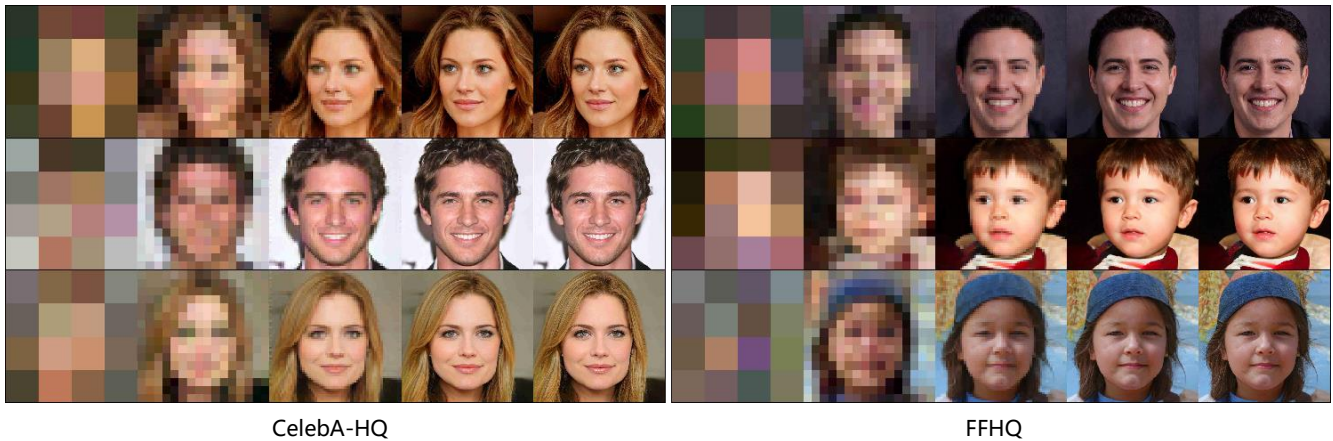


MSG-GAN: 生成对抗网络的多尺度梯度

Animesh Karnewar
TomTom

animesh.karnewar@tomtom.com

Oliver Wang
Adobe Research
owang@adobe.com



CelebA-HQ

FFHQ

图 1: 我们提出的 MSG-GAN 技术的效果, 其中生成器同时合成所有分辨率的图像, 并且梯度从单个判别器直接流到所有级别。第一列的分辨率为 4×4 , 向右增加, 达到最终输出分辨率 1024×1024 。在屏幕上放大观看最佳。

摘要

尽管生成对抗网络 (GAN) 在图像合成任务中取得了巨大成功, 但众所周知它们很难适应于不同的数据集, 部分原因是训练期间的不稳定和对超参数探测的敏感性。引起这种不稳定的一个普遍接受的原因是, 当真实分布和假分布的范围没有足够的重叠时, 从判别器到生成器的梯度将变得毫无意义。在这项工作中, 我们提出了多尺度梯度生成对抗网络 (MSG-GAN), 这是一种简单但有效的技术, 通过允许从判别器到生成器的梯度流在多个尺度上来解决此问题。该技术为高分辨率图像合成提供了一种稳定的方法, 并且可以替代常用的渐进式生成技术。我们表明, MSG-GAN 在各种大小, 分辨率和域以及各种类型的损失函数和体系结构的图像数据集上稳定收敛,

所有这些都具有相同的一组固定的超参数。与最先进的 GAN 相比, 我们的方法在大多数情况下都达到或超过了最佳性能。

1. 介绍

自从 Goodfellow 等人介绍以来 [9], 生成对抗网络 (GAN) 已成为高质量图像合成的事实上的标准。GAN 的成功来自于这样一个事实, 即它们不需要手动设计损失函数来进行优化, 因此可以学会生成复杂的数据分布, 而无需确保能明确定义它们。虽然基于流的模型 (例如 [5、6、24、15]) 和自回归模型 (例如 [29、28、26]) 允许直接使用最大似然估计 (分别是显式和隐式) 训练生成模型, 但高保真度的生成图像的数量尚未能够与最新的 GAN 模型相匹配 [13、14、3]。但是, GAN 训练存在两个突出的问题: (1) 模式坍塌

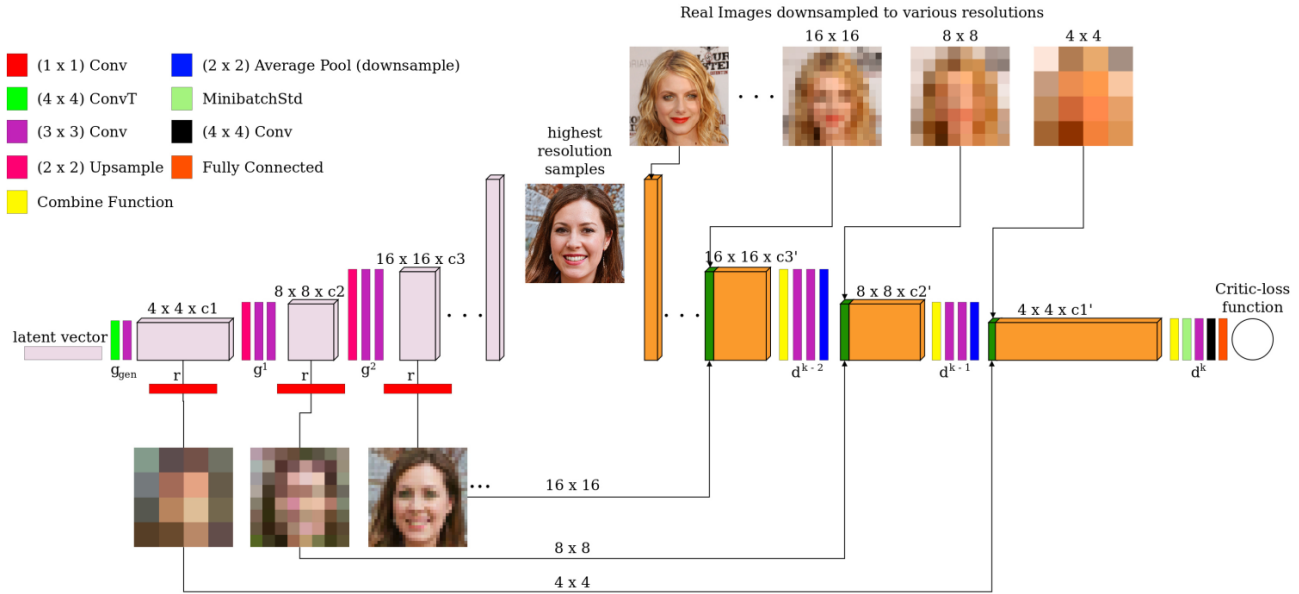


图 2: MSG-GAN 的体系结构, 此处显示为 ProGAN 中提出的基本模型[13]。我们的体系结构包括从生成器的中间层到判别器的中间层的连接。发送到判别器的多尺度图像与从卷积层主路径获得的相应激活值 (其后为连接函数) (以黄色显示) 连接在一起。

和 (2) 训练不稳定。

当生成器网络仅能够捕获数据分布中存在方差的子集时, 就会发生模式坍塌的问题。尽管已经提出了许多方法[25、36、13、18]来解决这个问题, 但它仍然是一个开放的研究领域。但是, 在这项工作中, 我们解决了训练不稳定的问题。这是 GAN 的一个基本问题, 在以前的著作中已广泛报道 [25、19、2、10、16、30、12、13、33、22]。我们提出了一种方法, 该方法通过研究如何使用多种尺度的梯度来生成高分辨率图像 (由于数据维数通常更具挑战性) 而无需依靠先前的贪婪方法, 例如渐进生长技术[13, 14], 从而解决图像生成任务的训练不稳定性。MSG-GAN 允许判别器不仅查看生成器的最终输出 (最高分辨率), 而且还查看中间层的输出 (图 2)。结果, 判别器成为生成器的多个尺度输出的函数, 重要的是, 将梯度同时传递给所有尺度 (第 1.1 节和第 2 节中有更多详细信息)。

