In-Domain GAN Inversion for Real Image Editing

Jiapeng Zhu*[1][0000−0001−9198−0304], Yujun Shen*[1][0000−0003−3801−6705], Deli Zhao[2][0000−0002−8838−578X], and Bolei Zhou[1][0000−0003−4030−0684]

[1] The Chinese University of Hong Kong
{jpzhu, sy116, bzhou}@ie.cuhk.edu.hk
[2] Xiaomi AI Lab
zhaodeli@gmail.com

Abstract. Recent work has shown that a variety of semantics emerge in the latent space of Generative Adversarial Networks (GANs) when being trained to synthesize images. However, it is difficult to use these learned semantics for real image editing. A common practice of feeding a real image to a trained GAN generator is to invert it back to a latent code. However, existing inversion methods typically focus on reconstructing the target image by pixel values yet fail to land the inverted code in the semantic domain of the original latent space. As a result, the reconstructed image cannot well support semantic editing through varying the inverted code. To solve this problem, we propose an *in-domain* GAN inversion approach, which not only faithfully reconstructs the input image but also ensures the inverted code to be semantically meaningful for editing. We first learn a novel *domain-guided* encoder to project a given image to the native latent space of GANs. We then propose *domain-regularized* optimization by involving the encoder as a regularizer to fine-tune the code produced by the encoder and better recover the target image. Extensive experiments suggest that our inversion method achieves satisfying real image reconstruction and more importantly facilitates various image editing tasks, significantly outperforming start-of-the-arts.[1]

1 Introduction

Generative Adversarial Networks (GANs) [12] are formulated as a two-player game between a generator to synthesize images and a discriminator to differentiate real data from fake data. Recent work [11, 16, 30] has shown that GANs spontaneously learn to encode rich semantics inside the latent space and that varying the latent code leads to the manipulation of the corresponding attributes occurring in the output image. However, it remains difficult to apply such manipulation capability to real images since GANs lack the ability of taking a particular image as the input to infer its latent code.

* denotes equal contribution.
[1] Code and models are available at https://genforce.github.io/idinvert/.

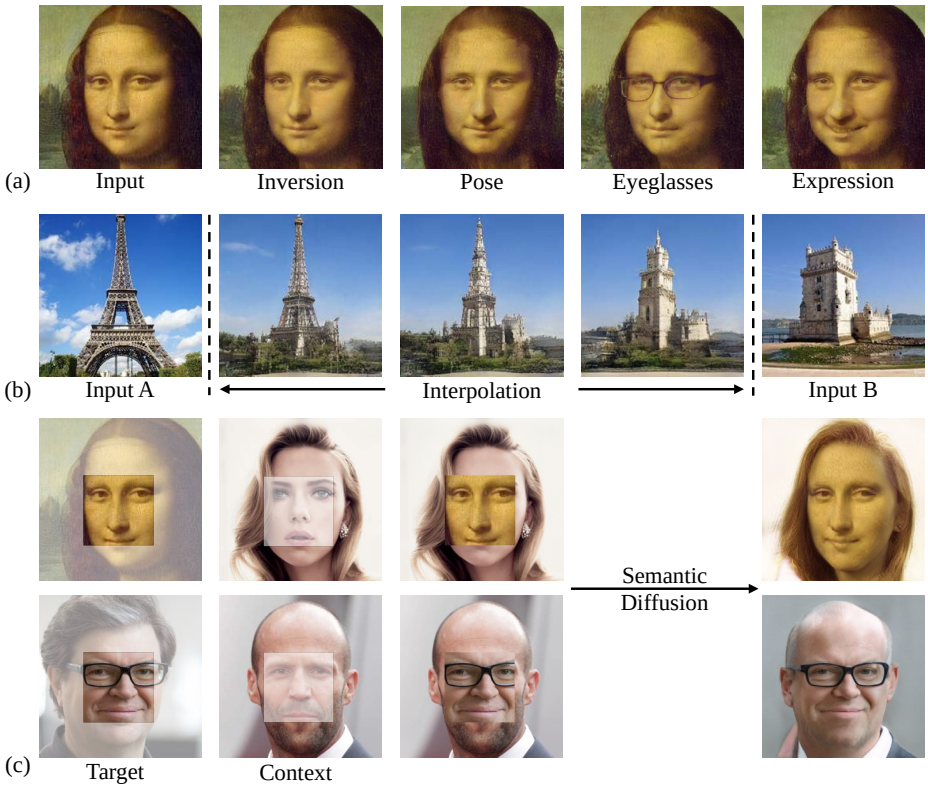

Fig. 1. Real image editing using the proposed *in-domain* GAN inversion with a *fixed* GAN generator. (a) Semantic manipulation with respect to various facial attributes. (b) Image interpolation by linearly interpolating two inverted codes. (c) Semantic diffusion which diffuses the target face to the context and makes them compatible.

Many attempts have been made to reverse the generation process by mapping the image space back to the latent space, which is widely known as *GAN inversion*. They either learn an extra encoder beyond the GAN [23, 36, 4] or directly optimize the latent code for an individual image [22, 24, 8]. However, existing methods mainly focus on reconstructing the pixel values of the input image, leaving some important open questions about the property of the inverted code. For example, does the inverted code lie in the original latent space of GANs? Can the inverted code semantically represent the target image? Does the inverted code support image editing by reusing the knowledge learned by GANs? Can we use a well-trained GAN to invert any image? Answering these questions not only deepens our understanding of the internal mechanism of GANs, but is also able to unleash the pre-trained GAN models for the versatile image editing capability.

