Disentangled Inference for GANs with Latently Invertible Autoencoder

Jiapeng Zhu[*1,2] · **Deli Zhao**[*1] · **Bo Zhang**[1] · **Bolei Zhou**[2]

Abstract Generative Adversarial Networks (GANs) play an increasingly important role in machine learning. However, there is one fundamental issue hindering their practical applications: the absence of capability for encoding real-world samples. The conventional way of addressing this issue is to learn an encoder for GAN via Variational Auto-Encoder (VAE). In this paper, we show that the entanglement of the latent space for the VAE/-GAN framework poses the main challenge for encoder learning. To address the entanglement issue and enable inference in GAN we propose a novel algorithm named Latently Invertible Autoencoder (LIA). The framework of LIA is that an invertible network and its inverse mapping are symmetrically embedded in the latent space of VAE. The decoder of LIA is first trained as a standard GAN with the invertible network and then the partial encoder is learned from a disentangled autoencoder by detaching the invertible network from LIA, thus avoiding the entanglement problem caused by the random latent space. Experiments conducted on the FFHQ face dataset and three LSUN datasets validate the effectiveness of LIA/GAN. [1]

Keywords GAN · Inference · Disentanglement

1 Introduction

Deep generative models play a more and more important role in cracking challenges in computer vision as well as in other disciplines, such as high-quality image generation (Isola et al, 2017; Zhu et al, 2017; Karras et al, 2018a,b; Brock et al, 2018), text-to-speech transformation (van den Oord et al, 2016, 2017), information retrieval (Wang et al, 2017), 3D rendering (Wu et al, 2016; Eslami et al, 2018), and signal-to-image acquisition (Zhu et al, 2018). In particular, Generative Adversarial Network (GAN) (Goodfellow et al, 2014) exhibits an extraordinary capability of learning distributions of high-dimensional imagery data. For example, the real faces and the generated human faces by Style-GAN algorithms (Karras et al, 2018b, 2019) become indistinguishable. Adversarial learning is also widely applied as a loss function in many machine learning applications.

However, a critical limitation for the vanilla GAN is the absence of the encoder for carrying out inference on real samples. Namely, we cannot directly derive the corresponding random variable z for a given sample x by the GAN architecture. This is an important problem because real applications via GANs usually depend on manipulating the latent variables such as domain adaptation, data augmentation, and image editing.

The existing approaches for addressing this issue fall into three categories, as summarized in Table 1. The common approach is called the GAN inversion that is based on the optimization of Mean Squared Error (MSE) between the generated samples and the associated real samples (Radford et al, 2016; Berthelot et al, 2017). This type of algorithms mainly suffers from two drawbacks: the sensitivity to initialization of z and the time-consuming computation. The second category is named as adversarial inference (Dumoulin et al, 2017; Donahue et al, 2017), which uses another GAN to infer z as latent variables in a framework of dual GANs. Adversarial inference generally highlights the high-level representation of objects rather than aims to faithfully

Table 1: Different approaches for enabling inference in GANs. f and g denote the encoder and the decoder/generator, respectively. c is the discriminator for GAN. $\boldsymbol{z}$ obeys a specific probabilistic prior, i.e. $\boldsymbol{z} \sim p(\boldsymbol{z})$ and $\boldsymbol{y} = \phi^{-1}(\boldsymbol{z})$, where ϕ is an invertible network.

Method	Formulation
GAN inversion	$\boldsymbol{z}^* = \arg\min_{\boldsymbol{z}} \|g(\boldsymbol{z}) - \boldsymbol{x}\|$
Adversarial inference	$\boldsymbol{x} \overset{f}{\mapsto} \tilde{\boldsymbol{z}},\ \{\tilde{\boldsymbol{z}}, \boldsymbol{z}\} \overset{c}{\mapsto} 0/1$
VAE/GAN	$\boldsymbol{x} \overset{f}{\mapsto} \boldsymbol{z} \overset{g}{\mapsto} \tilde{\boldsymbol{x}},\ \{\tilde{\boldsymbol{x}}, \boldsymbol{x}\} \overset{c}{\mapsto} 0/1$ $\max \log p(\boldsymbol{x})$
LIA/GAN (ours)	$\boldsymbol{x} \overset{f}{\mapsto} \boldsymbol{y} \overset{g}{\mapsto} \tilde{\boldsymbol{x}},\ \{\tilde{\boldsymbol{x}}, \boldsymbol{x}\} \overset{c}{\mapsto} 0/1$ $\min \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|$

reconstruct samples (Donahue and Simonyan, 2019). Another natural way of performing GAN inference is to finetune an encoder via the principle of the VAE algorithm, provided a pretrained GAN model (Luo et al, 2017) or to learn the whole VAE/GAN architecture end-to-end (Larsen et al, 2016). The framework of combining VAE and GAN with shared decoder/generator is actually an elegant solution to GAN inference. However, a mystery for VAE/GAN is that the reconstruction precision is usually worse than that of using VAE alone and the quality of reconstructed samples is always inferior to that of generated samples from sampling pretrained GANs. We will analyze it in more detail in section 6.

In this paper, we verify that the disentanglement of the latent space is the decisive factor of learning a high-quality encoder for VAE/GAN. Based on the disentanglement argument, we develop a new model called Latently Invertible Autoencoder (LIA). LIA utilizes an invertible network to bridge the encoder and the decoder in a symmetric manner in the latent space, thus forming the LIA/GAN framework via adversarial learning for GAN generation and inference. The architecture of LIA/GAN is illustrated in Figure 2. We summarize its key advantages as follows:

- We demonstrate the effect of the entanglement in the latent space ($\boldsymbol{z}$-space) and the disentanglement in the intermediate latent space ($\boldsymbol{y}$-space). We argue that the entanglement in the latent space formed by GAN training is the key reason why VAE/GAN cannot work well. Based on this analysis, we propose the LIA architecture and the corresponding two-stage training scheme.
- The symmetric design of the invertible network in LIA brings two benefits. The prior distribution can be *exactly* fitted from a *disentangled* and unfolded feature space, thus significantly easing the inference problem. Besides, since the latent space is detached when training the encoder, the encoder can

be trained *without variational inference*, as opposed to VAE.

- The two-stage adversarial learning decomposes the LIA framework into the vanilla GAN with an invertible network and a standard autoencoder without stochastic variables. Therefore the encoder training is *deterministic* and performed via the *disentangled latent space*, implying that our model will not suffer from the entanglement problem in training VAE.
- We compare LIA with state-of-the-art GAN models on inference and generation/reconstruction. The experimental results on FFHQ and LSUN datasets show the LIA/GAN achieves superior performance.

2 Entanglement in GANs

We analyze empirically the effect of entanglement and disentanglement in GANs and its mathematical principle in this section, as well as their implication for image reconstruction.

