A. Derivation of objectives

A.1. Discriminator objective

$$
\begin{aligned}
\theta^* &= \arg\max_\theta \left\{ \sum_{x \in \mathcal{X}} \log P_D\left(x\right) \right\} \\
&= \arg\max_\theta \left\{ \sum_{x \in \mathcal{X}} \log \frac{1}{\zeta} e^{D_\theta(x)} \right\} \\
&= \arg\max_\theta \left\{ \sum_{x \in \mathcal{X}} \left(D_\theta\left(x\right) - \log \zeta \right) \right\} \\
&= \arg\max_\theta \left\{ \sum_{x \in \mathcal{X}} \left[D_\theta\left(x\right) - \log \sum_{y \sim P_{G_\psi}} \frac{e^{D_\theta(y)}}{P_{G_\psi}\left(y\right)} + \log S \right] \right\} \\
&= \arg\max_\theta \left\{ \sum_{x \in \mathcal{X}} \left[D_\theta\left(x\right) - \log \sum_{y \sim P_{G_\psi}} e^{D_\theta(y) - \log P_{G_\psi}(y)} \right] \right\} \\
&= \arg\max_\theta \left\{ \sum_{x \in \mathcal{X}} \left[D_\theta\left(x\right) - \log \sum_{z \sim P_Z(z)} e^{D_\theta(G_\psi(z)) - \log P_Z(z) + J_{G_\psi}(z)} \right] \right\} \\
&= \arg\min_\theta \left\{ \sum_{x \in \mathcal{X}} \left[-\log e^{D_\theta(x)} + \log \sum_{z \sim P_Z(z)} e^{D_\theta(G_\psi(z)) - \log P_Z(z) + J_{G_\psi}(z)} \right] \right\} \\
&= \arg\min_\theta \left\{ \sum_{x \in \mathcal{X}} \left[\log \sum_{z \sim P_Z(z)} e^{D_\theta(G_\psi(z)) - D_\theta(x)} e^{J_{G_\psi}(z) - \log P_Z(z)} \right] \right\}
\end{aligned}
\tag{13}
$$

Where $J_{G_\psi}\left(z\right) = \log \left| \det \left(\frac{\partial G_\psi(z)}{\partial z^\intercal} \right) \right|$ is the log Jacobian determinant of G_ψ at z.

A.2. Generator objective

$$
\begin{aligned}
\psi^* &= \arg\min_\psi \mathrm{KL}\left[P_{G_\psi}\left(y\right) \parallel P_{D_\theta}\left(y\right) \right] \\
&= \arg\min_\psi \left\{ \mathbb{E}_{y \sim P_G(y)} \log \left(\frac{P_{G_\psi}\left(y\right)}{P_{D_\theta}\left(y\right)} \right) \right\} \\
&= \arg\min_\psi \left\{ \mathbb{E}_{y \sim P_G(y)} \left[\log \left(P_{G_\psi}\left(y\right) \right) \right] - \mathbb{E}_{y \sim P_G(y)} \left[\log \left(P_{D_\theta}\left(y\right) \right) \right] \right\} \\
\text{[Definition of entropy]} &= \arg\min_\psi \left\{ -H\left(y\right) - \frac{1}{m} \sum_{y \sim P_G(y)} \log \left(P_{D_\theta}\left(y\right) \right) \right\} \\
[y = G_\psi\left(z\right)] &= \arg\min_\psi \left\{ -H\left(G_\psi\left(z\right)\right) - \frac{1}{m} \sum_{z \sim P_Z(z)} \log \left(\frac{e^{D_\theta(G_\psi(z))}}{\zeta} \right) \right\} \\
&= \arg\min_\psi \left\{ -H\left(G_\psi\left(z\right)\right) - \frac{1}{m} \sum_{z \sim P_Z(z)} D_\theta\left(G_\psi\left(z\right)\right) \right\} \\
&= \arg\max_\psi \left\{ H\left(G_\psi\left(z\right)\right) + \frac{1}{m} \sum_{z \sim P_Z} D_\theta\left(G_\psi\left(z\right)\right) \right\}
\end{aligned}
\tag{14}
$$

B. Optimization using Jacobian approximation

Using a single random sample to approximate the Jacobian determinant as in Eq. (8) does not provide an accurate estimate. Yet, this approximation is effective at maximization of its determinant. We confirm this claim by optimizing the approximation of the Jacobian and measure the effect on the true Jacobian.