In this work, we show that a good GAN inversion method should not only reconstruct the target image at the *pixel* level, but also align the inverted

code with the *semantic* knowledge encoded in the latent space. We call such semantically meaningful codes as *in-domain* codes since they are subject to the semantic domain learned by GANs. We also find that in-domain codes can better support image editing by reusing the rich knowledge emerging in the GAN models. To this end, we propose an *in-domain* GAN inversion approach to recover the input image at *both the pixel level and the semantic level*. Concretely, we first train a novel *domain-guided* encoder to map the image space to the latent space such that all codes produced by the encoder are in-domain. We then perform instance-level *domain-regularized* optimization by involving the encoder as a regularizer to better reconstruct the pixel values without affecting the semantic property of the inverted code. We summarize our contributions as follows:

- We analyze an important issue in the GAN inversion task that the inverted code should go beyond merely recovering the per-pixel values of the input image by further considering the semantic information.
- We propose an *in-domain* GAN inversion approach by first learning a *domain-guided* encoder and further use this encoder as a regularizer for *domain-regularized* optimization.
- We evaluate our method on a variety of image editing tasks, as shown in Fig.1. Qualitative and Quantitative results suggest that our *in-domain* inversion can faithfully recover the target image from both the low-level pixels and the high-level semantics, significantly surpassing existing approaches.

1.1 Related Work

Generative Adversarial Networks (GANs). By learning the distribution of real images via adversarial training, GANs [12] have advanced image synthesis in recent years. Many variants of GANs are proposed to improve the synthesis quality [25, 34, 18, 7, 19] and training stability [2, 14, 6]. Recently, GANs are shown to spontaneously learn semantics inside the latent space, which can be further used to control the generation process. Goetschalckx *et al.* [11] explored how to make the synthesis from GANs more memorable, Jahanian *et al.* [16] achieved camera movements and color changes by shifting the latent distribution, Shen *et al.* [30] interpreted the latent space of GANs for semantic face editing, and Yang *et al.* [32] observed that hierarchical semantics emerge from the layer-wise latent codes of GANs for scene synthesis. However, due to the lack of inference capability in GANs, it remains difficult to apply the rich semantics encoded in the latent space to editing real images.

GAN Inversion. To better apply well-trained GANs to real-world applications, GAN inversion enables real image editing from the latent space [36, 27, 3]. Given a fixed GAN model, GAN inversion aims at finding the most accurate latent code to recover the input image. Existing inversion approaches typically fall into two types. One is learning-based, which first synthesizes a collection of images with randomly sampled latent codes and then uses the images and codes as inputs and supervisions respectively to train a deterministic model [27, 36]. The other is optimization-based, which deals with a single instance at one time by directly

optimizing the latent code to minimize the pixel-wise reconstruction loss [22, 8, 24, 29]. Some work combines these two ideas by using the encoder to generate an initialization for optimization [5, 4]. There are also some models that take invertibility into account at the training stage by designing new architectures [10, 9, 35, 21]. Some concurrent work improves GAN inversion with better reconstruction quality: Gu *et al.* [13] employs multiple latent codes to recover a single image, Pan *et al.* [26] optimizes the parameters of the generator together with the latent code, Karras *et al.* [20] and Abdal *et al.* [1] focus on inverting StyleGAN [19] models by exploiting the layer-wise noises.

Key Difference. One important issue omitted by existing inversion methods is that they merely focus on reconstructing the target image at the pixel level without considering the semantic information in the inverted code. If the code cannot align with the semantic domain of the latent space, even being able to recover the per-pixel values of the input image, it would still fail to reuse the knowledge learned by GANs for semantic editing. Therefore, in this work, we argue that only using the pixel-wise reconstruction loss as the metric to evaluate a GAN inversion approach is not proper enough. Instead, we deeply study the property of the inverted code from the *semantic* level and propose the *in-domain* GAN inversion that well supports real image editing.

2 In-Domain GAN Inversion

As discussed above, when inverting a GAN model, besides recovering the input image by pixel values, we also care about whether the inverted code is semantically meaningful. Here, the semantics refer to the emergent knowledge that GAN has learned from the observed data [11, 16, 30, 32]. For this purpose, we propose to first train a *domain-guided* encoder and then use this encoder as a regularizer for the further *domain-regularized* optimization, as shown in Fig.2.

Problem Statement. Before going into details, we briefly introduce the problem setting with some basic notations. A GAN model typically consists of a generator $G(\cdot) : \mathcal{Z} \to \mathcal{X}$ to synthesize high-quality images and a discriminator $D(\cdot)$ to distinguish real from synthesized data. GAN inversion studies the reverse mapping of $G(\cdot)$, which is to find the best latent code $\mathbf{z}^{inv}$ to recover a given real image $\mathbf{x}^{real}$. We denote the semantic space learned by GANs as $\mathcal{S}$. We would like $\mathbf{z}^{inv}$ to also align with the prior knowledge $\mathcal{S}$ in the pre-trained GAN model.

Choice of Latent Space. Typically, GANs sample latent codes $\mathbf{z}$ from a pre-defined distributed space $\mathcal{Z}$, such as normal distribution. The recent StyleGAN model [19] proposes to first map the initial latent space $\mathcal{Z}$ to a second latent space $\mathcal{W}$ with Multi-Layer Perceptron (MLP), and then feed the codes $\mathbf{w} \in \mathcal{W}$ to the generator for image synthesis. Such additional mapping has already been proven to learn more disentangled semantics [19, 30]. As a result, the disentangled space $\mathcal{W}$ is widely used for the GAN inversion task [29, 35, 1, 20]. Similarly, we also choose $\mathcal{W}$ space as the inversion space for three reasons: (i) We focus on the semantic (*i.e., in-domain*) property of the inverted codes, making $\mathcal{W}$ space more appropriate for analysis. (ii) Inverting to $\mathcal{W}$ space achieves better performance

