PERCEPTUAL GENERATIVE AUTOENCODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern generative models are usually designed to match target distributions directly in the data space, where the intrinsic dimensionality of data can be much lower than the ambient dimensionality. We argue that this discrepancy may contribute to the difficulties in training generative models. We therefore propose to map both the generated and target distributions to the latent space using the encoder of a standard autoencoder, and train the generator (or decoder) to match the target distribution in the latent space. The resulting method, perceptual generative autoencoder (PGA), is then incorporated with a maximum likelihood or variational autoencoder (VAE) objective to train the generative model. With maximum likelihood, PGAs generalize the idea of reversible generative models to unrestricted neural network architectures and arbitrary latent dimensionalities. When combined with VAEs, PGAs can generate sharper samples than vanilla VAEs. Compared to other autoencoder-based generative models using simple priors, PGAs achieve state-of-the-art FID scores on CIFAR-10 and CelebA.

1 INTRODUCTION

Recent years have witnessed great interest in generative models, mainly due to the success of generative adversarial networks (GANs) (Goodfellow et al., 2014; Radford et al., 2016; Karras et al., 2018; Brock et al., 2019). Despite their prevalence, the adversarial nature of GANs can lead to a number of challenges, such as unstable training dynamics and mode collapse. Since the advent of GANs, substantial efforts have been devoted to addressing these challenges (Salimans et al., 2016; Arjovsky et al., 2017; Gulrajani et al., 2017; Miyato et al., 2018), while non-adversarial approaches that are free of these issues have also gained attention. Examples include variational autoencoders (VAEs) (Kingma & Welling, 2014), reversible generative models (Dinh et al., 2014; 2017; Kingma & Dhariwal, 2018), and Wasserstein autoencoders (WAEs) (Tolstikhin et al., 2018).

However, non-adversarial approaches often have significant limitations. For instance, VAEs tend to generate blurry samples, while reversible generative models require restricted neural network architectures or solving neural differential equations (Grathwohl et al., 2019). Furthermore, to use the change of variable formula, the latent space of a reversible model must have the same dimensionality as the data space, which is unreasonable considering that real-world, high-dimensional data (e.g., images) tends to lie on low-dimensional manifolds, and thus results in redundant latent dimensions and variability. Intriguingly, recent research (Arjovsky et al., 2017; Dai & Wipf, 2019) suggests that the discrepancy between the intrinsic and ambient dimensionalities of data also contributes to the difficulties in training GANs and VAEs.

In this work, we present a novel framework for training autoencoder-based generative models, with non-adversarial losses and unrestricted neural network architectures. Given a standard autoencoder and a target data distribution, instead of matching the target distribution in the data space, we map both the generated and target distributions to the latent space using the encoder, and train the generator (or decoder) to minimize the divergence between the mapped distributions. We prove, under mild assumptions, that by minimizing a form of latent reconstruction error, matching the target distribution in the latent space implies matching it in the data space. We call this framework *perceptual generative autoencoder (PGA)*. We show that PGAs enable training generative autoencoders with maximum likelihood, without restrictions on architectures or latent dimensionalities. In addition, when combined with VAEs, PGAs can generate sharper samples than vanilla VAEs.[1]

[1] Code is available at `https://bit.ly/2U0kRYL`.

We summarize our main contributions as follows:

- A training framework, PGA, for generative autoencoders is developed to match the target distribution in the latent space, which, we prove, ensures the matching in the data space.
- We combine the PGA framework with maximum likelihood, and remove the restrictions of reversible generative models on neural network architectures and latent dimensionalities.
- We combine the PGA framework with VAE, which solves the problem of blurry samples, without introducing any auxiliary models or sophisticated model architectures.

2 Related Work

Autoencoder-based generative models are trained by minimizing an input reconstruction loss with regularizations. As an early approach, denoising autoencoders (DAEs) (Vincent et al., 2008) are trained to recover the original input from an intentionally corrupted input. Then a generative model can be obtained by sampling from a Markov chain (Bengio et al., 2013). To sample from a decoder directly, most recent approaches resort to mapping a simple prior distribution to a data distribution using the decoder. For instance, adversarial autoencoders (AAEs) (Makhzani et al., 2016) and Wasserstein autoencoders (WAEs) (Tolstikhin et al., 2018) attempt to match the aggregated posterior and the prior, either by adversarial training or by minimizing their Wasserstein distance. However, due to the use of deterministic encoders, there can be "holes" in the latent space that are not covered by the aggregated posterior, which would result in poor sample quality (Rubenstein et al., 2018). By using stochastic encoders and variational inference, variational autoencoders (VAEs) are likely to suffer less from this problem, but are known to generate blurry samples (Rezende & Viola, 2018; Dai & Wipf, 2019). Nevertheless, as we will show, the latter problem can be addressed by moving the VAE reconstruction loss from the data space to the latent space.

In a different line of work, reversible generative models (Dinh et al., 2014; 2017; Kingma & Dhariwal, 2018) are developed to enable exact inference. Consequently, by the change of variables theorem, the likelihood of each data sample can be exactly computed and optimized. However, to avoid expensive Jacobian determinant computations, reversible models can only be composed of restricted transformations, rather than general neural network architectures. While this restriction can be relaxed by utilizing recently developed neural ordinary differential equations (Chen et al., 2018; Grathwohl et al., 2019), they still rely on a shared dimensionality between latent and data spaces, which remains an unnatural restriction. In this work, we use the proposed training framework to trade exact inference for unrestricted neural network architectures and arbitrary latent dimensionalities, generalizing maximum likelihood training to autoencoder-based models.

3 Methods

3.1 Perceptual Generative Model

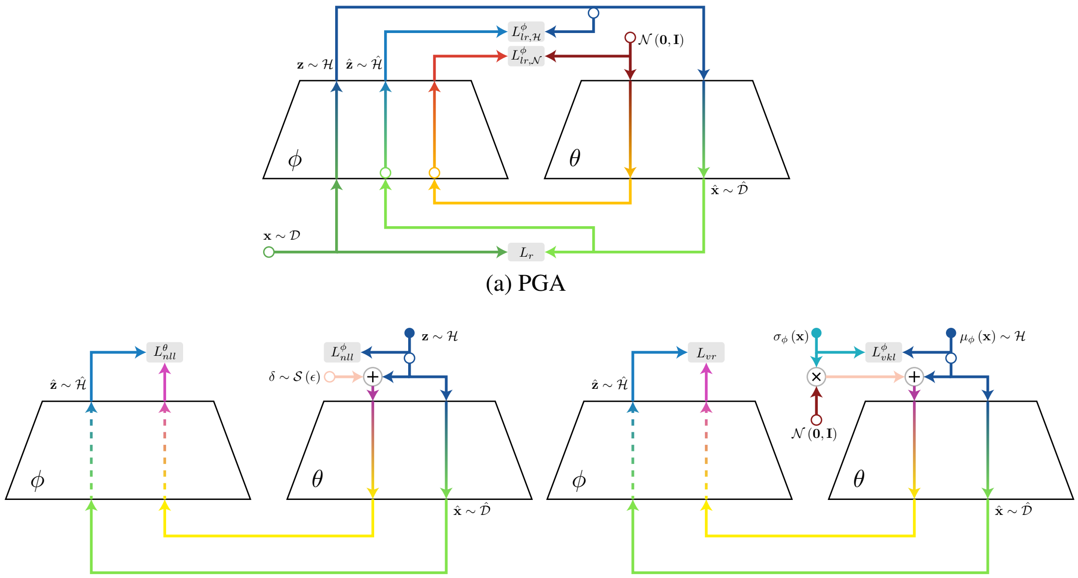
Let $f : \mathbb{R}^D \to \mathbb{R}^H$ be the encoder parameterized by ϕ, and $g : \mathbb{R}^H \to \mathbb{R}^D$ be the decoder parameterized by θ. Our goal is to obtain a generative model, which maps a simple prior distribution to a target data distribution, $\mathcal{D}$. Throughout this paper, we use $\mathcal{N}(\mathbf{0}, \mathbf{I})$ as the prior distribution. See Appendix A for a summary of notations.