此外, 我们的方法对不同的损失函数 (我们展示了 WGAN-GP 和非饱和 GAN 损失 (具有单侧梯度惩罚) 的结果), 数据集 (我们在广泛的常用数据集和最新创建的印度名人数据集上展示了结果具有鲁棒性) 和架构 (我们将 MSG 方法与 ProGAN

和 StyleGAN 作为基础架构进行结合)。就像渐进式增长 [13] 一样, 我们注意到, 多尺度梯度在 FID 得分方面比原始 DCGAN 体系结构有显著提高。但是, 我们的方法在大多数现有数据集上的训练时间与最新方法相当, 而无需逐步增长引入的额外超参数, 例如不同生成阶段的训练时间表和学习率。这种鲁棒性使得 MSG-GAN 方法可以很容易地在新数据集上“开箱即用”地使用, 在这种情况下, 使用与以前相同的超参数集会使基于渐进式增长的方法效果不佳 (请参见表 1 和 2)。我们还通过高分辨率 FFHQ 数据集上的消融实验显示了在所有分辨率上进行多尺度连接的重要性。

总而言之, 我们提出以下贡献:

1. 我们引入了一种用于图像合成的多尺度梯度技术, 该技术提高了先前工作中定义的训练的稳定性。
2. 我们证明, 我们能够在许多常用数据集上稳健地生成高质量样本, 包括 CIFAR10, Oxford102 花卉, CelebA-HQ, LSUN 教堂, Flickr Faces HQ 和我们的新印度名人 Celebs, 它们都具有相同的固定超参数。

1.1. 动机

Arjovsky 和 Bottou [1]指出 GAN 训练不稳定性的原因之一是由于当真实和虚假的分布空间之间存在实质性的重叠时, 从判别器到生成器的随机 (非信息性) 梯度传递导致的。自 GAN 诞生以来, 已针对此问题提出了许多解决方案。一个较早的示例提出将实例噪声添加到真实图像和伪图像中, 以使分布空间最小重叠[1, 27]。最近, Peng 等[22]提出了输入图像与判别器对这些输入图像的最深特征之间的互信息瓶颈, 称为变判判别器瓶颈 (VDB) [22], Karras 等人[13]提出了一种渐进式增长技术, 以添加分辨率更高的连续层。VDB 解决方案迫使判别器仅将注意力集中在图像的最明显特征上进行分类, 这可以视为实例噪声的自适应变体。我们的工作与 VDB 技术正交, 我们将对 MSG-GAN 和 VDB 的组合进行调查, 以供将来使用。

渐进式增长技术通过逐步使生成的图像的操作分辨率加倍来逐层训练 GAN, 从而解决了不稳定问题。每当将新层添加到训练中时, 它都会慢慢渐入进来, 以保留先前层的学习。从直觉上讲, 此技术可解决分布重叠问题, 因为它首先在较低分辨率上实现了良好的分布匹配, 而较低分辨率的数据维数较低, 然后使用这些先前训练的权重部分初始化了较高分辨率的训练, 重点是学习更好的细节。

尽管此方法能够生成最新的结果, 但由于要对每个数据集进行调整超参数 (包括不同的迭代次数, 学习率 (对于生成器和判别器[11]) 以及不同分辨率对应的衰减时间。

多尺度图像生成是一种成熟的技术, 伴随着在此任务流行之前的早已存在的深层网络方法[17, 31]。最近, 许多基于 GAN 的方法将高分辨率图像合成过程分解为较小的子任务[32、35、34、7、8、13]。例如, LR-GAN [32]使用单独的生成器为最终图像合成背景, 前景和合成器蒙版。GMAN 和 Stack-GAN 之类的作品分别采用了一个生成器和多个判别器来分别进行引导变化和多尺度生成[7、35、34]。而 MAD-GAN [8]是使用多种生成器来解决模式坍塌的问题, 通过使不同生成器在训练数据集中捕获不同的模式的方式训练这种多主体的任务设置。LapGAN [4]建立模型之间的差异

在使用单个生成器和用于不同尺度的多个判别器在图像的拉普拉斯金字塔的生成的多比例分量之间进行比较。

我们提出的方法从所有这些作品中汲取了构建灵感, 并以它们的指引和意识形态为基础, 但有一些关键区别。在 MSG-GAN 中, 我们使用单个判别器和带有多尺度连接的单个生成器, 从而允许梯度同时以多种分辨率流动。我们提议的方法有几个优点 (主要由简单性决定)。如果在每个分辨率下使用多个判别器[35、34、7、4], 则由于需要重复的下采样层, 因此总参数在各个尺度上呈指数增长, 而在 MSG-GAN 中, 该关系是线性的。除了需要较少的参数和设计选择之外, 我们的方法还避免了跨多个尺度生成的图像上需要显式的颜色一致性正则化项, 这是必要的, 例如在 StackGAN [34]中。

2. 多尺度梯度 GAN

我们将 MSG-GAN 框架应用于两个基本架构 ProGANs [13]和 Style-GAN [14]上进行了实验。我们分别将这两种方法称为 MSG-ProGAN 和 MSG-StyleGAN。尽管有名称, 但在任何 MSG 变体中都没有使用渐进式增长, 并且我们注意到没有渐进式增长的 ProGAN 本质上是 DCGAN [23]架构。图 2 显示了我们的 MSG-ProGAN 架构的概述, 我们将在本节中对其进行详细定义, 并将 MSG-StyleGAN 模型的详细信息包括在补充材料中。

将生成器函数 g_{gen} 的初始块定义为 $g_{gen} : Z \mapsto A_{begin}$, 这样 $Z = \mathbb{R}^{512}$, $Z \sim N(0, \mathbb{I})$ 和 $A_{begin} = \mathbb{R}^{4 \times 4 \times 512}$ 包含 $[4 \times 4 \times 512]$ 维激活。令 g^i 为一个通用函数, 它充当基本的生成器块, 在我们的实现中, 该函数包括一个上采样操作, 后跟两个转换层。

$$g^i : A_{i-1} \mapsto A_i \quad (1)$$

$$\text{where, } A_i = \mathbb{R}^{2^{i+2} \times 2^{i+2} \times c} \quad (2)$$

$$\text{and, } i \in \mathbb{N}; A_0 = A_{begin} \quad (3)$$

其中 c 是生成器的中间激活的通道数。我们在补充材料中提供所有图层中 c 的大小。完整的生成器 $GEN(z)$ 然后遵循标准格式, 并且可以定义为 k 个这样的 g 函数的合成的序列, 然后是具有 g_{gen} 的最终合成:

$$y' = GEN(z) = g^k \circ g^{k-1} \circ \dots \circ g^i \circ \dots \circ g^1 \circ g_{gen}(z). \quad (4)$$

现在, 我们定义函数 r , 该函数在生成器的不同阶段生成输出 (图 2 中的红色框),

其中输出对应于最终输出图像的不同下采样版本。我们将 r 简单地建模为 (1x1) 卷积, 将中间卷积激活值转换为图像。

$$r^i : A_i \mapsto O_i \quad (5)$$

$$\text{where, } O_i = \mathbb{R}_{[0-1]}^{2^{i+2} \times 2^{i+2} \times 3} \quad (6)$$

$$\text{hence, } r^i(g^i(z)) = r^i(a_i) = o_i. \quad (7)$$

换句话说, O_i 是从生成器的第 i 个中间层的输出合成的图像。类似于渐进式增长[13]的想法, r 充当正则化器, 要求学习的特征图能够直接投影到 RGB 空间。

