在训练集上和验证集上准确率差距过大且越差越大
 越少数据集模式崩塌越早（注意10%，20%，和100%数据集下图的比较）
 D在训练数据集上过高的准确率和验证集上过低的准确率意味着过拟合，即D记住了训练集的数据
 D的准确率过高，无法给G提供梯度





北京邮局

Part 02 前人工作

2.1 Related Work

训练GAN遇到的问题

- 1、计算负担较大（可以用压缩算法降低，本文不讨论）
- 2、数据效率过低，必须使用大量数据才能训练GAN，本文解决。

数据效率过低（数据不足，缺乏数据）

- 1、标签少（少监督、无监督学习干这事儿）
- 2、图片少（本文解决）

正则化方法

谱归一化、JS归一化、梯度惩罚等，本质上是防止D参数的突然变化

数据增强

- 1、经常用于分类任务，之前没有应用到GAN上
- 2、GAN是学习数据分布的，直接进行数据增强破坏掉数据分布



北京郵局

Part 03 使 用 方 法

3.1 方法概述

GAN的公式中，有3部分图片，一部分是D损失中接受的 x 和 $G(z)$ ，还有一部分是G损失中用于计算梯度的 $G(z)$

$$L_D = \mathbb{E}_{x \sim p(x)} [f_D(-D(x))] + \mathbb{E}_{z \sim p(z)} [f_D(D(G(z)))]$$

$$L_G = \mathbb{E}_{z \sim p(z)} [f_G(-D(G(z)))]$$

各种变换：平移、裁剪、加蒙版、颜色变换（对比度、饱和度、亮度）

1、如果只在训练数据集上做各种变换，实际上生成器容易错误地学到这些变换内容，进而产生伪影，比如如果某个地方被遮挡，那么很可能学到遮挡。

$$L_D = \mathbb{E}_{x \sim p(x)} [f_D(-D(T(x)))] + \mathbb{E}_{z \sim p(z)} [f_D(D(G(z)))]$$

$$L_G = \mathbb{E}_{z \sim p(z)} [f_G(-D(G(z)))]$$

2、同时增强判别器输入端的真图和生成图导致生成器判别器的平衡被打破，相当于改变了目标函数

$$L_D = \mathbb{E}_{x \sim p(x)} [f_D(-D(T(x)))] + \mathbb{E}_{z \sim p(z)} [f_D(D(T(G(z))))]$$

$$L_G = \mathbb{E}_{z \sim p(z)} [f_G(-D(G(z)))]$$

3、训练和测试时在三部分都进行增强，注意在训练G的时候的增强是通过梯度实现的。

$$L_D = \mathbb{E}_{x \sim p(x)} [f_D(-D(T(x)))] + \mathbb{E}_{z \sim p(z)} [f_D(D(T(G(z))))]$$

$$L_G = \mathbb{E}_{z \sim p(z)} [f_G(-D(T(G(z))))]$$

3.2 Baseline

一般的数据增强只增强数据集，在GAN中就是训练数据集。

一般的数据增强方法有：平移、裁剪、加蒙版、颜色变换（对比度、饱和度、亮度），这些都破坏了数据集分布
(GAN是在拟合数据集分布)

只有翻转能够保留数据集分布，因此使用在数据集中添加翻转的方法提升BigGAN, StyleGAN2等，作为
Baseline

在BigGAN中，FID 8.7->7.6

使用原始方法中采用的其他正则化方法：谱归一化、一致性损失、R1正则化

细节：

1、原文随机缩放随机裁剪，本文随机缩放中间裁剪

2、使用DiffAugment的情况下，过大的学习率会导致D的loss太大，因此从 $5 \times 10^{-4} \rightarrow 4 \times 10^{-4}$ ，在BigGAN，
100%Inagenet设定下

3、使用DiffAugment的情况下，小数据集需要小一点的lr， 2×10^{-4} 在10%和20%数据集，但是在50%数据集上
会让表现变差 FID/IS 9.64/89.9 -> 10.79/75.7

3.2 只增强训练数据集 $x \rightarrow T(x)$

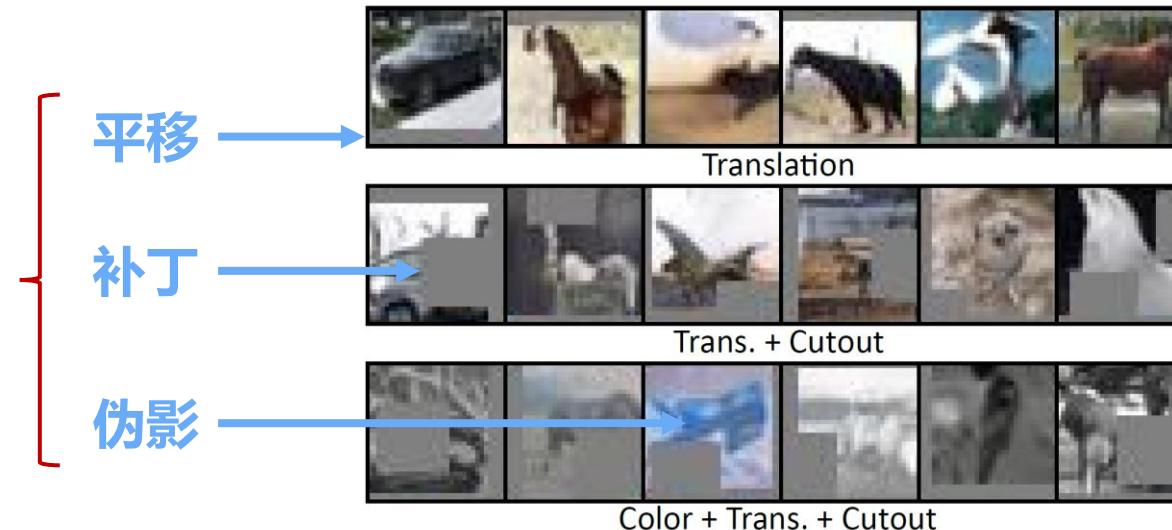
思路来源于分类任务，把数据集中的数据进行前面提到的各种增强

缺点：与GAN的原理 – 拟合数据分布相悖，GAN将学习到增强后的数据集（收敛的最优状态），而不是原数据集，因而会带上补丁、色彩伪影的不期望的东西

$$L_D = \mathbb{E}_{x \sim p_{\text{data}}(x)}[f_D(-D(\mathbf{T}(x)))] + \mathbb{E}_{z \sim p(z)}[f_D(D(G(z)))]$$

$$L_G = \mathbb{E}_{z \sim p(z)}[f_G(-D(G(z)))].$$

增强数据集带来的后果



(a) “Augment reals only”: the same augmentation artifacts appear on the generated images.