2.1 Entanglement from Random Mapping

For a GAN model, we have $\tilde{\boldsymbol{x}} = g(\boldsymbol{z})$, where g is the generator and $\boldsymbol{z}$ is usually sampled from a Gaussian prior with zero mean and unit variance, i.e. $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{1})$. To visualize the entanglement in GANs, we randomly sample $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ and linearly interpolate between them. The corresponding generated samples can be attained by $\tilde{\boldsymbol{x}} = g(\boldsymbol{z})$. As visualized in Figure 1(a) via StyleGAN on the FFHQ database, the face identities are significantly changed during the deformation from face $\tilde{\boldsymbol{x}}_i$ and face $\tilde{\boldsymbol{x}}_j$, meaning that the faces in the $\tilde{\boldsymbol{x}}$-space do not change continuously with linearity in the $\boldsymbol{z}$-space. As a comparison, we establish the mapping via an intermediate latent variable $\boldsymbol{y} = \varphi(\boldsymbol{z})$ and $\tilde{\boldsymbol{x}} = g(\boldsymbol{y})$, as StyleGAN does, where φ is a multilayer perceptron producing an output vector with the same dimension as $\boldsymbol{z}$. From Figure 1(c), it is clear that the face path associated with $\boldsymbol{y}$ interpolation approaches the straight line significantly nearer than that with $\boldsymbol{z}$ interpolation and the corresponding generated faces vary very smoothly as shown in Figure 1(b).

In fact, this is a very common phenomenon for $\boldsymbol{z}$ and $\boldsymbol{y}$ interpolations between arbitrary two faces with large variation. Following Karras et al (2018b), we call the folded $\boldsymbol{z}$ space with respect to the generated face the entangled latent space, and that the intermediate latent space is the disentangled one. The entanglement for $\boldsymbol{z}$ incurs from GAN training with random sampling, because there is no geometric constraint to guarantee

(a) Interpolation on z.

(b) Interpolation on y.
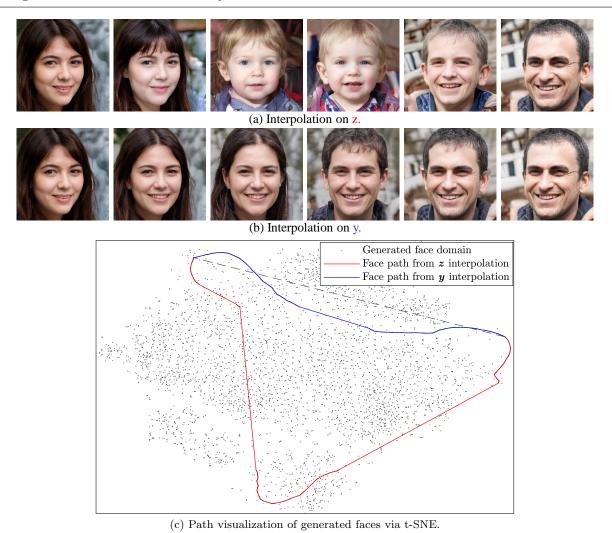

(c) Path visualization of generated faces via t-SNE.

Fig. 1: Illustration of the entanglement in latent spaces of GANs. The results are obtained by StyleGAN on FFHQ dataset. There are 4,000 randomly generated faces as the face domain (gray dots) and 100 interpolated faces for each red/blue path. The t-SNE method is used to obtain the 2D data points based on the VGG features of all faces.

the spatial adjacency correspondence between z and $\tilde{x}$. A consequence is that the spatial locations of the associated z_i and z_j are not necessarily adjacent if $\tilde{x}_i$ and face $\tilde{x}_j$ are perceptually similar. The disentanglement of the y-space will be analyzed in section 2.3

2.2 Inference Difficulty from Entanglement

Without loss of generality, we write the optimization as

$$z^{t+1} = z^t - \nabla L_g(z^t), \qquad (1)$$

where $\nabla L_g(z^t)$ denotes the gradient of z of step t. Here z may be the prior or the output of the decoder in VAE, i.e. $z = f(x)$. To make the problem easily understood, we suppose that the generated sample $\tilde{x} = g(z)$

is also a face. Let $\text{ID}(\tilde{x})$ denote the identity of face $\tilde{x}$ (or the category of an object). The entanglement occurs if $\text{ID}(g(z^t))$ is semantically distinctive from $\text{ID}(g(z^0))$ and $\text{ID}(g(z^T))$ shown in Figure 1(a), where z^0 and z^T are the initialization and the convergent target. Then the z^t path will be probably entangled during the optimization process shown in Figure 1(c), thus leading to the difficulty that the algorithm converges at the right minima. From the above analysis, we argue that the entanglement of the z space is the crux that VAE/GAN and the MSE-based optimization are incapable of performing inference for GANs well. Evidence from practice is that the entanglement causes the sharp gradient volatility during optimization, hindering the convergence of the algorithm, which will be analyzed in section 6.5.

Table 2: Disentanglement comparison on z and y.

	Path Length		Separability
	full	end	
z	207.48	206.56	132.19
y	74.65	67.11	7.75

2.3 Disentanglement and Lipschitz Continuity

The disentanglement for GANs can be characterized with Lipschitz continuity. For the GAN model, we can write

$$\|g(\boldsymbol{v}_i) - g(\boldsymbol{v}_j)\| \leq C\|\boldsymbol{v}_i - \boldsymbol{v}_j\|, \tag{2}$$

where C is Lipschitz constant, $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are variables. Inequality (2) says that the distance between two generated samples $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$ is no more than that between variables $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ with a constant factor. For the map $z \overset{g}{\mapsto} \tilde{\boldsymbol{x}}$ (i.e. $\boldsymbol{v} = \boldsymbol{z}$), we can easily find the failing case that inequality (2) holds, as is shown in Figure 1. For the map $\boldsymbol{y} \overset{g}{\mapsto} \tilde{\boldsymbol{x}}$ from the intermediate latent space, however, Lipschitz continuity can be satisfied in some condition, i.e.

$$\|g(\boldsymbol{y}_i) - g(\boldsymbol{y}_j)\| \leq C\|\boldsymbol{y}_i - \boldsymbol{y}_j\|. \tag{3}$$

To see this, we need the Jacobian $\boldsymbol{J}$ of $g(\boldsymbol{y})$, i.e. $J_{ij} = \partial g(\boldsymbol{y})_i / \partial y_j$. A isometric g-mapping has been revealed by (Karras et al, 2019) [2] that $\|g(\boldsymbol{y}_i) - g(\boldsymbol{y}_j)\| = C_y\|\boldsymbol{y}_i - \boldsymbol{y}_j\|$ holds if $\boldsymbol{J}^\top \boldsymbol{J} = a\boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix, a is a constant, and $\top$ denotes the transpose. Therefore, the Jacobian regularizer was applied in Karras et al (2019) to enhance the disentanglement of the generator with respect to $\boldsymbol{y}$. Besides, spectral normalization (Miyato et al, 2018) serves to Lipschitz Continuity as well.

In practice, we find that Lipschitz continuity after a mapping network is sufficient to establish a disentangled intermediate latent space via the deep mapping $\boldsymbol{y} = \varphi(\boldsymbol{z})$ embedded in GANs, as shown in Figures 1. The path length index applied in Karras et al (2019) can reflect the property of Lipschitz continuity. To further reveal this phenomenon, we compare the path lengths of generated faces from randomly sampled $\boldsymbol{z}$ and $\boldsymbol{y}$, respectively. Table 2 shows that the path length from $\boldsymbol{y}$ is much shorter than that from $\boldsymbol{z}$, verifying the fact that the $\boldsymbol{y}$-space approaches Lipschitz and thus is more disentangled.

