Abstract
Modern generative models are usually designed to match target distributions directly in the data space, where the intrinsic dimension of data can be much lower than the ambient dimension. 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 a latent space using the encoder of a standard autoencoder, and train the generator (or decoder) to match the target distribution in the latent space. Specifically, we enforce the consistency in both the data space and the latent space with theoretically justified data and latent reconstruction losses. The resulting generative model, which we call a perceptual generative autoencoder (PGA), is then trained with a maximum likelihood or variational autoencoder (VAE) objective. With maximum likelihood, PGAs generalize the idea of reversible generative models to unrestricted neural network architectures and arbitrary number of latent dimensions. When combined with VAEs, PGAs substantially improve over the baseline VAEs in terms of sample quality. 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 dimension 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 dimensions 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 a latent space using an encoder, while also minimizing 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

We summarize our main contributions as follows:

1University of Calgary, Canada 2MILA, Université de Montréal, Canada 3Wuhan University, China. Correspondence to: Zijun Zhang <zijun.zhang@ucalgary.ca>, Ruixiang Zhang <ruixiang.zhang@umontreal.ca>.


1Code is available at https://github.com/zj10/PGA.
• A training framework, PGA, for generative autoencoders is developed to match the target distribution in the latent space, which, we prove, ensures correct matching in data space.
• We combine PGA with the maximum likelihood objective, and remove the restrictions of reversible (flow-based) generative models on neural network architectures and latent dimensionalities.
• We combine PGA with the VAE objective, solving the VAE’s issue of blurry samples without introducing any auxiliary models or sophisticated model architectures.

2. Related Work

Autoencoder-based generative models are trained by minimizing an data 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, variational autoencoders (VAEs) (Makhzani et al., 2016) and Wasserstein autoencoders (WAEs) (Tolstikhin et al., 2018) work in the latent space to match the aggregated posterior with the data-space target distribution. Inspired by AAEs and WAEs, we develop a principled approach to matching data distributions in the latent space, aiming to improve the generative performance of AAEs and WAEs (Rubenstein et al., 2018), as well as that of VAEs (Rezende & Viola, 2018; Dai & Wipf, 2019). While previous work has explored the use of perceptual loss for a similar purpose (Hou et al., 2017), it relies on a VGG net pre-trained on ImageNet and provides no theoretical guarantees. In our work, the encoder of an autoencoder is jointly trained, such that matching the target distribution in the latent space guarantees the matching in the data space.

In a different line of work, reversible generative models (Dinh et al., 2014; 2017) are developed to enable exact inference. Recent work shows that they are capable of generating realistic images (Kingma & Dhariwal, 2018). 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 the 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 $f_\phi : \mathbb{R}^D \rightarrow \mathbb{R}^H$ be the encoder parameterized by $\phi$, and $g_\theta : \mathbb{R}^H \rightarrow \mathbb{R}^D$ be the decoder parameterized by $\theta$. Our goal is to obtain a decoder-based generative model, which maps a simple prior distribution to a target data distribution, $\mathcal{D}$. Throughout this paper, we use $\mathcal{N}(0, 1)$ as the prior distribution. This section will introduce several related but different distributions, which are illustrated in Fig. 1a. A summary of notations is provided in the supplementary material.

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

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

(1)

where $\hat{\mathbf{x}} = g_\theta(f_\phi(\mathbf{x}))$, then $\hat{\mathbf{x}}$ can be seen as a projection of the input data, $\mathbf{x}$, onto the manifold of $g_\theta(\mathbf{z})$. Let $\hat{\mathcal{D}}$ denote the reconstructed data distribution, i.e., $\hat{\mathbf{x}} \sim \hat{\mathcal{D}}$. Given enough capacity of the encoder, $\hat{\mathcal{D}}$ is the best approximation to $\mathcal{D}$ (in terms of $\ell_2$-distance), that we can obtain from the decoder, and thus can serve as a surrogate target distribution for training the decoder-based generative model.