3.3 增强D，包括训练数据集 $x \rightarrow T(x)$ ，和生成图 $G(z) \rightarrow T(G(z))$

站在D的角度，D在更新时同时考虑两部分，只对一部分进行变换肯定是不平衡的，所以对数据集和生成图都变换
 缺点：这意味着D无法给不patch的图片 ($G(z)$) 提供足够的信息，打破了D和G的训练平衡

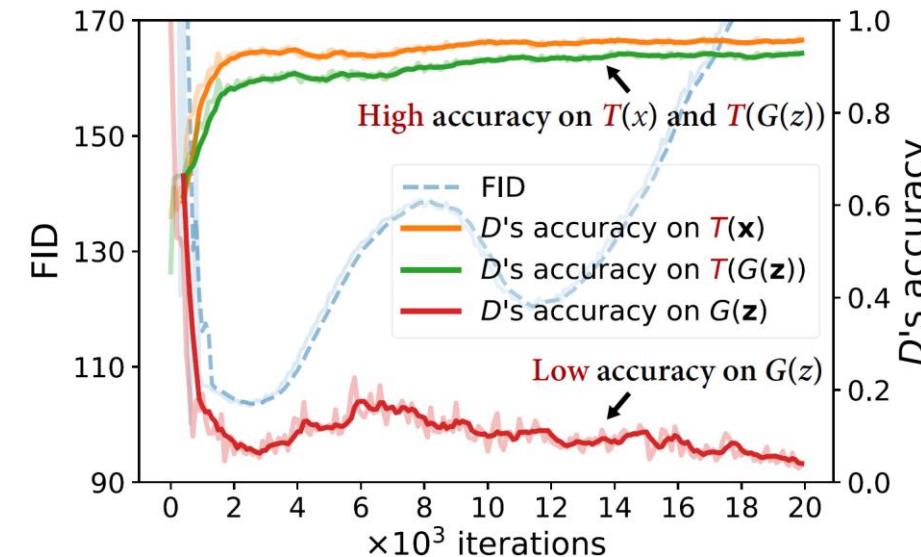
$$L_D = \mathbb{E}_{x \sim p_{\text{data}}(x)} [f_D(-D(\textcolor{red}{T}(x)))] + \mathbb{E}_{z \sim p(z)} [f_D(D(\textcolor{red}{T}(G(z))))],$$

$$L_G = \mathbb{E}_{z \sim p(z)} [f_G(-D(G(z)))].$$

GAN训练的两个步骤：①固定G训练D ②固定D训练G

在固定G训练D时，D对于增强后的图片分类准确率高

在固定D训练G时，D对于 $G(z)$ 判断准确率下降



(b) “Augment D only”: the unbalanced optimization between G and D cripples training.

3.3 增强D的同时，考虑G

刚刚的实验说明G和D的平衡不能被打破，且G不能忽视这种数据增强

方法：在固定D训练G时，也给D输入增强后的图片，使回传给G的梯度包含对增强操作的考虑

$$L_D = \mathbb{E}_{x \sim p_{\text{data}}(x)} [f_D(-D(\mathbf{T}(x)))] + \mathbb{E}_{z \sim p(z)} [f_D(D(\mathbf{T}(G(z))))],$$