Lipschitz continuity can guarantee the spatial consistency between the latent space and the generated sample space, which is more favorable of inference task. Therefore, we devise our algorithm for GAN inference based on the disentangled $\boldsymbol{y}$-space instead of the $\boldsymbol{z}$-space.

---

[2] The theoretical analysis was detailed in section C in (Karras et al, 2019).

3 Latently Invertible Autoencoder

As analyzed in the preceding section, to acquire the disentangled latent space, we need to embed a mapping network in the architecture of the vanilla GAN, i.e. $\boldsymbol{z} \overset{\varphi}{\mapsto} \boldsymbol{y} \overset{g}{\mapsto} \tilde{\boldsymbol{x}}$. At the same time, the encoder f has to directly infer $\boldsymbol{y}$ to favor the disentanglement. So, we also need to establish a reverse mapping to obtain $\boldsymbol{z}$ from $\boldsymbol{y}$, i.e. $\boldsymbol{z} = \phi(\boldsymbol{y})$. This implies $\varphi = \phi^{-1}$. Therefore, an invertible neural network can establish the reversibility between $\boldsymbol{z}$ and $\boldsymbol{y}$. The LIA framework is designed according to this rule. The details are described as follows.

3.1 Neural Architecture of LIA

As shown in Figure 2(a), we symmetrically embed an invertible neural network in the latent space of VAE, following the diagram of mapping process as

$$\boldsymbol{x} \overset{f}{\longmapsto} \boldsymbol{y} \leftrightarrow \underbrace{\boldsymbol{y} \overset{\phi}{\longmapsto}}_{\text{invertible}} \boldsymbol{z} \underbrace{\overset{\phi^{-1}}{\longmapsto} \boldsymbol{y}}_{\text{invertible}} \leftrightarrow \boldsymbol{y} \overset{g}{\longmapsto} \tilde{\boldsymbol{x}}, \tag{4}$$

where ϕ denotes the deep composite mapping of the invertible network. LIA first performs nonlinear dimensionality reduction on the input data $\boldsymbol{x}$ and transforms them into the low-dimensional disentangled feature space $\mathbb{R}^{d_y}$. The role of $f(\boldsymbol{x})$ for LIA can be regarded to unfold the underlying data manifold. Therefore, Euclidean operations such as linear interpolation and vector arithmetic are more reliable and continuous in this disentangled feature space. Then we establish an invertible mapping $\phi(\boldsymbol{y})$ from the feature $\boldsymbol{y}$ to the latent variable $\boldsymbol{z}$, as opposed to VAEs that directly map original data to latent variables. The feature $\boldsymbol{y}$ can be exactly recovered via the invertibility of ϕ from $\boldsymbol{z}$, which is the advantage of using invertible networks. The recovered feature $\boldsymbol{y}$ is then fed into a partial decoder $g(\boldsymbol{y})$ to generate the corresponding data $\tilde{\boldsymbol{x}}$. If the maps ϕ and ϕ^{-1} of the invertible network are bypassed, LIA reduces to a standard autoencoder, i.e. $\boldsymbol{x} \overset{f}{\mapsto} \boldsymbol{y} \overset{g}{\mapsto} \tilde{\boldsymbol{x}}$.

In general, any invertible networks are applicable in the LIA framework. We find in practice that a simple invertible network used in Dinh et al (2015) is sufficiently capable of constructing the mapping from the feature space $\mathbb{R}^{d_y}$ to the latent space $\mathbb{R}^{d_z}$. Let $\boldsymbol{x} = [\boldsymbol{x}_t; \boldsymbol{x}_b]$ and $\boldsymbol{z} = [\boldsymbol{z}_t; \boldsymbol{z}_b]$ be the forms of the top and bottom fractions of $\boldsymbol{x}$ and $\boldsymbol{z}$, respectively. Then the invertible network can be built as

$$\boldsymbol{z}_t = \boldsymbol{x}_t, \quad \boldsymbol{z}_b = \boldsymbol{x}_b + \tau(\boldsymbol{x}_t), \tag{5}$$
$$\boldsymbol{x}_t = \boldsymbol{z}_t, \quad \boldsymbol{x}_b = \boldsymbol{z}_b - \tau(\boldsymbol{z}_t), \tag{6}$$

(a) Neural architecture of LIA.

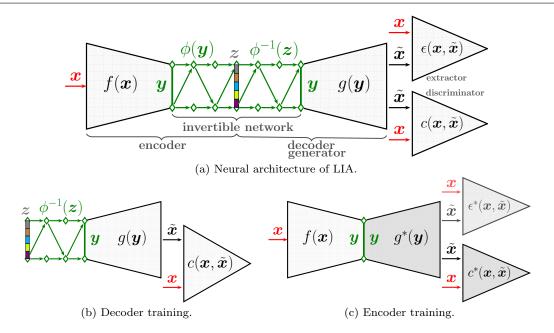
(b) Decoder training.

(c) Encoder training.

Fig. 2: Latently invertible autoencoder (LIA) with adversarial learning. (a) LIA consists of five functional modules: an encoder to unfold the manifold $\boldsymbol{y} = f(\boldsymbol{x})$, an invertible network ϕ to reshape feature embeddings to match the prior distribution $\boldsymbol{z} = \phi(\boldsymbol{y})$ and ϕ^{-1} to map latent variables to disentangled feature vectors $\boldsymbol{y} = \psi^{-1}(\boldsymbol{z})$, a decoder to produce output $\tilde{\boldsymbol{x}} = g(\tilde{\boldsymbol{y}})$, a feature extractor ϵ to perform reconstruction measure, and a discriminator c to distinguish real/fake distributions. The training of LIA proceeds in the two-stage way: (b) first training the decoder via a GAN model and (c) then the encoder by detaching the invertible network from LIA. The parameters of modules in dark gray in (c) are frozen in this stage.

where τ is the transformation that can be an arbitrary differentiable function. Alternatively, one can attempt to exploit the complex invertible network with affine coupling mappings for more challenging tasks (Dinh et al, 2017; Kingma and Dhariwal, 2018). As conducted in Dinh et al (2015), we set τ for simplicty as a multi-layer perceptron with the leaky ReLU activation.

3.2 Reconstruction Loss and Adversarial Learning

To guarantee the precise reconstruction $\tilde{\boldsymbol{x}}$, the conventional way by (variational) autoencoders is to use the distance $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|$ or the cross entropy directly between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$. Here, we utilize both the pixel loss and the perceptual loss that is proven to be more robust to variations of image details (Johnson et al, 2016). Let ϵ denote the feature extractor, e.g. VGG (Simonyan and Zisserman, 2014). Then we can write the loss

$$L(\epsilon, \boldsymbol{x}, \tilde{\boldsymbol{x}}) = \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2 + \beta_1 \|\epsilon(\boldsymbol{x}) - \epsilon(\tilde{\boldsymbol{x}})\|_2. \quad (7)$$

where β_1 the hyper-parameter to balance those two losses. The feasibility for this type of mixed reconstruction loss is actually evident in diverse image-to-image

translation tasks. It suffices to emphasize that the functionality of ϵ here is in essence to produce the representations of the input $\boldsymbol{x}$ and the output $\tilde{\boldsymbol{x}}$. The acquisition of ϵ is fairly flexible. It can be attained by supervised *or* unsupervised learning, meaning that ϵ can be trained with class labels like VGG (Simonyan and Zisserman, 2014) or without class labels (van den Oord et al, 2018).