For $\mathbf{z} \in \mathbb{R}^H$, the output of the decoder, $g(\mathbf{z})$, lies in a manifold that is at most H-dimensional. Therefore, if we train the autoencoder to minimize

$$L_r = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\|\hat{\mathbf{x}} - \mathbf{x}\|_2^2\right], \tag{1}$$

where $\hat{\mathbf{x}} = g(f(\mathbf{x}))$, then $\hat{\mathbf{x}}$ can be seen as a projection of the input data, $\mathbf{x}$, onto the manifold of $g(\mathbf{z})$. Let $\hat{\mathcal{D}}$ denote the distribution of $\hat{\mathbf{x}}$. Given enough capacity of the encoder, $\hat{\mathcal{D}}$ is the best approximation to $\mathcal{D}$ (in terms of ℓ_2-distance), that we can obtain from the decoder, and thus can serve as a surrogate target for training the generator.

Due to the difficulty of directly training the generator to match $\hat{\mathcal{D}}$, we seek to map $\hat{\mathcal{D}}$ to the latent space, and train the generator to match the mapped distribution, $\hat{\mathcal{H}}$, in the latent space. To this end,


$\mathbf{z}\sim\mathcal{H}$
$\hat{\mathbf{z}}\sim\hat{\mathcal{H}}$
$L^{\phi}_{lr,\mathcal{H}}$
$L^{\phi}_{lr,\mathcal{N}}$
$\mathcal{N}(\mathbf{0},\mathbf{I})$
ϕ
θ
$\hat{\mathbf{x}}\sim\hat{\mathcal{D}}$
$\mathbf{x}\sim\mathcal{D}$
L_r
$\hat{\mathbf{z}}\sim\hat{\mathcal{H}}$
L^{θ}_{nll}
L^{ϕ}_{nll}
$\mathbf{z}\sim\mathcal{H}$
$\delta\sim\mathcal{S}(\epsilon)$
ϕ
θ
$\hat{\mathbf{x}}\sim\hat{\mathcal{D}}$
$\hat{\mathbf{z}}\sim\hat{\mathcal{H}}$
L_{vr}
$\sigma_{\phi}(\mathbf{x})$
L^{ϕ}_{vkl}
$\mu_{\phi}(\mathbf{x})\sim\mathcal{H}$
$\mathcal{N}(\mathbf{0},\mathbf{I})$
ϕ
θ
$\hat{\mathbf{x}}\sim\hat{\mathcal{D}}$

(a) PGA

(b) LPGA

(c) VPGA

Figure 1: Illustration of the training process of PGAs. The overall loss function consists of (a) the basic PGA losses, and either (b) the LPGA-specific losses or (c) the VPGA-specific losses. Circles indicate where the gradient is truncated, and dashed lines indicate where the gradient is ignored when updating parameters.

we reuse the encoder for mapping $\hat{\mathcal{D}}$ to $\hat{\mathcal{H}}$, and train the generator such that $h(\cdot) = f(g(\cdot))$ maps $\mathcal{N}(\mathbf{0},\mathbf{I})$ to $\hat{\mathcal{H}}$. In addition, to ensure that g maps $\mathcal{N}(\mathbf{0},\mathbf{I})$ to $\hat{\mathcal{D}}$, we minimize the following latent reconstruction loss w.r.t. ϕ:

$$L^{\phi}_{lr,\mathcal{N}} = \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\|h(\mathbf{z}) - \mathbf{z}\|_2^2\right]. \tag{2}$$

Formally, let $Z(\mathbf{x})$ be the set of all $\mathbf{z}$'s that are mapped to the same $\mathbf{x}$ by the decoder, we have the following theorem:

Theorem 1. *Assuming $\mathbb{E}[\mathbf{z}|\mathbf{x}] \in Z(\mathbf{x})$ for all $\mathbf{x}$ generated by the decoder, and sufficient capacity of the encoder; for $\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$, if Eq. (2) is minimized and $h(\mathbf{z})\sim\hat{\mathcal{H}}$, then $g(\mathbf{z})\sim\hat{\mathcal{D}}$.*

Proof sketch. We first show that any different $\mathbf{x}$'s generated by the decoder are mapped to different $\mathbf{z}$'s by the encoder. Let $\mathbf{x}_1 = g(\mathbf{z}_1)$, $\mathbf{x}_2 = g(\mathbf{z}_2)$, and $\mathbf{x}_1 \neq \mathbf{x}_2$. Since the encoder has sufficient capacity and Eq. (2) is minimized, we have $f(\mathbf{x}_1) = \mathbb{E}[\mathbf{z}_1 \mid \mathbf{x}_1]$ and $f(\mathbf{x}_2) = \mathbb{E}[\mathbf{z}_2 \mid \mathbf{x}_2]$. By assumption, $f(\mathbf{x}_1) \in Z(\mathbf{x}_1)$ and $f(\mathbf{x}_2) \in Z(\mathbf{x}_2)$. Therefore, since $Z(\mathbf{x}_1) \cap Z(\mathbf{x}_2) = \varnothing$, we have $f(\mathbf{x}_1) \neq f(\mathbf{x}_2)$.

For $\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$, denote the distributions of $g(\mathbf{z})$ and $h(\mathbf{z})$, respectively, by $\widetilde{\mathcal{D}}$ and $\widetilde{\mathcal{H}}$. We then consider the case where $\widetilde{\mathcal{D}}$ and $\hat{\mathcal{D}}$ are discrete distributions. If $g(\mathbf{z}) \nsim \hat{\mathcal{D}}$, then there exists an $\mathbf{x}$ that is generated by the decoder, such that $p_{\widetilde{\mathcal{H}}}(f(\mathbf{x})) = p_{\widetilde{\mathcal{D}}}(\mathbf{x}) \neq p_{\hat{\mathcal{D}}}(\mathbf{x}) = p_{\hat{\mathcal{H}}}(f(\mathbf{x}))$, contradicting that $h(\mathbf{z})\sim\hat{\mathcal{H}}$. The result still holds when $\widetilde{\mathcal{D}}$ and $\hat{\mathcal{D}}$ approach continuous distributions, in which case $\widetilde{\mathcal{D}} = \hat{\mathcal{D}}$ almost everywhere. □

Note that the two distributions compared in Theorem 1, $\widetilde{\mathcal{D}}$ and $\hat{\mathcal{D}}$, are mapped respectively from $\mathcal{N}(\mathbf{0},\mathbf{I})$ and $\mathcal{H}$, the aggregated posterior of $f(\mathbf{x})$ given $\mathbf{x}\sim\mathcal{D}$. While $\mathcal{N}(\mathbf{0},\mathbf{I})$ is supported on the whole $\mathbb{R}^H$, there can be $\mathbf{z}$'s with low probabilities in $\mathcal{N}(\mathbf{0},\mathbf{I})$, but with high probabilities in $\mathcal{H}$, which are not well covered by Eq. (2). Therefore, it is sometimes helpful to minimize another latent reconstruction loss on $\mathcal{H}$:

$$L^{\phi}_{lr,\mathcal{H}} = \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim\mathcal{H}}\left[\|\hat{\mathbf{z}} - \mathbf{z}\|_2^2\right], \tag{3}$$

where $\hat{\mathbf{z}} = h(\mathbf{z}) \sim \hat{\mathcal{H}}$. In practice, we observe that $L^{\phi}_{lr,\mathcal{H}}$ is often small without explicit minimization, which we attribute to its consistency with the minimization of L_r.