$$L_G = \mathbb{E}_{z \sim p(z)} [f_G(-D(G(z)))].$$

从D的表现判断过拟合

D在训练集上准确率较低，但是验证集上准确率很高，说明D没有过拟合

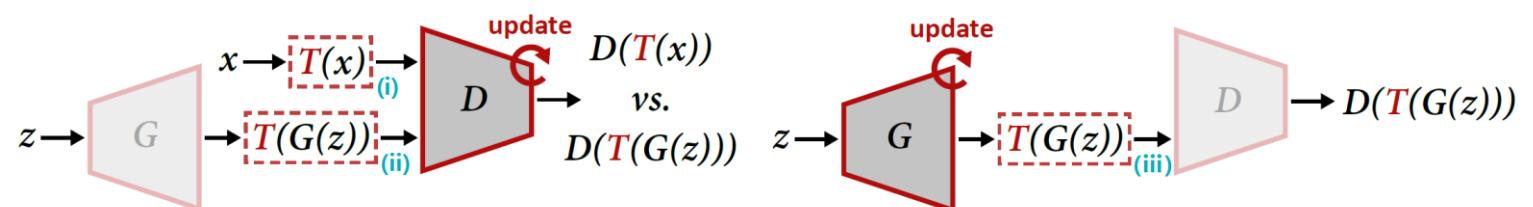
D在训练集上准确率很高，但在验证集上准确率很低，说明D过拟合，是在背训练集图片

调整包括：平移

平移： $[-\frac{1}{8}, \frac{1}{8}]$ ，空白补0

裁剪：添加图像大小一半的正方形

色彩：亮度 (L) $[-0.5, 0.5]$ ，对比度 $[0.5, 1.5]$ ，饱和度 (S) $[0, 2]$



3.4 应用较为简单 – 对原模型的G和D的输出套娃处理

data-efficient-gans/DiffAugment-stylegan2/training/loss.py

Lines 19 to 20 in ce3c9c4

```
19     real_scores = D.get_output_for(DiffAugment(reals, policy=policy, channels_first=True), is_training=True)
20     fake_scores = D.get_output_for(DiffAugment(fakes, policy=policy, channels_first=True), is_training=True)
```

```
5 import torch
6 import torch.nn.functional as F
7
8 def DiffAugment(x, policy='', channels_first=True):
9     if policy:
10         if not channels_first:
11             x = x.permute(0, 3, 1, 2)
12         for p in policy.split(','):
13             for f in AUGMENT_FNS[p]:
14                 x = f(x)
15         if not channels_first:
16             x = x.permute(0, 2, 3, 1)
17         x = x.contiguous()
18     return x
19
20 def rand_brightness(x):
21     x = x + (torch.rand(x.size(0), 1, 1, 1, dtype=x.dtype, device=x.device) - 0.5)
22     return x
23
24 def rand_saturation(x):
25     x_mean = x.mean(dim=1, keepdim=True)
26     x = (x - x_mean) * (torch.rand(x.size(0), 1, 1, 1, dtype=x.dtype, device=x.device) * 2) + x_mean
27     return x
28
29 def rand_contrast(x):
30     x_mean = x.mean(dim=[1, 2, 3], keepdim=True)
31     x = (x - x_mean) * (torch.rand(x.size(0), 1, 1, 1, dtype=x.dtype, device=x.device) + 0.5) + x_mean
32
33
34
35
36
37
38
39     def rand_translation(x, ratio=(1, 8)):
40         shift_x, shift_y = x.size(2) * ratio[0] // ratio[1], x.size(3) * ratio[0] // ratio[1]
41         translation_x = torch.randint(-shift_x, shift_x + 1, size=[x.size(0), 1, 1], device=x.device)
42         translation_y = torch.randint(-shift_y, shift_y + 1, size=[x.size(0), 1, 1], device=x.device)
43         grid_batch, grid_x, grid_y = torch.meshgrid(
44             torch.arange(x.size(0), dtype=torch.long, device=x.device),
45             torch.arange(x.size(2), dtype=torch.long, device=x.device),
46             torch.arange(x.size(3), dtype=torch.long, device=x.device),
47         )
48         grid_x = torch.clamp(grid_x + translation_x + 1, 0, x.size(2) + 1)
49         grid_y = torch.clamp(grid_y + translation_y + 1, 0, x.size(3) + 1)
50         x_pad = F.pad(x, [1, 1, 1, 1, 0, 0, 0, 0])
51         x = x_pad.permute(0, 2, 3, 1).contiguous()[grid_batch, grid_x, grid_y].permute(0, 3, 1, 2)
52     return x
53
54
55     def rand_cutout(x, ratio=(1, 2)):
56         cutout_size = x.size(2) * ratio[0] // ratio[1], x.size(3) * ratio[0] // ratio[1]
57         offset_x = torch.randint(0, x.size(2) + (1 - cutout_size[0] % 2), size=[x.size(0), 1, 1], device=x.device)
58         offset_y = torch.randint(0, x.size(3) + (1 - cutout_size[1] % 2), size=[x.size(0), 1, 1], device=x.device)
59         grid_batch, grid_x, grid_y = torch.meshgrid(
60             torch.arange(x.size(0), dtype=torch.long, device=x.device),
61             torch.arange(cutout_size[0], dtype=torch.long, device=x.device),
62             torch.arange(cutout_size[1], dtype=torch.long, device=x.device),
63         )
64         grid_x = torch.clamp(grid_x + offset_x - cutout_size[0] // 2, min=0, max=x.size(2) - 1)
65         grid_y = torch.clamp(grid_y + offset_y - cutout_size[1] // 2, min=0, max=x.size(3) - 1)
66         mask = torch.ones(x.size(0), x.size(2), x.size(3), dtype=x.dtype, device=x.device)
67         mask[grid_batch, grid_x, grid_y] = 0
68         x = x * mask.unsqueeze(1)
69     return x
```

