HyperInverter: Improving StyleGAN Inversion via Hypernetwork

Tan M. Dinh Anh Tuan Tran Rang Nguyen Binh-Son Hua

VinAI Research, Hanoi, Vietnam

{v.tandm3,v.anhtt152,v.rangnhm,v.sonhb}@vinai.io

Abstract

Real-world image manipulation has achieved fantastic progress in recent years as a result of the exploration and utilization of GAN latent spaces. GAN inversion is the first step in this pipeline, which aims to map the real image to the latent code faithfully. Unfortunately, the majority of existing GAN inversion methods fail to meet at least one of the three requirements listed below: high reconstruction quality, editability, and fast inference. We present a novel two-phase strategy in this research that fits all requirements at the same time. In the first phase, we train an encoder to map the input image to StyleGAN2 $\mathcal{W}$-space, which was proven to have excellent editability but lower reconstruction quality. In the second phase, we supplement the reconstruction ability in the initial phase by leveraging a series of hypernetworks to recover the missing information during inversion. These two steps complement each other to yield high reconstruction quality thanks to the hypernetwork branch and excellent editability due to the inversion done in the $\mathcal{W}$-space. Our method is entirely encoder-based, resulting in extremely fast inference. Extensive experiments on two challenging datasets demonstrate the superiority of our method. [1]

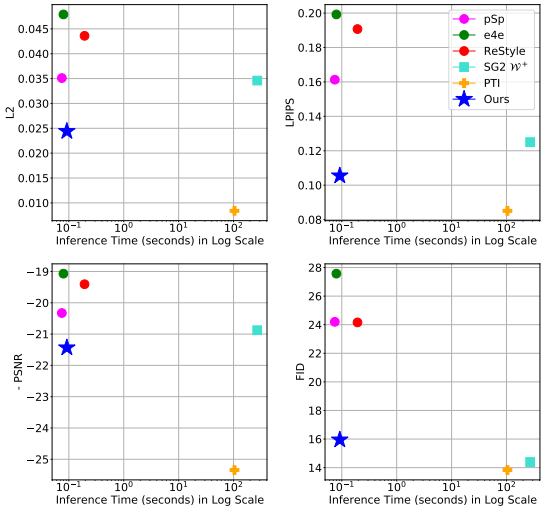

Figure 1. Our end-to-end encoder-based method has more accurate reconstruction while having fast inference (toward the bottom left of each plot). As can be seen, our work outperform other encoder-based inversion methods (pSp [37], e4e [44], ReStyle [3]) significantly. Comparing with per-image optimization inversion technique [1], our method is on par of quality but 3000× faster. Only PTI [38] has reconstruction quality better than our work. However, PTI requires per-image generator fine-tuning during the inference phase, which took a long time, 1100× longer.

1. Introduction

Generative adversarial networks (GANs) [13] in modern deep learning have allowed us to synthesize expressively realistic images, a trend that has continued to flourish in recent years. GANs can now be trained to generate images of high-resolution [22] with diverse styles [23–25] and apparently fewer artifacts [26]. Moreover, the latent spaces learned by these models also encode a diverse set of interpretable semantics. These semantics provide a tool to manipulate the synthesized images. Therefore, understanding and exploring a well-trained GAN model is an important and active research area. Several studies [7, 16, 40, 41, 50] have been conducted to examine the latent spaces learned by GANs, which convey a wide range of interpretable semantics.

To apply the semantic directions explored from GAN latent space to real-world images, the common-used practice is the "invert first, edit later" pipeline. *GAN Inversion* is a typical line of work that aims to first map a real photograph to a latent code of a GAN model so that the model can accurately reconstruct the photograph. Then, we can manipulate the latent code to edit different attributes of the reconstructed picture. There are two general approaches to determining a latent code of a GAN model given an input image: by an iterative optimization [1, 10, 21, 25, 32] and by inference with an encoder [35, 37, 44, 58]. The optimization-based method tends to perform reconstruction more accurately than encoder-based one but requires significantly more computa-

[1] Project page: https://di-mi-ta.github.io/HyperInverter

tion time, hindering use cases for interactive editing.