Due to the difficulty in directly matching the generated distribution with the data-space target distribution, $\mathcal{D}$, we reuse the encoder to map $\hat{\mathcal{D}}$ to a latent-space target distribution, $\mathcal{H}$. We then transform the problem of matching $\hat{\mathcal{D}}$ in the data space into matching $\mathcal{H}$ in the latent space. In other words, we aim to ensure that for $\mathbf{z} \sim \mathcal{N}(0, 1)$, if $f_\phi(g_\theta(\mathbf{z})) \sim \mathcal{H}$, then $g_\theta(\mathbf{z}) \sim \hat{\mathcal{D}}$. In the following, we define $h = f_\phi \circ g_\theta$ for notational convenience.

To this end, we minimize the following latent reconstruction loss w.r.t. $\phi$:

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

(2)

Let $Z(\mathbf{x})$ be the set of all $\mathbf{z}$’s that are mapped to the same $\mathbf{x}$ by $g_\theta$, we have the following theorem:

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

Perceptual Generative Autoencoders
We defer the proof to the supplementary material. Note that Theorem 1 requires that different x’s generated by $g_\theta$ (from $\mathcal{N}(0, I)$ and $\mathcal{H}$) are mapped to different z’s by $f_\phi$. In theory, minimizing Eq. (2) would suffice, since $\mathcal{N}(0, I)$ is supported on the whole $\mathbb{R}^H$. However, there can be z’s with low probabilities in $\mathcal{N}(0, I)$, but with high probabilities in $\mathcal{H}$ that are not well covered by Eq. (2). Therefore, it is sometimes helpful to minimize another latent reconstruction loss on $\mathcal{H}$:

$$L_{r,H}^\phi = \frac{1}{2} \mathbb{E}_{z \sim \mathcal{H}} [\|h(z) - z\|_2^2].$$ (3)

In practice, we observe that $L_{r,H}^\phi$ is often small without explicit minimization, which we attribute to its consistency with the minimization of $L_r$. Moreover, minimizing the latent reconstruction losses w.r.t. $\theta$ is not required by Theorem 1, and it degrades the performance empirically. In addition, the use of $\ell_2$-norm in the reconstruction losses is not a necessity, and the framework can be easily extended to other norm definitions.

By Theorem 1, the problem of training the generative model reduces to training $h$ to map $\mathcal{N}(0, I)$ to $\mathcal{H}$, which we refer to as the perceptual generative model. The basic loss function of PGAs is given by

$$L_{pga} = L_r + \alpha L_{r,N}^\phi + \beta L_{r,H}^\phi,$$ (4)

where $\alpha$ and $\beta$ are hyperparameters to be tuned. Eq. (4) is also illustrated in Fig. 1b.

In the subsequent subsections, we present a maximum likelihood approach, as well as a VAE-based approach to train the perceptual generative model. To build intuition before delving into the details, we note that both of these two approaches work by attracting the latent representations of data samples to the origin, while expanding the volume occupied by each sample in the latent space. These two tendencies together push $\mathcal{H}$ closer to $\mathcal{N}(0, I)$, such that $\mathcal{H}$ matches $\mathcal{H}$. This observation further leads to a unified view of the two approaches.

### 3.2. A Maximum Likelihood Approach

We first assume the invertibility of $h$. For $\hat{x} \sim \hat{\mathcal{D}}$, let $\hat{z} = f_\phi(\hat{x}) = h(z) \sim \mathcal{H}$. We can train $h$ directly with maximum likelihood using the change of variables formula...
The inequality is tight if \( h \) becomes infinite, in which case Eq. (5) cannot be optimized. Consequently, when \( h \) is not invertible at some point \( z \), the logarithm of the Jacobian determinant at \( z \) becomes infinite, in which case Eq. (5) cannot be optimized.

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

\[
L_{lpga} = L_{pga} + \gamma \left( \ell_{nll}^0 + \ell_{nll}^\theta \right).
\]  

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 \( \log p(x) \) as

\[
\log p(x) \geq \log p(x) - \KL(q(z'|x) \parallel p(z'|x)) = \mathbb{E}_{z' \sim q(z'|x)} \left[ \log p(x|z') - \KL(q(z'|x) \parallel p(z')) \right],
\]

where \( p(x|z') \) is modeled with the decoder, and \( q(z'|x) \) is modeled with the encoder. Note that \( z' \) denotes the stochastic version of \( z \), whereas \( z \) 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 \( \log p(z) \), where \( z = h(x) \). Therefore, we replace \( p(x|z') \) on the r.h.s. of Eq. (10) with \( p(z|z') \), and derive a lower bound on \( \log p(z) \) as

\[
\log p(z) \geq \log p(z) - \KL(q(z'|x) \parallel p(z'|z)) = \mathbb{E}_{z' \sim q(z'|x)} \left[ \log p(z|z') - \KL(q(z'|x) \parallel p(z')) \right].
\]