By Theorem 1, the problem of training the generative model reduces to training h to map $\mathcal{N}(\mathbf{0}, \mathbf{I})$ to $\hat{\mathcal{H}}$, which we refer to as the perceptual generative model. In the subsequent subsections, we present a maximum likelihood approach, a VAE-based approach, and a unified approach to train the perceptual generative model. The basic loss function of PGAs is given by

$$L_{pga} = L_r + \alpha L^{\phi}_{lr,\mathcal{N}} + \beta L^{\phi}_{lr,\mathcal{H}}, \tag{4}$$

where α and β are hyperparameters to be tuned. Eq. (4) is also illustrated in Fig. 1a.

3.2 A Maximum Likelihood Approach

We first assume the invertibility of h. For $\hat{\mathbf{x}} \sim \hat{\mathcal{D}}$, let $\hat{\mathcal{H}}$ be the distribution of $f(\hat{\mathbf{x}})$. We can train h directly with maximum likelihood using the change of variables formula as

$$\mathbb{E}_{\hat{\mathbf{z}} \sim \hat{\mathcal{H}}} [\log p(\hat{\mathbf{z}})] = \mathbb{E}_{\mathbf{z} \sim \mathcal{H}} \left[\log p(\mathbf{z}) - \log \left| \det \left(\frac{\partial h(\mathbf{z})}{\partial \mathbf{z}} \right) \right| \right]. \tag{5}$$

Ideally, we would like to maximize Eq. (5) w.r.t. the parameters of the generator (or decoder), θ. However, directly optimizing the first term in Eq. (5) requires computing $\mathbf{z} = h^{-1}(\hat{\mathbf{z}})$, which is usually unknown. Nevertheless, for $\hat{\mathbf{z}} \sim \hat{\mathcal{H}}$, we have $h^{-1}(\hat{\mathbf{z}}) = f(\mathbf{x})$ and $\mathbf{x} \sim \mathcal{D}$, and thus we can minimize the following loss function w.r.t. ϕ instead:

$$L^{\phi}_{nll} = -\mathbb{E}_{\mathbf{z} \sim \mathcal{H}} [\log p(\mathbf{z})] = \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim \mathcal{D}} \left[\| f(\mathbf{x}) \|_2^2 \right]. \tag{6}$$

To avoid computing the Jacobian in the second term of Eq. (5), which is slow for unrestricted architectures, we approximate the Jacobian determinant and derive a loss function for the decoder as

$$L^{\theta}_{nll} = \frac{H}{2} \mathbb{E}_{\mathbf{z} \sim \mathcal{H}, \delta \sim \mathcal{S}(\epsilon)} \left[\log \frac{\| h(\mathbf{z} + \delta) - h(\mathbf{z}) \|_2^2}{\| \delta \|_2^2} \right] \approx \mathbb{E}_{\mathbf{z} \sim \mathcal{H}} \left[\log \left| \det \left(\frac{\partial h(\mathbf{z})}{\partial \mathbf{z}} \right) \right| \right], \tag{7}$$

where $\mathcal{S}(\epsilon)$ can be either $\mathcal{N}\left(\mathbf{0}, \epsilon^2 \mathbf{I}\right)$, or a uniform distribution on a small $(H-1)$-sphere of radius ϵ centered at the origin. The latter choice is expected to introduce slightly less variance. We show below that the approximation gives an upper bound when $\epsilon \to 0$. Eqs. (6) and (7) are illustrated in Fig. 1b.

Proposition 1. *For $\epsilon \to 0$,*

$$\log \left| \det \left(\frac{\partial h(\mathbf{z})}{\partial \mathbf{z}} \right) \right| \leq \frac{H}{2} \mathbb{E}_{\delta \sim \mathcal{S}(\epsilon)} \left[\log \frac{\| h(\mathbf{z} + \delta) - h(\mathbf{z}) \|_2^2}{\| \delta \|_2^2} \right]. \tag{8}$$

The inequality is tight if h is a multiple of the identity function around $\mathbf{z}$.

We defer the proof to Appendix B. We note that the above discussion relies on the invertibility of h, which, however, is not required by the resulting method. Indeed, when h is invertible at some point $\mathbf{z}$, the latent reconstruction loss ensures that h is close to the identity function around $\mathbf{z}$, and hence the tightness of the upper bound in Eq. (8). Otherwise, when h is not invertible at some $\mathbf{z}$, the logarithm of the Jacobian determinant at $\mathbf{z}$ becomes infinite, in which case Eq. (5) cannot be optimized. Nevertheless, since $\| h(\mathbf{z} + \delta) - h(\mathbf{z}) \|_2^2$ is unlikely to be zero if the model is properly initialized, the approximation in Eq. (7) remains finite, and thus can be optimized regardless.

To summarize, we train the autoencoder to obtain a generative model by minimizing the following loss function:

$$L_{lpga} = L_{pga} + \gamma \left(L^{\phi}_{nll} + L^{\theta}_{nll} \right). \tag{9}$$

We refer to this approach as maximum likelihood PGA (LPGA).

3.3 A VAE-based Approach

The original VAE is trained by maximizing the evidence lower bound on $\log p(\mathbf{x})$ as

$$\begin{aligned}\log p(\mathbf{x}) &\geq \log p(\mathbf{x}) - \mathbb{KL}(q(\mathbf{z}' \mid \mathbf{x}) \;||\; p(\mathbf{z}' \mid \mathbf{x})) \\ &= \mathbb{E}_{\mathbf{z}' \sim q(\mathbf{z}' \mid \mathbf{x})}\left[\log p(\mathbf{x} \mid \mathbf{z}')\right] - \mathbb{KL}(q(\mathbf{z}' \mid \mathbf{x}) \;||\; p(\mathbf{z}')),\end{aligned} \tag{10}$$

where $p(\mathbf{x} \mid \mathbf{z}')$ is modeled with the decoder, and $q(\mathbf{z}' \mid \mathbf{x})$ is modeled with the encoder. Note that $\mathbf{z}'$ denotes the stochastic version of $\mathbf{z}$, whereas $\mathbf{z}$ remains deterministic for the basic PGA losses in Eqs. (2) and (3). In our case, we would like to modify Eq. (10) in a way that helps maximize $\log p(\hat{\mathbf{z}})$. Therefore, we replace $p(\mathbf{x} \mid \mathbf{z}')$ on the r.h.s. of Eq. (10) with $p(\hat{\mathbf{z}} \mid \mathbf{z}')$, and derive a lower bound on $\log p(\hat{\mathbf{z}})$ as

$$\begin{aligned}\log p(\hat{\mathbf{z}}) &\geq \log p(\hat{\mathbf{z}}) - \mathbb{KL}(q(\mathbf{z}' \mid \mathbf{x}) \;||\; p(\mathbf{z}' \mid \hat{\mathbf{z}})) \\ &= \mathbb{E}_{\mathbf{z}' \sim q(\mathbf{z}' \mid \mathbf{x})}\left[\log p(\hat{\mathbf{z}} \mid \mathbf{z}')\right] - \mathbb{KL}(q(\mathbf{z}' \mid \mathbf{x}) \;||\; p(\mathbf{z}')).\end{aligned} \tag{11}$$