In this experiment, we randomized 50 neural networks (using a random order of linear, convolution, LeakyRelu or batchnorm layers). Each network was trained to maximize the approximation of the log determinant of the Jacobian. After each optimization step, we computed the difference in the value of the log determinant of the true Jacobian from the previous step. We repeated this experiment with different vector size (8, 16, 32) and different network sizes (1, 4, 16, 32 layers, comparable with DCGAN) with a learning rate of 5e-4. In Fig. 9 we mark the differences between consecutive steps of the true log Jacobian. We define the success rate to be the percentage of times the log determinant increases, and present this metric in each graph. As can be seen, in all cases the success rate is above 85%.

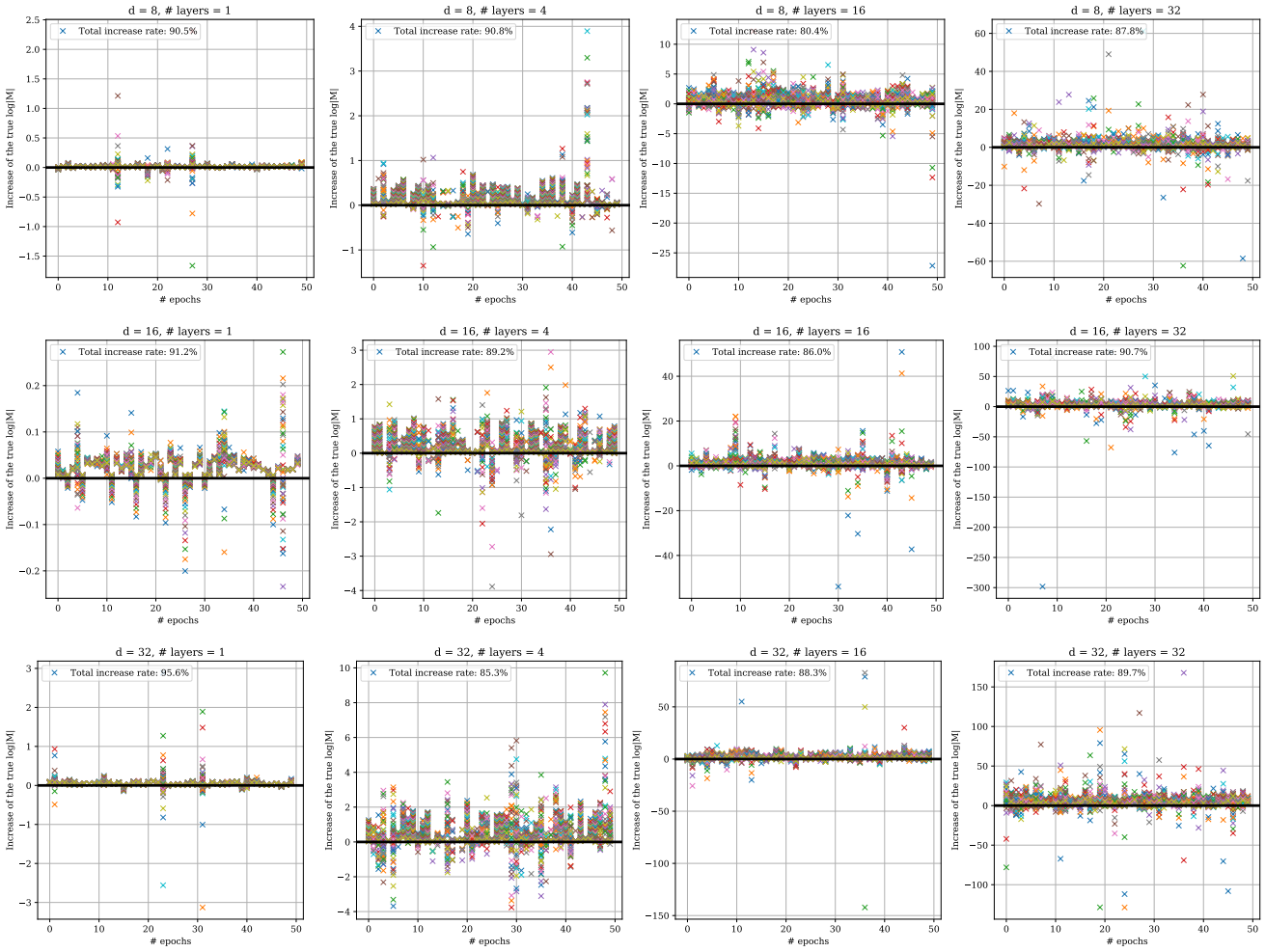

Figure 9. The difference between iterations of the true log determinant of a network trained to maximize the 1-sample approximation of the Jacobian log determinant. Each column represents a different network size and each row corresponds to a different latent size. The black bold line marks 0, where every mark above it means the true value increased.