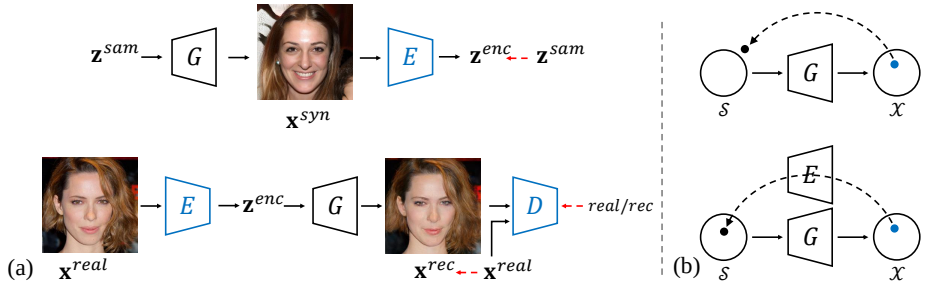

Fig. 2. (a) The comparison between the training of conventional encoder and *domain-guided* encoder for GAN inversion. Model blocks in **blue** are trainable and **red** dashed arrows indicate the supervisions. Instead of being trained with synthesized data to recover the latent code, our *domain-guided* encoder is trained with the objective to recover the real images. The *fixed* generator is involved to make sure the codes produced by the encoder lie in the native latent space of the generator and stay semantically meaningful. (b) The comparison between the conventional optimization and our *domain-regularized* optimization. The well-trained *domain-guided* encoder is included as a regularizer to land the latent code in the semantic domain during the optimization process.

than $\mathcal{Z}$ space [35]. (iii) It is easy to introduce the $\mathcal{W}$ space to any GAN model by simply learning an extra MLP ahead of the generator. Hence, it will not harm the generalization ability of our approach. In this work, we conduct all experiments on the $\mathcal{W}$ space, but our approach can be performed on the $\mathcal{Z}$ space as well. For simplicity, we use $\mathbf{z}$ to denote the latent code in the following sections.

2.1 Domain-Guided Encoder

Training an encoder is commonly used for GAN inversion problem [27, 36, 5, 4] considering its fast inference speed. However, existing methods simply learn a deterministic model with no regard to whether the codes produced by the encoder align with the semantic knowledge learned by $G(\cdot)$. As shown on the top of Fig.2(a), a collection of latent codes $\mathbf{z}^{sam}$ are randomly sampled and fed into $G(\cdot)$ to get the corresponding synthesis $\mathbf{x}^{syn}$. Then, the encoder $E(\cdot)$ takes $\mathbf{x}^{syn}$ and $\mathbf{z}^{sam}$ as inputs and supervisions respectively and is trained with

$$\min_{\Theta_E} \mathcal{L}_E = ||\mathbf{z}^{sam} - E(G(\mathbf{z}^{sam}))||_2, \tag{1}$$

where $||\cdot||_2$ denotes the l_2 distance and Θ_E represents the parameters of the encoder $E(\cdot)$. We argue that the supervision by only reconstructing $\mathbf{z}^{sam}$ is not powerful enough to train an accurate encoder. Also, the generator is actually omitted and cannot provide its domain knowledge to guide the training of encoder since the gradients from $G(\cdot)$ are not taken into account at all.

To solve these problems, we propose to train a *domain-guided* encoder, which is illustrated in the bottom row of Fig.2(a). There are three main **differences** compared to the conventional encoder: (i) The output of the encoder is fed into

the generator to reconstruct the input image such that the objective function comes from the image space instead of latent space. This involves semantic knowledge from the generator in training and provides more informative and accurate supervision. The output code is therefore guaranteed to align with the semantic domain of the generator. (ii) Instead of being trained with synthesized images, the *domain-guided* encoder is trained with real images, making our encoder more applicable to real applications. (iii) To make sure the reconstructed image is realistic enough, we employ the discriminator to compete with the encoder. In this way, we can acquire as much information as possible from the GAN model (*i.e.*, both two components of GAN are used). The adversarial training manner also pushes the output code to better fit the semantic knowledge of the generator. We also introduce perceptual loss [17] by using the feature extracted by VGG [31]. Hence, the training process can be formulated as

$$\begin{aligned}\min_{\Theta_E} \mathcal{L}_E = ||\mathbf{x}^{real} - G(E(\mathbf{x}^{real}))||_2 \ &+ \lambda_{vgg}||F(\mathbf{x}^{real}) - F(G(E(\mathbf{x}^{real})))||_2 \\ &- \lambda_{adv} \mathop{\mathbb{E}}_{\mathbf{x}^{real}\sim P_{data}} [D(G(E(\mathbf{x}^{real})))],\end{aligned} \tag{2}$$

$$\begin{aligned}\min_{\Theta_D} \mathcal{L}_D = \mathop{\mathbb{E}}_{\mathbf{x}^{real}\sim P_{data}} [D(G(E(\mathbf{x}^{real})))] \ &- \mathop{\mathbb{E}}_{\mathbf{x}^{real}\sim P_{data}} [D(\mathbf{x}^{real})] \\ &+ \frac{\gamma}{2} \mathop{\mathbb{E}}_{\mathbf{x}^{real}\sim P_{data}} [||\nabla_{\mathbf{x}} D(\mathbf{x}^{real})||_2^2],\end{aligned} \tag{3}$$

where P_{data} denotes the distribution of real data and γ is the hyper-parameter for the gradient regularization. λ_{vgg} and λ_{adv} are the perceptual and discriminator loss weights. $F(\cdot)$ denotes the VGG feature extraction model.

2.2 Domain-Regularized Optimization

Unlike the generation process of GANs which learns a mapping at the distribution level, *i.e.* from latent distribution to real image distribution, GAN inversion is more like an instance-level task which is to best reconstruct a given individual image. From this point of view, it is hard to learn a perfect reverse mapping with an encoder alone due to its limited representation capability. Therefore, even though the inverted code from the proposed *domain-guided* encoder can well reconstruct the input image based on the pre-trained generator and ensure the code itself to be semantically meaningful, we still need to refine the code to make it better fit the target individual image at the pixel values.

