Training Generative Adversarial Networks in One Stage
— *Supplementary Material* —

Chengchao Shen[1], Youtan Yin[1], Xinchao Wang[2,5], Xubin Li[3], Jie Song[1,4,*], Mingli Song[1]
[1]Zhejiang University, [2]National University of Singapore, [3]Alibaba Group,
[4]Zhejiang Lab, [5]Stevens Institute of Technology
{chengchaoshen,youtanyin,sjie,brooksong}@zju.edu.cn,
xinchao@nus.edu.sg,lxb204722@alibaba-inc.com

We provide in this document technical details and visual results that cannot fit into the main manuscript due to the page limit.

S1. Adversarial Objectives

In this section, we list the adversarial objectives evaluated in our experiments. As shown in Tab. S1, the adversarial objectives can be categorized into two classes, symmetric and asymmetric ones, based on the term $G(z)$ in the discriminator and generator loss.

Additionally, we find that the objectives for both RelGAN [4] and FisherGAN [9] can not be explicitly split into the two terms, one on fake sample $G(z)$ and another on real sample x. However, since the derivation with respect to $G(z)$ regards x as a constant, in reality, we can omit the term about x when carrying out the derivation with respect to $G(z)$. Therefore, we may still stick to the two-term-split of the adversarial objective, as done in Eq. 4 of our main manuscript.

Method	**Discriminator Loss $\mathcal{L}_{\mathcal{D}}$**	**Generator Loss $\mathcal{L}_{\mathcal{G}}$**
DCGAN [10] (sym)	$-\log \mathcal{D}(x) - \log\left(1-\mathcal{D}(\mathcal{G}(z))\right)$	$\log\left(1-\mathcal{D}(\mathcal{G}(z))\right)$
WGAN [8] (sym)	$\mathcal{D}^*(\mathcal{G}(z)) - \mathcal{D}^*(x)$	$-\mathcal{D}^*(\mathcal{G}(z))$
DCGAN [1] (asym)	$-\log \mathcal{D}(x) - \log\left(1-\mathcal{D}(\mathcal{G}(z))\right)$	$-\log\left(\mathcal{D}(\mathcal{G}(z))\right)$
LSGAN [7] (asym)	$\mathcal{L}_{\mathcal{D}} = \left(\mathcal{D}^*(x)-1\right)^2 + \mathcal{D}^*(\mathcal{G}(z))^2$	$\left(\mathcal{D}^*(\mathcal{G}(z))-1\right)^2$
GeoGAN [5] (asym)	$\max\left\{1-\mathcal{D}^*(x), k\cdot(1-\mathcal{D}^*(x))\right\}$ $+\max\left\{1+\mathcal{D}^*(\mathcal{G}(z)), k\cdot(1+\mathcal{D}^*(\mathcal{G}(z)))\right\}$	$\max\left\{1-\mathcal{D}^*(\mathcal{G}(z))), k\cdot(1-\mathcal{D}^*(\mathcal{G}(z)))\right\}$
RelGAN [4] (asym)	$-\log\left(\mathcal{D}(x)-\mathcal{D}(\mathcal{G}(z))\right)$ $-\log\left(1-\left(\mathcal{D}(\mathcal{G}(z))-\mathcal{D}(x)\right)\right)$	$-\log\left(\mathcal{D}(\mathcal{G}(z))-\mathcal{D}(x)\right)$ $-\log\left(1-\left(\mathcal{D}(x)-\mathcal{D}(\mathcal{G}(z))\right)\right)$
FisherGAN [9] (asym)	$\mathcal{D}^*(\mathcal{G}(z)) - \mathcal{D}^*(x) - \lambda\cdot\mathcal{R} + 0.5\cdot\rho\cdot\mathcal{R}^2$, where $\mathcal{R} = 1 - 0.5\cdot\left(\mathcal{D}^*(x)^2 + \mathcal{D}^*(\mathcal{G}(z))^2\right)$	$-\mathcal{D}^*(\mathcal{G}(z))$
BGAN [2] (asym)	$-\log \mathcal{D}(x) - \log\left(1-\mathcal{D}(\mathcal{G}(z))\right)$	$0.5\cdot\left(\log\mathcal{G}(z) - \log\left(1-\mathcal{D}(\mathcal{G}(z))\right)\right)^2$

Table S1: Various adversarial objectives. Specifically, "sym" and "asym" respectively denote Symmetric GANs and Asymmetric GANs. $\mathcal{D}^*(\cdot)$ denotes the discriminator whose output is not normalized by sigmoid operation.

