Training Generative Adversarial Networks in One Stage

Chengchao Shen,Youtan Yin,Xinchao Wang,Xubin Li,Jie Song,Mingli Song

摘要

摘要：生成对抗网络 (GANs) 在各种图像生成任务中取得了前所未有的成功。然而，这些结果是以繁琐的训练过程为代价的，在此过程中，生成器和鉴别器在两个阶段交替更新。在本文中，作者研究了一种通用的训练方案，可以在一个阶段有效地训练 GANs。基于生成器和判别器的对抗损失，我们将 GANs 分为两类，对称 GANs 和非对称 GANs，并引入了一种新的梯度分解方法来统一这两类 GANs，这使得我们能够在一个阶段训练这两个类，从而减轻训练工作量。我们还计算分析了所提方法的效率，并通过经验证明，所提方法在各种数据集和网络架构上产生了 1.5 倍的整体加速度。

关键词：生成对抗网络；单阶段

1 引言

生成对抗网络 (GANs) 自从在[1]中引入以来，在各种图像生成任务上产生了前所未有的令人印象深刻的结果。由于生成器和判别器这两个关键组件的对抗性性质，GANs 提供的合成图像在视觉上很吸引人，在许多情况下与真实图像难以区分。最近，许多 GANs 的变体引入并专注于设计的不同方面，包括图像质量、训练稳定性和多样性。除了生成图像作为最终目标外，GANs 还被应用于其他任务，如无数据的知识蒸馏和领域适应。

然而，GANs 提供的有希望的结果是以繁重的训练过程为代价的。现有的 GANs 依赖于耗时的两阶段训练过程，作者称之为两阶段 GANs (TSGANs)。在第一阶段，由生成器合成的假图像与真实图像一起输入判别器进行训练; 在此过程中，判别器被更新，但生成器是固定的。在第二阶段，判别器将从损失函数得到的梯度传递给生成器，在此期间生成器更新，但判别器是固定的。因此，在每个对抗回合中，生成器和判别器都执行两次前向传播，而判别器执行另两次反向传播步骤，涉及许多重复的计算。

为了努力减少 GANs 繁琐的训练过程，前辈们进行了许多工作。例如,[2]的工作采用了一种有效的对抗训练策略进行无监督域适应，其中学习特征提取器和分类器只需要一轮正向推理和反向传播。[3]的方法还利用了单步优化，一次更新生成器和判别器的参数，并展示了其生成视觉逼真图像的能力。

尽管这种方法提高了效率，但只适用于 GANs 的一个子集，其损失函数采用特定的形式。具体而言，在这种 GANs 中，生成器和判别器中的对抗损失项是相同的。因此，我们将这种模型称为对称 GANs。

然而，许多其他流行的 GANs 采用了对生成器和判别器持有不同对抗项的损失函数，我们将这些模型称为非对称 GANs。不幸的是，所采用的加速优化技术已无法处理此类非对称模型。为了解决非对称模型训练繁琐的过程，有必要对此进行研究。

2 相关工作

2.1 生成对抗网络

GAN 的开创性工作引入了一种对抗策略，其中生成器被训练合成假图像来混淆鉴别器，判别器试图将合成图像与真实图像区分开来。当生成器和判别器收敛到竞争平衡时，生成器最终可以从潜在变量中合成真实图像。主流的 GAN 研究主要集中在提高图像质量、训练稳定性和多样性。最近，GANs 在各种图像生成任务中展示了其有前景的结果，如超分辨率，图像编辑，图像补全[4]和图像翻译。

2.2 一阶段及两阶段框架

目前的目标检测方法可以分为单阶段框架和两阶段框架。对于两阶段框架[5]，先实现类别无关的区域推理模块，查找图像中可能的对象位置，然后由类别特定的分类器为每个位置分配类别标签，并在此基础上实现多个前向反馈。而单阶段检测方法将目标包围盒和类预测统一在一个前向反馈中，大大降低了计算成本。

与目标检测类似，实例分割也可以分为单阶段框架和两阶段框架。对于两阶段框架，模型首先对每个对象实例进行对象检测，获得边界框，然后在每个边界框内进行二进制分割，得到最终结果。单阶段实例分割采用与检测相似的动机，将实例掩码预测和类别预测结合起来，有效提高了推理效率。而分类等其他任务主要依赖于一阶段方案。