Similar to the original VAE, we make the assumption that \( q(z'|x) \) and \( p(z|z') \) are Gaussian; i.e., \( q(z'|x) = \mathcal{N}(z' \mid \mu_{\theta}(x), \sigma^2_{\theta}(x)) \), and \( p(z|z') = \mathcal{N}(z \mid \mu_{\theta,\phi}(z'), \sigma^2_{\phi}) \). Here, \( \mu_{\theta}(\cdot) = f_{\theta}(\cdot) \), \( \mu_{\theta,\phi}(\cdot) = h(\cdot) \), and \( \sigma > 0 \) is a tunable scalar. Note that if \( \sigma \) is fixed, the first term on the r.h.s. of Eq. (11) has a trivial maximum, where \( z, \bar{z}, \) and \( \mu_{\theta,\phi}(z') \) are all close to zero. To circumvent this, we set \( \sigma \) proportional to the \( \ell_2 \)-norm of \( z \).

The VAE variant is trained by minimizing

\[
L_{vae} = L_{vpr} + L_{vkl}^0
\]

where \( L_{vpr} \) and \( L_{vkl}^0 \) correspond, respectively, to the reconstruction and KL divergence losses of VAE, as illustrated in Fig. 1d. In \( L_{vpr} \), while the gradient through \( \sigma^2_{\phi}(x) \) remains unchanged, we ignore the gradient passed directly from \( L_{vpr} \) to the encoder, due to a similar reason discussed for Eq. (7).

Accordingly, the overall loss function is given by

\[
L_{vpga} = L_{pga} + \eta L_{vae}.
\]  

We refer to this approach as variational PGA (VPGA).
3.4. A High-Level View of the PGA Framework

We summarize what each loss term achieves, and explain from a high-level how they work together.

**Data reconstruction loss** (Eq. (1)): For Theorem 1 to hold, we need to use the reconstructed data distribution \( \mathcal{D} \), instead of the original data distribution \( \mathcal{D} \), as the target distribution. Therefore, minimizing the data reconstruction loss ensures that the target distribution is close to the data distribution.

**Latent reconstruction loss** (Eqs. (2) and (3)): The encoder \( f_\theta \) is reused to map data-space distributions to the latent space. As shown by Theorem 1, minimizing the latent reconstruction loss (w.r.t. the parameters of the encoder) ensures that if the generated distribution and the target distribution can be mapped to the same distribution \( \mathcal{H} \) in the latent space by the encoder, then the generated distribution and the target distribution are the same.

**Maximum likelihood loss** (Eqs. (6) and (7)) or **VAE loss** (Eq. (12)): The decoder \( g_\phi \) is trained to map \( \mathcal{H} \) to the latent-space target distribution \( \mathcal{H} \) by minimizing either the maximum likelihood loss or the VAE loss.

The first loss allows to use the reconstructed data distribution as the target distribution. The second loss transforms the problem of matching the target distribution in the data space into matching it in the latent space. The latter problem is then solved by the third loss. Therefore, the three losses together ensure that the generated distribution is close to the data distribution.

3.5. 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. 1c and 1d). Intuitively, observe that

\[
L_{vkl}^\phi = L_{nil}^\phi + \frac{1}{2} \sum_{x \sim \mathcal{D}} \left[ \sigma^2_{\phi, i}(x) - \log \left( \sigma^2_{\phi, i}(x) \right) \right],
\]

which means both \( L_{nil}^\phi \) and \( L_{vkl}^\phi \) tend to attract the latent representations of data samples to the origin. In addition, by minimizing \( \log |\det \left( \partial h(z) / \partial z \right)| \), \( L_{nil}^\phi \) expands the volume occupied by each sample in the latent space, which can be also achieved by \( L_{vr} \) with the second term of Eq. (14).