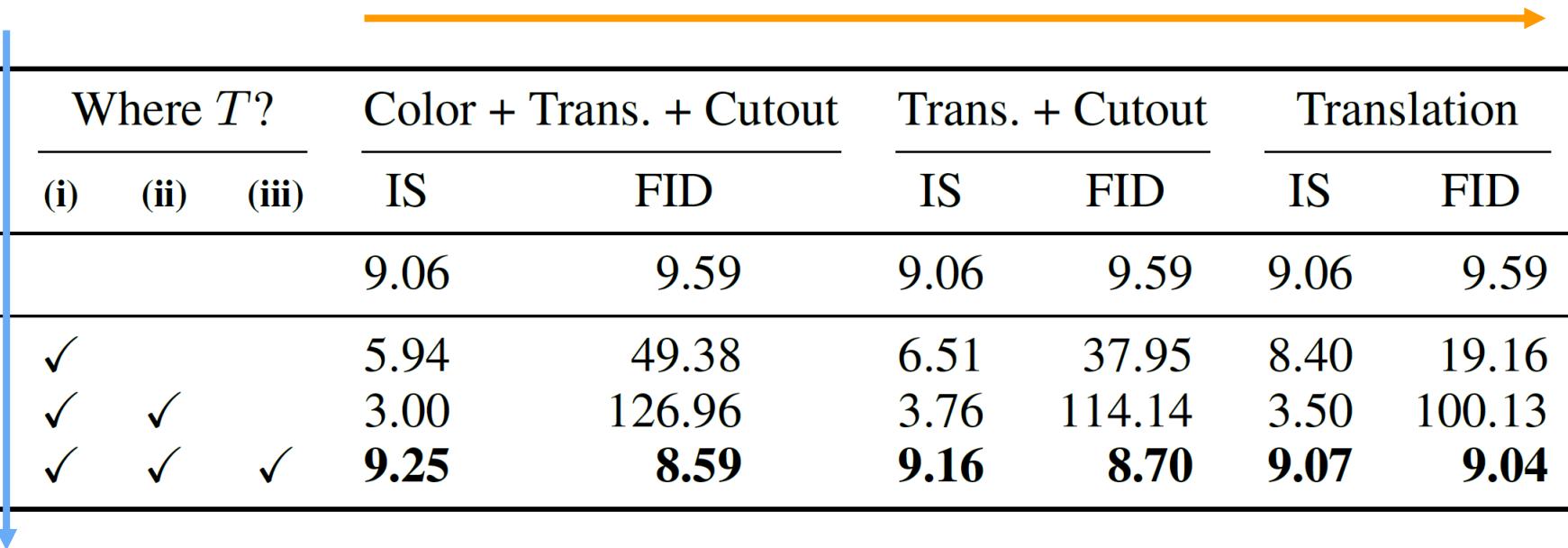


北京邮局

Part 04 实验结果

4.1 在三处添加增强操作得到的效果最好

- 1、最好的情况只在三处都添加增强效果最好，只在D添加效果最差
- 2、在三处都添加增强的情况下，添加的增强手段越多效果越好



Method	Where T ?			Color + Trans. + Cutout		Trans. + Cutout		Translation	
	(i)	(ii)	(iii)	IS	FID	IS	FID	IS	FID
BigGAN (baseline)				9.06	9.59	9.06	9.59	9.06	9.59
Aug. reals only	✓			5.94	49.38	6.51	37.95	8.40	19.16
Aug. reals + fakes (D only)	✓	✓		3.00	126.96	3.76	114.14	3.50	100.13
DiffAugment ($D + G$, ours)	✓	✓	✓	9.25	8.59	9.16	8.70	9.07	9.04

4.2 使用DiffAugmentation在多种模型小数据集上使模型效果提升

只使用翻转、正则化方法，在小数据集上（10%）表现都不好，用DiffAugmentation好（目前最好）

尤其是在小数据集上，针对StyleGAN2，使用20%的数据+DiffAugmentation就达到了使用100%数据的效果

Method	100% training data		20% training data		10% training data	
	IS	FID	IS	FID	IS	FID
BigGAN [2]	9.06	9.59	8.41	21.58	7.62	39.78
+ DiffAugment	9.16	8.70	8.65	14.04	8.09	22.40
CR-BigGAN [46]	9.20	9.06	8.43	20.62	7.66	37.45
+ DiffAugment	9.17	8.49	8.61	12.84	8.49	18.70
StyleGAN2 [16]	9.18	11.07	8.28	23.08	7.33	36.02
+ DiffAugment	9.40	9.89	9.21	12.15	8.84	14.50

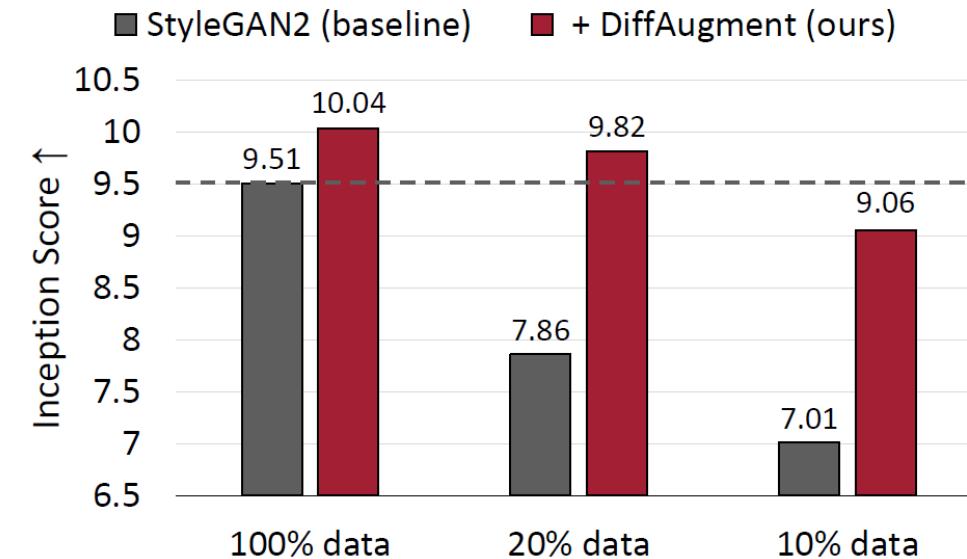
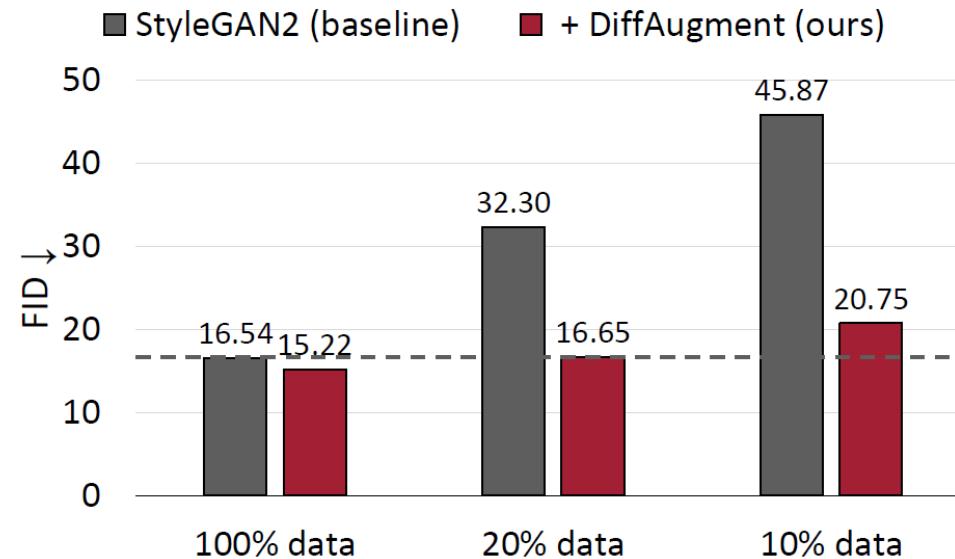
Table 5: **CIFAR-10 results.** IS and FID are measured using 10k samples; the validation set is used as the reference distribution. We report the best FID of each method and its corresponding IS averaged over 5 evaluation runs; all standard deviations are less than 1% relatively.