现在我们继续定义判别器。由于判别器的最终判别损失不仅是生成器 y' 的最终输出的函数, 而且是中间输出 O_i 的函数, 因此梯度可以从判别器的中间层流到生成器的中间层。我们用字母 d 表示判别函数的所有组件。我们将判别器的最后一层 (提供判别分数) 命名为 $d_{critic}(z')$, 并将定义判别式 $d^0(y)$ 或 $d^0(y')$ 的第一层的函数命名为实像 y (真实样本) 或最高分辨率的合成图像 y' (假样本) 作为输入。类似地, 令 d^j 代表判别器的中间层函数。注意, 当 $j = k - i$ 时, i 和 j 总是彼此相关。因此, 判别器的第 j 个中间层的输出激活值 a'_j 定义为:

$$a'_j = d^j(\phi(o_{k-j}, a'_{j-1})) \quad (8)$$

$$= d^j(\phi(o_i, a'_{j-1})), \quad (9)$$

其中 ϕ 是将生成器的第 (i) 个中间层的输出 O_i (或相应的最高分辨率实像 y 的下采样版本) 与在判别器中第 $(j - 1)$ 中间层的相应输出进行组合的函数。在我们的实验中, 我们实验了此连接函数的三个不同变体:

$$\phi_{simple}(x_1, x_2) = [x_1; x_2] \quad (10)$$

$$\phi_{lin_cat}(x_1, x_2) = [r'(x_1); x_2] \quad (11)$$

$$\phi_{cat_lin}(x_1, x_2) = r'([x_1; x_2]) \quad (12)$$

其中, r' 是又一个 (1x1) 卷积运算, 类似于 r , 并且 $[\cdot]$ 是一个简单的通道级联操作。我们将在第 4 节中比较这些不同的连接函数。

最终的判别函数定义为:

$$DIS(y', o_0, o_1, \dots, o_i, \dots, o_{k-1}) = \quad (13)$$

$$d_{critic} \circ d^k(\cdot, o_0) \circ d^{k-1}(\cdot, o_1) \circ \dots \circ d^j(\cdot, o_i) \circ \dots \circ d^0(y') \quad (14)$$

我们为 d_{critic} 函数试验了两种不同的损失函数, 即 ProGAN [13] 使用的 WGAN-GP [10] 和 StyleGAN [14] 使用的 1-sided GP [9, 20] 的非饱和 GAN 损失。请注意, 由于判别器现在是生成器生成的多个输入图像的函数, 因此我们将梯度惩罚修改为每个输入的惩罚平均值。

3. 实验

虽然评估 GAN 生成的图像的质量并不是一件容易的事, 但当今最常用的指标是初始得分 (IS, 越高越好) [25] 和 Frechet' 初始距离 (FID, 越低越好) [11]。为了将我们的结果与以前的工作进行比较, 我们将 IS 用于 CIFAR10 实验, 将 FID 用于其余实验, 并报告“显示的真实图像数量”, 如先前的工作[13, 14]所示。

新的印度名人数据集 除了现有的数据集外, 我们还收集了一个由印度名人组成的新数据集。我们采购新数据集的目的是尝试使用很小的数据集 (以图像数为单位), 因为 GAN 社区已经表明, 数据集的大小对于创造良好的生成模型很重要[14]。为此, 我们使用类似于 CelebA-HQ 的过程收集了图像。首先, 我们通过抓取相关搜索查询的网页来下载印度名人的图像。然后, 我们使用现成的面部检测器检测面部, 并裁剪所有图像并将其调整为 256x256。最后, 我们通过滤除低质量, 错误和低光照的图像来手动清理图像。最后, 数据集仅包含 3K 个样本, 比 CelebA-HQ 小一个数量级。该数据集将被公开以供研究。

3.1 实施细节

我们在分辨率和大小 (图像数量) 不同的各种数据集上评估我们的方法。CIFAR10 (60K 图像, 分辨率为 32x32); Oxford flowers (8K 图像, 分辨率为 256x256), LSUN 教堂 (126K 图像, 分辨率为 256x256), 印度名人 (3K 图像, 分辨率为 256x256), CelebA-HQ (3 万图像, 分辨率为 1024x1024) 和 FFHQ (70K 图像, 分辨率为 1024x1024)。

对于每个数据集, 我们使用相同的初始潜在维数 512, 从标准正态分布 $N(0, I)$ 提取, 然后进行超球面归一化 [13]。对于所有实验, 我们对 MSG-ProGAN 和 MSG-StyleGAN 使用相同的超参数设置 ($lr = 0.003$), 唯一的区别是上采样层数 (对于较低分辨率的数据集更少)。

所有模型都使用 RMSprop 进行了训练, 生成器和判别器的学习率均为 0.003。我们



(a) LSUN churches

(b) Indian Celebs

(c) Oxford Flowers

图 3: 由 MSG-StyleGAN 在不同的中分辨率 (256x256) 数据集上生成的随机, 未整理的样本。我们的方法可在具有相同超参数的所有数据集上生成高质量的结果。在屏幕上放大观看最佳。

Dataset	Size	Method	# Real Images	GPUs used	Training Time	FID (\downarrow)
Oxford Flowers (256x256)	8K	ProGANs*	12M	4 GTX1080-8GB	75 hrs	58.60
		MSG-ProGAN	1.7M	1 V100-32GB	44 hrs	28.27
		StyleGAN*	7.2M	2 V100-32GB	33 hrs	64.70
		MSG-StyleGAN	1.6M	2 V100-32GB	16 hrs	19.60
Indian Celebs (256x256)	3K	ProGANs*	9M	2 V100-32GB	37 hrs	67.49
		MSG-ProGAN	2M	2 V100-32GB	34 hrs	36.72
		StyleGAN*	6M	4 V100-32GB	18 hrs	61.22
		MSG-StyleGAN	1M	4 V100-32GB	7 hrs	28.44
LSUN Churches (256x256)	126K	StyleGAN*	25M	8 V100-16GB	47 hrs	6.58
		MSG-StyleGAN	24M	8 V100-16GB	50 hrs	5.2

表 1: 中分辨率 (即 256x256) 数据集的实验。我们会尽可能使用作者提供的分数, 否则将使用官方代码和建议的超参数 (表示为“*”) 来训练模型

根据标准正态分布 $N(0, 1)$ 初始化生成器和判别器的参数。为了与先前发表的工作相匹配, 所有 StyleGAN 和 MSG-StyleGAN 模型都通过 1-sided GP 的非饱和 GAN 损失进行了训练, 而 ProGAN 和 MSG-ProGAN 模型则通过 WGAN-GP 损失函数进行了训练。

我们还扩展了 MinBatchStdDev 技术[13, 14], 将一批激活的平均标准偏差反馈到判别器, 以改善样本多样性, 从而达到我们的多尺度设置。为此, 我们在判别器中每个块的开头添加一个单独的 MinBatchStdDev 层。这样, 判别器获得了所生成样本的批次统计以及每个尺度上的直线路径激活, 并且可以检测到生成器在某种程度上的模式坍塌。

当我们自己训练模型时, 我们会报告训练时间和使用的 GPU, 并尽可能尝试

使用相同的机器, 以便可以进行直接的训练时间比较 (除 Oxford Flowers Pro-GAN 与 MSG-ProGAN 之外的所有情况)。显示的实际图像数量和训练时间的差异是由于以下事实: 按照惯例, 我们报告了在固定数量的迭代中获得的最佳 FID 得分, 以及达到该得分所花费的时间。复制我们的作品所需的所有代码和受训练的模型可用于研究目的, 在以下获得: <https://github.com/akanimax/msg-stylegan-tf>。

3.2. 结果

质量 表 1 显示了我们的方法在各种中分辨率数据集上的定量结果。在 Oxford Flowers, LSUN 教堂和印度名人的 (256x256) 分辨率数据集上, 我们的 MSG-ProGAN 和 MSG-StyleGAN 模型均获得了比 ProGAN 和 StyleGAN 基线更高的 FID 分数。虽然