上述一阶段方法侧重于提高其任务的推理效率，而作者提出的方法更注重提高 GANs 的训练效率。

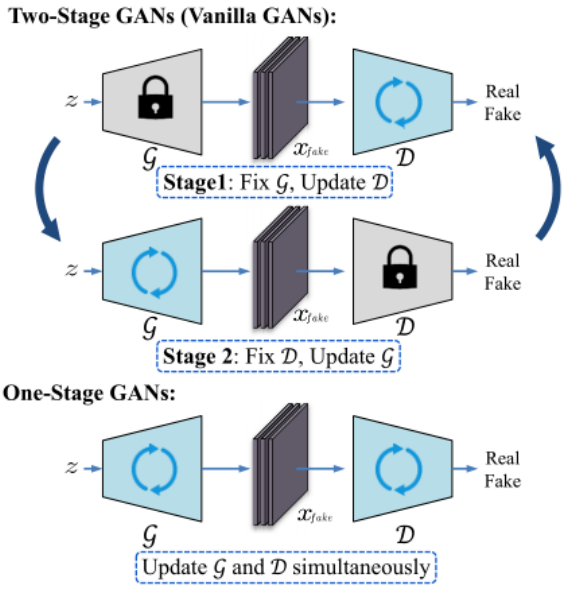

图 1: 两阶段和单阶段对比

3 本文方法

3.1 本文方法概述

GAN 通过引入判别器网络和生成器网络之间最大化最小化的博弈过程，使得生成网络实现了真实性样本合成。GAN 所采用的目标函数如下：

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathbb{E}_{x \sim p_d} [\log \mathcal{D}(x)] + \mathbb{E}_{z \sim p_z} [\log(1 - \mathcal{D}(\mathcal{G}(z)))]$$

为了后续讨论的方便，作者将上述目标函数拆解成分别针对判别器 D 和生成器 G 的损失函数，其

形式如下：

$$\mathcal{L}_{\mathcal{D}} = -\log \mathcal{D}(x) - \log(1 - \mathcal{D}(\mathcal{G}(z)))$$

$$\mathcal{L}_{\mathcal{G}} = -\log(\mathcal{D}(\mathcal{G}(z)))$$

其中，生成器损失函数和判别器损失函数包含了关于相同的对抗损失项。因此，将这种生成对抗网络称为对称型 GAN。但有人发现，使用对称型损失函数会造成梯度消失问题，为了缓解训练过程中生成器网络梯度消失[6]的问题，有学者又提出了非饱和对抗损失函数：

$$\mathcal{L}_{\mathcal{D}} = -\log \mathcal{D}(x) - \log(1 - \mathcal{D}(\mathcal{G}(z)))$$

$$\mathcal{L}_{\mathcal{G}} = -\log(\mathcal{D}(\mathcal{G}(z)))$$

其中生成器网络和判别器网络关于的假样本的对抗损失项并不一致，因此将上述生成对抗网络称为非对称型 GAN。

如上述公式所述，GAN 目标函数中的对抗项通常是关于假样本的。因此，为了分析的方便性，将一般化判别器损失函数的分成两个部分：关于真实样本的损失项和关于假样本的损失项。最终生成对抗网络一般化的目标函数可以表示为：

$$\mathcal{L}_{\mathcal{D}}(x, \hat{x}) = \mathcal{L}_{\mathcal{D}}^{r}(x) + \mathcal{L}_{\mathcal{D}}^{f}(\hat{x})$$

$$\mathcal{L}_{\mathcal{G}}(\hat{x}) = \mathcal{L}_{\mathcal{G}}(\mathcal{G}(z))$$

在本文中，作者提出了一种新的单阶段训练方案，称为单阶段 GANs (OSGANs)，它适用于对称和非对称 GANs。作者的关键思想是在前向传播中集成生成器和判别器的优化，在反向传播中分解它们的梯度，分别在一个阶段更新它们。对于对称情况，由于判别损失具有与生成器损失相同的项，我们只需要计算这一项的梯度一次，并将其用于两个损失。通过这种方式，生成器和判别器的更新可以安全地在一个前进和后退步骤中进行。