More concretely, we observe that both \( L_{nil}^\phi \) and \( L_{vr} \) are minimizing the difference between \( h(z) \) and \( h(z + \delta') \), where \( \delta' \) is some additive zero-mean noise. However, they differ in that the variance of \( \delta' \) is fixed for \( L_{nil}^\phi \), but is trainable for \( L_{vr} \); and the distance between \( h(z) \) and \( h(z + \delta') \) are defined in two different ways. In fact, \( L_{vr} \) is a squared \( \ell_2 \)-distance derived from the Gaussian assumption on \( z \), whereas \( L_{nil}^\phi \) can be derived similarly by assuming that \( d^H = \| z - h(z + \delta') \|^2_H \) follows a reciprocal distribution as

\[
p(\delta^H; a, b) = \frac{1}{d^H (\log(b) - \log(a))},
\]

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 \( p(\delta^H; a, b) \).

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

\[
L_{vpga} = L_{pga} + \gamma L_{vr} + \gamma L_{nil}^\phi + \eta L_{vkl}^\phi.
\]

When \( \gamma' = \gamma \) and \( \eta = 0 \), Eq. (16) is equivalent to Eq. (9), considering that \( \sigma^2_{\phi}(x) \) will be optimized to approach \( \epsilon^2 \). 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. This approach mainly serves to demonstrate the connection between LPGA and VPGA, and is less practical due to the extra hyperparameters. 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, \( \sigma \) 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
Figure 2. Random samples generated by LPGA, VPGA, and VAE. Note how LPGA and VPGA images are less blurry than those from the VAE.

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) \cite{Heusel2017} 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
Perceptual Generative Autoencoders

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.

<table>
<thead>
<tr>
<th>Model</th>
<th>MNIST</th>
<th>CIFAR-10</th>
<th>CelebA</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAE</td>
<td>19.21</td>
<td>106.37</td>
<td>48.12</td>
</tr>
<tr>
<td>CV-VAE</td>
<td>33.79</td>
<td>94.75</td>
<td>48.87</td>
</tr>
<tr>
<td>WAE</td>
<td>20.42</td>
<td>117.44</td>
<td>53.67</td>
</tr>
<tr>
<td>RAE-L2</td>
<td>22.22</td>
<td>80.80</td>
<td>51.13</td>
</tr>
<tr>
<td>RAE-SN</td>
<td>19.67</td>
<td>84.25</td>
<td>44.74</td>
</tr>
<tr>
<td>VPGA</td>
<td>13.83 ± 0.06</td>
<td>115.74 ± 0.63</td>
<td>43.60 ± 0.33</td>
</tr>
<tr>
<td>LPGA</td>
<td>10.34 ± 0.15</td>
<td>55.87 ± 0.25</td>
<td>14.53 ± 0.52</td>
</tr>
<tr>
<td>LVPGA</td>
<td>6.32 ± 0.16</td>
<td>52.94 ± 0.89</td>
<td>13.80 ± 0.20</td>
</tr>
</tbody>
</table>

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 all three datasets. 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 show further results on 140x140 crops and latent space interpolations in the supplementary material. While PGA has largely bridged the performance gap between generative autoencoders and GANs, there is still a noticeable gap between them especially on CIFAR-10. For instance, the FIDs of WGAN-GP and SN-GAN on CIFAR-10 using a similar architecture are respectively 40.2 and 25.5 (Miyato et al., 2018), as compared to 51.5 of VPGA.

Empirically, different PGA variants share the same optimal values of $\alpha$ and $\beta$ (Eq. (4)) when trained on the same dataset. For LPGA, $\gamma$ (Eq. (9)) tends to vary in a small range for different datasets (e.g., $1.5e-2$ for MNIST and CIFAR-10, and $1e-2$ for CelebA). For VPGA, $\eta$ (Eq. (13)) can vary widely (e.g., $2e-2$ for MNIST, $3e-2$ for CIFAR-10, and $2e-3$ for CelebA), and thus is slightly more difficult to tune. 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 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 decoder (e.g., $H$ and $\mathcal{N}(0, I)$) 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.

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 data reconstruction loss minimizes...
Perceptual Generative Autoencoders

Figure 4. Random samples generated by LPGA without the latent reconstruction losses ($\alpha = \beta = 0$). Compared to the samples in Fig. 2, we observe a degradation.

the discrepancy between the input data and its reconstruction. Since the reconstructed data distribution serves as the surrogate target distribution, removing the data 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. 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.

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


Perceptual Generative Autoencoders