(a) CelebA-HQ

(b) FFHQ

图 4: MSG-StyleGAN 在高分辨率 (1024x1024) 数据集上生成的随机, 未整理的样本。在屏幕上放大观看最佳。

Dataset	Size	Method	# Real Images	GPU Used	Training Time	FID (↓)
CelebA-HQ (1024x1024)	30K	ProGANs [14]	12M	-	-	7.79
		MSG-ProGAN	3.2M	8 V100-16GB	1.5 days	8.02
		StyleGAN [14]	25M	-	-	5.17
		MSG-StyleGAN	11M	8 V100-16GB	4 days	6.37
FFHQ (1024x1024)	70K	ProGANs*	12M	4 V100-32GB	5.5 days	9.49
		ProGANs [13]	12M	-	-	8.04
		MSG-ProGAN	6M	4 V100-32GB	6 days	8.36
		StyleGAN*	25M	4 V100-32GB	6 days	4.47
		StyleGAN [14]	25M	-	-	4.40
		MSG-StyleGAN	9.6M	4 V100-32GB	6 days	5.8

表 2: 针对高分辨率 (1024x1024) 数据集的实验。我们会尽可能使用作者提供的分数, 并使用其他带有官方代码和建议的超参数 (表示为“*”) 的训练模型。

由于所有层都一起训练, 因此 MSG-GAN 的每次迭代速度较慢, 但是它趋于在更少的迭代次数中收敛, 从而需要更少的 GPU 训练总时间来达到这些分数。图 3 显示了在这些数据集上生成的用于定性评估的随机样本。

对于高分辨率实验 (表 2), MSG-ProGAN 模型训练的时间相当, 并且在 CelebA-HQ 和 FFHQ 数据集上分别获得了相似分数 (8.02 vs 7.79) 和 (8.36 vs 8.04)。我们注意到作者报告的分数与作者提供的代码所能达到的成绩略有不同,

这可能是由于细微的硬件差异或运行之间的差异。我们的 MSG-StyleGAN 模型无法在 CelebA-HQ 数据集 (6.37 vs 5.17) 和 FFHQ 数据集 (5.8 vs 4.40) 上超过 StyleGAN 的 FID 得分。我们讨论了为什么会出现在第 4 节中的一些假设, 但是请注意, 我们的方法还有其他优点, 即, 如我们的其他实验所示, 将其更好地归因于不同的数据集似乎更容易。

训练期间的稳定性 为了比较 MSG-ProGAN 和 ProGAN 在训练期间的稳定性, 我们测量了

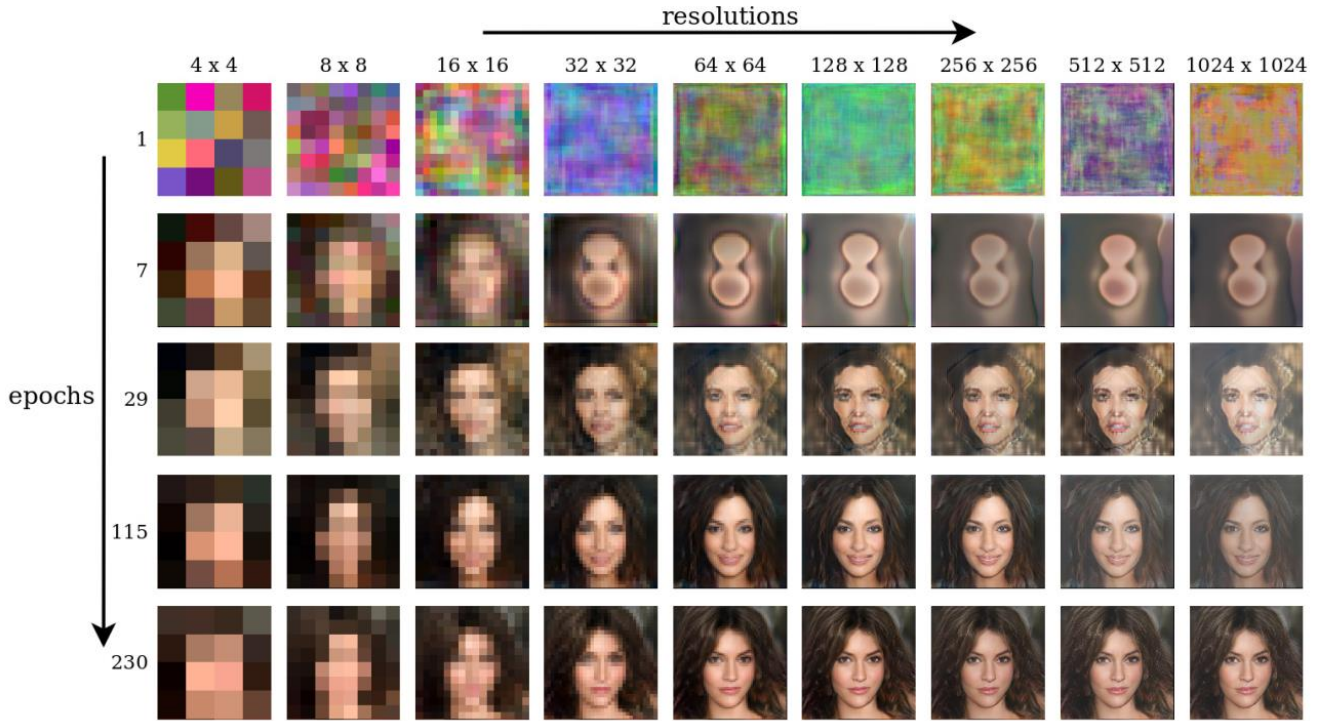


图 5: 在训练过程中, MSG-GAN 中的所有层在训练的早期就跨生成的分辨率进行了同步, 随后同时所有比例下提高了生成图像的质量。在整个训练过程中, 生成器仅对固定潜码生成的图像进行了最小的增量改进。

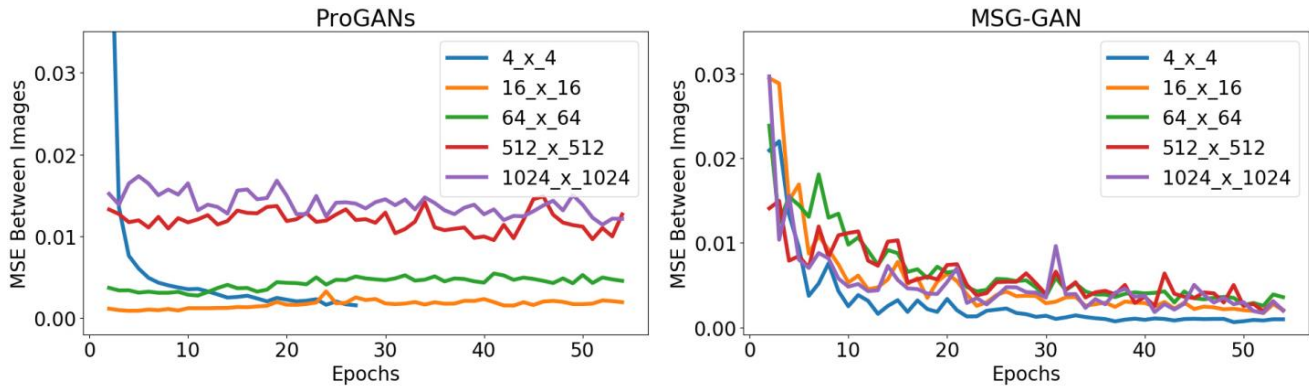


图 6: 训练期间的图像稳定性。这些图显示了 CelebA-HQ 数据集上连续迭代开始时 (取 36 个潜码样本的平均值) 从相同潜码生成的图像之间的 MSE。MSG-ProGAN 会随着时间稳定收敛, 而 ProGANs [13] 在各个时期仍会持续变化很大。

对于相同的固定潜码, 随着迭代的进行, 生成的样本中的变化 (CelebA-HQ 数据集)。这种方法是 [33] 引入的一种在训练过程中测量稳定性的方法, 该方法通过计算两个连续样本之间的均方误差来量化。图 6 显示, 尽管 ProGAN 仅在较低分辨率下趋于收敛 (变化较小), 但 MSG-ProGAN 对所有分辨率均显示出相同的收敛特性。针对 ProGAN 的训练迭代会在每种分辨率下依次进行, 而针对 MSG-ProGAN