Previous methods [8, 24, 29] propose to gradient descent algorithm to optimize the code. The top row of Fig.2(b) illustrates the optimization process where the latent code is optimized "freely" based on the generator only. It may very likely produce an out-of-domain inversion since there are no constraints on the latent code at all. Relying on our *domain-guided* encoder, we design a *domain-regularized* optimization with two improvements, as shown at the bottom of Fig.2(b): (i) We use the output of the *domain-guided* encoder as an ideal starting point which avoids the code from getting stuck at a local minimum and also significantly shortens the optimization process. (ii) We include the *domain-guided* encoder

as a regularizer to preserve the latent code within the semantic domain of the generator. To summarize, the objective function for optimization is

$$\begin{aligned}\mathbf{z}^{inv} = \arg\min_{\mathbf{z}} \ ||\mathbf{x} - G(\mathbf{z})||_2 \ &+ \lambda_{vgg}||F(\mathbf{x}) - F(G(\mathbf{z}))||_2 \\ &+ \lambda_{dom}||\mathbf{z} - E(G(\mathbf{z}))||_2,\end{aligned} \tag{4}$$

where $\mathbf{x}$ is the target image to invert. λ_{vgg} and λ_{dom} are the loss weights corresponding to the perceptual loss and the encoder regularizer respectively.

3 Experiments

In this section, we experimentally show the superiority of the proposed *in-domain* GAN inversion over existing methods in terms of semantic information preservation, inversion quality, inference speed, as well as real image editing.

3.1 Experimental Settings

We conduct experiments on FFHQ dataset [19], which contains 70,000 high-quality face images, and LSUN dataset [33], which consists of images from 10 different scene categories. Only results on the tower category are shown in the main paper. More results can be found in the **supplementary material**. The GANs to invert are pre-trained following StyleGAN [19].[2] When training the encoder, the generator is *fixed* and we only update the encoder and discriminator according to Eq.(2) and Eq.(3). As for the perceptual loss in Eq.(2), we take `conv4_3` as the VGG [31] output. Loss weights are set as $\lambda_{vgg} = 5e^{-5}$, $\lambda_{adv} = 0.1$, and $\gamma = 10$. We set $\lambda_{dom} = 2$ in Eq.(4) for the *domain-regularized* optimization.

3.2 Semantic Analysis of the Inverted Codes

In this part, we evaluate how the inverted codes can semantically represent the target images. As pointed out by prior work [19, 30], the latent space of GANs is linearly separable in terms of semantics. In particular, for a binary attribute (*e.g.*, male *v.s.* female), it is possible to find a latent hyperplane such that all points from the same side correspond to the same attribute. We use this property to evaluate the alignment between the inverted codes and the latent semantics.

We collect 7,000 real face images and use off-the-shelf attribute classifiers to predict age (young *v.s.* old), gender (female *v.s.* male), eyeglasses (absence *v.s.* presence), and pose (left *v.s.* right). These predictions are considered as ground-truth. Then, we use the state-of-the-art GAN inversion method, Image2StyleGAN [29], and our proposed *in-domain* GAN inversion to invert these images back to the latent space of a *fixed* StyleGAN model trained on FFHQ dataset [19]. InterFaceGAN [30] is used to search the semantic boundaries for the aforementioned attributes in the latent space. Then, we use these boundaries as

[2] Different from StyleGAN, we use different latent codes for different layers.

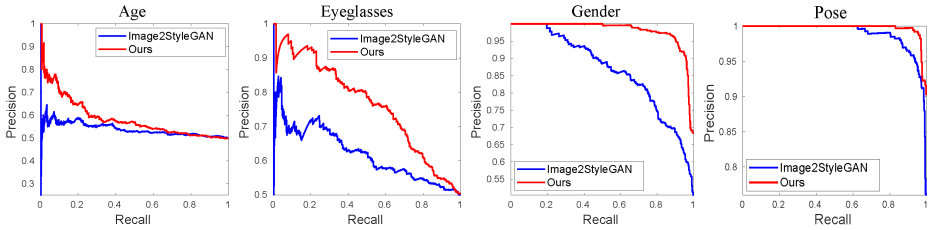

Fig. 3. Precision-recall curves by directly using the inverted codes for facial attribute classification. Our *in-domain* inversion shows much better performance than Image2StyleGAN [29], suggesting a stronger semantic preservation.

Table 1. Quantitative comparison between different inversion methods. For each model, we invert 500 images for evaluation. ↓ means lower number is better.

		Face			Tower		
Method	**Speed**	**FID↓**	**SWD↓**	**MSE↓**	**FID↓**	**SWD↓**	**MSE↓**
Traditional Encoder [36]	**0.008s**	88.48	100.5	0.507	73.02	69.19	0.455
MSE-based Optimization [29]	290s	58.04	29.19	**0.026**	69.16	55.35	0.068
Domain-Guided Encoder (Ours)	0.017s	52.85	**13.02**	0.062	46.81	27.13	0.071
In-Domain Inversion (Ours)	8s	**42.64**	13.44	0.030	**44.77**	**26.44**	**0.052**

well as the inverted codes to evaluate the attribute classification performance. Fig.3 shows the precision-recall curves on each semantic. We can easily tell that the codes inverted by our method are more semantically meaningful. This quantitatively demonstrates the effectiveness of our proposed *in-domain* inversion for preserving the semantics property of the inverted code.

3.3 Inversion Quality and Speed

As discussed above, our method can produce *in-domain* codes for the GAN inversion task. In this part, we would like to verify that the improvement of our approach from the semantic aspect does not affect its performance on the traditional evaluation metric, *i.e.*, image reconstruction quality. Fig.4 shows the qualitative comparison between different inversion methods including training traditional encoder [36], MSE-based optimization [29], as well as our proposed *domain-guided* encoder and the *in-domain* inversion. Comparison between Fig.4(b) and Fig.4(d) shows the superiority of our *domain-guided* encoder in learning a better mapping from the image space to the latent space. Also, our full algorithm (Fig.4(e)) shows the best reconstruction quality. Tab.1 gives the quantitative comparison results, where *in-domain* inversion surpasses other competitors from all metrics, including Frchet Inception Distance (FID) [15], Sliced Wasserstein Discrepancy (SWD) [28], and Mean-Square Error (MSE). The inference speed is also shown in Tab.1. Our *domain-guided* encoder can produce much better reconstruction results compared to the traditional encoder with comparable inference time. It also provides a better initialization for further the *domain-regularized* optimization, leading to a significantly faster speed (∼35X faster) than the state-of-the-art optimization-based method [29].