The norm-based reconstruction constraints usually incur the blurry output images in autoencoder-like architectures (Lehtinen et al, 2018). This problem can be handled via the adversarial learning (Goodfellow et al, 2014). To do so, a discriminator c is employed to balance the loss of the comparison between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$. Using the Wasserstein GAN (Arjovsky et al, 2017; Gulrajani et al, 2017), we can write the optimization objective as

$$L(c) = \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim p_{\tilde{x}}} [c(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim p_x} [c(\boldsymbol{x})] + \frac{\gamma}{2} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim p_x} \left[\|\nabla_{\boldsymbol{x}} c(\boldsymbol{x})\|_{\ell_2}^2 \right],$$
$$(8)$$

where p_x and $p_{\tilde{x}}$ denote the probability distributions of the real data and the generated data, respectively. γ is the hyper-parameter of the regularization. The R_1 regularizer is formulated in (Mescheder et al, 2018), which is proven more stable for convergence. In practice, the sliced Wasserstein distance that is approximated by

Monte Carlo sampling is preferred to perform comparison between p_x and $p_{\tilde{x}}$ (Karras et al, 2018a).

4 Two-Stage Training

We propose a scheme of two-stage training, which decomposes the framework into two parts that can be well trained end-to-end respectively, as shown in Figure 2(b) and (c). First, the decoder of LIA is trained as a GAN model with invertible network. Second, the invertible network that connects feature space and latent space is detached from the architecture, reducing the framework to a standard autoencoder *without* variational inference. Thus this two-stage design prevents the entanglement issue.

4.1 Decoder Training

ProGAN (Karras et al, 2018a), StyleGAN (Karras et al, 2018b, 2019), and BigGAN (Brock et al, 2018) are capable of generating photo-realistic images from random noise sampled from some prior distribution. Then it is naturally supposed that such GAN models are applicable to recover a precise $\tilde{x}$ if we can find the latent variable z for the given x. Namely, we may train the associated GAN model separately in the LIA framework. To conduct this, we single out a standard GAN model for the first-stage training, as displayed in Figure 2(b), the diagram of which can be formalized by

$$z \xrightarrow{\phi^{-1}} y \xrightarrow{g} \tilde{x} \leftrightarrow \begin{Bmatrix} \tilde{x} \\ x \end{Bmatrix} \xrightarrow{c} L(c, x, \tilde{x}), \tag{9}$$

where z is directly sampled from a pre-defined prior. According to the principle of Wasserstein GAN, the optimization objective can be written as

$$\{\phi^*, g^*, c^*\} = \min_{\phi, g} \max_c L(c), \tag{10}$$

where the superscript $*$ denotes that the parameters of corresponding mappings have already been learned. It is worth noting that the role of the invertible network here is just its transformation invertibility. *We do not pose any constraints on the probabilities of z and $\phi(y)$ in contrast to normalizing flows.* Our strategy of attaching an invertible network in front of the generator can be potentially applied to any GAN models.

4.2 Encoder Training

In the LIA architecture, the invertible network is embedded in the latent space in a symmetric fashion, i.e.

$f(x) = y = \phi^{-1}(z)$. This unique characteristic of the invertible network allows us to detach the invertible network ϕ from the LIA framework. Thus we attain a conventional autoencoder without stochastic variables, as shown in Figure 2(c). We can write the diagram

$$x \xrightarrow{f} y \xrightarrow{g^*} \tilde{x} \leftrightarrow \begin{Bmatrix} \tilde{x} \\ x \end{Bmatrix} \begin{matrix} \xrightarrow{\epsilon^*} L(\epsilon^*, x, \tilde{x}) \\ \xrightarrow{c^*} L(c^*, x, \tilde{x}) \end{matrix}. \tag{11}$$

In practice, the feature extractor ϵ in perceptual loss is the VGG weight up to conv4 pretrained on the ImageNet dataset. After the first-stage GAN training, the parameter of f is learned as

$$\{f^*, c^{**}\} = \min_f \beta_2 L(\epsilon^*, x, \tilde{x}) + L(c^*, x, \tilde{x}), \tag{12}$$

where β_2 is the hyper-parameter and c^{**} is the fine-tuned parameter of the discriminator, meaning that the discriminator is fine-tuned with the training of the encoder while the generator is frozen. The above optimization serving to the architecture in Figure 2(c) is widely applied in computer vision. It is the backbone framework of various GANs for diverse image processing tasks (Isola et al, 2017; Zhu et al, 2017). For LIA, however, it is much simpler because we only need to learn the partial encoder f. This simplicity brought by the two-stage training is able to enforce the encoder to converge with more precise inference.

5 Related Work

Our LIA model is relevant to the works that solve the inference problem for VAEs with adversarial learning as well as the works that design encoders for GANs. The integration of GAN with VAE can be traced back to the work of VAE/GAN (Larsen et al, 2016) and implicit autoencoders (Makhzani et al, 2015; Makhani, 2018). These methods encounter the difficulty of end-to-end training, because the gradients are prone to becoming unstable after going through the latent space in deep complex architectures (Bowman et al, 2015; Kingma et al, 2016). Besides, there is an intriguing attempt of training VAE in the adversarial manner (Ulyanov et al, 2017; Heljakka et al, 2018). These approaches confront the trade-off between the roles of the encoder that performs inference and compares the real/fake distributions. This is difficult to tune. So we prefer the framework of the vanilla GAN with an indispensable discriminator.

The related works to LIA are the models of combining VAE and the inverse autoregressive flow (Kingma et al, 2016) and the latent-flow-based VAE approach that are VAEs with latent variables conditioned by normalizing flows (Su and Wu, 2018; Xiao et al, 2019). These
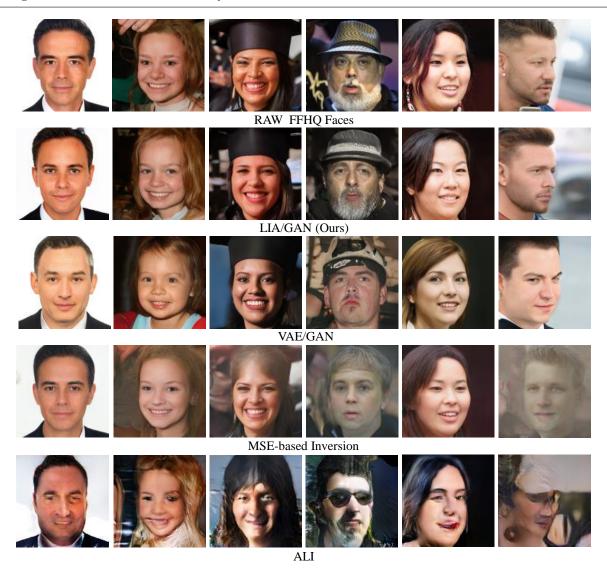
Fig. 3: Comparison of reconstructed faces on FFHQ database.

three models all need to optimize the log-likelihood of normalizing flows, which is essentially different from LIA. The invertible network in LIA only serves to establish the invertibility between the entangled z-space and the disentangled y-space. There is no probabilistic optimization for normalizing flows involved in LIA. There are alternative attempts of specifying the generator of GAN with normalizing flow (Grover et al, 2017) or mapping images into feature space with partially invertible network (Lucas et al, 2019). These approaches suffer from high complexity computation for high dimensions. The approach of two-stage training in Luo et al (2017) suffers the entanglement problem.