训练非对称 GAN 更加棘手，因为我们不能再将从判别器派生的梯度复制到生成器。为此，作者仔细研究了判别器的梯度的组成，最后作者发现，从不同的对抗项衍生的梯度，在现实中，保持它们在总梯度中的比例，从最后一层一直回到判别器的第一层。梯度的这种性质反过来为作者提供了一种可行的解决方案，可以分解不同对抗项的梯度，然后更新判别器和生成器，从而实现非对称 GAN 的单阶段训练。最后，作者完成了两类 GANs 的优化统一，并表明对称 GANs 实际上可以被视为非对称 GANs 的退化情况。

3.2 对称型 GAN 的单阶段训练

对于对称型 GAN，生成器损失函数与判别器损失函数包含相同的关于假样本的损失项。生成器关于假样本的梯度与判别器成相反数关系，因此我们可以通过判别器关于假样本的梯度乘上负一得到生成器关于假样本的梯度，从而在更新判别器 D 的同时，更新生成器 G。综上所述，通过上述方法就可以将对称型 GAN 的两阶段训练过程简化成单阶段过程。

3.3 非对称型 GAN 的单阶段训练

对于非对称型 GAN，由于生成器与判别器关于假样本的损失项不相关，生成器关于假样本的梯度无法像对称型 GAN 一样从判别器中获得。作者发现一种直接的思路是将生成器的损失函数与判别

器的损失函数整合在一起，整合形式如下：

$$\mathcal{L} = \mathcal{L}_{\mathcal{D}} - \mathcal{L}_{\mathcal{G}} = \mathcal{L}_{\mathcal{D}}^{r} + \mathcal{L}_{\mathcal{D}}^{f} - \mathcal{L}_{\mathcal{G}}$$

其中生成器的损失项取负号是因为生成器关于假样本的梯度与判别器关于假样本的梯度符号通常是相反的，取负号可以避免产生梯度抵消。这样做就可以从判别器关于假样本的梯度中获得生成器关于假样本的梯度。然而，这种方式也会产生另外一个问题：如何从混合的梯度中恢复出生成器关于假样本的梯度。

为了解决上述问题，作者对判别器网络的反向传播进行了调研，发现了主流神经网络模块中有一个很有意思的反向传播性质。除了批归一化模块，其他模块对应的损失函数关于输入的梯度和关于输出的梯度之间的关系可以表示为：

$$\nabla_{\text{in}} \mathcal{L} = P \cdot \mathcal{F}\left(\nabla_{\text{out}} \mathcal{L}\right) \cdot Q$$

其中 P 和 Q 是由对应神经模块或者其输出决定的矩阵；F(.) 是一个满足如下关系的函数：

$$\mathcal{F}\left(y_1 + y_2\right) = \mathcal{F}\left(y_1\right) + \mathcal{F}\left(y_2\right)$$

上述梯度满足公式的神经网络模块主要包括卷积模块、全连接模块以及非线性激活函数、以及池化模块等。需要说明的是，虽然非线性激活函数和池化模块是非线性操作，但是其关于输入的梯度和输出的梯度仍然满足上述公式。

根据上述梯度公式，作者得到判别器网络中关于假样本的梯度和生成器关于的梯度之间的关系：

$$\frac{\nabla_{\hat{x}_i} \mathcal{L}_{\mathcal{G}}}{\nabla_{\hat{x}_i} \mathcal{L}_{\mathcal{D}}} = \cdots = \frac{\nabla_{\hat{x}_i^I} \mathcal{L}_{\mathcal{G}}}{\nabla_{\hat{x}_i^l} \mathcal{L}_{\mathcal{D}}} = \cdots = \frac{\nabla_{\hat{x}_i^L} \mathcal{L}_{\mathcal{G}}}{\nabla_{\hat{x}_i^L} \mathcal{L}_{\mathcal{D}}} = \gamma_i$$

其中 L 表示判别器网络的层数；γi 是关于假样本的样本比例标量。

关于样本的 γi 对于判别器 D 不同的网络层是一个常数。同时对于每一个样本都有一个对应的 γi。也就是说，一般不同样本有不同的 γi。一方面，我们只需要计算最后一层的 γi，即

$$\nabla_{\hat{x}_i^L}^{L} \mathcal{L}_{\mathcal{G}} / \nabla_{\hat{x}_i^L}^{L} \mathcal{L}_{\mathcal{D}}$$