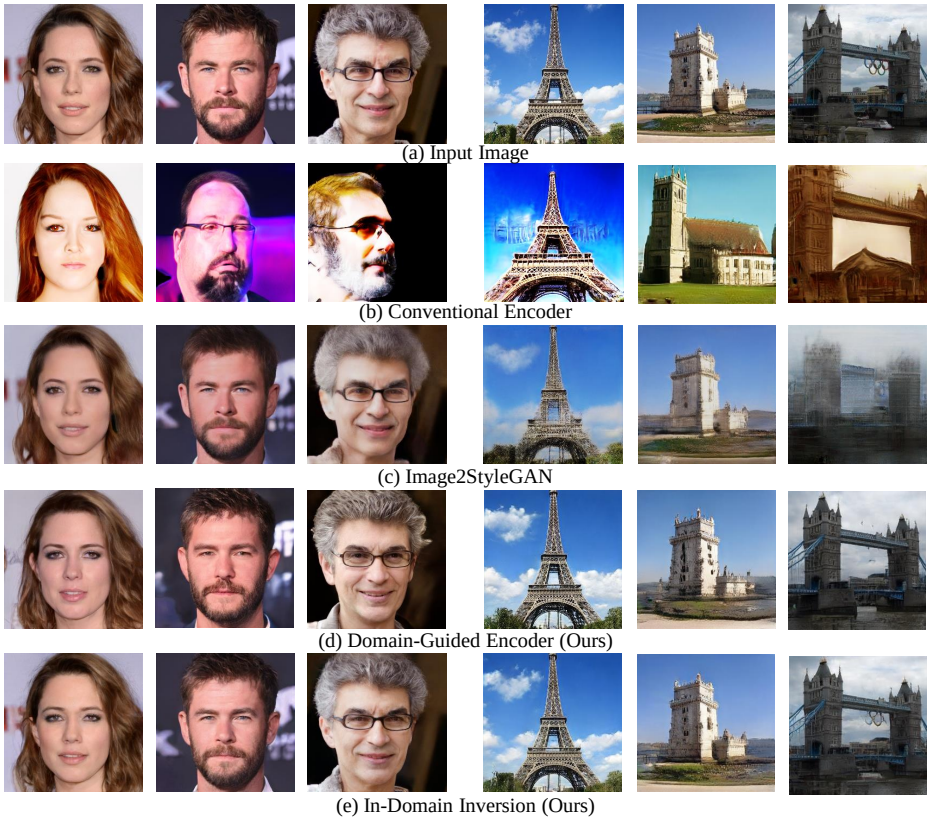

Fig. 4. Qualitative comparison on image reconstruction with different GAN inversion methods. (a) Input image. (b) Conventional encoder [36]. (c) Image2StyleGAN [29]. (d) Our proposed *domain-guided* encoder. (e) Our proposed *in-domain* inversion.

3.4 Real Image Editing

In this section, we evaluate our *in-domain* GAN inversion approach on real image editing tasks, including image interpolation and semantic image manipulation. We also come up with a novel image editing task, called *semantic image diffusion*, to see how our approach is able to adapt the content from one image into another and keep the results semantically meaningful and seamlessly compatible.

Image Interpolation. Image interpolation aims at semantically interpolating two images, which is suitable for investigating the semantics contained in the inverted codes. In other words, for a good inversion, the semantic should vary continuously when interpolating two inverted codes. Fig.5 shows the comparison results on the image interpolation task between Image2StyleGAN [29] and our *in-domain* inversion. We do experiments on both face and tower datasets to more comprehensively analyze the semantic property. For the face dataset, our method achieves much smoother interpolated faces than Image2StyleGAN. For example,

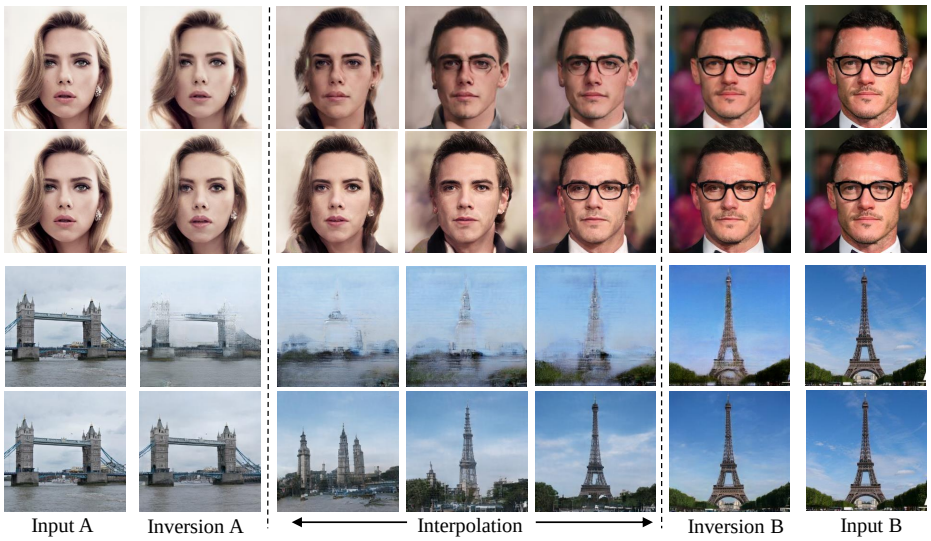

Fig. 5. Qualitative comparison on image interpolation between Image2StyleGAN [29] (odd rows) and our *in-domain* inversion (even rows).

Table 2. Quantitative comparison on image interpolation and manipulation between Image2StyleGAN [29] and our *in-domain* inversion. ↓ means lower number is better.