While the goal of GAN inversion is not only to reconstruct the input image faithfully but also to effectively perform image editing later, there is the so-called reconstruction-editing trade-off raised by multiple previous works [44, 59]. This trade-off is shown to depend on the embedding space where an input image is mapped to. The native StyleGAN $\mathcal{W}$ space and the extended version $\mathcal{W}^+$ space [1] are two most popular embedding spaces for StyleGAN inversion. Specifically, inverting an image to $\mathcal{W}$ space usually has excellent editability, but they are proved to be infeasible to reconstruct the input image faithfully [1]. On the contrary, $\mathcal{W}^+$ space allows to obtain more accurate reconstructions but it surfers from editing ability [44]. To mitigate the effects of this trade-off, a diverse set of methods has been proposed. Some of them [44, 59] introduce the ways (e.g., regularizer or adversarial training) to select the latent code in the high editable region of $\mathcal{W}^+$ space and accept a bit of sacrifice in the reconstruction quality. Another option is to utilize a two-stage approach. PIE [43] opts first to use an optimization process to locate the latent code in $\mathcal{W}$ space to preserve the editing ability and then optimize further the latent in the $\mathcal{W}^+$ space to enhance the reconstruction quality. PTI [38] approaches the same as PIE in the first stage. However, in the second stage, they choose the generator fine-tuning option to improve reconstruction results further. As can be seen, such methods require either optimization or fine-tuning process for each new image, leading to expensive inference time.

Motivated from above weakness, we propose a novel pure encoder-based two-phase method for StyleGAN inversion. Our method not only runs very fast but also obtains robust reconstruction quality. Specifically, in the first phase, we change to use a standard encoder to regress the image to the latent code in $\mathcal{W}$ space instead of utilizing optimization process as PTI and PIE. Then, in the second phase, we leverage the hypernetworks to predict the *residual weights*, which can recover the lost details of the input image after the first phase. We then use the residual weights to update the original generator for synthesizing the final reconstructed image. Our design would help migrate optimization or fine-tuning procedures in the inference phase, significantly reducing processing time. In summary, our contributions are:

- A completely encoder-based two-phase GAN inversion approach allows faithful reconstruction of real-world images while keeping editability and having fast inference;
- A novel network architecture that consists of a series of hypernetworks to update the weights of a pre-trained StyleGAN generator, thereby improving reconstruction quality;
- An extensive benchmark of GAN inversion that demonstrates the superior performance of our method compared to the existing works;
- A new method for real-world image interpolation that interpolates both latent code and generator weights.

2. Related Work

Latent Space Manipulation The latent space of a well-trained GAN generator (e.g., StyleGAN) contains various interpretable semantics of real-world images, which provides a wonderful tool to perform a diverse set of semantic image editing tasks. Therefore, understanding and exploring the semantic directions encoded in the pre-trained GAN latent space have been the subject of numerous studies. Some researches [12, 41, 51] use complete supervision in the form of semantic labels. Such methods require either a well-trained attribute classifier or the images annotated with editing attributes. Therefore, this condition prevents the applications of supervised methods to a limited of known attributes. As a result, other studies [16, 40, 46, 47] propose unsupervised approaches to accomplish the same aim without the need for manual annotations. These types of methods allow exploring various fancy editing directions that we did not know before. Besides, some researches [36, 53] also use the idea of contrastive learning to analyze the GAN latent space.

GAN Inversion. To apply such latent manipulations to the real-world image, we first need to locate the latent code represented for that image in the pre-trained GAN latent space. This process is known as *GAN inversion* [58]. Existing GAN inversion methods can be categorized into two general groups: optimization-based and encoder-based approaches. Optimization-based methods [1, 10, 21, 25, 32] directly optimize the latent vector by minimizing the reconstruction error for each given image. These methods usually provide high-quality reconstructions but require too much time to perform, and it is tough to apply on real-time applications. Encoder-based works [35, 37, 44, 49, 58] employ an encoder to learn a direct mapping from a given picture to its corresponding latent vector, which allows for fast inference. However, the reconstruction quality of encoder-based techniques is usually worse than that of optimization-based approaches. Therefore, some hybrid works [5, 6, 14, 57] have been proposed, which combine both above general approaches to partly balance the trade-off of reconstruction quality and inference time. They first use an encoder to encode the image to the initial latent code and then use this latent as an initial point for the later optimization process. Hybrid methods can be considered as the subset of the two-stage methods and they use both optimization and encoder techniques in the method. In addition, there are other two-stage approaches but are not hybrid ones. For example, PIE [43] uses optimization methods on both phases, while PTI [38] combines optimization process with the generator fine-tuning technique. In comparison, our work is also a two-stage method but differs from the above ones in that our approach is only based purely on the encoder-based manner in both phases.