It is worth noting that the reconstruction task we focus here is different from the recent work of representation learning which learns features for recognition and classification using adversarial inference (Dumoulin et al, 2017; Donahue et al, 2017; Donahue and Simonyan, 2019). Our primary goal is to learn latent codes of real images and further faithfully to reconstruct real images from latent codes for downstream tasks like image editing. The performance of image editing completely depends on reconstruction precision whereas the work of adversarial inference such as Dumoulin et al (2017); Donahue et al (2017); Donahue and Simonyan (2019) prefers to learn high-level semantic features that are favorable of classification task. The goals are essentially different.

We are aware that a concurrent work, called Adversarial Latent Auto-Encoders (ALAE) (Pidhorskyi et al, 2020), proposed a similar idea to ours. There are four critical differences between ALAE and our LIA method.

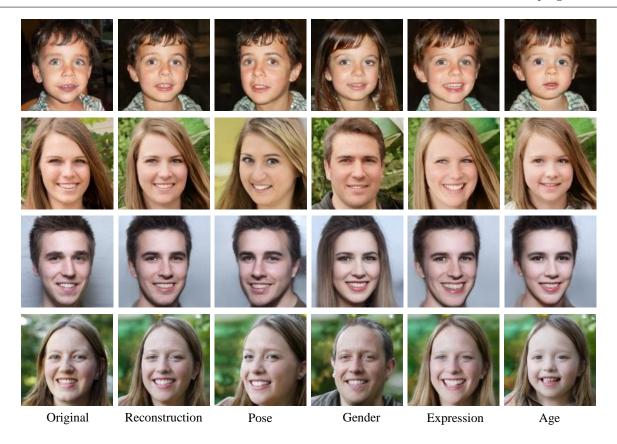| | Original | Reconstruction | Pose | Gender | Expression | Age |

Fig. 4: Manipulating real faces from latent codes. The algorithm in (Shen et al, 2020) is applied to manipulate the face reconstructed from the latent codes $\boldsymbol{y}$ given by our LIA algorithm. Each row shows the original image, the reconstruction, pose, gender, expression and age.

Table 3: Quantitative comparison of image reconstruction.

Metric	LIA/GAN	ALI	MSE	VAE/GAN
FID	**19.26**	74.98	44.79	22.26
SWD	**12.16**	15.09	43.44	15.82
MSE	**11.79**	32.61	18.81	23.18

First, ALAE uses the style-based encoder that is more complex than ours. For LIA, there is no special constraint to the architecture of the encoder. Second, the encoder of ALAE is also the main module for feature extraction used in reconstruction loss and the discriminator, which allows the end-to-end training of the whole algorithm. However, LIA can be integrated with any GAN frameworks due to its much more flexible modular design and the scheme of the two-stage training. Third, we explicitly interpret and reveal the underlying reason why GAN inference needs to be performed in the deterministic $\boldsymbol{y}$-space instead of the stochastic $\boldsymbol{z}$-space. Finally, the $\boldsymbol{y}$-space and the $\boldsymbol{z}$-space are exactly invertible for LIA, which provides a convenient way of investigating one from another. We will demonstrate

the application of this invertibility in the experiment section.

6 Experiments

For experimental setup, we instantiate the decoder of LIA with the generator of StyleGAN (Karras et al, 2018b). The difference is that we replace the mapping network (MLP) in StyleGAN with the invertible network. The layer number of the invertible network is 8. The hyper-parameters for the discriminator are $\gamma = 10$ (equation (6)), $\beta_1 = 5e\text{-}5$ and $\beta_2 = 0.1$ (equation (5) and equation (8) respectively). For perceptual loss in equation (5), we take $\epsilon = \mathsf{conv4\_3}$ from the VGG weight.

The generative models we compare are the combination of VAE and GAN (VAE/GAN)[3], the MSE-based GAN inversion (Radford et al, 2016; Berthelot et al, 2017; Lipton and Tripathi, 2017), and the adversarially learned inference (ALI) (Dumoulin et al, 2017). To evaluate the necessity of the invertible network, we also

---
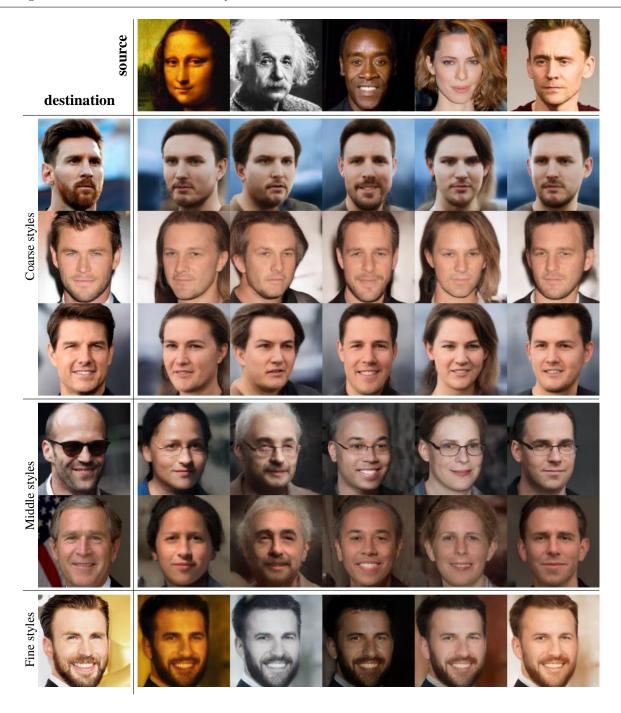[3] Code comes from `https://github.com/Puzer/stylegan-encoder`

Fig. 5: Style mixing for real faces. First column indicates source images and first row shows destination images. The coarse styles, middle styles and the fine style show the destination's latent codes were replaced using source latent codes at resolution 4^2-8^2, 16^2-32^2, 64^2-128^2, respectively.

train an encoder and a StyleGAN with its original multilayer perceptron, which is the last column in Figure 3. The two-stage training scheme is used as LIA does. The generator and discriminator of the StyleGAN is exactly same to that of StyleGAN.

For quantitative evaluation metrics, we use Fréchet inception distance (FID), sliced Wasserstein distance (SWD), and mean square error (MSE). These three metrics are commonly used to measure the numerical accuracy of GANs (Ulyanov et al, 2017; Karras et al, 2018a; Donahue et al, 2017; Karras et al, 2018b). We
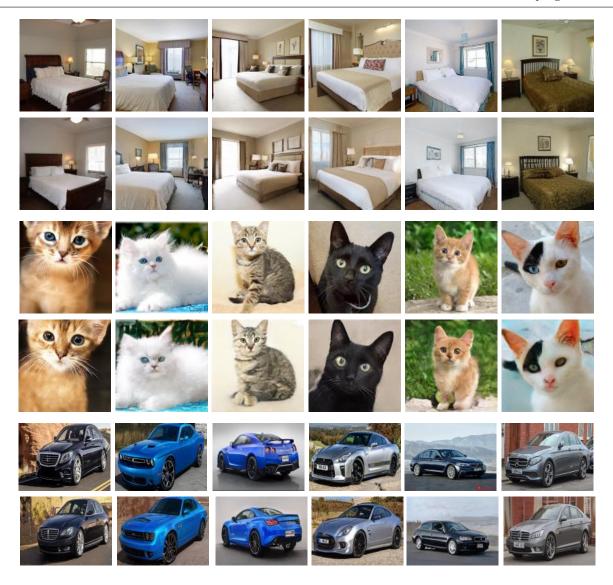
Fig. 6: The exemplar real images of objects and scenes from LSUN database validation set and their reconstructed images by LIA. Three categories are tested, *i.e.* cat, bedroom, and car. For each group, the first row is the original images, the second row shows the reconstructed images.

directly use the code released by the authors of Pro-GAN (Karras et al, 2018a). The Gaussian prior for z is of dimension 512.