	Interpolation				Manipulation			
	Face		Tower		Face		Tower	
Method	**FID↓**	**SWD↓**	**FID↓**	**SWD↓**	**FID↓**	**SWD↓**	**FID↓**	**SWD↓**
MSE-based Optimization [29]	112.09	38.20	121.38	67.75	83.69	28.48	113	52.91
In-Domain Inversion (Ours)	**91.18**	**33.91**	**57.22**	**28.24**	**76.43**	**17.99**	**57.92**	**31.50**

in the first two rows of Fig.5, eyeglasses are distorted during the interpolation process with Image2StyleGAN and the change from female to male is unnatural. For tower images, which are much more diverse than faces, the interpolation results from Image2StyleGAN exhibit artifacts and blurriness. By contrast, our inverted codes lead to more satisfying interpolation. One noticeable thing is that during interpolating two towers with different types (*e.g.*, one with one spire and the other with multiple spires), the interpolated images using our approach are still high-quality towers. This demonstrates the *in-domain* property of our algorithm. Quantitative evaluation in Tab.2 gives the same conclusion.

Semantic Manipulation. Image manipulation is another way to examine whether the embedded latent codes align with the semantic knowledge learned by GANs. As pointed out by prior work [30, 32], GANs can learn rich semantics in the latent space, enabling image manipulation by linearly transforming the latent representation. This can be formulated as

$$\mathbf{x}^{edit} = G(\mathbf{z}^{inv} + \alpha \mathbf{n}), \tag{5}$$

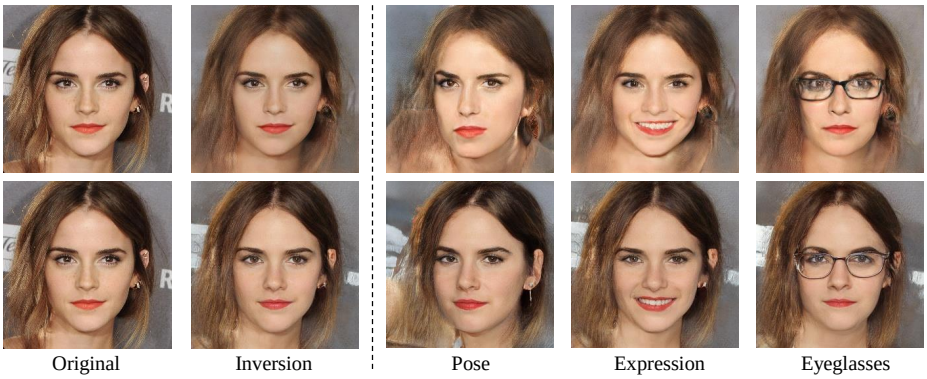

Fig. 6. Comparison of Image2StyleGAN [29] (top row) and our *in-domain* inversion (bottom row) on facial attribute manipulation.

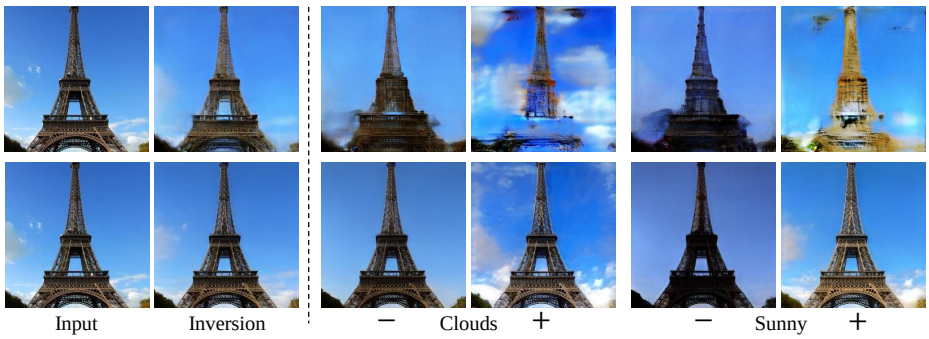

Fig. 7. Comparison of Image2StyleGAN [29] (top row) and our *in-domain* inversion (bottom row) on tower image editing. For the manipulation of each attribute, we show the results by either decreasing or increasing the semantic degree.

where $\mathbf{n}$ is the normal direction corresponding to a particular semantic in the latent space and α is the step for manipulation. In other words, if a latent code is moved towards this direction, the semantics contained in the output image should vary accordingly. We follow [30] to search the semantic direction $\mathbf{n}$.

Fig.6 and Fig.7 show the comparison results of manipulating faces and towers using Images2StyleGAN [29] and our *in-domain* GAN inversion. We can see that our method shows more satisfying manipulation results than Image2StyleGAN. Taking face manipulation (Fig.6) as an example, the hair of the actress becomes blurred after the pose rotation using Image2StyleGAN and the identity changes a lot when editing expression and eyeglasses with the codes from Image2StyleGAN. That is because it only focuses on the reconstruction of the per-pixel values yet omits the semantic information contained in the inverted codes. On the contrary, our *in-domain* inversion can preserve most other details when editing a

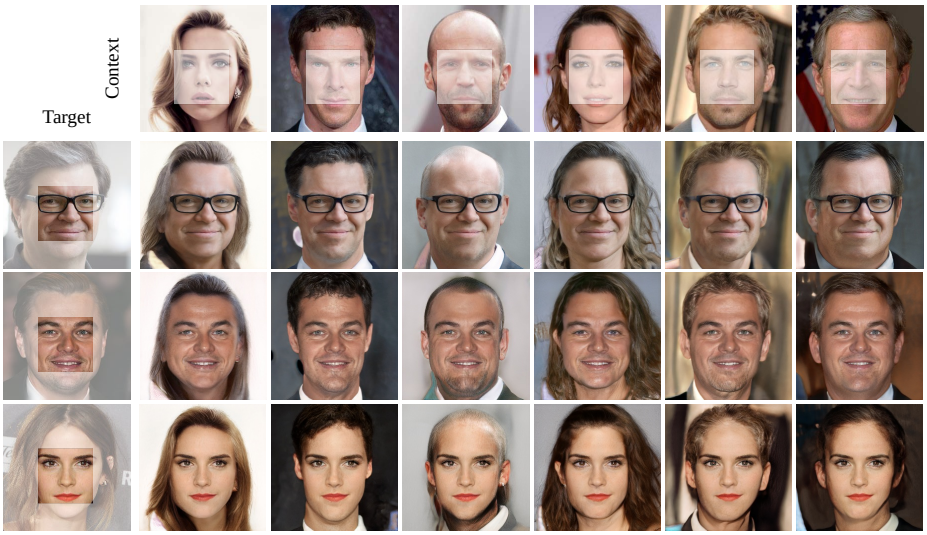

Fig. 8. Semantic diffusion result using our *in-domain* GAN inversion. Target images (first column) are seamlessly diffused into context images (first row) while the identify remains the same as the target.