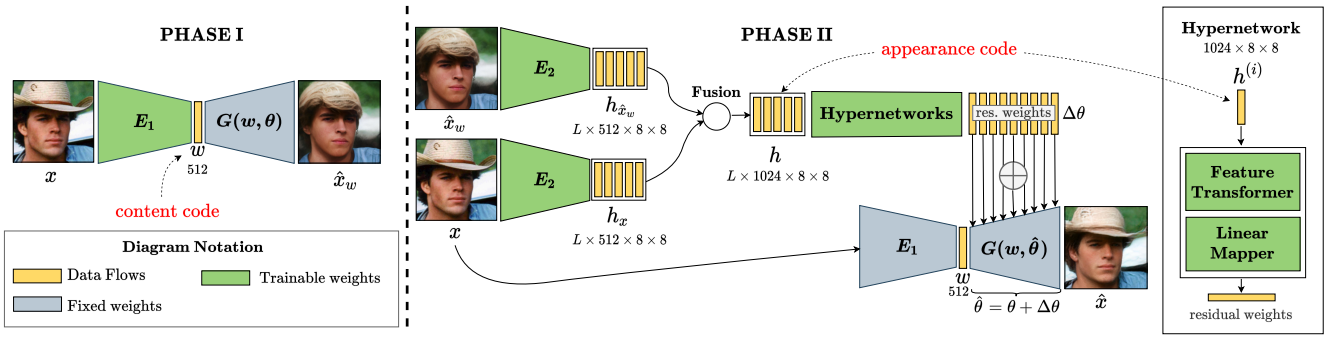

Figure 2. Our method contains **two** sequential phases: (1) We first train an encoder E_1 to encode the input image x to a *content* code w in $\mathcal{W}$-space; w represents the main semantics of the image, therefore used in editing later. The output image of this phase is $\hat{x}_w$. (2) We further regress the *residual weights* to update the generator to faithfully reconstruct the input details. First, we use another encoder E_2 and a fusion operator to extract the *appearance* code h from the input image x and the initial image $\hat{x}_w$, where L is the number of style layers of StyleGAN. Then, we employ a series of hypernetworks to embed the appearance code h to the generator G by predicting the residual weights $\Delta\theta$. The final reconstructed image $\hat{x}$ is generated by G with updated weights $\hat{\theta} = \theta + \Delta\theta$ and w content code from Phase I.

Hypernetwork. Hypernetwork is the auxiliary neural network that produces the weights for other network (often called as *primary network*). They were first proposed by Ha et al. [15] and have been used in a wide range of applications from semantic segmentation [34], 3D scene representation [29, 42], neural architecture search (NAS) [54] to continual learning [45]. In this study, we design the hypernetworks to improve the quality of our GAN inversion method. Specifically, we leverage hypernetworks to update the weights of the pre-trained GAN generator instead of fine-tuning procedure, which took a lot of processing time. Interestingly, a concurrent work named HyperStyle [4] also leverages the hypernetworks to solve the StyleGAN inversion task at the same time as ours.

3. Method

Our method is an end-to-end encoder-based GAN inversion approach with two consecutive phases. The first phase is a standard encoder that first maps input image to the GAN latent space (Section 3.1). The second phase then refines the generator so that it adapts to the reconstructed latent code to preserve original image details while allowing editability (Section 3.2). Figure 2 provides an overview of our approach. Let us now describe the details of each phase below.

3.1. Phase I: From image to content code

In StyleGAN, $\mathcal{W}$ space is empirically proven to have excellent editing ability compared to the popular $\mathcal{W}^+$ space for inversion [44, 59]. Therefore, we first choose $\mathcal{W}$ space to locate our latent code. Specifically, given the input image x, we train an encoder E_1 to regress x to the corresponding latent code $w \in \mathcal{W}$:

$$w = E_1(x), \tag{1}$$

where w has the size of $\mathbb{R}^{512}$. This w code has a role as the *content* code encoding the main semantics in the image. For image editing task later, one could traversal this w code on the interpretable semantic directions to manipulate the image. The reconstructed image $\hat{x}_w$ can be computed by passing the latent code w to the generator:

$$\hat{x}_w = G(w, \theta), \tag{2}$$

where G is the well-trained StyleGAN generator with convolutional weights θ.