6.1 FFHQ Face Database

All models are first tested on the Flickr-Faces-HQ (FFHQ) database[4] created by the authors of StyleGAN as the benchmark. FFHQ contains 70,000 high-quality face images. We take the first 65,000 faces as the training set and the remaining 5,000 faces as the reconstruction test according to the exact order of the dataset. We do

---

[4] https://github.com/NVlabs/ffhq-dataset

not split the dataset by random sampling for interested readers can precisely reproduce all the reported results with our experimental protocol.

Figure 3 shows the reconstructed faces of all models. It is clear that LIA significantly outperforms others. The reconstructed faces by ALI and AGE look correct, but the quality is mediocre. The ideas of ALI and AGE are elegant. Their performance may be improved with the new techniques such as progressive growing of neural architecture or style-based one. The method of the MSE-based optimization produces facial parts of comparable quality with LIA when the faces are normal. But this approach fails when the variations of faces become large. For example, the failure comes from the long fair, hats,

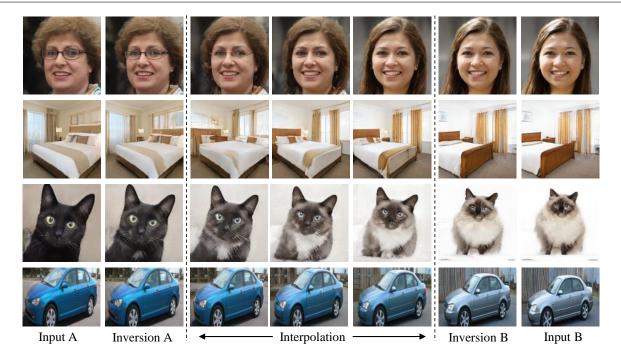Input A   Inversion A   ←——— Interpolation ———→   Inversion B   Input B

Fig. 7: Interpolation on real images using LIA.

beard, and large pose. The interesting phenomenon is that the StyleGAN with encoder only does not succeed in recovering the target faces using the same training strategy as LIA, even though it is capable of generating photo-realistic faces in high quality due to the StyleGAN generator. This indicates that the invertible network plays the crucial role to make the LIA work. The quantitative result in Table 3 shows the consistent superiority of LIA.

The reconstruction from LIA facilitates semantic photo editing. Figure 4 shows the manipulation results on attributes of reconstructed faces and Figure 5 displays the style mixing of real faces.

6.2 LSUN Database

To further evaluate LIA on the data with large variations, we use the three categories from the large-scale LSUN database (Yu et al, 2015), i.e. cat, car, and bedroom. For each category, the 0.1 million images are selected by ranking algorithm (Zhou et al, 2003) from the first 0.5 million images in the dataset. Each cat and bedroom images are resized to be 128×128 and the size of the car image is 128×96 for training. We take subsets because it does not take too long for training to converge while still maintains the data complexity. These subsets will be made available for evaluation.

Figure 6 shows that the reconstructed objects by LIA faithfully maintain the semantics as well as the

appearance of the original ones. For example, the cats' whiskers are recovered, indicating that LIA is able to recover very detailed information. We can see that LIA significantly improves the reconstruction quality. The improvement mainly comes from the two-stage training of LIA. The decoder trained with adversarial learning guarantees that the generated images are photo-realistic. The encoder deterministically trained with perceptual and adversarial losses ensures that latent feature vectors can be obtained more precisely. This two-stage training is enabled by the design that the invertible network detachs the encoder and decoder, thus avoiding the optimization of the posterior probability when learning the encoder. Figure 7 shows the interpolation results on those reconstructed objects. We can see that the objects are smoothly deformed and the photo-realistic effect is well preserved during deformation. The experimental results on FFHQ and LSUN databases verify that the symmetric design of the invertible network and the two-stage training successfully handles the issue of GAN inference.

6.3 Finetuning the Latent Code to a Specific Image

The reconstruction precision can be further improved by two ways: increasing the dimension of the intermediate latent space and finetuning the model to fit a specific image. A large latent space can preserve more semantic information of images and the image-wise finetuning can
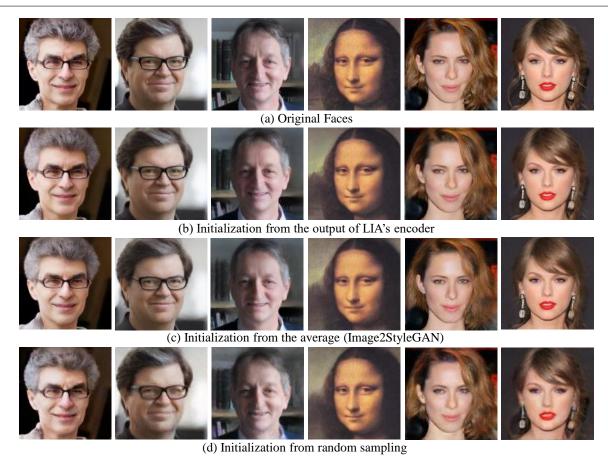
(a) Original Faces

(b) Initialization from the output of LIA's encoder

(c) Initialization from the average (Image2StyleGAN)

(d) Initialization from random sampling

Fig. 8: Reconstruction from finetuning the high-dimensional latent code to a specific image. Different initialized codes are compared.

(a)   (b)   (c)   (d)

Fig. 9: Detailed comparison on reconstructed faces from finetuning. (a) Original faces. (b) Initialization from the output of LIA's encoder. (c) Initialization from the average (Image2StyleGAN). (d) Initialization from random sampling.

recover specific features of objects or scenes. The Image2StyleGAN algorithm (Abdal et al, 2019) is such an example, which assigns the different ys to the different layers of the StyleGAN generator with the optimization of the MSE-based GAN inversion. The different ys span a high-dimensional latent space. As analyzed in section 2, the GAN inversion heavily depends on the initialization of y, even for the disentangled case. The

original Image2StyleGAN method uses the average y as the initial value to ameliorate the problem. To show the superiority of inference capability of LIA, we compare the three cases of initialization for model finetuning via GAN inversion: random sampling, the average, and the output of LIA's encoder. The associated optimization is
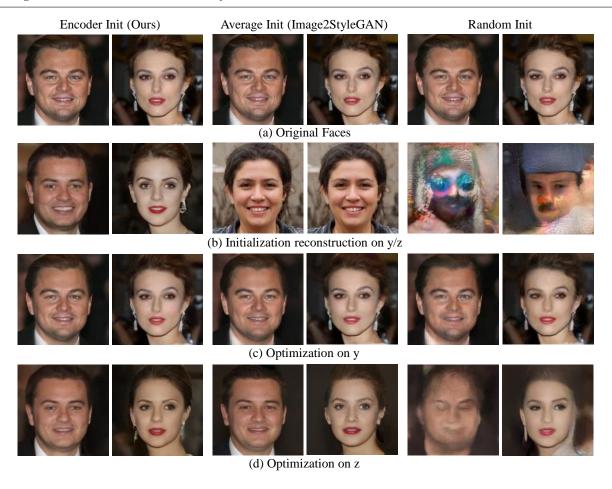
Encoder Init (Ours)   Average Init (Image2StyleGAN)   Random Init