particular facial attribute. As for tower manipulation, we observe from Fig.7 that our *in-domain* approach surpasses MSE-based optimization by both decreasing and increasing the semantic level. For example, when removing or adding clouds in the sky, Image2StyleGAN will blur the tower together with the sky, since it only recovers the image at the pixel level without considering the semantic meaning of the recovered objects. Therefore, the cloud is added to the entire image regardless whether a particular region belongs to sky or tower. By contrast, our algorithm barely affects the tower itself when editing clouds, suggesting that our *in-domain* inversion can produce semantically informative latent codes for image reconstruction. We also include the quantitative evaluation on the manipulation task in Tab.2. We can tell that our *in-domain* inversion outperforms Image2StyleGAN from all evaluation metrics.

Semantic Diffusion. Semantic diffusion aims at diffusing a particular part (usually the most representative part) of the target image into the context of another image. We would like the fused result to keep the characteristics of the target image (*e.g.*, identity of face) and adapt the context information at the same time. Fig.8 shows some examples where we successfully diffuse various target faces into diverse contexts using our *in-domain* GAN inversion approach. We can see that the results well preserve the identity of the target face and reasonably integrate into different surroundings. This is different from style mixing since the center region of our resulting image is kept the same as that of the target image. More detailed analysis on the semantic diffusion operation can be found in the **supplementary material**.

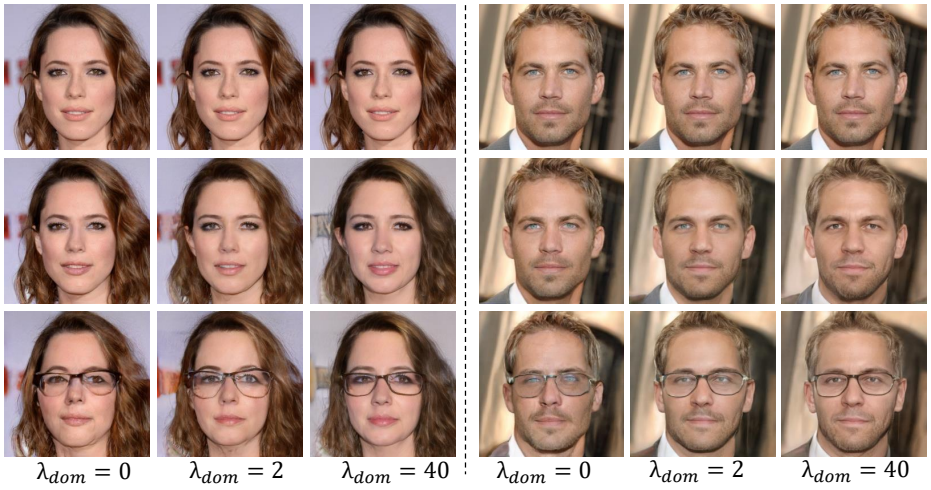

Fig. 9. Ablation study on the loss weight in Eq.(4) for the *domain-regularized* optimization. From top to bottom: original images, reconstructed images, and manipulation results (wearing eyeglasses). For each group of images, the weight λ_{dom} is set to be 0, 2, 40. When λ_{dom} equals to 0, it produces the best reconstructed results but relatively poor manipulation results. When λ_{dom} equals to 40, we get worse reconstruction but more satisfying manipulation.

3.5 Ablation Study

In this part, we conduct an ablation study to analyze the proposed *in-domain* inversion. After the initial training of the encoder, we perform the *domain-regularized* optimization on each image to further improve the reconstruction quality. Different from the previous MSE-based optimization, we involve the learned *domain-guided* encoder as a regularizer to land the inverted code inside the semantic domain, as described in Eq.(4). Here, we study the role of the encoder in the optimization process by varying the weight λ_{dom} in Eq.(4). Fig.9 shows the comparison between $\lambda_{dom} = 0, 2, 40$. We observe the trade-off between the image reconstruction quality and the manipulation quality. Larger λ_{dom} will bias the optimization towards the domain constraint such that the inverted codes are more semantically meaningful. Instead, the cost is that the target image cannot be ideally recovered for per-pixel values. In practice, we set $\lambda_{dom} = 2$.

4 Discussion and Conclusion

In this work, we explore the *semantic* property of the inverted codes in the GAN inversion task and propose a novel *in-domain* inversion method. To the best of our knowledge, this is the first attempt to invert a pre-trained GAN model *explicitly* considering the semantic knowledge encoded in the latent space. We show that the code that simply recovers the pixel value of the target image is not sufficient

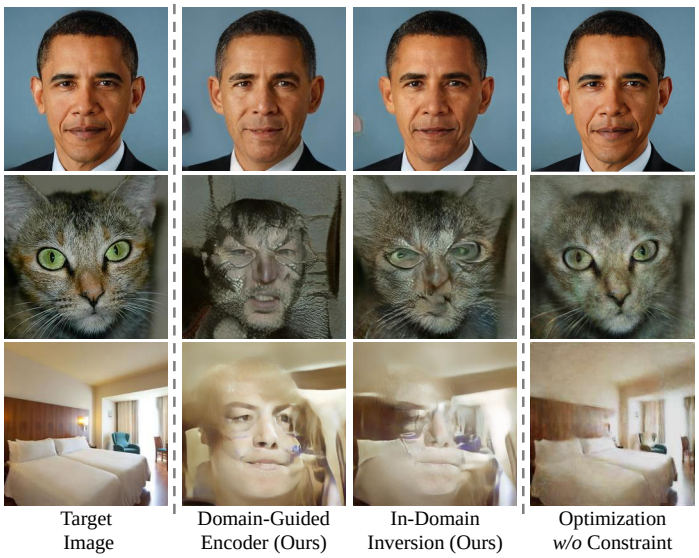

Fig. 10. Results on inverting face, cat face, and bedroom using the same face synthesis model. From left to right: target images, reconstruction results with the outputs from the *domain-guided* encoder, reconstruction results with the proposed *in-domain* inversion, reconstruction results by directly optimizing the latent code *w/o* considering domain alignment [29].