A typical problem of the aforementioned inversion is that the image $\hat{x}_w$ is perceptually different from the input x, making the editing results less convincing. This phenomenon is caused by the relatively low dimensionality of the latent code w, which makes it challenging to represent all features from the input image. To address this weakness, we propose a novel phase II focusing on improving further the reconstruction quality by refining generator G with new weights.

3.2. Phase II: Generator refinement via hypernetworks from appearance code.

Our goal in this phase is to recover the missing information of input x and thus reduce the discrepancy between x and $\hat{x}_w$. Other two-stage methods use either per-image optimization process [43] or per-image fine-tuning G [38] to improve reconstruction quality further. However, the drawback of such these approaches is the slow inference time. We instead opt for a different approach that aims to create a single-forward pass network to learn to refine the generator G by inspecting the current input x and the reconstruction $\hat{x}_w$. Using single-forward pass network can guarantee the very fast running time. This network first encodes x and $\hat{x}_w$ into intermediate features and fuses them into a common feature tensor that can be used by a set of hypernetworks to output a set of *residual weights*. We then use the residual

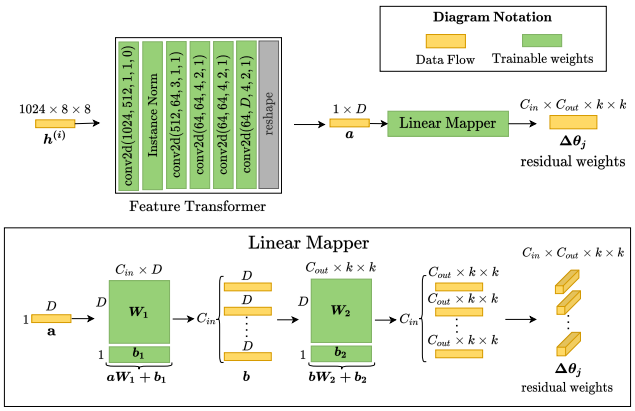

Figure 3. **Hypernetwork design** for predicting the residual weights $\Delta\theta_j$ of the convolutional layer j in the StyleGAN generator. Here we assume layer j receives the style vector indexed by i. Diagram notation: conv2d(in_channels, out_channels, kernel_size, stride, padding). For simplicity, we omit in the figure the ReLU [2] activation layer after each conv2d layer.

weights to update the corresponding convolutional layers in the generator G to yield a new set of weights $\hat{\theta}$. It is expected that the refined reconstruction from $G(\cdot, \hat{\theta})$ should be closer to the input x than the current reconstruction $\hat{x}_w$.

Particularly, we first use a shared encoder E_2 to transform both image x and $\hat{x}_w$ into the intermediate features:

$$h_{\hat{x}_w} = E_2(\hat{x}_w), \qquad h_x = E_2(x), \tag{3}$$

where $h_{\hat{x}_w}$, $h_x \in \mathbb{R}^{L\times 512\times 8\times 8}$, and L is the number of style layers from the pretrained StyleGAN generator. Recall that in StyleGAN generator network, each convolutional layer receives an input that corresponds to one of L style vectors. Therefore, we choose to regress to L feature components rather than a common one motivated by the StyleGAN generator's design. Then we use a fusion operator to combine these two features forming an appearance code h, with $h \in \mathbb{R}^{L\times 1024\times 8\times 8}$.

To embed the appearance code h to the reconstructed image, we opt to use h to update the weights of the pretrained StyleGAN generator G. Motivated by the hypernetwork's idea [15], which use an small extra neural network to predict the weights for the primary network, we leverage a series of hypernetworks to predict the residual weights, which then are added to refine the weights of G. Here we only consider predicting the weights for the main convolutional layers of G, skipping biases and other layers.