Method	100% training data		20% training data		10% training data	
	IS	FID	IS	FID	IS	FID
BigGAN [2]	10.92	12.87	9.11	33.11	5.94	66.71
+ DiffAugment	10.66	12.00	9.47	22.14	8.38	33.70
CR-BigGAN [46]	10.95	11.26	8.44	36.91	7.91	47.16
+ DiffAugment	10.81	11.25	9.12	20.28	8.70	26.90
StyleGAN2 [16]	9.51	16.54	7.86	32.30	7.01	45.87
+ DiffAugment	10.04	15.22	9.82	16.65	9.06	20.75

Table 6: **CIFAR-100 results.** IS and FID are measured using 10k samples; the validation set is used as the reference distribution. We report the best FID of each method and its corresponding IS averaged over 5 evaluation runs; all standard deviations are less than 1% relatively.

4.2 使用DiffAugmentation在多种模型小数据集上使模型效果提升

在CIFAR-100和StyleGAN2上，使用20%数据+DiffAugment就可以达到原版StyleGAN2的效果



4.3 与其他少样本生成方法对比

少样本条件下（如100张图），从头训练达到或超过基于预训练模型的迁移学习方法的效果
 同时能够应用在其他迁移学习方法中，以提高效果

Method	Pre-training?	100-shot			AnimalFace [35]	
		Obama	Grumpy cat	Panda	Cat	Dog
Scale/shift [29]	Yes	50.72	34.20	21.38	54.83	83.04
MineGAN [41]	Yes	235.00	287.96	331.86	279.48	254.08
TransferGAN [42]	Yes	48.73	34.06	23.20	52.61	82.38
+ DiffAugment	Yes	39.85	29.77	17.12	49.10	65.57
FreezeD [28]	Yes	41.87	31.22	17.95	47.70	70.46
+ DiffAugment	Yes	35.75	29.34	14.50	46.07	61.03
StyleGAN2 [16]	No	89.18	61.97	90.96	95.75	164.54
+ DiffAugment	No	54.39	29.90	13.21	46.51	62.78