就可以得到所有网络层的 γi 值。这个方式只需要计算两个标量之间的比值，计算过程十分简单高效。另一方面，γi 的值根据样本的不同而发生变化，因此，对于每一个样本都要重新计算一次 γi。由于 γi 是两个标量：某一层生成器关于假样本的梯度与判别器关于假样本的梯度的比值，因此其计算代价在整个网络训练中可以忽略不计。

结合上述公式，作者发现可以按比例从混合的梯度中分解得到关于假样本的生成器和判别器的梯度，具体如下：

$$\nabla_{\hat{x}_i^l} \mathcal{L}_{\mathcal{D}}^{f} = \frac{1}{1-\gamma_i} \nabla_{\hat{x}_i^l} \mathcal{L}_f,$$
$$\nabla_{\hat{x}_i^l} \mathcal{L}_{\mathcal{G}} = \frac{\gamma_i}{1-\gamma_i} \nabla_{\hat{x}_i^l} \mathcal{L}_f.$$

也就是说，可以通过对联合梯度进行尺度缩放分别得到生成器和判别器关于假样本的梯度。为了方便计算，作者将上面这种尺度缩放操作应用到了损失函数上 L，以实现和上述梯度分解公式相同的

效果。最后作者得到判别器网络 D 和生成器网络 G 的实例损失函数：

$$\mathcal{L}_{\mathcal{D}}^{ins} = \mathcal{L}_{\mathcal{D}}^{r} + \frac{1}{1-\gamma_i}\left(\mathcal{L}_{\mathcal{D}}^{f} - \mathcal{L}_{\mathcal{G}}\right),$$
$$\mathcal{L}_{\mathcal{G}}^{ins} = \frac{\gamma_i}{1-\gamma_i}\left(\mathcal{L}_{\mathcal{D}}^{f} - \mathcal{L}_{\mathcal{G}}\right),$$

通过这种方式，非对称型 GAN 可以转化为对称型 GAN。因此，可以将对称型 GAN 中采用的单阶段训练策略应用到非对称型 GAN 中。

3.4 优化效率分析

作者进一步分析了单阶段生成对抗网络和两阶段生成对抗网络的效率。主要从三个角度对这个问题进行了分析：1）在一个数据批训练中，真实样本的耗时和生成样本的耗时；2）前向推理的耗时和反向传播的耗时；3）关于网络参数梯度计算的耗时和反向传播的耗时。