S2. Back-Propagation Property

Here, we show an interesting observation on back-propagation, which allows us to derive the gradient-propagation equations in Tab. S2 and further enables our solution to training Asymmetric OSGANs.

*Corresponding author

Specifically, the gradient propagation from l-th layer to $(l-1)$-th layer can be expressed as follows:

$$\nabla_{x^{l-1}}\mathcal{L} = P \cdot \mathcal{F}\left(\nabla_{x^l}\mathcal{L}\right) \cdot Q. \tag{S1}$$

The transformation is composed of three parts: P, Q and, $\mathcal{F}(\cdot)$. Both P and Q here are matrices that change the size of the gradient tensor $\nabla_{x^l}\mathcal{L}$, and their values depend on the specific module of the corresponding layer. For example, in a linear module, P is the transpose of its weights $(W^l)^{\mathrm{T}}$, which transforms the size of $\nabla_{x^l}\mathcal{L}$ into the one of $\nabla_{x^{l-1}}\mathcal{L}$, while Q is an identity matrix, $\mathcal{F}(\cdot)$ depends on the corresponding module and its features: x^{l-1} or x^l. Since x^{l-1} and x^l remain constant for different loss functions in the same one forward step, $\mathcal{F}(\cdot)$ can be treated as a univariate function about $\nabla_{x^l}\mathcal{L}$. Except batch normalization module, we can find $\mathcal{F}(\cdot)$ in Tab. S2 satisfies

$$\mathcal{F}(\boldsymbol{y_1} + \boldsymbol{y_2}) = \mathcal{F}(\boldsymbol{y_1}) + \mathcal{F}(\boldsymbol{y_2}). \tag{S2}$$

For batch normalization module, due to the involvement of the gradients from other samples: x_j, it does not meet Eq. S2.