Similar to the original VAE, we make the assumption that $q(\mathbf{z}' \mid \mathbf{x})$ and $p(\hat{\mathbf{z}} \mid \mathbf{z}')$ are Gaussian; i.e., $q(\mathbf{z}' \mid \mathbf{x}) = \mathcal{N}\big(\mathbf{z}' \;\big|\; \mu_\phi(\mathbf{x}), \operatorname{diag}\big(\sigma_\phi^2(\mathbf{x})\big)\big)$, and $p(\hat{\mathbf{z}} \mid \mathbf{z}') = \mathcal{N}\left(\hat{\mathbf{z}} \;\middle|\; \mu_{\theta,\phi}(\mathbf{z}'), \sigma^2\mathbf{I}\right)$. Here, $\mu_\phi(\cdot) = f(\cdot)$, $\mu_{\theta,\phi}(\cdot) = h(\cdot)$, and $\sigma > 0$ is a tunable scalar. Note that if σ is fixed, the first term on the r.h.s. of Eq. (11) has a trivial minimum, where $\mathbf{z}$, $\hat{\mathbf{z}}$, and the reconstruction of $\mathbf{z}'$ are all close to zero. To circumvent this, we set σ proportional to the ℓ_2-norm of $\mathbf{z}$.

The VAE variant is trained by minimizing

$$L_{vae} = L_{vr} + L_{vkl}^{\phi} = -\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\mathbb{E}_{\mathbf{z}' \sim q(\mathbf{z}' \mid \mathbf{x})}\left[\log p(\hat{\mathbf{z}} \mid \mathbf{z}')\right] - \mathbb{KL}(q(\mathbf{z}' \mid \mathbf{x}) \;||\; p(\mathbf{z}'))\right], \tag{12}$$

where L_{vr} and L_{vkl}^{ϕ} are, respectively, the reconstruction and KL divergence losses of VAE, as illustrated in Fig. 1c. Accordingly, the overall loss function is given by

$$L_{vpga} = L_{pga} + \eta L_{vae}. \tag{13}$$

We refer to this approach as variational PGA (VPGA).

3.4 A Unified Approach

While the loss functions of maximum likelihood and VAE seem completely different in their original forms, they share remarkable similarities when considered in the PGA framework (see Figs. 1b and 1c). Intuitively, observe that

$$L_{vkl}^{\phi} = L_{nll}^{\phi} + \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \sum_{i\in[H]} \left[\sigma_{\phi,i}^2(\mathbf{x}) - \log\left(\sigma_{\phi,i}^2(\mathbf{x})\right)\right], \tag{14}$$

which means both L_{nll}^{ϕ} and L_{vkl}^{ϕ} tend to attract the latent representations of data samples to the origin. In addition, L_{nll}^{θ} expands the volume occupied by each sample in the latent space, which can be also done by L_{vr} with the second term of Eq. (14).

More concretely, we observe that both L_{nll}^{θ} and L_{vr} are minimizing the difference between $h(\mathbf{z})$ and $h(\mathbf{z}+\delta')$, where δ' is some additive zero-mean noise. However, they differ in that the variance of δ' is fixed for L_{nll}^{θ}, but is trainable for L_{vr}; and the distance between $h(\mathbf{z})$ and $h(\mathbf{z}+\delta')$ are defined in two different ways. In fact, L_{vr} is a squared ℓ_2-distance derived from the Gaussian assumption on $\hat{\mathbf{z}}$, whereas L_{nll}^{θ} can be derived similarly by assuming that $d^H = \|\hat{\mathbf{z}} - h(\mathbf{z}+\delta)\|_2^H$ follows a reciprocal distribution as

$$p\left(d^H; a, b\right) = \frac{1}{d^H(\log(b) - \log(a))}, \tag{15}$$

where $a \leq d^H \leq b$, and $a > 0$. The exact values of a and b are irrelevant, as they only appear in an additive constant when we take the logarithm of $p\left(d^H; a, b\right)$.

Since there is no obvious reason for assuming Gaussian $\hat{\mathbf{z}}$, we can instead assume $\hat{\mathbf{z}}$ to follow the distribution defined in Eq. (15), and multiply H by a tunable scalar, γ', similar to σ. Furthermore, we can replace δ in Eq. (7) with $\delta' \sim \mathcal{N}\big(\mathbf{0}, \operatorname{diag}\big(\sigma_\phi^2(\mathbf{x})\big)\big)$, as it is defined for VPGA with a subtle difference that $\sigma_\phi^2(\mathbf{x})$ is constrained to be greater than ϵ^2. As a result, LPGA and VPGA are unified into a single approach, which has a combined loss function as

$$L_{lvpga} = L_{pga} + \gamma' L_{vr} + \gamma L_{nll}^{\phi} + \eta L_{vkl}^{\phi}. \tag{16}$$

When $\gamma' = \gamma$ and $\eta = 0$, Eq. (16) is equivalent to Eq. (9), considering that $\sigma_\phi^2(\mathbf{x})$ will be optimized to approach ϵ^2. Similarly, when $\gamma = 0$, Eq. (16) is equivalent to Eq. (13). Interestingly, it also becomes possible to have a mix of LPGA and VPGA by setting all three hyperparameters to positive values. We refer to this approach as LVPGA.