Assume that the pretrained generator G has N convolutional layers with weights $\theta = (\theta_1, \theta_2, ..., \theta_N)$. In StyleGAN architecture, each convolution layer $j \in \{1..N\}$ receives a corresponding style vector indexed by $i \in \{1..L\}$ as input where L is the total number of style vectors. We therefore propose to use a small hypernetwork H_j to predict the residual weights $\Delta\theta_j$ of each convolutional layer j, which results in a total of N hypernetworks. The residual weights $\Delta\theta_j$ for convolutional layer j is computed as

$$\Delta\theta_j = H_j(h^{(i)}). \tag{4}$$

The details of the design of hypernetwork is illustrated in Figure 3. Specifically, each hypernetwork contains two main modules, which are feature transformer and linear mapper. The feature transformer is a small convolutional neural network to transform the appearance code $h^{(i)}$ to the hidden features. The linear mapper is used for the final mapping from the hidden features to the convolutional weights. We adopt the technique proposed by Ha et al. [15] that uses two small matrices instead of a single large matrix to the weight mapping to keep the number of parameters in the hypernetwork manageable.

Finally, the updated generator has the convolutional weights as $\hat{\theta} = (\theta_1 + \Delta\theta_1, \theta_2 + \Delta\theta_2, ..., \theta_N + \Delta\theta_N)$. The final reconstructed image $\hat{x}$ could be obtained from the updated generator and the content code w from Phase I:

$$\hat{x} = G(w, \hat{\theta}). \tag{5}$$

3.3. Loss functions

We train our network in the two phases independently. In Phase I, we employ a set of loss functions to ensure faithful reconstruction. Given x as the input image, $\hat{x}$ is the reconstructed image, we define our reconstruction loss $\mathcal{L}_{rec}$ as a weighted sum of

$$\mathcal{L}_{rec} = \lambda_{pixel}\mathcal{L}_2 + \lambda_{perc}\mathcal{L}_{LPIPS} + \lambda_{id}\mathcal{L}_{ID}, \tag{6}$$

where λ_{pixel}, λ_{perc}, λ_{id} are the hyper-parameters. Each loss is defined as follows:

- $\mathcal{L}_2 = ||x - \hat{x}||_2$ measures the pixel-wise similarity,
- $\mathcal{L}_{LPIPS} = ||\mathcal{F}_{LP}(x) - \mathcal{F}_{LP}(\hat{x})||_2$ is the perceptual loss [56], where $\mathcal{F}_{LP}$ indicates a perceptual feature extractor [56]. We use the pre-trained AlexNet [28] version, similar to previous GAN inversion works.
- $\mathcal{L}_{ID} = 1 - \langle \mathcal{F}_{ID}(x), \mathcal{F}_{ID}(\hat{x}) \rangle$, where $\mathcal{F}_{ID}$ is a class-specific feature extractor network, which is a pre-trained ArcFace [11] for human facial domain or a ResNet-50 [17] pre-trained with MOCOv2 [9] for Churches domain, $\langle . \rangle$ denotes cosine similarity between two feature embedding vectors.

In Phase II, we further use the non-saturating GAN loss [13] in addition to the reconstruction loss in Phase I. This loss helps to ensure the realism of generated images

since we modify the weights of the generator in this phase:

$$\mathcal{L}_{enc} = \mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv} \tag{7}$$

where

$$\mathcal{L}_{adv} = -\mathbb{E}_{\hat{x}\sim p_{\hat{X}}}[\log(D(\hat{x}))], \tag{8}$$

D is initialized with the weights from the well-trained StyleGAN discriminator, λ_{adv} is the hyperparameter balancing two losses. Discriminator D is trained along with our network in a adversarial manner. We further impose $\mathcal{R}_1$ regularization [33] to D loss. The final loss for D is:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p_X}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_{\hat{X}}}[\log(1 - D(\hat{x}))] + \frac{\gamma}{2}\mathbb{E}_{x\sim p_X}[||\nabla_x D(x)||_2^2]. \tag{9}$$

4. Experiments

4.1. Experimental Settings

Datasets. We conduct the experiments on two challenging domains, which are *human faces* and *churches*. Those are selected because reconstructing human faces is a popular task in GAN inversion while reconstructing churches is a common test for generating outdoor scene images. For human facial domain, we employ the FFHQ [23] dataset as our training set, and the official test set of CelebA-HQ [22, 31] as our test set. For churches domain, we use the LSUN Church [52] dataset. We adopt the official train/test split of LSUN Church for our training and testing sets, respectively.