C. Implementation

C.1. Numerical stability with large Jacobian determinant

During training, since the determinant of the Jacobian is generally a different order of magnitude than the discriminator's response (*e.g.* in the objectives in Eqs. (3) and (5)), it can cause instability in the gradients. To solve this, we add a scalar w and re-define the probability as

$$P_D(y) = \frac{e^{\frac{1}{w}D(y)}}{\zeta}. \tag{15}$$

This results in a slightly modified discriminator objective (Eq. (3)):

$$\begin{aligned}
\theta^* &= \arg\max_\theta \left\{ \sum_{x\in\mathcal{X}} \log P_D(x) \right\} \\
&= \arg\max_\theta \left\{ \sum_{x\in\mathcal{X}} \log \frac{1}{\zeta} e^{\frac{1}{w}D_\theta(x)} \right\} \\
&= \arg\max_\theta \left\{ \sum_{x\in\mathcal{X}} \left(\frac{1}{w} D_\theta(x) - \log\zeta \right) \right\} \\
&= \arg\max_\theta \left\{ \sum_{x\in\mathcal{X}} \left[\frac{1}{w} D_\theta(x) - \log \sum_{y\sim P_{G_\psi}} \frac{e^{\frac{1}{w}D_\theta(x)}}{P_{G_\psi}(y)} \right] \right\} \\
&= \arg\min_\theta \left\{ \sum_{x\in\mathcal{X}} \left[-\frac{1}{w} D_\theta(x) + \log \sum_{y\sim P_G(y)} \exp\left(\frac{1}{w} D_\theta(y) - \log P_{G_\psi}(y) \right) \right] \right\}.
\end{aligned} \tag{16}$$

For the generator, we use a scaled KL-divergence (f-divergence with $f(t) = wt\log t$. This results in the modified generator objective (Eq. (5)):

$$\begin{aligned}
\mathcal{D}_f(P_G \parallel P_D) &= \int f\left(\frac{P_G(y)}{P_D(y)}\right) P_D(y)\,\mathrm{d}y \\
&= \int w\frac{P_G(y)}{P_D(y)} \log\left(\frac{P_G(y)}{P_D(y)}\right) P_D(y)\,\mathrm{d}y \\
&= \mathbb{E}_{y\sim P_G}\left[w\log\left(\frac{P_G(y)}{P_D(y)}\right)\right] \\
&= \frac{1}{S}\sum_{y\sim P_G}\left[w\log\left(P_G(y)\right) - w\log\left(\frac{e^{\frac{1}{w}D(y)}}{\zeta}\right)\right] \\
&= \sum_{y\sim P_G}\left[-w\log|J(y)| - D(y)\right].
\end{aligned} \tag{17}$$

C.2. Implementation Details

We trained the generator and discriminator using the PyTorch ADAM optimizer [20]. We set the learning rate of the discriminator to 1e-5 and of the generator to 5e-4. We trained our model on the CelebA [27] for 30 epochs and CIFAR-10 [25] for 180 epochs. We generated images using a latent vector of size 100.

C.3. Architecture

C.3.1 Synthetic data network

For the synthetic data problems in Sec. 4.1, our generator is a 2-layered MLP. We used the pytorch autograd *jacobian* function to compute the Jacobian and its determinant.

Table 3. Comparison of time per training iteration in seconds between our model and WGAN with DCGAN architecture. The time is formatted as ⟨mean⟩ ± ⟨standard deviation⟩

	CELEBA	CIFAR-10
WGAN-GP	0.031 ± 0.00072	0.025 ± 0.00067
OURS - 1 SAMPLE	0.053 ± 0.00181	0.044 ± 0.00078
OURS - 2 SAMPLES	0.063 ± 0.00183	0.045 ± 0.00231

C.3.2 DC-GAN based generator architecture

The generator architecture we used for CIFAR-10 and CelebA is based on the DC-GAN architecture [30]. Given a latent vector, we concatenate a random normal vector. We then pass this vector to the DC-GAN layers and return the output. In order to compute $\|Jv\|$ from Eq. (9), we use the *jvp* (Jacobian-vector multiplication) function from pytorch.

C.4. Run times

To compare the effect of the model on the run time of training, we used a NVIDIA A100-SXM4-40GB GPU for all training on CelebA (64 pixels per edge) and CIFAR-10 (32 pixels per edge). Tab. 3 shows the run time per iteration in seconds. The table shows that applying the one-way flow almost doubles the time per iteration, but using a few additional samples does not increase the run time significantly further.