4 Experiments

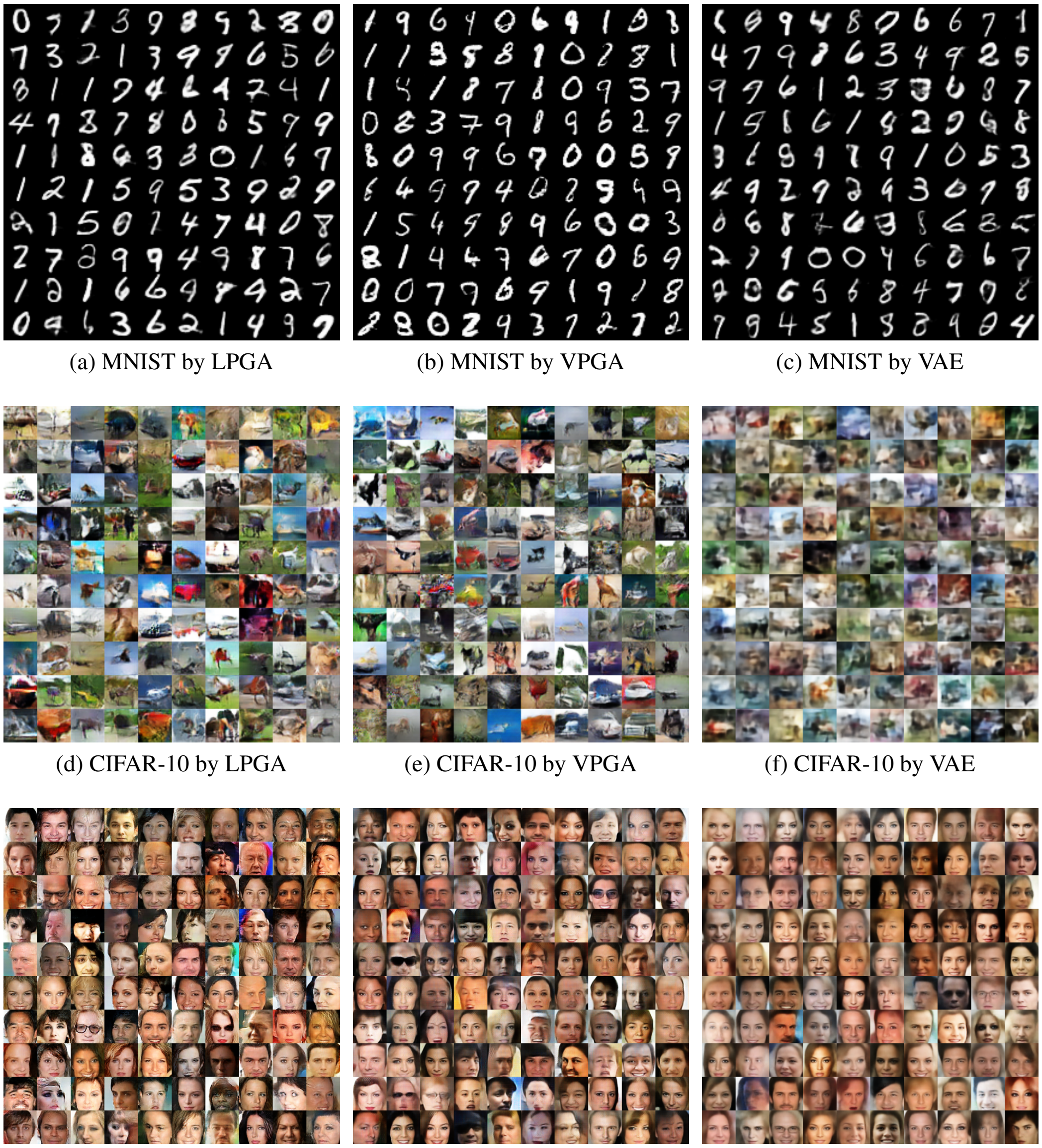
In this section, we evaluate the performance of LPGA and VPGA on three image datasets, MNIST (LeCun et al., 1998), CIFAR-10 (Krizhevsky & Hinton, 2009), and CelebA (Liu et al., 2015). For CelebA, we employ the discriminator and generator architecture of DCGAN (Radford et al., 2016) for the encoder and decoder of PGA. We half the number of filters (i.e., 64 filters for the first convolutional layer) for faster experiments, while more filters are observed to improve performance. Due to smaller input sizes, we reduce the number of convolutional layers accordingly for MNIST and CIFAR-10, and add a fully-connected layer of 1024 units for MNIST, as done in Chen et al. (2016). SGD with a momentum of 0.9 is used to train all models. Other hyperparameters are tuned heuristically, and could be improved by a more extensive grid search. For fair comparison, σ is tuned for both VAE and VPGA. All experiments are performed on a single GPU.

As shown in Fig. 2, the visual quality of the PGA-generated samples is significantly improved over that of VAEs. In particular, PGAs generate much sharper samples on CIFAR-10 and CelebA compared to vanilla VAEs. The results of LVPGA much resemble that of either LPGA or VPGA, depending on the hyperparameter settings. In addition, we use the Fréchet Inception Distance (FID) (Heusel et al., 2017) to evaluate the proposed methods, as well as VAE. For each model and each dataset, we take 5,000 generated samples to compute the FID score. The results (with standard errors of 3 or more runs) are summarized in Table. 1. Compared to other autoencoder-based non-adversarial approaches (Tolstikhin et al., 2018; Kolouri et al., 2019; Ghosh et al., 2019), where similar but larger architectures are used, we obtain substantially better FID scores on CIFAR-10 and CelebA. Note that the results from Ghosh et al. (2019) shown in Table. 1 are obtained using slightly different architectures and evaluation protocols. Nevertheless, their results of VAE align well with ours, suggesting a good comparability of the results. Interestingly, as a unified approach, LVPGA can indeed combine the best performances of LPGA and VPGA on different datasets. For CelebA, we further show results on 140x140 crops in Fig. 5, and latent space interpolations in Fig. 6 (see Appendix C).

Table 1: FID scores of autoencoder-based generative models. The first block shows the results from Ghosh et al. (2019), where CV-VAE stands for constant-variance VAE, and RAE stands for regularized autoencoder. The second block shows our results of LPGA, VPGA, LVPGA, and VAE.

Model	MNIST	CIFAR-10	CelebA
VAE	19.21	106.37	48.12
CV-VAE	33.79	94.75	48.87
WAE	20.42	117.44	53.67
RAE-L2	22.22	80.80	51.13
RAE-SN	19.67	84.25	44.74
VAE	15.55 ± 0.18	115.74 ± 0.63	43.60 ± 0.33
LPGA	12.06 ± 0.12	55.87 ± 0.25	14.53 ± 0.52
VPGA	11.67 ± 0.21	$\mathbf{51.51} \pm 1.16$	24.73 ± 1.25
LVPGA	$\mathbf{11.45} \pm 0.25$	52.94 ± 0.89	$\mathbf{13.80} \pm 0.20$

The training process of PGAs is stable in general, given the non-adversarial losses. As shown in Fig. 3a, the total losses change little after the initial rapid drops. This is due to the fact that the encoder and decoder are optimized towards different objectives, as can be observed from Eqs. (4), (9), and (12). In contrast, the corresponding FIDs, shown in Fig. 3b, tend to decrease monotonically during training. However, when trained on CelebA, there is a significant performance gap between



(a) MNIST by LPGA (b) MNIST by VPGA (c) MNIST by VAE

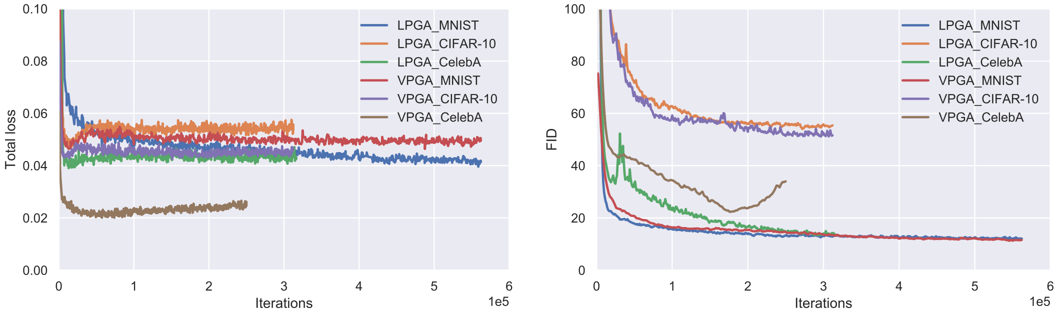
(d) CIFAR-10 by LPGA (e) CIFAR-10 by VPGA (f) CIFAR-10 by VAE

(g) CelebA by LPGA (h) CelebA by VPGA (i) CelebA by VAE

Figure 2: Random samples generated by LPGA, VPGA, and VAE.