它们是同时发生的 (图 5)。尽管不一定确保产生良好的结果, 但是具有高稳定性的方法可能会具有优势, 因为它更容易通过在训练过程中可视化快照来获得最终结果的合理估计, 这将有助于训练工作将几周的时间压缩到几天。

学习率的鲁棒性 先前的工作 [25, 12, 21, 20] 以及我们的经验表明, 训练过程中 GAN 的收敛非常依赖

Method	# Real Images	Learning rate	IS (\uparrow)
Real Images	-	-	11.34
MSG-ProGAN	12M	0.003	8.63
MSG-ProGAN	12M	0.001	8.24
MSG-ProGAN	12M	0.005	8.33
MSG-ProGAN	12M	0.01	7.92

表 3: CIFAR-10 的学习率稳健性。我们看到, 在一系列学习率范围内, 我们的方法收敛到相似的 IS 分数。

Level of Multi-scale connections	FID (\downarrow)
No connections (DC-GAN)	14.20
Coarse Only	10.84
Middle Only	9.17
Fine Only	9.74
All (MSG-ProGAN)	8.36
ProGAN*	9.49

表 4: 高分辨率 (1024x1024) FFHQ 数据集上不同程度的多尺度梯度连接的消融实验。粗略包含 (4x4) 和 (8x8) 的连接, 中间包含 (16x16) 和 (32x32) 的连接; 并以 (64x64) 精细化直到 (1024x1024)。

Method	Combine function	FID (\downarrow)
MSG-ProGAN	$\phi_{lin.cat}$	11.88
	$\phi_{cat.lin}$	9.63
	ϕ_{simple}	8.36
MSG-StyleGAN	ϕ_{simple}	6.46
	$\phi_{lin.cat}$	6.12
	$\phi_{cat.lin}$	5.80

表 5: 对高分辨率 (1024x1024) FFHQ 数据集使用不同连接函数的实验。

在选择超参数方面, 学习率尤其重要。为了验证 MSG-ProGAN 对超参数变化的鲁棒性, 我们针对 CIFAR-10 数据集以四种不同的学习率 (0.001、0.003、0.005 和 0.01) 训练了网络 (表 3)。我们可以看到, 即使学习率发生较大变化, 这四个模型也会收敛, 产生出清晰的图像和相似的初始分数。健壮的训练方案非常重要, 因为它们表明可以轻松地将一种方法推广到未曾见过的数据集上。

4. 讨论

消融研究 我们对 MSG-ProGAN 架构进行了两种消融。

表 4 总结

我们在消融版本的多尺度渐变中进行的实验, 其中我们仅将生成器中的连接子集添加到不同尺度的判别器上。我们可以看到, 在原始 ProGANs / DCGAN 体系结构的任何级别上添加多尺度梯度已经可以改善 FID 得分。有趣的是, 仅添加中间级别的连接要比仅添加粗糙或精细级别的连接稍好, 但是在所有级别都存在连接的情况下, 可以实现总体最佳性能。

表 5 展示了我们在 MSG-ProGAN 和 MSG-StyleGAN 体系结构上使用连接函数 ϕ 的不同变体的实验。 ϕ_{simple} (Eq 10) 在 MSG-ProGAN 架构上表现最佳, 而 $\phi_{cat.lin}$ (Eq 12) 在 MSG-StyleGAN 架构上表现最佳。本工作中显示的所有结果均采用了这些各自的连接函数。

局限性和未来工作 我们的方法并非没有局限性。我们注意到, 使用渐进式训练, 以较低的分辨率进行的第一组迭代要快得多, 而 MSG-GAN 的每次迭代花费的时间相同。但是, 我们观察到 MSG-GAN 需要更少的总迭代次数才能达到相同的 FID, 并且通常在总训练时间相似的情况下才这样做。

最后, 我们注意到在 FFHQ 和 CelebA-HQ 的面部数据集上, 我们没有超过 StyleGAN 的生成质量。造成这种情况的原因可能有很多, 包括不正确的超参数选择, 或者 StyleGANs 架构更适合这些数据集。另外, 由于我们在 MSG-StyleGAN 中进行了多尺度修改, 因此我们的方法无法利用混合正则化技巧[14], 在混合技巧中, 将多个潜在向量进行混合, 并且所生成的图像被判别器逼真。这样做是为了允许在测试时在不同级别混合不同样式, 同时也提高了整体质量。有趣的是, 即使我们没有明确强制执行混合正则化, 我们的方法仍然能够产生合理的混合结果 (请参阅补充材料)。尽管没有提高 FFHQ 上的 FID 分数, 但我们的方法在其他数据集上的分数更高, 并且引入了一种易于使用的高分辨率合成新方法, 可能会激发后续工作, 从而进一步提高结果质量。

结论 尽管在迈向逼真的高分辨率图像合成方面已取得了长足的进步 [3, 14], 但仍未实现真正的逼真的图像, 特别是在外观方面存在很大差异的领域。在这项工作中, 我们介绍了 MSG-GAN 技术, 该技术通过一种简单的方法为这些工作做出了贡献, 从而可以利用 GAN 生成高分辨率的多尺度图像。

5. 致谢

我们要感谢 Alexia Jolicoeur-Martineau (美国 MILA 的博士研究生) 对 GAN 中理论的指导以及对本文的校对提供了指导。我们也非常感谢 TomTom 和 Adobe 向我们提供 GPU 资源。最后, 我们还要特别感谢 Michael Hoffman (TomTom 高级软件工程经理) 的支持和鼓励。

参考文献

- [1] Mart'ın Arjovsky and Leon' Bottou. Towards principled meth-ods for training generative adversarial networks. CoRR, 2017.
- [2] Mart'ın Arjovsky, Soumith Chintala, and Leon' Bottou. Wasserstein generative adversarial networks. In ICML, 2017.
- [3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthe-sis. In International Conference on Learning Representa-tions, 2019.
- [4] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In NIPS, 2015.
- [5] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. CoRR, 2014.
- [6] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. CoRR, 2016.
- [7] Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks. arXiv preprint arXiv:1611.01673, 2016.
- [8] Arnab Ghosh, Viveka Kulharia, Vinay P. Nambodiri, Philip H.S. Torr, and Puneet K. Dokania. Multi-agent di-verse generative adversarial networks. In CVPR, June 2018.
- [9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, 2014.
- [10] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein GANs. In Advances in Neural Information Pro-cessing Systems, 2017.
- [11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equi-librium. In Advances in Neural Information Processing Sys-tems, 2017.
- [12] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard GAN. In International Conference on Learning Representations, 2019.
- [13] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In ICLR, 2018.
- [14] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In CVPR, pages 4401–4410, 2019.
- [15] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In Advances in Neural Information Processing Systems, 2018.
- [16] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. On convergence and stability of GANs. arXiv preprint arXiv:1705.07215, 2017.
- [17] Sylvain Lefebvre and Hugues Hoppe. Parallel controllable texture synthesis. In ACM Transactions on Graphics (ToG), volume 24. ACM, 2005.
- [18] Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. PacGAN: The power of two samples in generative adversar-ial networks. In Advances in Neural Information Processing Systems, 2018.
- [19] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, and Zhen Wang. Multi-class generative adversarial networks with the l2 loss function. ArXiv, abs/1611.04076, 2016.
- [20] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Which training methods for GANs do actually converge? In ICML, 2018.
- [21] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks, 2016.
- [22] Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. Variational discriminator bottleneck: Im-proving imitation learning, inverse RL, and GANs by con-straining information flow. In ICLR, 2019.
- [23] Alec Radford, Luke Metz, and Soumith Chintala. Un-supervised representation learning with deep convolu-tional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.
- [24] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In ICML, 2015.
- [25] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In Advances in neural information pro-cessing systems, 2016.
- [26] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. PixelCNN++: A PixelCNN implementation with discretized logistic mixture likelihood and other modifica-tions. In ICLR, 2017.
- [27] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wen-zhe Shi, and Ferenc Huszar'. Amortised MAP inference for image super-resolution. ArXiv, abs/1610.04490, 2016.
- [28] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In Advances in Neural Information Processing Systems, 2016.
- [29] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. CoRR, abs/1601.06759, 2016.
- [30] Ruohan Wang, Antoine Cully, Hyung Jin Chang, and Yiannis Demiris. Magan: Margin adaptation for generative adversar-ial networks. ArXiv, abs/1704.03817, 2017.
- [31] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time completion of video. IEEE Transactions on Pattern Analysis & Machine Intelligence, (3), 2007.
- [32] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. Lr-gan: Layered recursive generative adversarial net-works for image generation. ICLR, 2017.