to represent the image at the semantic level. For example, in Fig.10, we invert different types of image instances (*i.e.*, face, cat face, and bedroom) with the face synthesis model. The last column shows the results from Image2StyleGAN [29] which recovers a cat or a bedroom with the domain knowledge learned to synthesis human faces. By contrast, the face outline can still be observed in the reconstructions using our *in-domain* inversion (third column). This demonstrates, from a different angle, the superiority of our approach in producing semantically meaningful codes. Taking inverting bedroom (third row) as an example, the bedroom image is outside the domain of the training data and the GAN model should not be able to learn the bedroom-related semantics. Accordingly, reusing the face knowledge to represent a bedroom is ill-defined. Even though we can always use more parameters to over-fit the pixel values of the bedroom (*e.g.*, the last column), such over-fitting would fail to support semantic image manipulation. From this viewpoint, our *in-domain* inversion lands the inverted code inside the original domain to make it semantically meaningful. In other words, we aim at finding the most adequate code to recover the target image from *both the pixel level and the semantic level*. Such *in-domain* inversion significantly facilitates real image editing.

Acknowledgement. This work is supported in part by the Early Career Scheme (ECS) through the Research Grants Council (RGC) of Hong Kong under Grant No.24206219, CUHK FoE RSFS Grant, and SenseTime Collaborative Grant.

References

1. Abdal, R., Qin, Y., Wonka, P.: Image2stylegan++: How to edit the embedded images? arXiv preprint arXiv:1911.11544 (2019)
2. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: ICML (2017)
3. Bau, D., Strobelt, H., Peebles, W., Wulff, J., Zhou, B., Zhu, J.Y., Torralba, A.: Semantic photo manipulation with a generative image prior. SIGGRAPH (2019)
4. Bau, D., Zhu, J.Y., Wulff, J., Peebles, W., Strobelt, H., Zhou, B., Torralba, A.: Inverting layers of a large generator. In: ICLR Workshop (2019)
5. Bau, D., Zhu, J.Y., Wulff, J., Peebles, W., Strobelt, H., Zhou, B., Torralba, A.: Seeing what a gan cannot generate. In: ICCV (2019)
6. Berthelot, D., Schumm, T., Metz, L.: Began: Boundary equilibrium generative adversarial networks. arXiv preprint arXiv:1703.10717 (2017)
7. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: ICLR (2019)
8. Creswell, A., Bharath, A.A.: Inverting the generator of a generative adversarial network. TNNLS (2018)
9. Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning. In: ICLR (2017)
10. Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., Courville, A.: Adversarially learned inference. In: ICLR (2017)
11. Goetschalckx, L., Andonian, A., Oliva, A., Isola, P.: Ganalyze: Toward visual definitions of cognitive image properties. In: ICCV (2019)
12. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NeurIPS (2014)
13. Gu, J., Shen, Y., Zhou, B.: Image processing using multi-code gan prior. In: CVPR (2020)
14. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. In: NeurIPS (2017)
15. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: NeurIPS (2017)
16. Jahanian, A., Chai, L., Isola, P.: On the "steerability" of generative adversarial networks. arXiv preprint arXiv:1907.07171 (2019)
17. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV (2016)
18. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. In: ICLR (2018)
19. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: CVPR (2019)
20. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. arXiv preprint arXiv:1912.04958 (2019)
21. Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. In: NeurIPS (2018)
22. Lipton, Z.C., Tripathi, S.: Precise recovery of latent vectors from generative adversarial networks. In: ICLR Workshop (2017)
23. Luo, J., Xu, Y., Tang, C., Lv, J.: Learning inverse mapping by autoencoder based generative adversarial nets. In: ICNIP (2017)

24. Ma, F., Ayaz, U., Karaman, S.: Invertibility of convolutional generative networks from partial measurements. In: NeurIPS (2018)
25. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. In: ICLR (2018)
26. Pan, X., Zhan, X., Dai, B., Lin, D., Loy, C.C., Luo, P.: Exploiting deep generative prior for versatile image restoration and manipulation. arXiv preprint arXiv:2003.13659 (2020)
27. Perarnau, G., Van De Weijer, J., Raducanu, B., Álvarez, J.M.: Invertible conditional gans for image editing. In: NeurIPS Workshop (2016)
28. Rabin, J., Peyré, G., Delon, J., Bernot, M.: Wasserstein barycenter and its application to texture mixing. In: International Conference on Scale Space and Variational Methods in Computer Vision (2011)
29. Rameen, A., Yipeng, Q., Peter, W.: Image2stylegan: How to embed images into the stylegan latent space? In: ICCV (2019)
30. Shen, Y., Gu, J., Tang, X., Zhou, B.: Interpreting the latent space of gans for semantic face editing. In: CVPR (2020)
31. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. ICLR (2015)
32. Yang, C., Shen, Y., Zhou, B.: Semantic hierarchy emerges in deep generative representations for scene synthesis. arXiv preprint arXiv:1911.09267 (2019)
33. Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., Xiao, J.: Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365 (2015)
34. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-attention generative adversarial networks. In: ICML (2019)
35. Zhu, J., Zhao, D., Zhang, B.: Lia: Latently invertible autoencoder with adversarial learning. arXiv preprint arXiv:1906.08090 (2019)
36. Zhu, J.Y., Krähenbühl, P., Shechtman, E., Efros, A.A.: Generative visual manipulation on the natural image manifold. In: ECCV (2016)