Module	$\boldsymbol{P}$	$\boldsymbol{\mathcal{F}(\nabla_{x^l}\mathcal{L})}$	$\boldsymbol{Q}$	**Equation**
Convolution	$\boldsymbol{I}$	$\nabla_{x^l}\mathcal{L}$	$rot180(W^l)$	$\nabla_{x^{l-1}}\mathcal{L} = \nabla_{x^l}\mathcal{L} * rot180(W^l)$
Linear	$(W^l)^{\mathrm{T}}$	$\nabla_{x^l}\mathcal{L}$	$\boldsymbol{I}$	$\nabla_{x^{l-1}}\mathcal{L} = (W^l)^{\mathrm{T}}\nabla_{x^l}\mathcal{L}$
Pooling	$\boldsymbol{I}$	$upsample(\nabla_{x^l}\mathcal{L})$	$\boldsymbol{I}$	$\nabla_{x^{l-1}}\mathcal{L} = upsample(\nabla_{x^l}\mathcal{L})$
Activation	$\boldsymbol{I}$	$\nabla_{x^l}\mathcal{L} \odot \sigma'(x^{l-1})$	$\boldsymbol{I}$	$\nabla_{x^{l-1}}\mathcal{L} = \nabla_{x^l}\mathcal{L} \odot \sigma'(x^{l-1})$
BatchNorm	$\boldsymbol{I}$	$\frac{1}{N\cdot\sqrt{\sigma^2+\epsilon}}[N\nabla_{x_i^l}\mathcal{L} - \sum_{j=1}^{N}\nabla_{x_j^l}\mathcal{L} - x_i^l\sum_{j=1}^{N}\left\{\nabla_{x_j^l}\mathcal{L}\cdot x_j^l\right\}]$	$\boldsymbol{I}$	$\nabla_{x_i^{l-1}}\mathcal{L} = \frac{1}{N\cdot\sqrt{\sigma^2+\epsilon}}[N\nabla_{x_i^l}\mathcal{L} - \sum_{j=1}^{N}\nabla_{x_j^l}\mathcal{L} - x_i^l\sum_{j=1}^{N}\left\{\nabla_{x_j^l}\mathcal{L}\cdot x_j^l\right\}]$
Group Norm	$\boldsymbol{I}$	$\frac{1}{G\cdot\sqrt{\sigma^2+\epsilon}}[G\cdot\nabla_{x^{(l,i)}}\mathcal{L} - \sum_{j\in S_i}\nabla_{x^{(l,i,j)}}\mathcal{L} - \sum_{j\in S_i}\left\{\nabla_{x^{(l,i,j)}}\mathcal{L}\odot x^{(l,i,j)}\right\}\odot x^{(l,i)}]$	$\boldsymbol{I}$	$\nabla_{x^{(l-1,i)}}\mathcal{L} = \frac{1}{G\cdot\sqrt{\sigma^2+\epsilon}}[G\cdot\nabla_{x^{(l,i)}}\mathcal{L} - \sum_{j\in S_i}\nabla_{x^{(l,i,j)}}\mathcal{L} - \sum_{j\in S_i}\left\{\nabla_{x^{(l,i,j)}}\mathcal{L}\odot x^{(l,i,j)}\right\}\odot x^{(l,i)}]$
Layer Norm	$\boldsymbol{I}$	$\frac{1}{C\cdot H\cdot W\cdot\sqrt{\sigma^2+\epsilon}}[C\cdot H\cdot W\cdot\nabla_{x^l}\mathcal{L} - \sum_j\nabla_{x^l}\mathcal{L} - \sum_j\left\{\nabla_{x^l}\mathcal{L}\odot x^l\right\}\odot x^l]$	$\boldsymbol{I}$	$\nabla_{x^{l-1}}\mathcal{L} = \frac{1}{C\cdot H\cdot W\cdot\sqrt{\sigma^2+\epsilon}}[C\cdot H\cdot W\cdot\nabla_{x^l}\mathcal{L} - \sum_j\nabla_{x^l}\mathcal{L} - \sum_j\left\{\nabla_{x^l}\mathcal{L}\odot x^l\right\}\odot x^l]$
Instance Norm	$\boldsymbol{I}$	$\frac{1}{H\cdot W\cdot\sqrt{\sigma^2+\epsilon}}[H\cdot W\cdot\nabla_{x^{(l,j)}}\mathcal{L} - \sum_j\nabla_{x^{(l,j)}}\mathcal{L} - \sum_j\left\{\nabla_{x^{(l,j)}}\mathcal{L}\odot x^{(l,j)}\right\}\odot x^{(l)}]$	$\boldsymbol{I}$	$\nabla_{x^{(l-1,j)}}\mathcal{L} = \frac{1}{H\cdot W\cdot\sqrt{\sigma^2+\epsilon}}[H\cdot W\cdot\nabla_{x^{(l,j)}}\mathcal{L} - \sum_j\nabla_{x^{(l,j)}}\mathcal{L} - \sum_j\left\{\nabla_{x^{(l,j)}}\mathcal{L}\odot x^{(l,j)}\right\}\odot x^{(l)}]$

Table S2: Gradient propagation equations for mainstream neural modules. Each equation is split into three parts, P, $\mathcal{F}(\nabla_{x^l}\mathcal{L})$, and Q. "$\boldsymbol{I}$" denotes identity matrix, "$*$" and "$\odot$" respectively denote convolution operation and element-wise multiplication, and "$\sigma(\cdot)$" denotes activation operation, "G" denotes group size, "S_i" denotes the group set for sample x_i.

S3. Derivation of Gradient Decomposition

Since all above neural modules are differentiable, $\mathcal{F}(\cdot)$ is also differentiable. We set $\boldsymbol{y_1} = \mathbf{0}$ and $\boldsymbol{y_2} = \mathbf{0}$, then we have

$$\mathcal{F}(\mathbf{0}+\mathbf{0}) = \mathcal{F}(\mathbf{0}) + \mathcal{F}(\mathbf{0}) \Rightarrow \mathcal{F}(\mathbf{0}) = \mathbf{0}. \tag{S3}$$