Block	Operation	Act.	Output Shape
1.	Latent Vector	Norm	512 x 1 x 1
	Conv 4 x 4	LReLU	512 x 4 x 4
	Conv 3 x 3	LReLU	512 x 4 x 4
2.	Upsample	-	512 x 8 x 8
	Conv 3 x 3	LReLU	512 x 8 x 8
	Conv 3 x 3	LReLU	512 x 8 x 8
3.	Upsample	-	512 x 16 x 16
	Conv 3 x 3	LReLU	512 x 16 x 16
	Conv 3 x 3	LReLU	512 x 16 x 16
4.	Upsample	-	512 x 32 x 32
	Conv 3 x 3	LReLU	512 x 32 x 32
	Conv 3 x 3	LReLU	512 x 32 x 32
Model 1 ↑			
5.	Upsample	-	512 x 64 x 64
	Conv 3 x 3	LReLU	256 x 64 x 64
	Conv 3 x 3	LReLU	256 x 64 x 64
6.	Upsample	-	256 x 128 x 128
	Conv 3 x 3	LReLU	128 x 128 x 128
	Conv 3 x 3	LReLU	128 x 128 x 128
Model 2 ↑			
7.	Upsample	-	128 x 256 x 256
	Conv 3 x 3	LReLU	64 x 256 x 256
	Conv 3 x 3	LReLU	64 x 256 x 256
Model 3 ↑			
8.	Upsample	-	64 x 512 x 512
	Conv 3 x 3	LReLU	32 x 512 x 512
	Conv 3 x 3	LReLU	32 x 512 x 512
9.	Upsample	-	32 x 1024 x 1024
	Conv 3 x 3	LReLU	16 x 1024 x 1024
	Conv 3 x 3	LReLU	16 x 1024 x 1024
Model full ↑			

表 6: 用于训练的 MSG-ProGAN 模型的生成器体系架构。

Block	Operation	Act.	Output Shape	
Model full ↓				
1.	Raw RGB images 0	-	3 x 1024 x 1024	
	FromRGB 0	-	16 x 1024 x 1024	
	MinBatchStd	-	17 x 1024 x 1024	
	Conv 3 x 3	LReLU	16 x 1024 x 1024	
	Conv 3 x 3	LReLU	32 x 1024 x 1024	
2.	AvgPool	-	32 x 512 x 512	
	Raw RGB images 1	-	3 x 512 x 512	
	Concat/ ϕ_{simple}	-	35 x 512 x 512	
	MinBatchStd	-	36 x 512 x 512	
	Conv 3 x 3	LReLU	32 x 512 x 512	
3.	Conv 3 x 3	LReLU	64 x 512 x 512	
	Conv 3 x 3	LReLU	64 x 256 x 256	
	AvgPool	-	64 x 256 x 256	
	Model 3 ↓			
	3.	Raw RGB images 2	-	3 x 256 x 256
Concat/ ϕ_{simple}		-	67 x 256 x 256	
MinBatchStd		-	68 x 256 x 256	
Conv 3 x 3		LReLU	64 x 256 x 256	
Conv 3 x 3		LReLU	128 x 256 x 256	
4.	AvgPool	-	128 x 128 x 128	
	Model 2 ↓			
	4.	Raw RGB images 3	-	3 x 128 x 128
		Concat/ ϕ_{simple}	-	131 x 128 x 128
		MinBatchStd	-	132 x 128 x 128
Conv 3 x 3		LReLU	128 x 128 x 128	
Conv 3 x 3		LReLU	256 x 128 x 128	
5.	AvgPool	-	256 x 64 x 64	
	5.	Raw RGB images 4	-	3 x 64 x 64
		Concat/ ϕ_{simple}	-	259 x 64 x 64
		MinBatchStd	-	260 x 64 x 64
		Conv 3 x 3	LReLU	256 x 64 x 64
Conv 3 x 3		LReLU	512 x 64 x 64	
6.	AvgPool	-	512 x 32 x 32	
	Model 1 ↓			
	6.	Raw RGB images 5	-	3 x 32 x 32
		Concat/ ϕ_{simple}	-	515 x 32 x 32
		MinBatchStd	-	516 x 32 x 32
Conv 3 x 3		LReLU	512 x 32 x 32	
Conv 3 x 3		LReLU	512 x 32 x 32	
7.	AvgPool	-	512 x 16 x 16	
	7.	Raw RGB images 6	-	3 x 16 x 16
		Concat/ ϕ_{simple}	-	515 x 16 x 16
		MinBatchStd	-	516 x 16 x 16
		Conv 3 x 3	LReLU	512 x 16 x 16
Conv 3 x 3		LReLU	512 x 16 x 16	
8.	AvgPool	-	512 x 8 x 8	
	8.	Raw RGB images 7	-	3 x 8 x 8
		Concat/ ϕ_{simple}	-	515 x 8 x 8
		MinBatchStd	-	516 x 8 x 8
		Conv 3 x 3	LReLU	512 x 8 x 8
Conv 3 x 3		LReLU	512 x 8 x 8	
9.	AvgPool	-	512 x 4 x 4	
	9.	Raw RGB images 7	-	3 x 4 x 4
		Concat/ ϕ_{simple}	-	515 x 4 x 4
		MinBatchStd	-	516 x 4 x 4
		Conv 3 x 3	LReLU	512 x 4 x 4
Conv 4 x 4		LReLU	512 x 1 x 1	
Fully Connected	Linear	1 x 1 x 1		

表 7: 用于训练的 MSG-ProGAN 和 MSG-StyleGAN 模型的判别器体系架构。

- [33] Yasin Yazıcı, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, and Vijay Chandrasekhar. The unusual effectiveness of averaging in GAN training. In International Conference on Learning Representations, 2019.
- [34] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, XiaoLei Huang, and Dimitris Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. arXiv: 1710.10916, 2017.
- [35] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, XiaoLei Huang, and Dimitris N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In ICCV, Oct 2017.
- [36] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In Advances in Neural Information Processing Systems, 2017.

6. 附录

6.1. 架构细节

MSG-ProGAN 表 6 和表 7 分别提供了 MSG-ProGAN 生成器和判别器的详细配置。在生成器中的每个块之后,

1 x 1 转换层用于将输出激活值转换为 RGB 图像, 然后将其传递到判别器上。在判别器方面, 这些 RGB 图像使用连接函数 ϕ 与直线路径激活值结合在一起。在 ϕ_{simple} 的情况下, 将使用简单的逐级串联操作 (请参见表 7)。对于连接函数的 $\phi_{cat.lin}$ 变体, 使用 1 x 1 的转换层将 RGB 图像投影到激活空间中, 然后进行逐级连接操作。1 x 1 转换层输出的通道数等于判别器该块中输出通道的一半, 例如对于 block 3 (请参见表 7), 1 x 1 转换层的输出为 32 x 256 x 256, 线性运算的输出为 96 x 256 x 256 (32 + 64)。最后, 对于 $\phi_{cat.lin}$, 首先将 RGB 图像与直线路径激活值连接起来, 然后是 1 x 1 的转换层。由该 1x1 转换层输出的通道数再次等于该块中的通道数 (例如, block 3 为 64)。