Baselines. We compare our method with various GAN inversion approaches including optimization-based, encoder-based, and two-stage methods. For optimization-based works, we compare our method with the inversion technique proposed by [1], denoted as SG2 $\mathcal{W}^+$. For encoder-based approaches, we choose pSp [37], e4e [44], and ReStyle [3] (apply over e4e backbone) to compare with our results. For other two-stage methods, we compare with PTI [38]. We use the official pre-trained weights and configurations released from the authors to perform our evaluation experiments.

Implementation Details. In our experiments, the pre-trained StyleGAN generators and discriminators being used are obtained directly from StyleGAN2 [25] repository. To implement the encoders E_1 and E_2, we adopt the backbone design of [37, 44]. For E_2, we modify the style block of original backbone to output $512\times8\times8$ tensors instead of 512 vectors. In Phase I, we follow the previous encoder-based methods [3, 37, 44] and use the Ranger optimizer, which combines the Lookahead [55] and the Rectified Adam [30] optimizer, for training. In Phase II, we use Adam [27] with standard settings, which we found to make the training of the hypernetworks converges faster. In both phases, we set the learning rate to a constant of 0.0001. For hyperparameters in the loss functions, we set $\lambda_{pixel} = 1.0$ and $\lambda_{perc} = 0.8$ for both domains. For other hyperparameters, we use $\lambda_{id} = 0.1$, $\lambda_{adv} = 0.005$, $\gamma = 10$ for the human facial domain, and $\lambda_{id} = 0.5$, $\lambda_{adv} = 0.15$, $\gamma = 100$ for the churches domain. Empirically, we found that applying adversarial loss and $\mathcal{R}_1$ regularization later in the training process leads to better stability than using them from the beginning. Specifically, we first train the model with batch size 8 for the first $200,000$ iterations. Then, we add the adversarial loss and $\mathcal{R}_1$ regularization to the training process and continue the training with batch size 4 until convergence. All evaluation experiments are conducted using a single NVIDIA Tesla V100 GPU.

Figure 4. **Qualitative reconstruction comparison** of our method with the current state-of-the-art StyleGAN inversion approaches. More examples are in supplementary. Best viewed in zoom.

4.2. Reconstruction Results

Quantitative Results. We use a diverse set of metrics to measure the reconstruction quality of our method compared with existing approaches. Specifically, we use the pixel-wise L2, perceptual LPIPS [56], MS-SSIM [48], and PSNR metrics. We also evaluate the realisticity of reconstructed images by using KID [8] and FID [18] metrics. For the human facial domain, we measure the ability to preserve the subject identity of each inversion method by computing the identity similarity between the source images and the reconstructed ones using an off-the-shelf face recognition model CurricularFace [19]. Besides, we also consider the *quality-time trade-off* raised by [3] in our evaluation pipeline by reporting the inference time of each approach.

The results can be found in Table 1. Our method significantly outperforms other encoder-based methods for reconstruction quality on both domains. Compared to such methods using optimization (and fine-tuning) technique, we have comparable performance. On the human facial domain, we win against SG2 $\mathcal{W}+$ on L2, LPIPS, and PSNR. On the churches domain, we significantly outmatchs other method except PTI on all metrics. PTI still surfers from distortion-