(a) Original Faces

(b) Initialization reconstruction on y/z

(c) Optimization on y

(d) Optimization on z

Fig. 10: Optimization on the latent code $\boldsymbol{z}$ and the feature $\boldsymbol{y}$. The faces in the second row are reconstructed by the different initial values of $\boldsymbol{z}/\boldsymbol{y}$. The third row shows the result on $\boldsymbol{y}$ and the fourth row shows the result on $\boldsymbol{z}$.

based on the reconstruction loss

$$\boldsymbol{y}^* = \arg\min_{\boldsymbol{y}} L(\epsilon, \boldsymbol{x}, \tilde{\boldsymbol{x}}), \quad \tilde{\boldsymbol{x}} = g(\boldsymbol{y}), \tag{13}$$

where the synthesis network g is well trained and frozen during the inversion task. For our method, we take the initial value of $\boldsymbol{y}$ as the encoded latent code $\boldsymbol{y}_0 = f(\boldsymbol{x})$.

As shown in Figure 8, the reconstructed faces from LIA are the best. For example, Monnalisa's hair style near the right eye is correctly recovered from the latent codes of LIA whereas Image2StyleGAN and random sampling both fail. Figure 9 compares the detailed reconstruction quality from local facial parts. The consistent superiority of LIA can be quantitatively verified from the losses shown in Figure 11.

6.4 Disentanglement vs. Entanglement for GAN Inversion

To further validate the analysis about disentanglement and entanglement in section 2, we also conduct the experiment in the preceding section on the entangled $\boldsymbol{z}$-space. The same optimization method of GAN inversion in (13) is employed. For the $\boldsymbol{z}$-space, the initial $\boldsymbol{z}_0$s for three algorithms are derived from $\boldsymbol{y}$-to-$\boldsymbol{z}$ mapping via the invertible network, i.e. $\boldsymbol{z}_0 = \phi(\boldsymbol{y}_0)$. Namely, three algorithms use the exactly same initial reconstruction for the initialization in the $\boldsymbol{z}$-space and the $\boldsymbol{y}$-space, as the second row shows in Figure 10. From Figure 10(c) and (d), we can see the significantly better reconstruction results in the $\boldsymbol{y}$-space than in the $\boldsymbol{z}$-space. These three algorithms all fail to converge at the correct minima in the $\boldsymbol{z}$-space. An intriguing observation is that for random initialization, the inversion algorithm is capable of attaining the nearly correct reconstruction in the $\boldsymbol{y}$-space, whereas it totally fails to produce the meaningful face for the DiCaprio case in the $\boldsymbol{z}$-space. This is evidence that the entanglement of variables deteriorates the convergence of the algorithm.
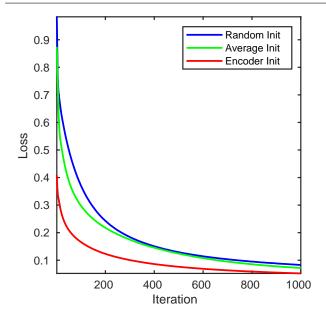
Fig. 11: Reconstruction loss when finetuning different latent codes as initialization. Three initialization codes are compared: randomly sampled $\boldsymbol{y}$, the average of $\boldsymbol{y}$s (used in the original Image2StyleGAN (Abdal et al, 2019)), and the output code $f(\boldsymbol{x})$ of the encoder from our LIA. Using the code from our method result in faster convergence and better reconstruction.

6.5 Disentanglement Facilitates Optimization

To further reveal the importance of the disentanglement in LIA, we investigate the gradients of the encoders under two different cases of decoders when learning the encoder in the two-stage training. The first case is to directly apply the KL-divergence to optimize the random latent space as VAE does. The mapping of the encoder and the decoder is $\boldsymbol{x} \overset{f}{\mapsto} \boldsymbol{z} \overset{\varphi}{\mapsto} \boldsymbol{y} \overset{g}{\mapsto} \tilde{\boldsymbol{x}}$, where φ is the mapping network (MLP) in StyleGAN. This is the conventional way of learning encoders for GAN algorithms via variational inference. The second one is our architecture of LIA for the second-stage training, i.e. $\boldsymbol{x} \overset{f}{\mapsto} \boldsymbol{y} \overset{g}{\mapsto} \tilde{\boldsymbol{x}}$, where the latent space is removed due to the symmetric design of the invertible network. Figure 12 clearly illustrates the difference of gradients between these two cases. The gradient volatility for variational inference is high and the associated loss is not effectively reduced, meaning that the gradients during training are noisy and not always informative. This may indicate that the entanglement in the latent space causes problems for training encoder via variational inference. Instead, the encoder's gradients for LIA are rather stable across different layers and the loss decreases monotonically,

showing the importance of the disentangled training via the invertible network.

7 Conclusion

A new generative model, named Latently Invertible Autoencoder (LIA), has been proposed for generating image sample from a probability prior and simultaneously inferring accurate latent code for a given sample. The core idea of LIA is to symmetrically embed an invertible network in an autoencoder. Then the neural architecture is trained with adversarial learning as two decomposed modules. With the design of two-stage training, the decoder can be replaced with any GAN generator for high-resolution image generation. The role of the invertible network is to remove any probability optimization and bridge the prior with unfolded feature vectors. The effectiveness of LIA is validated with experiments of reconstruction (inference and generation) on FFHQ and LSUN datasets. With accurate inference of latent codes by LIA, future work will be on facilitating various applications of GAN-based models, e.g. image editing, data augmentation, few-shot learning, and 3D graphics.

References

Abdal R, Qin Y, Wonka P (2019) Image2StyleGAN: How to embed images into the StyleGAN latent space? In: Proceedings of International Conference on Computer Vision (ICCV)

Arjovsky M, Chintala S, Bottou L (2017) Wasserstein GAN. In: arXiv:1701.07875

Berthelot D, Schumm T, Metz L (2017) BEGAN: boundary equilibrium generative adversarial networks. arXiv:170310717

Bowman SR, Vilnis L, Vinyals O, Dai AM, Jozefowicz R, Bengio S (2015) Generating sentences from a continuous space. In: arXiv:1511.06349

Brock A, Donahue J, Simonyan K (2018) Large scale GAN training for high fidelity natural image synthesis. In: arXiv:1809.11096

Dinh L, Krueger D, Bengio Y (2015) NICE: Non-linear independent components estimation. In: International Conference on Learning Representations (ICLR)

Dinh L, Sohl-Dickstein J, Bengio S (2017) Density estimation using Real NVP. In: International Conference on Learning Representations (ICLR)

Donahue J, Simonyan K (2019) Large scale adversarial representation learning. arXiv:190702544
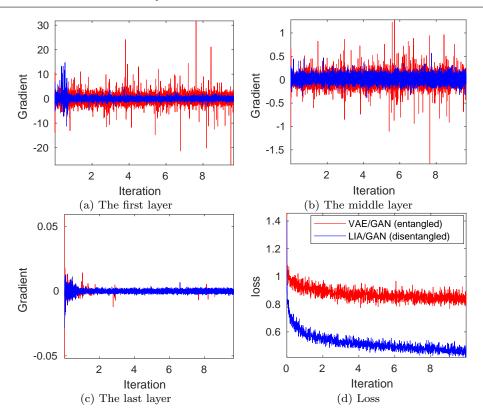
Fig. 12: Illustration of gradients and losses for training encoders on FFHQ dataset. For VAE, the decoder is exactly the generator of StyleGAN. The two-stage training of VAE is the same as that of LIA. The first and last layers in (a) and (b) refer to the neural architecture of the encoder. The y-axis indicates the average value of the gradients. The x-axis is re-scaled by 10,000. The gradients for LIA case are much more stable than the ones for VAE.