生成器 (表 6) 的模型 1, 模型 2 和模型 3 块分别用于合成 32 x 32、128 x 128 和 256 x 256 尺寸的图像。并且, 每进行 3 x 3 卷积操作后, 就会根据 PixNorm [13] 方案 (仅针对生成器) 对特征向量进行归一化。

MSG-StyleGAN MSG-StyleGAN 模型使用由 StyleGAN [14] 对 ProGANs [13] 体系结构提出的所有修改, 除了混合正则化。与 MSG-ProGAN 类似, 我们使用 1 x 1 转换层获取 Style-GAN 生成器每个块的 RGB 图像输出, 而其他所有内容 (映射网络, 非传统输入和样式 AdaIN) 均保持不变。判别器架构与 ProGAN (以及因此产生的 MSG-ProGAN, 表 7) 判别器相同。

6.2. 其他定性结果

在这里, 我们包括其他结果, 用于进一步的经验验证。我们针对 256 x 256 Oxford102 花卉数据集显示了 MSG-StyleGAN 的完整分辨率结果, 针对 128 x 128 CelebA 和 LSUN 卧室数据集显示了 MSG-ProGAN 体系架构。CelebA 模型经过训练, 可获取 2 千 8 百万张真实图像, 并且 FID 为 8.86。由于 LSUN 卧室数据集庞大 (3000 万个), 我们对其进行了训练, 以获取 1.5 亿个真实图像 (大约 5 个迭代), 从而 FID 为 18.32。图 12 和 13 分别显示了为 CelebA 和 LSUN 卧室数据集生成的 128 x 128 (最高分辨率) 样本。图 10 和图 11 分别显示了 MSG-StyleGAN 模型在所有分辨率下在 Oxford Flowers 和 Cifar-10 数据集上生成的样本。图 14 显示了 CelebA-HQ 数据集的其他定性结果 (随机样本), 使用我们的完整模型模型进行了训练

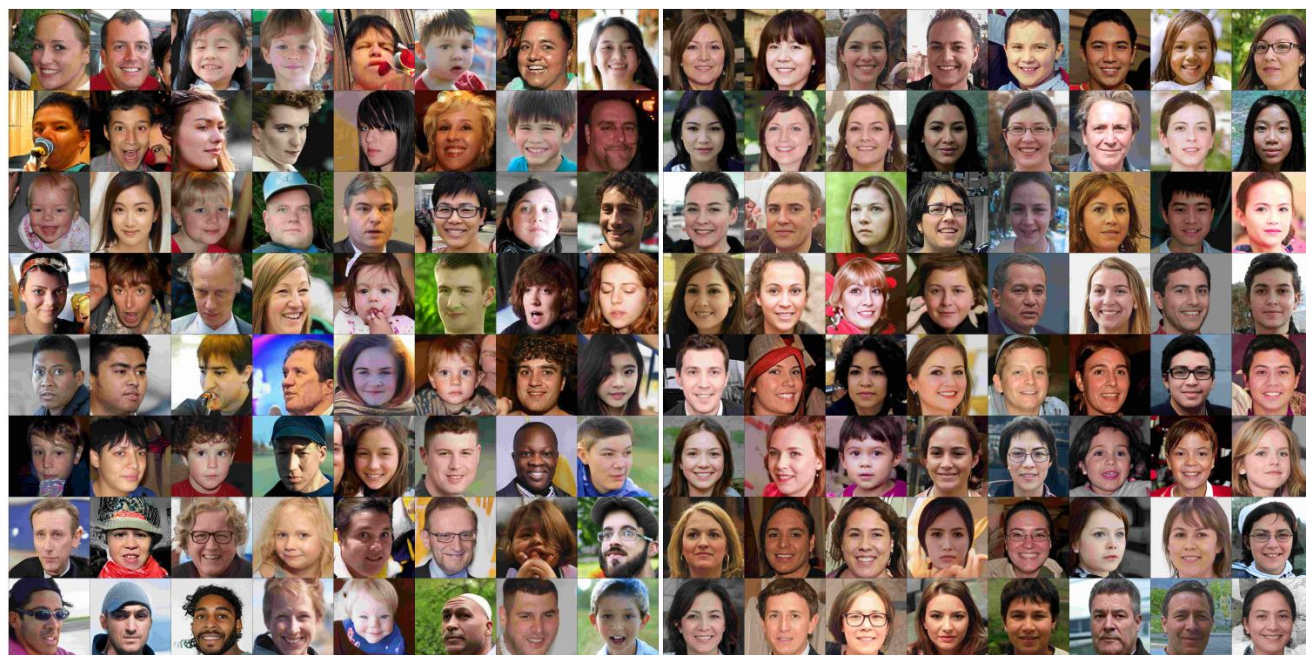
在 1024*1024 的分辨率上。

6.3. 观察

在本节中, 我们介绍一些有关我们的方法和 StyleGAN 产生的结果差异的观察和假设。我们在图 7 中显示了从两个模型中随机选择的样本的概述。在对结果的分析中, 我们发现, 尽管实际结果图像质量非常接近, 但 StyleGAN 样本在姿态方面表现出稍高的变化。相比之下, MSG-StyleGAN 的结果在全局上更一致, 更现实。多样性和结果质量之间的这种折衷被广泛报道过[36], 并且可以解释 FID 分数的某些差异。进一步研究控制任一轴的方法 (现实性与多样性) 及其对 FID 分数的影响, 将是未来工作的有趣方向。

我们还进行了实验, 研究了添加到 Style-GAN 生成器每个块中的像素噪声在图像生成中的作用。我们发现, 在非人脸数据集上, 这些噪声层不仅为随机变化建模, 而且还为图像的语义方面建模, 就像它们的最初意图一样[14] (见图 8)。我们观察到, MSG-StyleGAN 也显示出这种类型的效果, 尽管程度较小。我们推测, 对于面部建模任务 (例如, 在 CelebA-HQ 和 FFHQ 数据集上), 随机特征和语义特征之间的这种纠缠更为直接, 并且我们还在脸部和非脸部数据集上观察到不同的模型对此噪声的敏感性可能会导致某些性能差异。

如主文讨论部分所述, 我们不使用 StyleGAN [14] 中描述的混合正则化技术 (如何集成这样的正则化问题是未来工作的一个有趣方向)。但是, 我们注意到, 尽管没有使用它, 但是由于基于比例的约束, 模型仍然学会解开图像的高级语义特征 (见图 9)。从图中可以明显看出, 高级混合更加连贯, 并产生了更加逼真的视觉效果。而较低级别的混合通常会产生错误的视觉提示, 例如照明不正确和头发不平衡。这表明, 通过确保在低 (粗粒度) 生成水平上进行适当的基于样式的混合, 可能会提高性能。



(a) StyleGAN 生成图像

(b) MSG-StyleGAN 生成图像

图 7: 随机生成的样本, 用于 StyleGAN [14]和 MSG-StyleGAN 之间的定性比较。因为 FID 计算是在非截断的潜在空间上完成的, 所以生成的所有样本都不会截断输入的潜在空间。观看放大最佳。



图 8: 由 StyleGAN (顶部) 和 MSG-StyleGAN (底部) 使用每像素噪声的不同实现方式生成的 LSUN Church 图像, 同时保持输入潜码不变。



图 9: 通过将来自两个不同潜向量的样式以不同的生成级别 (粒度) 混合而生成的图像。与 StyleGAN [14] 中一样, 第一列图像是源 1, 第一行图像是源 2。编号 2、3 和 4 的行以分辨率 (4 x 4 和 8 x 8) 进行混合, 而 5 和 6 的行分辨率为 (16 x 16 和 32 x 32), 并通过以分辨率交换源 2 潜码来生成第 6 行图像 (64 x 64 至 1024 x 1024)。