LPGA and VPGA, and the FID of the latter starts to increase after a certain point of training. We suspect this phenomenon is related to the limited expressiveness of the variational posterior, which is not an issue for LPGA.

It is worth noting that stability issues can occur when batch normalization (Ioffe & Szegedy, 2015) is introduced, since both the encoder and decoder are fed with multiple batches drawn from different distributions. At convergence, different input distributions to the generator (e.g., $\mathcal{H}$ and $\mathcal{N}(\mathbf{0}, \mathbf{I})$) are expected to result in similar distributions of the internal representations, which, intriguingly, can be imposed to some degree by batch normalization. Therefore, it is observed that when batch normalization does not cause stability issues, it can substantially accelerate convergence and lead to slightly better generative performance. Furthermore, we observe that LPGA tends to be more stable than VPGA in the presence of batch normalization.

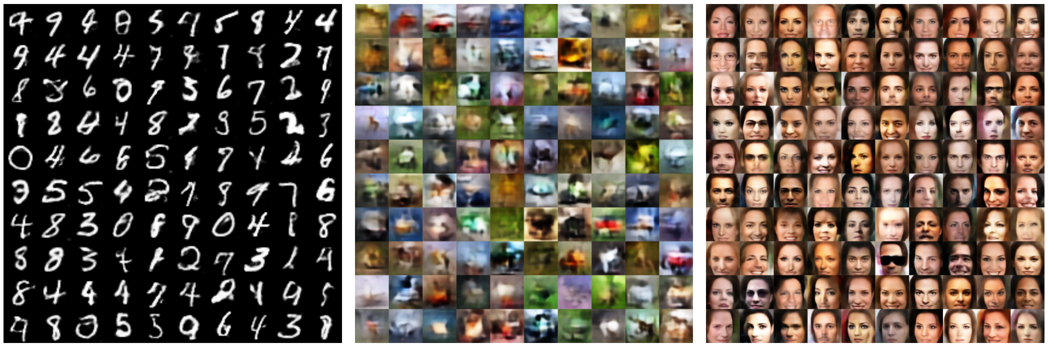


(a) Total loss

(b) FID

Figure 3: Training curves of LPGA and VPGA.

Finally, we conduct an ablation study. While the loss functions of LPGA and VPGA both consist of multiple components, they are all theoretically motivated and indispensable. Specifically, the input reconstruction loss minimizes the discrepancy between the input data and its reconstruction. Since the reconstructed data distribution serves as the surrogate target distribution, removing the input reconstruction loss will result in a random target. Moreover, removing the maximum likelihood loss of LPGA or the VAE loss of VPGA will leave the perceptual generative model untrained. In both cases, no valid generative model can be obtained. Nevertheless, it is interesting to see how the latent reconstruction loss contributes to the generative performance. Therefore, we retrain the LPGAs without the latent reconstruction loss and report the results in Fig. 4. Compared to Fig. 2a, 2d, 2g, and the results in Table 1, the performance significantly degrades both visually and quantitatively, confirming the importance of the latent reconstruction loss.



(a) MNIST, FID $= 18.13$

(b) CIFAR-10, FID $= 114.19$

(c) CelebA, FID $= 48.02$

Figure 4: Random samples generated by LPGA with $\alpha = \beta = 0$.

5 Conclusion

We proposed a framework, PGA, for training autoencoder-based generative models, with non-adversarial losses and unrestricted neural network architectures. By matching target distributions in the latent space, PGAs trained with maximum likelihood generalize the idea of reversible generative models to unrestricted neural network architectures and arbitrary latent dimensionalities. In addition, it improves the performance of VAE when combined together. Under the PGA framework, we further show that maximum likelihood and VAE can be unified into a single approach.

In principle, the PGA framework can be combined with any method that can train the perceptual generative model. While we have only considered non-adversarial approaches, an interesting future work would be to combine it with an adversarial discriminator trained on latent representations. Moreover, the compatibility issue with batch normalization deserves further investigation.

REFERENCES

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, 2017.

Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pp. 899–907, 2013.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pp. 6571–6583, 2018.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pp. 2172–2180, 2016.

Bin Dai and David Wipf. Diagnosing and enhancing vae models. In *International Conference on Learning Representations*, 2019.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. In *International Conference on Learning Representations Workshop*, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2017.

Partha Ghosh, Mehdi S. M. Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. *arXiv preprint arXiv:1903.12436*, 2019.

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*, pp. 2672–2680, 2014.

Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2019.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

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 equilibrium. In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10236–10245, 2018.

Soheil Kolouri, Phillip E Pope, Charles E Martin, and Gustavo K Rohde. Sliced wasserstein autoencoders. In *International Conference on Learning Representations*, 2019.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Yann LeCun, Corinna Cortes, and Chris J. C. Burges. The mnist handwritten digit database, 1998.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *International Conference on Computer Vision*, 2015.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. In *International Conference on Learning Representations Workshop*, 2016.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016.

Danilo Jimenez Rezende and Fabio Viola. Taming vaes. *arXiv preprint arXiv:1810.00597*, 2018.

Paul K Rubenstein, Bernhard Schoelkopf, and Ilya Tolstikhin. On the latent space of wasserstein auto-encoders. *arXiv preprint arXiv:1802.03761*, 2018.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pp. 2234–2242, 2016.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, pp. 1096–1103. ACM, 2008.

A NOTATIONS

Table 2: Notations and definitions

f/g	encoder/decoder of an autoencoder
h	$h = f \circ g$
ϕ/θ	parameters of the encoder/decoder
D/H	dimensionality of the data/latent space
$\mathcal{D}$	distribution of data samples denoted by $\mathbf{x}$
$\mathcal{H}$	distribution of $f(\mathbf{x})$ for $\mathbf{x} \sim \mathcal{D}$
$\hat{\mathcal{D}}$	distribution of $\hat{\mathbf{x}} = g(f(\mathbf{x}))$ for $\mathbf{x} \sim \mathcal{D}$
$\hat{\mathcal{H}}$	distribution of $\hat{\mathbf{z}} = h(\mathbf{z})$ for $\mathbf{z} \sim \mathcal{H}$
L_r	standard reconstruction loss of the autoencoder
$L_{lr,\mathcal{N}}^{\phi}$	latent reconstruction loss of PGA for $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, applied on the encoder
$L_{lr,\mathcal{H}}^{\phi}$	latent reconstruction loss of PGA for $\mathbf{z} \sim \mathcal{H}$, applied on the encoder
L_{nll}^{ϕ}	part of the negative log-likelihood loss of LPGA, applied on the encoder
L_{nll}^{θ}	part of the negative log-likelihood loss of LPGA, applied on the decoder
L_{vr}	VAE reconstruction loss of VPGA
L_{vkl}	VAE KL-divergence loss of VPGA
L_{vae}	$L_{vae} = L_{vr} + L_{vkl}$, VAE loss of VPGA