Donahue J, Krahenbuhl P, Darrell T (2017) Adversarial feature learning. In: International Conference on Learning Representations (ICLR)

Dumoulin V, Belghazi I, Poole B, Mastropietro O, Lamb A, Arjovsky M, Courville A (2017) Adversarially learned inference. In: International Conference on Learning Representations (ICLR)

Eslami SMA, Rezende DJ, Besse F, Viola F, Morcos AS, Garnelo M, Ruderman A, Rusu AA, Danihelka I, Gregor K, Reichert DP, Buesing L, Weber T, Vinyals O, Rosenbaum D, Rabinowitz N, King H, Hillier C, Botvinick M, Wierstra D, Kavukcuoglu K, Hassabis D (2018) Neural scene representation and rendering. Science 360(6394):1204–1210

Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial networks. In: Advances in Neural Information Processing Systems (NeurIPS)

Grover A, Dhar M, Ermon S (2017) Flow-GAN: Combining maximum likelihood and adversarial learning in generative models. In: arXiv:1705.08868

Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville A (2017) Improved training of Wasserstein GANs. In: arXiv:1704.00028

Heljakka A, Solin A, Kannala J (2018) Pioneer networks: Progressively growing generative autoencoder. In: arXiv:1807.03026

Isola P, Zhu JY, Zhou T, Efros AA (2017) Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Johnson J, Alahi A, Fei-Fei L (2016) Perceptual losses for real-time style transfer and super-resolution. arXiv:160308155

Karras T, Aila T, Laine S, Lehtinen J (2018a) Progressive growing of GANs for improved quality, stability, and variation. In: Proceedings of the 6th International Conference on Learning Representations (ICLR)

Karras T, Laine S, Aila T (2018b) A style-based generator architecture for generative adversarial networks. arXiv:181204948

Karras T, Laine S, Aittala M, Hellsten J, Lehtinen J, Aila T (2019) Analyzing and improving the image quality of stylegan. arXiv:191204958

Kingma DP, Dhariwal P (2018) Glow: Generative flow with invertible 1x1 convolutions. In: arXiv:1807.03039

Kingma DP, Salimans T, Jozefowicz R, Chen X, Sutskever I, Welling M (2016) Improving variational inference with inverse autoregressive flow. In: arXiv:1606.04934

Larsen ABL, Sønderby SK, Larochelle H, Winther O (2016) Autoencoding beyond pixels using a learned similarity metric. In: International Conference on Machine Learning (ICML), pp 1558–1566

Lehtinen J, Munkberg J, Hasselgren J, Laine S, Karras T, Aittala M, Aila T (2018) Noise2Noise: Learning image restoration without clean data. In: Proceedings of the 35th International Conference on Machine Learning (ICML)

Lipton ZC, Tripathi S (2017) Precise recovery of latent vectors from generative adversarial networks. In: International Conference on Learning Representations (ICLR)

Lucas T, Shmelkov K, Alahari K, Schmid C, Verbeek J (2019) Adversarial training of partially invertible variational autoencoders. arXiv:190101091

Luo J, Xu Y, Tang C, Lv J (2017) Learning inverse mapping by autoencoder based generative adversarial nets. In: arXiv:1703.10094

Makhani A (2018) Implicit autoencoders. In: arXiv:1805.09804

Makhzani A, Shlens J, Jaitly N, Goodfellow I, Frey B (2015) Adversarial autoencoders. In: arXiv:1511.05644

Mescheder L, Geiger A, Nowozin S (2018) Which training methods for GANs do actually converge? arXiv:180104406

Miyato T, Kataoka T, Koyama M, Yoshida Y (2018) Spectral normalization for generative adversarial networks. arXiv:180205957

van den Oord A, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior A, Kavukcuoglu K (2016) WaveNet: A generative model for raw audio. In: arXiv:1609.03499

van den Oord A, Li Y, Babuschkin I, Simonyan K, Vinyals O, Kavukcuoglu K, van den Driessche G, Lockhart E, Cobo LC, Stimberg F, Casagrande N, Grewe D, Noury S, Dieleman S, Elsen E, Kalchbrenner N, Zen H, Graves A, King H, Walters T, Belov D, Hassabis D (2017) Parallel WaveNet: Fast high-fidelity speech synthesis. In: arXiv:1711.10433

van den Oord A, Li Y, Vinyals O (2018) Representation learning with contrastive predictive coding. arXiv:180703748

Pidhorskyi S, Adjeroh DA, Doretto G (2020) Adversarial latent autoencoders. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Radford A, Metz L, Chintala S (2016) Unsupervised representation learning with deep convolutional generative adversarial networks. In: Proceedings of the 4th International Conference on Learning Representations (ICLR)

Shen Y, Gu J, Tang X, Zhou B (2020) Interpreting the latent space of gans for semantic face editing. Computer Vision and Pattern Recognition

Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556

Su J, Wu G (2018) f-VAEs: Improve VAEs with conditional flows. In: arXiv:1809.05861

Ulyanov D, Vedaldi A, Lempitsky V (2017) It takes (only) two: Adversarial generator-encoder networks. In: arXiv:1704.02304

Wang J, Yu L, Zhang W, Gong Y, Xu Y, Wang B, Zhang P, Zhang D (2017) IRGAN: A minimax game for unifying generative and discriminative information retrieval models. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval

Wu J, Zhang C, Xue T, Freeman WT, Tenenbaum JB (2016) Learning a probabilistic latent space of object shapes via 3D generative adversarial modeling. In: Advances in Neural Information Processing Systems (NeurIPS), pp 82–90

Xiao Z, Yan Q, Amit Y (2019) Generative latent flow. arXiv:190510485

Yu F, Seff A, Zhang Y, Song S, Funkhouser T, Xiao J (2015) LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv:150603365

Zhou D, Weston J, Gretton A, Bousquet O, Schölkopf B (2003) Ranking on data manifolds. In: Proceedings of the 16th International Conference on Neural Information Processing Systems (NeurIPS)

Zhu B, Liu JZ, Cauley SF, Rosen BR, Rosen MS (2018) Image reconstruction by domain-transform manifold learning. Nature 555:487–492

Zhu JY, Park T, Isola P, Efros AA (2017) Unpaired image-to-image translation using cycle-consistent adversarial networks. In: International Conference on Computer Vision (ICCV)