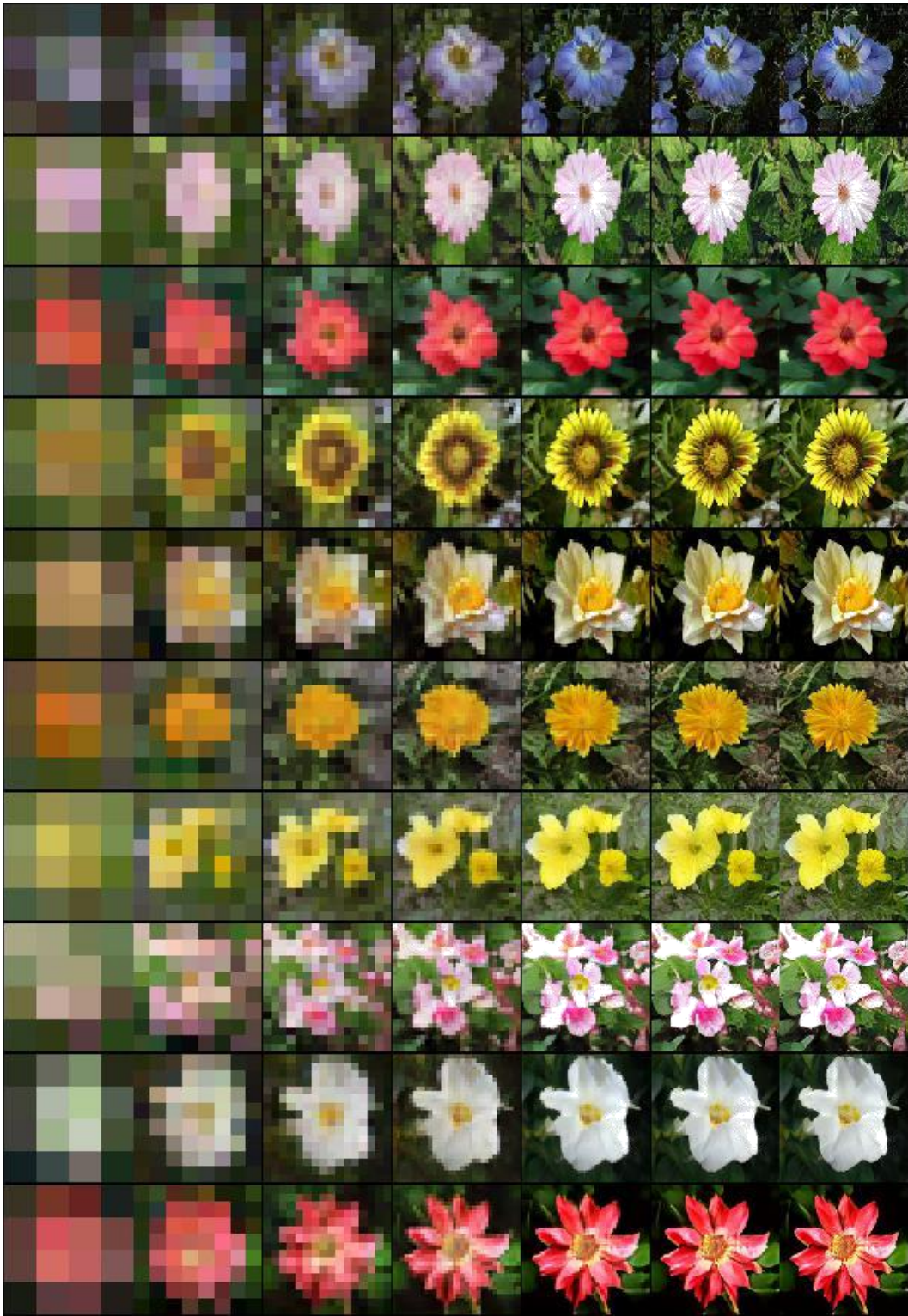


图 10: Oxford102 花卉数据集按所有 7 种分辨率生成的随机样本。

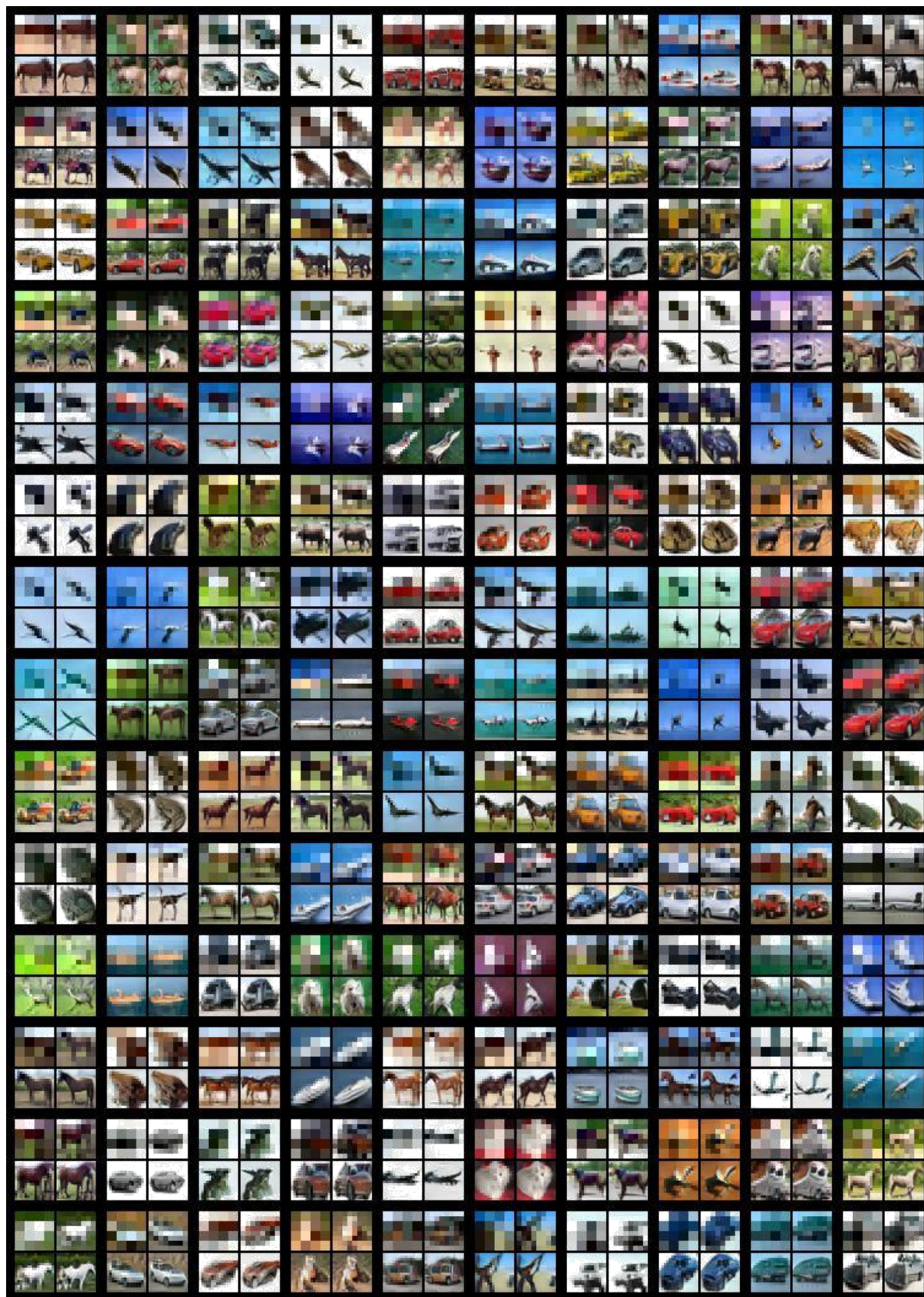


图 11: 按所有 4 种分辨率为 CIFAR-10 数据集生成的随机样本。



图 12: 随机生成的 CelebA Faces 分辨率为 128 x 128。



图 13: 随机生成的 LSUN 卧室, 分辨率为 128 x 128。



图 14: 分辨率为 1024 x 1024 的随机生成的 CelebA-HQ 人脸。