B Proof of Proposition 1

Proof. Let $\mathbf{J}(\mathbf{z}) = \partial h(\mathbf{z}) / \partial \mathbf{z}$, $\mathbf{P} = [\delta_1 \quad \delta_2 \quad \cdots \quad \delta_H]$, and $\hat{\mathbf{P}} = \mathbf{J}(\mathbf{z}) \mathbf{P} = \left[\hat{\delta}_1 \quad \hat{\delta}_2 \quad \cdots \quad \hat{\delta}_H\right]$, where $\Delta = \{\delta_1, \delta_2, \ldots, \delta_H\}$ is an orthogonal set of H-dimensional vectors. Since $\det\left(\hat{\mathbf{P}}\right) = \det(\mathbf{J}(\mathbf{z})) \det(\mathbf{P})$, we have

$$\log|\det(\mathbf{J}(\mathbf{z}))| = \log\left|\det\left(\hat{\mathbf{P}}\right)\right| - \log|\det(\mathbf{P})|. \tag{17}$$

By the geometric interpretation of determinants, the volume of the parallelotope spanned by Δ is

$$\operatorname{Vol}(\Delta) = |\det(\mathbf{P})| = \prod_{i \in [H]} \|\delta_i\|_2, \tag{18}$$

where $[H] = \{1, 2, \ldots, H\}$. While $\hat{\Delta} = \left\{\hat{\delta}_1, \hat{\delta}_2, \ldots, \hat{\delta}_H\right\}$ is not necessarily an orthogonal set, an upper bound on $\operatorname{Vol}\left(\hat{\Delta}\right)$ can be derived in a similar fashion. Let $\hat{\Delta}_k = \left\{\hat{\delta}_1, \hat{\delta}_2, \ldots, \hat{\delta}_k\right\}$, and a_k be the included angle between $\hat{\delta}_k$ and the plane spanned by $\hat{\Delta}_{k-1}$. We have

$$\operatorname{Vol}\left(\hat{\Delta}_2\right) = \left\|\hat{\delta}_1\right\|_2 \left\|\hat{\delta}_2\right\|_2 \sin a_2, \text{and } \operatorname{Vol}\left(\hat{\Delta}_k\right) = \operatorname{Vol}\left(\hat{\Delta}_{k-1}\right) \left\|\hat{\delta}_k\right\|_2 \sin a_k. \tag{19}$$

Given fixed $\left\|\hat{\delta}_k\right\|_2, \forall k \in [H]$, $\operatorname{Vol}\left(\hat{\Delta}_2\right)$ is maximized when $a_2 = \pi/2$, i.e., $\hat{\delta}_1$ and $\hat{\delta}_2$ are orthogonal; and $\operatorname{Vol}\left(\hat{\Delta}_k\right)$ is maximized when $\operatorname{Vol}\left(\hat{\Delta}_{k-1}\right)$ is maximized and $a_k = \pi/2$. By induction on k, we can conclude that $\operatorname{Vol}\left(\hat{\Delta}\right)$ is maximized when $\hat{\Delta} = \hat{\Delta}_H$ is an orthogonal set, and therefore

$$\operatorname{Vol}\left(\hat{\Delta}\right) = \left|\det\left(\hat{\mathbf{P}}\right)\right| \leq \prod_{i \in [H]} \left\|\hat{\delta}_i\right\|_2. \tag{20}$$

Combining Eq. (17) with Eqs. (18) and (20), we obtain

$$\log|\det(\mathbf{J}(\mathbf{z}))| \leq \sum_{i \in [H]} \left(\log\left\|\hat{\delta}_i\right\|_2 - \log\|\delta_i\|_2\right). \tag{21}$$

We proceed by randomizing Δ. Let $\Delta_k = \{\delta_1, \delta_2, \ldots, \delta_k\}$. We inductively construct an orthogonal set, $\Delta = \Delta_H$. In step 1, δ_1 is sampled from $\mathcal{S}(\epsilon)$, a uniform distribution on a $(H-1)$-sphere of radius ϵ, $S(\epsilon)$, centered at the origin of an H-dimensional space. In step k, δ_k is sampled from $\mathcal{S}(\epsilon; \Delta_{k-1})$, a uniform distribution on an $(H-k)$-sphere, $S(\epsilon; \Delta_{k-1})$, in the orthogonal complement of the space spanned by Δ_{k-1}. Step k is repeated until H mutually orthogonal vectors are obtained.

Obviously, when $k = H - 1$, for all $j > k$ and $j \leq H$, $p(\delta_j \mid \Delta_k) = p(\delta_j \mid \Delta_{H-1}) = \mathcal{S}(\delta_j \mid \epsilon; \Delta_{H-1}) = \mathcal{S}(\delta_j \mid \epsilon; \Delta_k)$. When $1 \leq k < H$, assuming for all $j > k$ and $j \leq H$, $p(\delta_j \mid \Delta_k) = \mathcal{S}(\delta_j \mid \epsilon; \Delta_k)$, we get

$$p(\delta_j \mid \Delta_{k-1}) = \int_{S(\epsilon; \Delta_{k-1} \cup \{\delta_j\})} p(\delta_k \mid \Delta_{k-1}) \, p(\delta_j \mid \Delta_k) \, d\delta_k, \tag{22}$$

where $S(\epsilon; \Delta_{k-1} \cup \{\delta_j\})$ is in the orthogonal complement of the space spanned by $\Delta_{k-1} \cup \{\delta_j\}$. Since $p(\delta_k \mid \Delta_{k-1})$ is a constant on $S(\delta_k \mid \epsilon; \Delta_{k-1})$, and $S(\epsilon; \Delta_{k-1} \cup \{\delta_j\}) \subset S(\epsilon; \Delta_{k-1})$, $p(\delta_k \mid \Delta_{k-1})$ is also a constant on $S(\epsilon; \Delta_{k-1} \cup \{\delta_j\})$. In addition, $\delta_k \in S(\epsilon; \Delta_{k-1} \cup \{\delta_j\})$ implies that $\delta_j \in S(\epsilon; \Delta_k)$, on which $p(\delta_j \mid \Delta_k)$ is also a constant. Then it follows from Eq. (22) that, for all $\delta_j \in S(\epsilon; \Delta_{k-1})$, $p(\delta_j \mid \Delta_{k-1})$ is a constant. Therefore, for all $j > k - 1$ and $j \leq H$, $p(\delta_j \mid \Delta_{k-1}) = \mathcal{S}(\delta_j \mid \epsilon; \Delta_{k-1})$. By backward induction on k, we conclude that the marginal probability density of δ_k, for all $k \in [H]$, is $p(\delta_k) = \mathcal{S}(\delta_k \mid \epsilon)$.