4.4 生成图质量比较

少样本条件下的生成图对比，baseline分别为TransferGAN, FreezeD, StyleGAN2，在AnimalFace数据集



TransferGAN + DiffAugment
FID: 52.61 FID: 49.10

FreezeD + DiffAugment
FID: 47.70 FID: 46.07

StyleGAN2 + DiffAugment
FID: 95.75 FID: 46.51



TransferGAN + DiffAugment
FID: 82.38 FID: 65.57

FreezeD + DiffAugment
FID: 70.46 FID: 61.03

StyleGAN2 + DiffAugment
FID: 164.54 FID: 62.78

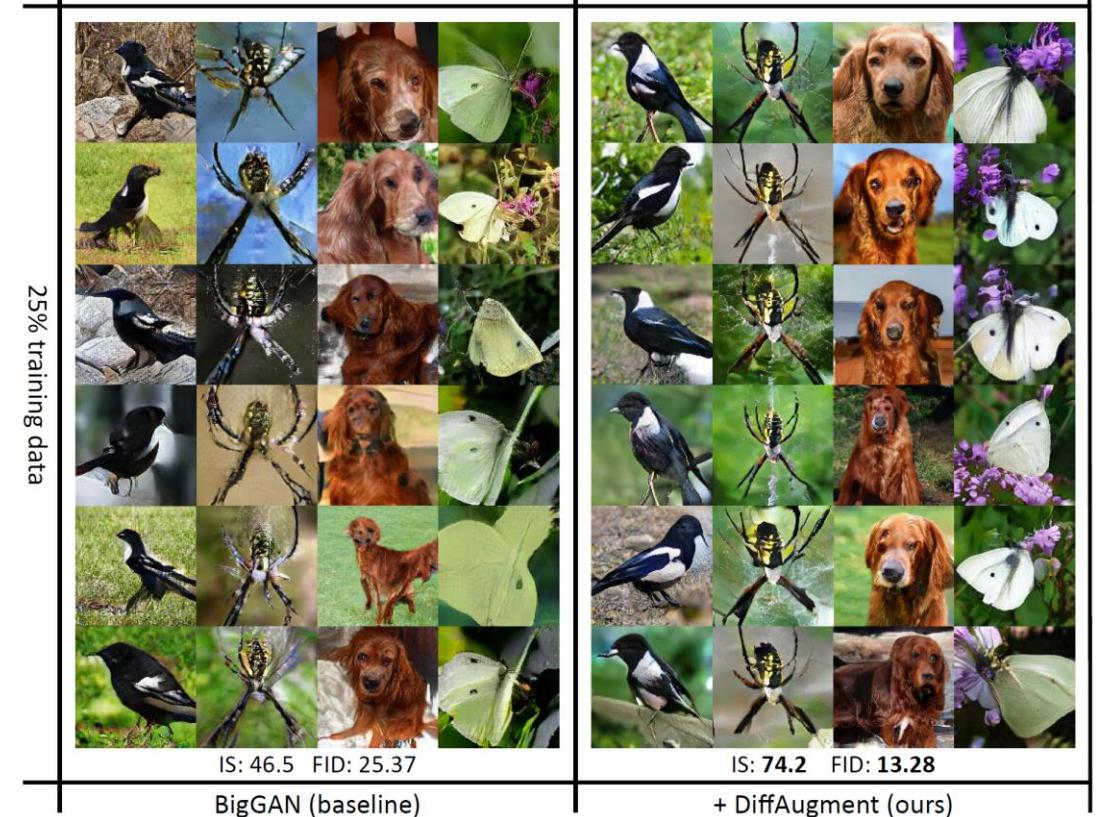
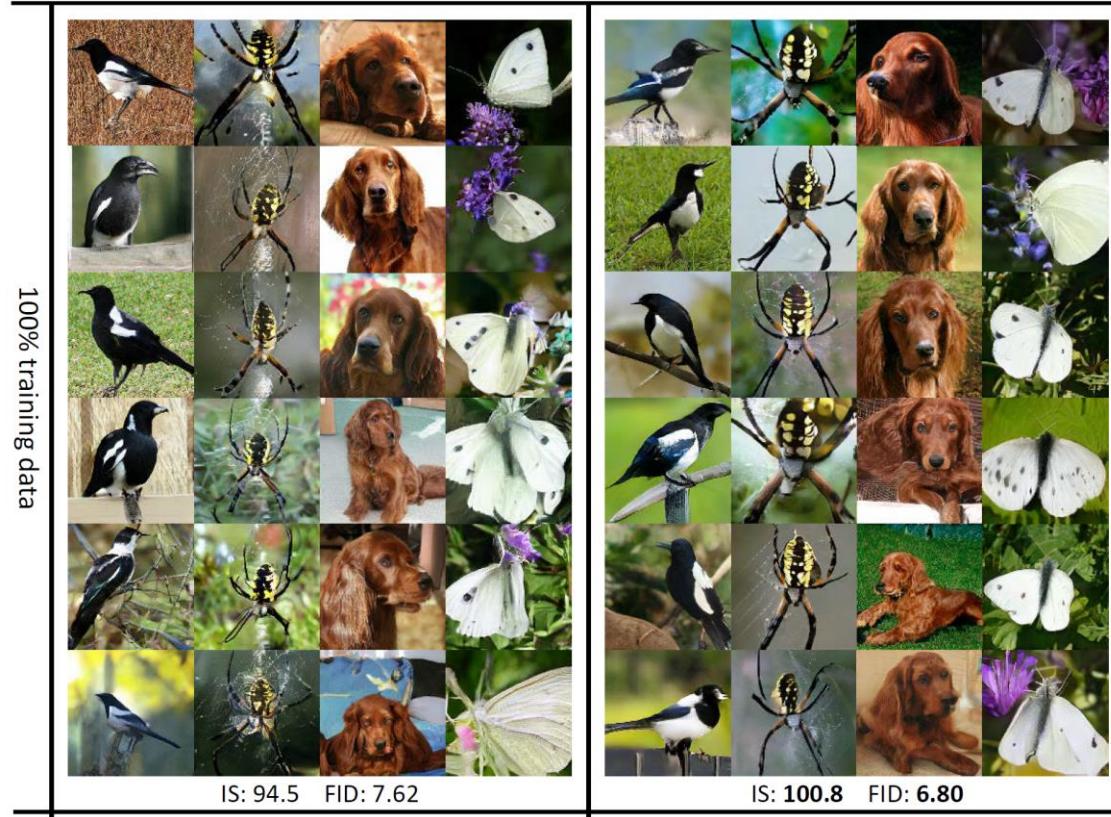
4.4 生成图质量比较