Combining with Eq. S2 and Eq. S3, we obtain

$$\begin{aligned}
&\frac{\mathcal{F}(\boldsymbol{y_1}+\boldsymbol{y_2})-\mathcal{F}(\boldsymbol{y_1})}{(\boldsymbol{y_1}+\boldsymbol{y_2})-\boldsymbol{y_1}}=\frac{\mathcal{F}(\boldsymbol{y_2})}{\boldsymbol{y_2}}=\frac{\mathcal{F}(\boldsymbol{y_2})-\mathcal{F}(\boldsymbol{0})}{\boldsymbol{y_2}-\boldsymbol{0}}\\
&\Rightarrow \lim_{\boldsymbol{y_2}\to\boldsymbol{0}}\frac{\mathcal{F}(\boldsymbol{y_1}+\boldsymbol{y_2})-\mathcal{F}(\boldsymbol{y_1})}{(\boldsymbol{y_1}+\boldsymbol{y_2})-\boldsymbol{y_1}}=\lim_{\boldsymbol{y_2}\to\boldsymbol{0}}\frac{\mathcal{F}(\boldsymbol{y_2})-\mathcal{F}(\boldsymbol{0})}{\boldsymbol{y_2}-\boldsymbol{0}}\\
&\Rightarrow \mathcal{F}'(\boldsymbol{y_1})=\mathcal{F}'(\boldsymbol{0})\\
&\Rightarrow \int_{\boldsymbol{0}}^{\boldsymbol{y}}\mathcal{F}'(\boldsymbol{y_1})d\boldsymbol{y_1}=\int_{\boldsymbol{0}}^{\boldsymbol{y}}\mathcal{F}'(\boldsymbol{0})d\boldsymbol{y_1}\\
&\Rightarrow \mathcal{F}(\boldsymbol{y})=\mathcal{F}'(\boldsymbol{0})\odot\boldsymbol{y}\\
&\Rightarrow \frac{\mathcal{F}(\boldsymbol{y_1})}{\mathcal{F}(\boldsymbol{y_2})}=\frac{\boldsymbol{y_1}}{\boldsymbol{y_2}},
\end{aligned}\tag{S4}$$

where all divisions are element-wise operation. Based on Eq. S1, we obtain

$$\begin{aligned}
\nabla_{x^{l-1}}\mathcal{L}_1&=P\cdot\mathcal{F}\left(\nabla_{x^l}\mathcal{L}_1\right)\cdot Q,\\
\nabla_{x^{l-1}}\mathcal{L}_2&=P\cdot\mathcal{F}\left(\nabla_{x^l}\mathcal{L}_2\right)\cdot Q,
\end{aligned}\tag{S5}$$

where both $\mathcal{L}_1$ and $\mathcal{L}_2$ are loss terms for sample x in the one forward inference.

Combining with Eq. S4 and Eq. S5, we obtain

$$\begin{aligned}
\frac{\nabla_{x^{l-1}}\mathcal{L}_1}{\nabla_{x^{l-1}}\mathcal{L}_2}&=\frac{P\cdot\mathcal{F}\left(\nabla_{x^l}\mathcal{L}_1\right)\cdot Q}{P\cdot\mathcal{F}\left(\nabla_{x^l}\mathcal{L}_2\right)\cdot Q}\\
&=\frac{\mathcal{F}\left(\nabla_{x^l}\mathcal{L}_1\right)}{\mathcal{F}\left(\nabla_{x^l}\mathcal{L}_2\right)}=\frac{\nabla_{x^l}\mathcal{L}_1}{\nabla_{x^l}\mathcal{L}_2}.
\end{aligned}\tag{S6}$$

Finally, for a fake sample $\hat{x}_i$ in the discriminator, we obtain

$$\frac{\nabla_{\hat{x}_i}\mathcal{L}_1}{\nabla_{\hat{x}_i}\mathcal{L}_2}=\cdots=\frac{\nabla_{\hat{x}_i^l}\mathcal{L}_1}{\nabla_{\hat{x}_i^l}\mathcal{L}_2}=\cdots=\frac{\nabla_{\hat{x}_i^L}\mathcal{L}_1}{\nabla_{\hat{x}_i^L}\mathcal{L}_2}=\gamma_i,\tag{S7}$$

where L is the total number of layers, and γ_i is an instance-wise scalar for $\hat{x}_i$.