Since Eq. (21) holds for any randomly (as defined above) sampled Δ, we have

$$\begin{aligned} \log|\det(\mathbf{J}(\mathbf{z}))| &\leq \mathbb{E}_\Delta \left[\sum_{i \in [H]} \left(\log\left\|\hat{\delta}_i\right\|_2 - \log\|\delta_i\|_2\right)\right] \\ &= H \mathbb{E}_{\delta \sim \mathcal{S}(\epsilon)} \left[\log\left\|\hat{\delta}\right\|_2 - \log\|\delta\|_2\right]. \end{aligned} \tag{23}$$

If h is a multiple of the identity function around $\mathbf{z}$, then $\mathbf{J}(\mathbf{z}) = C\mathbf{I}$, where $C \in \mathbb{R}$ is a constant. In this case, $\hat{\Delta}$ becomes an orthogonal set as Δ, and therefore the inequalities in Eqs. (20), (21), and (23) become tight. Furthermore, it is straightforward to extend the above result to the case $\delta \sim \mathcal{N}\left(\mathbf{0}, \epsilon^2\mathbf{I}\right)$, considering that $\mathcal{N}\left(\mathbf{0}, \epsilon^2\mathbf{I}\right)$ is a mixture of $\mathcal{S}(\epsilon)$ with different ϵ's.

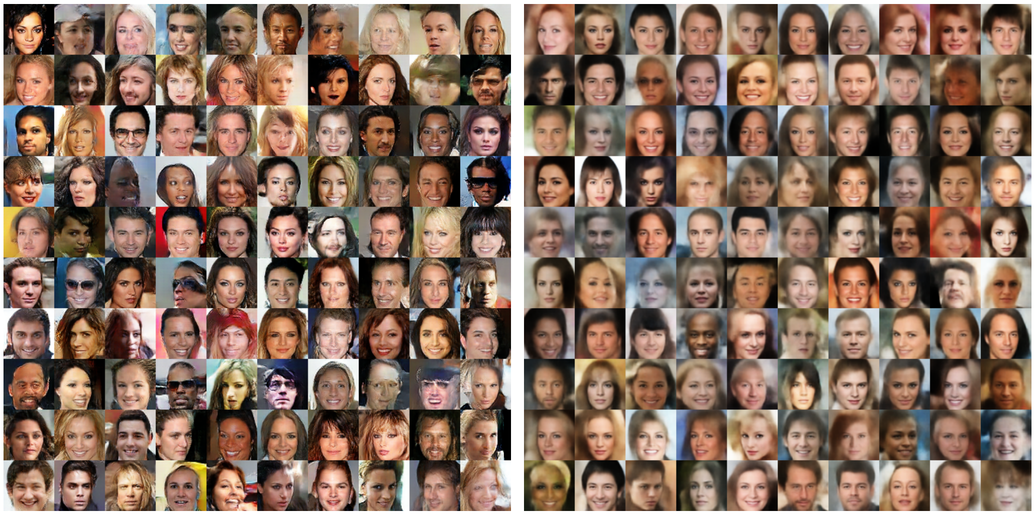
The Taylor expansion of h around $\mathbf{z}$ gives

$$h(\mathbf{z} + \delta) = h(\mathbf{z}) + \mathbf{J}(\mathbf{z})\,\delta + \mathcal{O}\left(\delta^2\right). \tag{24}$$

Therefore, for $\delta \to \mathbf{0}$ or $\epsilon \to 0$, we have $\hat{\delta} = \mathbf{J}(\mathbf{z})\,\delta = h(\mathbf{z} + \delta) - h(\mathbf{z})$. The result follows. □

C More Results on CelebA

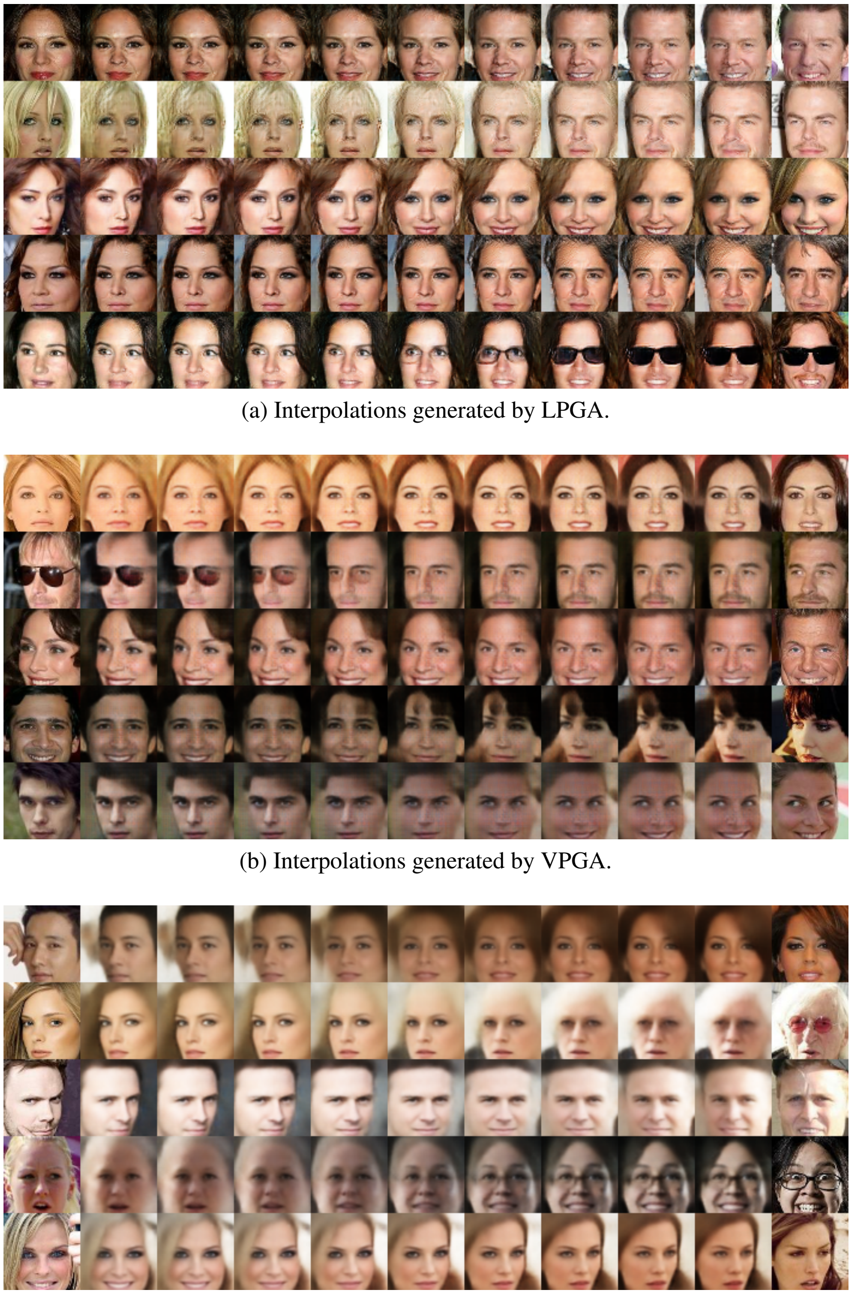
In Fig. 5, we compare the generated samples and FID scores of LPGA and VAE on 140x140 crops. In this experiment, we use the full DCGAN architecture (i.e., 128 filters for the first convolutional layer) for both LPGA and VAE. Other hyperparameter settings remain the same as for 108x108 crops. In Fig. 6, we show latent space interpolations of CelebA samples.



(a) LPGA, FID = 21.35

(b) VAE, FID = 54.25

Figure 5: Random CelebA (140x140 crops) samples generated by LPGA and VAE.



(a) Interpolations generated by LPGA.

(b) Interpolations generated by VPGA.

(c) Interpolations generated by VAE.

Figure 6: Latent space interpolations on CelebA.