少样本条件下的生成图对比，baseline分别为TransferGAN, FreezeD, StyleGAN2，在3个100张图片数据集



4.4 生成图质量比较

在ImageNet中，视觉差异在25%数据时最大



4.5 差值实验

非常平滑的变化，证明没有overfit

来自小数据集，小数据集更容易overfit，但使用论文方法可以不overfit

奥巴马：100张



grumpy cat：100张



熊猫：100张



AnimalFace - 猫：160张



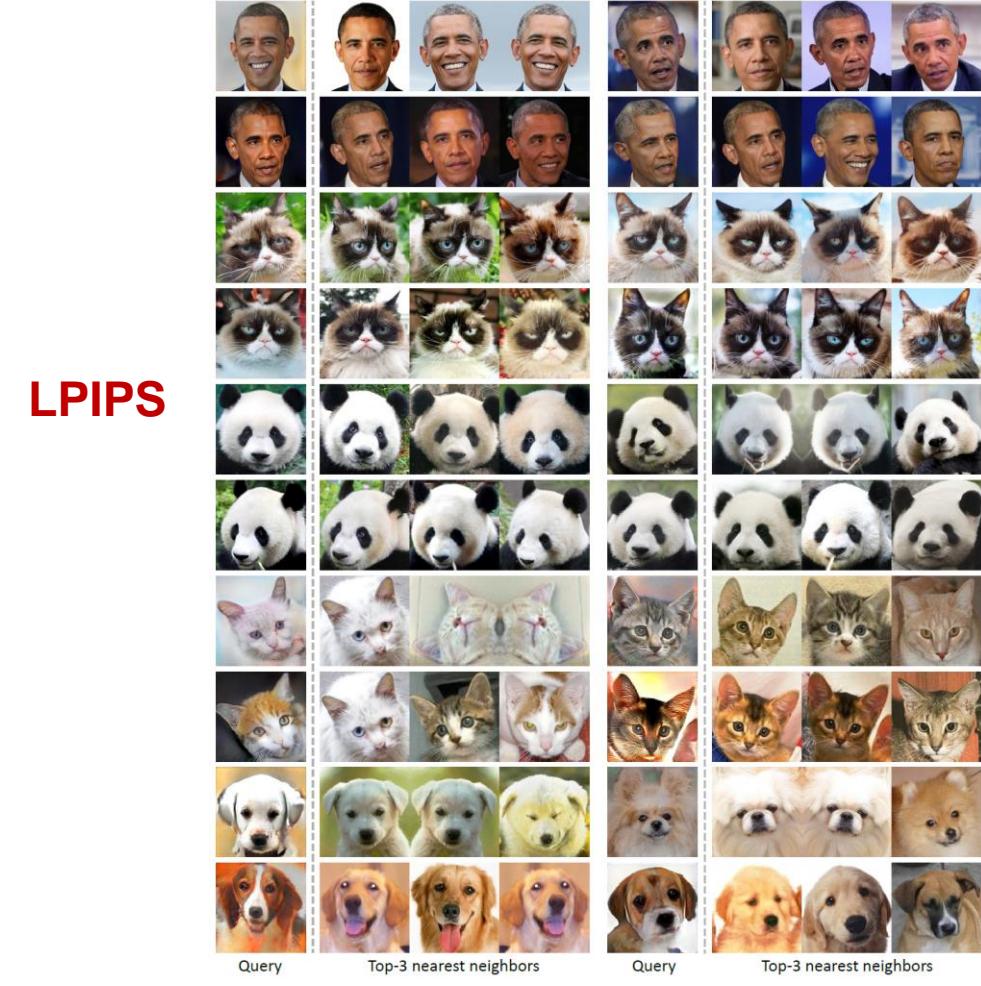
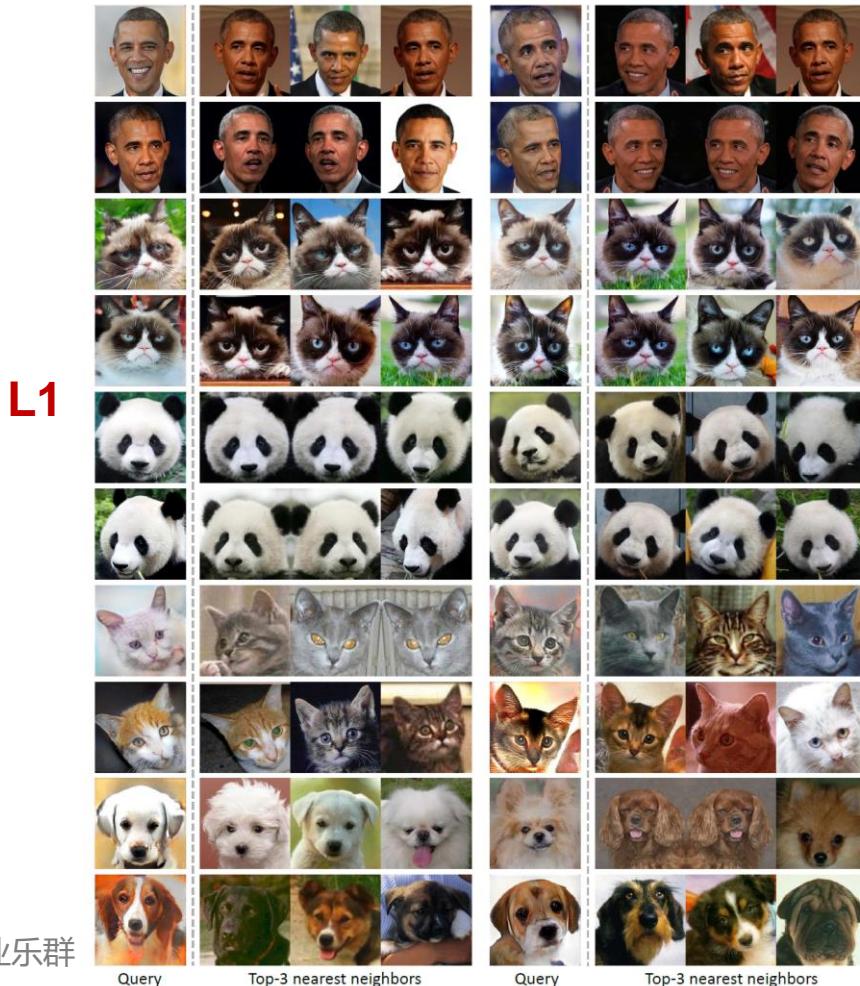
AnimalFace – 狗：389张



4.6 最近邻实验 – pixel wise L1最近邻 & LPIPS最近邻

最近邻都与生成图不相似，说明没有过拟合数据集，不存在背数据集的情况

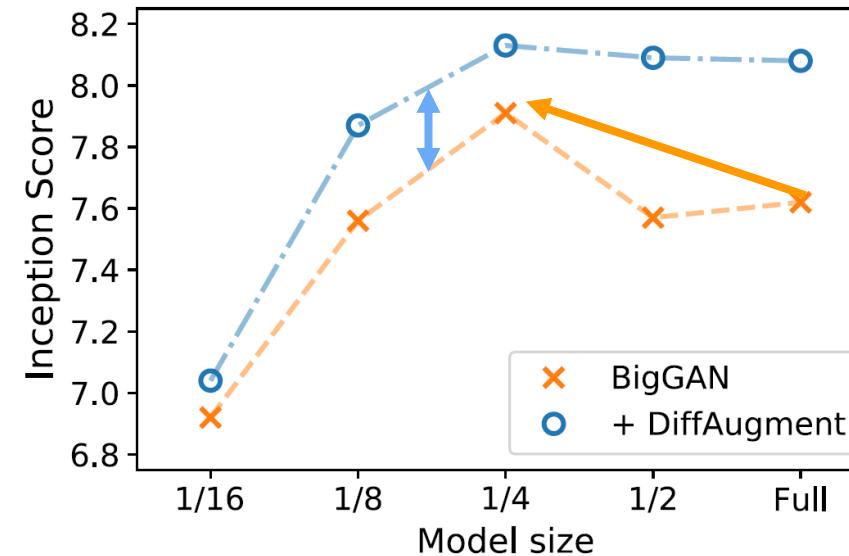
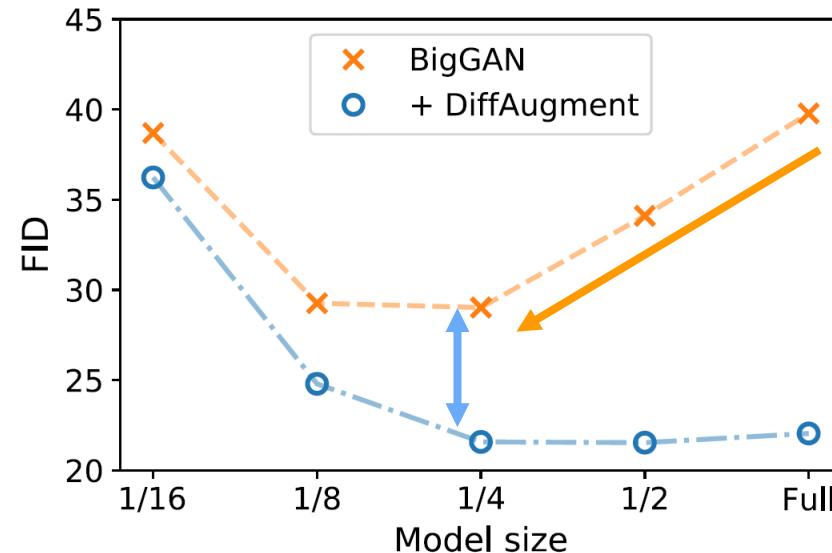
实验采用从头训练的 DiffAugmentation + StyleGAN2，且使用小数据集