S4. Time Analysis

In general, mainstream Convolutional Neural Networks (CNNs) spend more than 95% of their computation time on convolution and linear operations [3, 6]. Hence, we focus on the analysis of time cost for CNNs on convolution and linear operations. Convolution and linear operations can be regarded as the matrix multiplications between inputs and weights. As shown by the gradient propagation equation in Tab. S2, in both the forward inference and back-propagation stages, and they take an identical number of multiplication and addition operations. As a result, the time costs on the forward inference and back-propagation for these modules are approximately equal, i.e., $\mathcal{T}_{forw}(x)=\mathcal{T}_{back}(x)$.

We also analyze the number of float point operations of linear modules, such as convolution and linear operation, for the computation of parameter gradients and gradients backward. We find that parameter gradients computation and gradients backward cost the same number of float point multiply operations. Hence, the time for parameter gradients computation: $\mathcal{T}_{para}(x)$ and the one for gradients backward: $\mathcal{T}_{back}(x)$ are approximately equal, i.e., $\mathcal{T}_{para}(x)=\mathcal{T}_{back}(x)$.

S5. Simple One-Stage Adversarial Learning

The instance loss function as Eq. 9 is somewhat complex. We rethink the objective of one-stage adversarial learning from another point of view. For the discriminator loss function in asymmetric GAN, the aim of our one-stage adversarial is identical to the conventional two-stage one. Hence, the $\mathcal{L}_{\mathcal{D}}^{ins}$ in Eq. 9 can be replaced by plain discriminator loss: $\mathcal{L}_{\mathcal{D}}=\mathcal{L}_{\mathcal{D}}^{r}+\mathcal{L}_{\mathcal{D}}^{f}$ equivalently, which is simpler than Eq. 9. For the generator loss function in asymmetric GAN, its gradients are different from

the gradients of discriminator. With consideration of the property as Eq. 7, we find an alternative generator loss function, $\mathcal{L}_{\mathcal{G}}^{ins} = \gamma \mathcal{L}_{\mathcal{D}}^{f}$ to achieve the same target as Eq. 9. In summary, we can simplify the loss function in Eq. 9 as

$$\begin{aligned} \mathcal{L}_{\mathcal{D}}^{ins} &= \mathcal{L}_{\mathcal{D}}^{r} + \mathcal{L}_{\mathcal{D}}^{f}, \\ \mathcal{L}_{\mathcal{G}}^{ins} &= \gamma \mathcal{L}_{\mathcal{D}}^{f}, \end{aligned} \tag{S8}$$

which produces the same gradient information as Eq. 9 during adversarial learning.