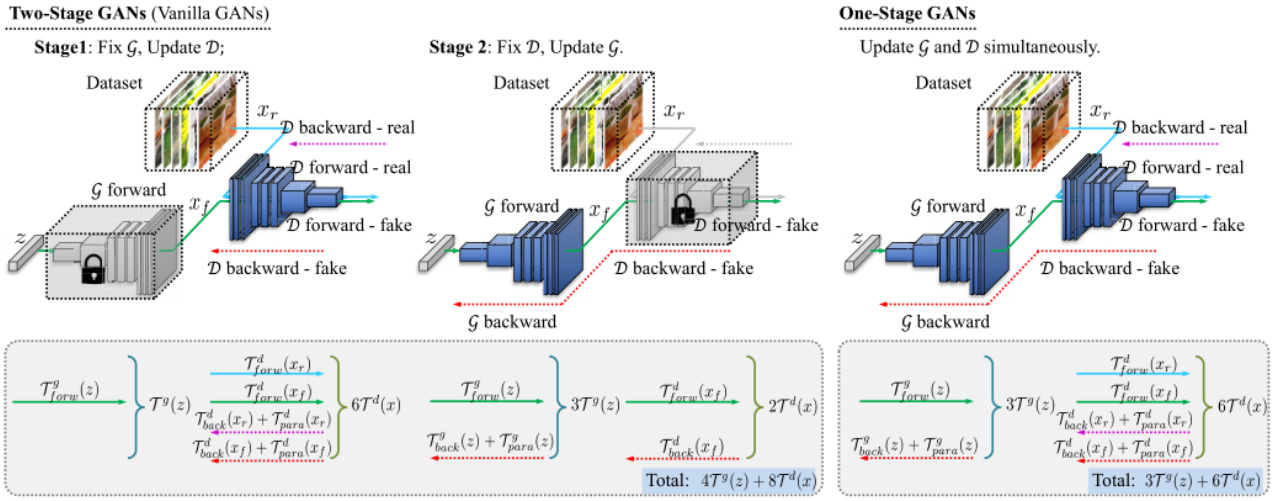

图 2: 优化效率分析

经过如上图 2 所示的分析发现，普通 GAN 训练的两阶段耗时分别为：

$$\mathcal{T}_{\text{stage 1}}(z,x) = \mathcal{T}^g(z) + 6\mathcal{T}^d(x)$$
$$\mathcal{T}_{\text{stage 2}}(z,x) = 3\mathcal{T}^g(z) + 2\mathcal{T}^d(x)$$

通过和两阶段生成对抗网络相同的方式，本文计算得到单阶段生成对抗网络的总耗时为:

$$\mathcal{T}_{\text{one}}(z,x) = 3\mathcal{T}^g(z) + 6\mathcal{T}^d(x)$$

最终，我们得到了在最坏的情况下单阶段生成对抗网络相对于两阶段生成对抗网络的加速比：

$$S = \frac{\mathcal{T}_{\text{two}}(z,x)}{\mathcal{T}_{\text{one}}(z,x)} = \frac{4\mathcal{T}^g(z) + 8\mathcal{T}^d(x)}{3\mathcal{T}^g(z) + 6\mathcal{T}^d(x)} = \frac{4}{3}$$

也就是说，相比于两阶段训练的 GAN 来说，单阶段 GAN 的训练时间相当于两阶段的 0.75。

4 复现细节

4.1 与已有开源代码对比

作者在这篇论文提出了一种生成对抗网络的优化方案，可以将 GAN 繁琐的两阶段训练过程简化为单阶段，大大减少了训练时间。与此同时，作者在其开源的代码中基于这项工作实现了两种 GAN 的优化，分别是 WGAN[7]和 DCGAN。通过学习这篇论文及作者的实现思路，我实现了两阶段的 LSGAN[8]与优化后的单阶段 LSGAN。

4.2 实验环境搭建

实验环境：python3.7

数据集：MINST 以及 CelebA

4.3 创新点

通过对这篇文章及其代码的学习，将这种梯度分解的思路运用到 LSGAN 当中，再复现 LSGAN 的基础上完成了单阶段的 LSGAN，并对比了两阶段 LSGAN 与单阶段 LSGAN 的运行效率。

5 实验结果分析

以下是作者分别在两阶段 GAN 和单阶段 GAN 在 CelebA 下的生成结果：

图 3: 实验结果示意

下图是使用实现的单阶段 LSGAN 得到的运行结果

图 4: LSGAN 测试结果

图 5 是 OSGANs 和 TSGANs 的比较结果。所有结果的平均值超过五次运行和误差条对应的标准偏差。具体来说，“sym”和“asym”分别表示对称 GANs 和不对称 GANs，“one”和“two”分别表示一阶段和两阶段策略。+ 表示在 [1] 中采用不对称对抗损失;↓ 和 ↑ 分别表示作者的策略优于和劣于两阶段策略。