4.6 减小模型容量

思路来源于更多的数据需要更大的模型拟合，那么更少的数据是不是使用更小的模型就可以使用小模型（减小channel）能够减小FID，减小过拟合，实验采用BigGAN，10% CIFAR-10

在一定区间内，小模型能够提高小数据集上的生成质量

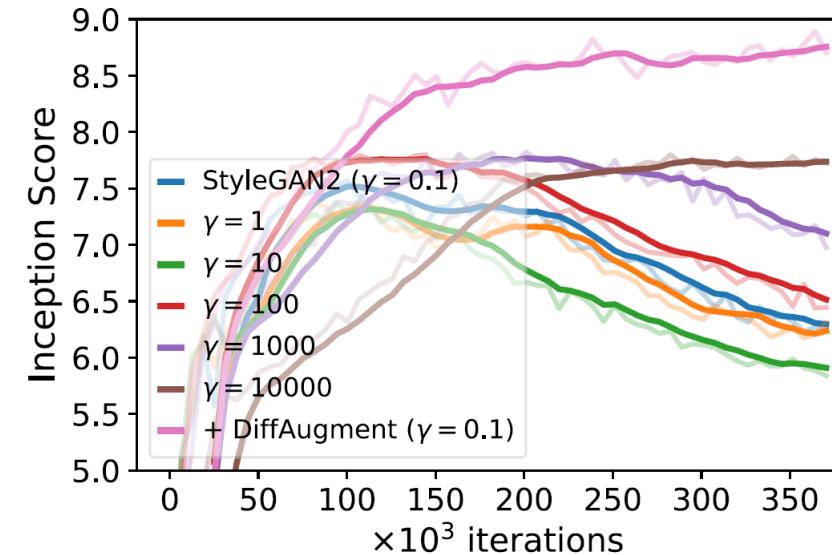
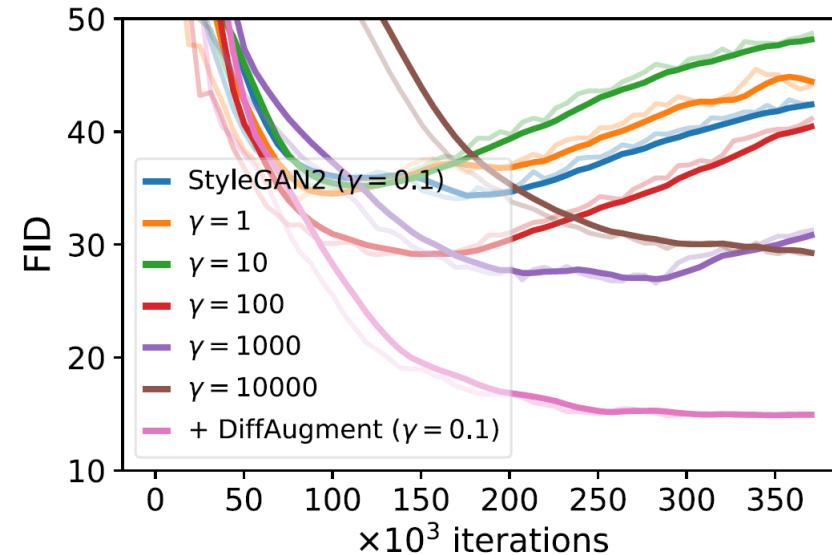


何种情况下，使用DiffAugmentation都有更好的结果，且在小数据集大模型上更明显

4.7 增强正则化项

正则化项本来就是抑制模型过拟合的，那么出现过拟合，增大正则化项也是种方法

随着正则化项的增大，最小的FID减小，最大的IS增大，但是都比不上DiffAugment, 10% CIFAR-10





北京邮局

Part 05 附录

5.1 所使用数据集大小

ImageNet – 1,281,167 Train, 50,000 Eval, 100,000 Test

AnimalFace – 160张猫 389张狗

CIFAR-10 – 该数据集共有60000张彩色图像，这些图像是32*32，分为10个类，每类6000张图。这里有50000张用于训练，构成了5个训练批，每一批10000张图；另外10000用于测试，单独构成一批。测试批的数据里，取自10类中的每一类，每一类随机取1000张。抽剩下的就随机排列组成了训练批。注意一个训练批中的各类图像并不一定数量相同，总的来看训练批，每一类都有5000张图。

CIFAR-100 - 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs).

作者自制数据集：3类每类100张



北京邮局

谢谢

THANK YOU