S6. More Synthetic Results

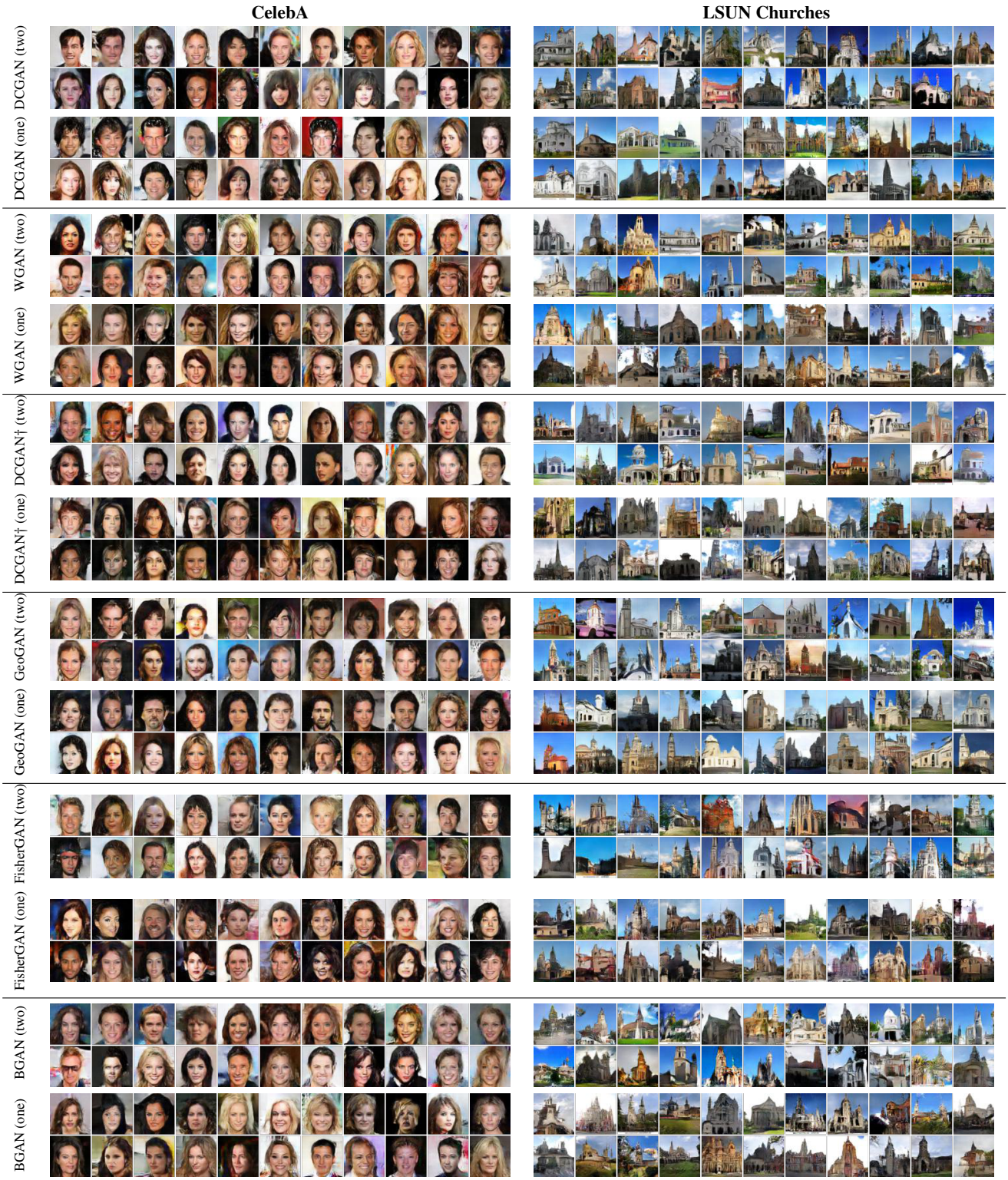

Figure S1: Results synthesized by TSGANs ("two") and OSGANs ("one") on CelebA and LSUN Churches. † adopts asymmetric adversarial loss in [1];