Method	CelebA		LSUN Churches		FFHQ	
	FID	KID	FID	KID	FID	KID
DCGAN [52] (sym, two)	23.78±0.12	0.019±0.001	46.09±0.25	0.043±0.002	40.56±0.32	0.040±0.001
DCGAN (sym, one)	22.66±0.09(↓)	0.018±0.001(↓)	38.99±0.21(↓)	0.037±0.001(↓)	29.38±0.24(↓)	0.026±0.001(↓)
CGAN [45] (sym, two)	39.85±0.27	0.035±0.000	/	/	/	/
CGAN (sym, one)	30.97±0.33(↓)	0.013±0.000(↓)	/	/	/	/
WGAN [43] (sym, two)	33.30±0.23	0.028±0.001	36.29±0.14	0.034±0.002	35.66±0.13	0.032±0.001
WGAN (sym, one)	27.23±0.18(↓)	0.021±0.001(↓)	35.58±0.30(↓)	0.033±0.002(↓)	39.77±0.14(↑)	0.036±0.001(↑)
DCGAN [1]† (asym, two)	25.34±0.20	0.022±0.001	33.35±0.40	0.030±0.001	31.50±0.15	0.031±0.001
DCGAN† (asym, one)	24.20±0.17(↓)	0.019±0.001(↓)	25.41±0.30(↓)	0.024±0.001(↓)	31.19±0.21(↓)	0.028±0.001(↓)
LSGAN [41] (asym, two)	23.00±0.17	0.019±0.001	22.63±0.16	0.021±0.001	30.29±0.23	0.029±0.001
LSGAN (asym, one)	19.31±0.18(↓)	0.015±0.001(↓)	22.39±0.37(↓)	0.019±0.001(↓)	29.11±0.12(↓)	0.026±0.001(↓)
GeoGAN [35] (asym, two)	21.92±0.13	0.018±0.001	18.85±0.41	0.016±0.001	30.55±0.21	0.031±0.001
GeoGAN (asym, one)	21.52±0.14(↓)	0.017±0.001(↓)	18.92±0.24(↑)	0.017±0.001(↑)	30.63±0.26(↑)	0.032±0.001(↑)
RelGAN [29] (asym, two)	20.91±0.15	0.018±0.001	24.95±0.19	0.023±0.002	35.81±0.18	0.033±0.001
RelGAN (asym, one)	20.79±0.18(↓)	0.015±0.001(↓)	25.09±0.25(↑)	0.024±0.001(↑)	35.72±0.16(↓)	0.031±0.001(↓)
FisherGAN [47] (asym, two)	27.88±0.29	0.024±0.001	29.25±0.32	0.026±0.001	52.65±0.16	0.050±0.001
FisherGAN (asym, one)	27.10±0.19(↓)	0.021±0.001(↓)	29.23±0.18(↓)	0.025±0.001(↓)	51.63±0.18(↓)	0.049±0.001(↓)
BGAN [23] (asym, two)	31.12±0.19	0.027±0.001	35.06±0.31	0.035±0.001	41.35±0.17	0.038±0.001
BGAN (asym, one)	25.34±0.17(↓)	0.022±0.001(↓)	34.42±0.21(↓)	0.035±0.001(↓)	42.64±0.13(↑)	0.039±0.001(↑)
SNGAN [46] (asym, two)	34.95±0.24	0.033±0.001	33.51±0.16	0.034±0.001	52.46±0.24	0.050±0.001
SNGAN (asym, one)	34.34±0.12(↓)	0.033±0.001(↓)	31.23±0.17(↓)	0.031±0.001(↓)	51.03±0.36(↓)	0.048±0.001(↓)

图 5: 性能比较

图 6 是 OSGANs 与 TSGANs 训练效率比较。当它们的性能达到平稳期时，所有实验将终止。

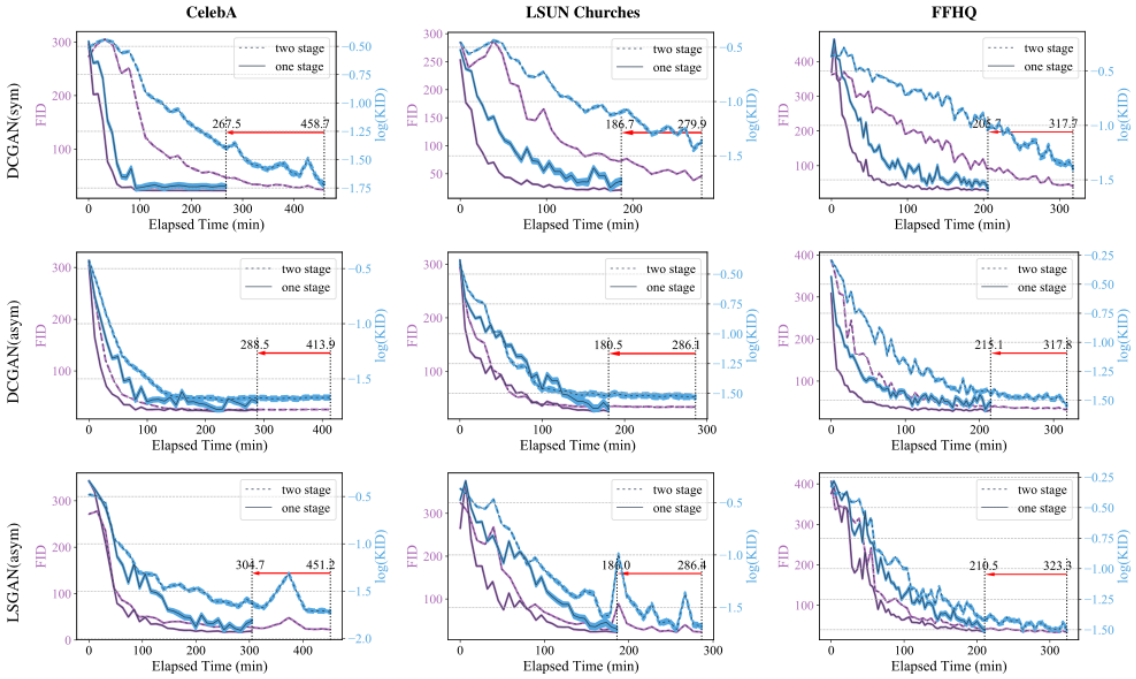

图 6: 训练效率比较

6 总结与展望

在本文中，作者研究了一种通用的单阶段训练策略，以提高 GANs 的训练效率。

文中将 GANs 分为两类，对称 GANs 和非对称 GANs。对于对称 GANs，在训练过程中通过判别器的反向传播获得生成器的梯度计算。因此，生成器和判别器的参数可以在一个阶段更新。对于非对称 GANs，作者提出了一种梯度分解方法，该方法整合了前向传播时的非对称对抗损失，并在反向传播时分解它们的梯度，在一个阶段分别更新生成器和鉴别器。

之后作者分析了上述解决方案之间的关系，并将其统一为单阶段训练策略。最后，文中计算分析了 OSGANs 与 TSGANs 的训练加速比。实验结果表明，OSGANs 比 TSGANs 的速度提高了 1.5 倍以上，同时能够实现 TSGANs 的结果。

参考文献

[1] GOODFELLOW I, POUGET-ABADIE J, MIRZA M, et al. Generative Adversarial Networks[J/OL]. Commun. ACM, 2020, 63(11): 139-144. https://doi.org/10.1145/3422622. DOI: 10.1145/3422622.

[2] GANIN Y, LEMPITSKY V. Unsupervised Domain Adaptation by Backpropagation[C/OL]//BACH F, BLEI D. Proceedings of Machine Learning Research: Proceedings of the 32nd International Conference on Machine Learning: vol. 37. Lille, France: PMLR, 2015: 1180-1189. https://proceedings.mlr.press/v37/ganin15.html.

[3] NOWOZIN S, CSEKE B, TOMIOKA R. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization[C/OL]//LEE D, SUGIYAMA M, LUXBURG U, et al. Advances in Neural Information Processing Systems: vol. 29. Curran Associates, Inc., 2016. https://proceedings.neurips.cc/paper/2016/file/cedebb6e872f539bef8c3f919874e9d7-Paper.pdf.

[4] PATHAK D, KRAHENBUHL P, DONAHUE J, et al. Context Encoders: Feature Learning by Inpainting [C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016.

[5] GIRSHICK R. Fast R-CNN[C]//Proceedings of the IEEE International Conference on Computer Vision (ICCV). 2015.

[6] ARJOVSKY M, BOTTOU L. Towards Principled Methods for Training Generative Adversarial Networks [EB/OL]. arXiv. 2017. https://arxiv.org/abs/1701.04862.

[7] ARJOVSKY M, CHINTALA S, BOTTOU L. Wasserstein Generative Adversarial Networks[C/OL]//PRECUP D, TEH Y W. Proceedings of Machine Learning Research: Proceedings of the 34th International Conference on Machine Learning: vol. 70. PMLR, 2017: 214-223. https://proceedings.mlr.press/v70/arjovsky17a.html.

[8] MAO X, LI Q, XIE H, et al. Least Squares Generative Adversarial Networks[C]//Proceedings of the IEEE International Conference on Computer Vision (ICCV). 2017.