S7. Efficiency Analysis

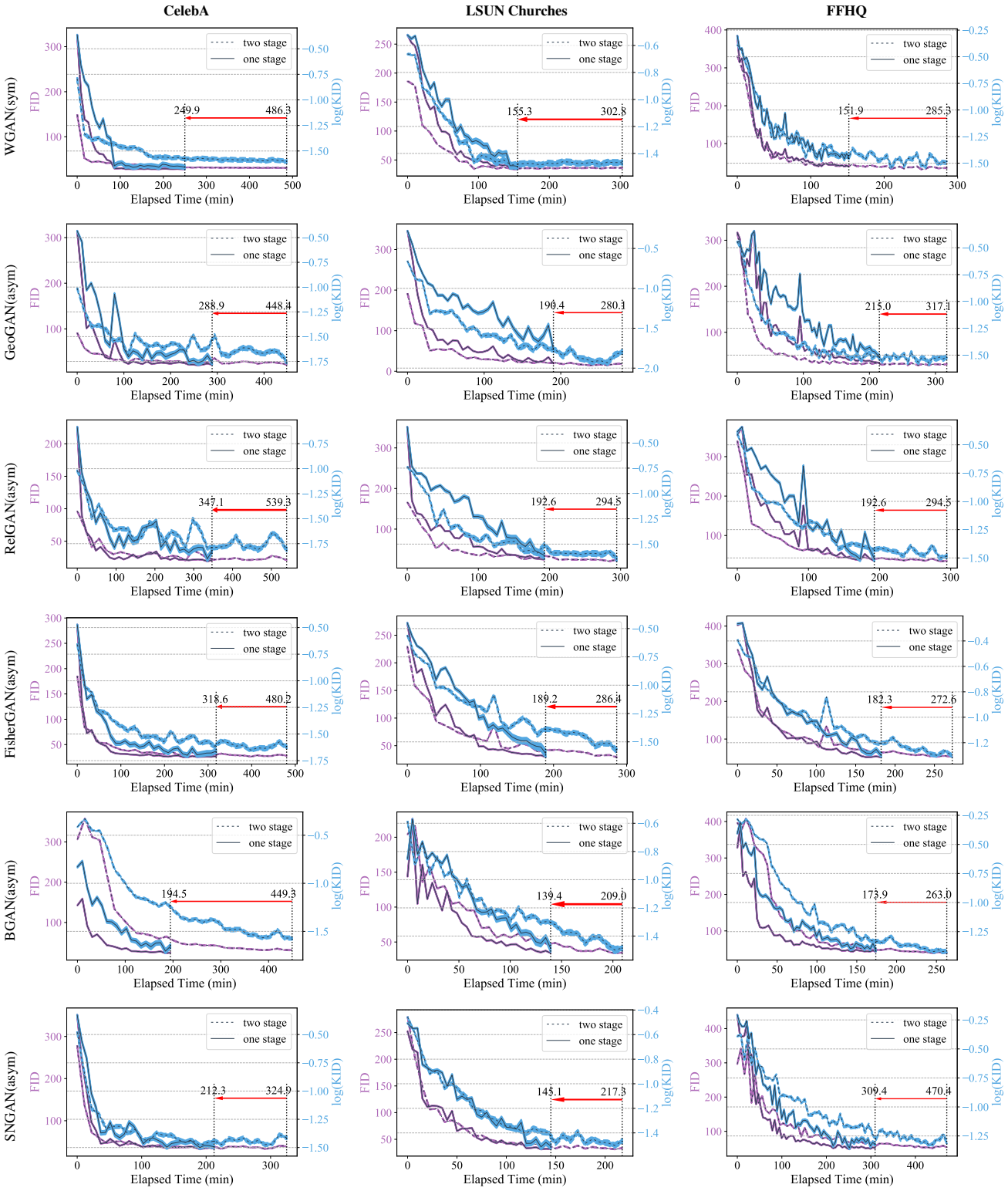

Figure S2: Comparison of training efficiency between OSGANs and TSGANs. All results are averaged over five runs and error bars correspond to the standard deviations. “sym” denotes symmetric GAN and “asym” denotes asymmetric GAN.

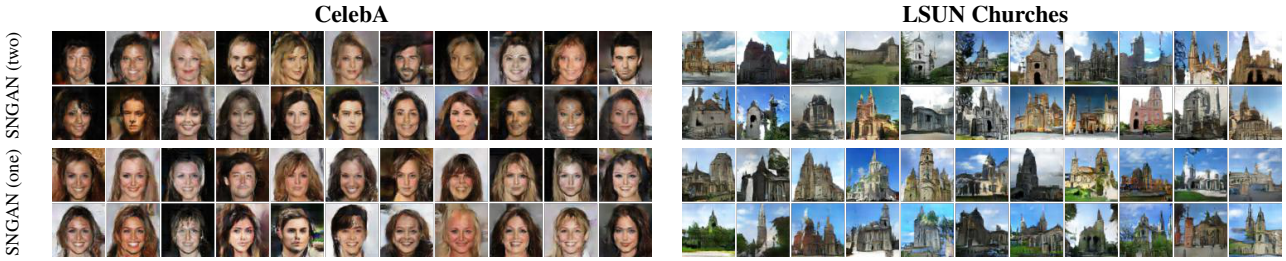

Figure S3: Results synthesized by TSGANs ("two") and OSGANs ("one") on CelebA and LSUN Churches.

References

[1] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2017. 1, 5

[2] R Devon Hjelm, Athul Paul Jacob, Tong Che, Adam Trischler, Kyunghyun Cho, and Yoshua Bengio. Boundary-seeking generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018. 1

[3] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 3

[4] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. In *International Conference on Learning Representations (ICLR)*, 2017. 1

[5] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017. 1

[6] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018. 3

[7] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international Conference on Computer Vision (ICCV)*, pages 2794–2802, 2017. 1

[8] SC Martin Arjovsky and Leon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017. 1

[9] Youssef Mroueh and Tom Sercu. Fisher gan. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2513–2523, 2017. 1

[10] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2015. 1