Domain	Method	L2 (↓)	LPIPS (↓)	ID (↑)	FID (↓)	KID$^{(\times 10^3)}$ (↓)	PSNR (↑)	MS-SSIM (↑)	Time (s) (↓)
Human Faces	SG2 $\mathcal{W}^+$ [1]	**0.0346**	**0.1250**	0.6669	14.39	5.41	**20.8771**	0.7181	270
	PTI [38]	0.0084	0.0851	0.8387	13.83	5.06	25.3447	0.7799	104
	pSp [37]	0.0351	0.1613	0.5596	24.20	12.13	20.3277	0.6496	0.0746
	e4e [44]	0.0479	0.1992	0.4963	27.57	13.90	19.0703	0.6231	0.0797
	ReStyle [3]	0.0436	0.1907	0.5036	24.16	10.80	19.4076	0.6309	0.1932
	Ours	**0.0243**	**0.1054**	**0.6013**	**15.88**	**5.63**	**21.4417**	**0.6725**	0.0920
Churches	SG2 $\mathcal{W}^+$ [1]	**0.1668**	**0.3250**	/	**50.00**	**7.32**	**14.8427**	**0.4797**	181
	PTI [38]	0.0506	0.0971	/	**76.21**	**28.29**	19.0105	0.6968	69
	pSp [37]	0.1107	0.3590	/	57.42	14.35	15.7509	0.4070	0.0521
	e4e [44]	0.1414	0.4209	/	55.40	11.03	14.7071	0.3481	0.0534
	ReStyle [3]	0.1272	0.3775	/	52.16	9.04	15.1849	0.3878	0.1028
	Ours	**0.0910**	**0.2226**	/	**46.96**	**6.82**	**16.7152**	**0.5762**	0.0661

Table 1. **Quantitative reconstruction results with inference time** of our method compared to the state-of-the-art StyleGAN inversion approaches. The **best** and runner-up results within *encoder-based methods* are marked in bold and underline, respectively. Values in **blue** and **red** highlight the cases that we outperform PTI [38] and SG2 $\mathcal{W}^+$ [1], respectively.

perception trade-off, exposed by the worst FID and KID. Overall, while our method does not outperform optimization-based methods completely, it is $3000\times$ and $1100\times$ faster than SG2 $\mathcal{W}+$ and PTI, respectively. Further reducing the performance gap for encoder-based and optimization-based (with generator fine-tuning) methods is left for future work.

Qualitative Results We visualize the reconstructions in Figure 4. For faces, we found that our method is particularly robust at preserving details of the accessories, e.g., see the cowboy hat example, and the background, e.g., see the hand example. These curated results also highlight that optimization-based methods do not perform well in such cases (despite they are better overall, which is reflected in the quantitative results in Table 1). In churches domain, our method reconstructed images significantly better than other methods including both PTI and SG2 $\mathcal{W}+$ in terms of both distortion and perception. Since PTI suffers from the distortion-perception trade-off as mentioned above, so the pictures from PTI are not completely realistic. Additional qualitative results can be found in the supplementary.

4.3. Editing Results

Quantitative Results. We first perform a quantitative evaluation for editing ability. Following previous works [38, 59], we present an experiment to test the effect of editing operator with the same editing magnitude on the latent code inverted by different inversion methods. We opt for age and rotation for two editing directions in this experiment. Given the latent code w, we apply the editing operator to obtain the new latent code as $w_{\text{edit}} = w + \gamma * d$ where γ is the editing magnitude and d is the semantic direction learned by InterFaceGAN [41]. To quantitatively evaluate the editing ability, we measure the amount of age change for age edit and yaw angle change (in degree) for pose edit when applying the same γ on each baseline. We employ the DEX VGG [39] model for age regression, and a pre-trained FacePoseNet [20] for head pose estimation. The results are shown in Table 2. As can be seen, as we expected, the inversion done by our Phase I $\mathcal{W}$ encoder achieves the most significant effects since it works on highly editable $\mathcal{W}$-space latent code. After Phase 2 refinement, our method still preserves the capability of editing, outperforming all previous methods despite not being as good as the results in Phase 1.

Dir.	γ	e4e	ReStyle	SG2 $\mathcal{W}^+$	PTI	**Ours** ($\mathcal{W}$ enc.)	**Ours** (full)
Age	-3	-9.51	-4.55	-5.45	-11.17	**-22.69**	-20.58
	-2	-5.6	-2.78	-3.79	-6.71	**-15.96**	-12.01
	-1	-2.44	-1.2	-1.92	-3.15	**-5.51**	-4.56
	1	2.72	1.24	2.17	2.68	**6.55**	4.78
	2	6.82	3.47	4.15	6.08	**13.04**	9.8
	3	11.64	5.61	6.79	10.3	**20.41**	14.25
Pose	-3	7.68	5.14	5.79	8.50	**13.63**	11.56
	-2	5.21	3.57	3.99	5.96	**9.01**	7.73
	-1	2.75	1.85	2.00	3.15	**4.64**	4.01
	1	-2.75	-1.77	-1.90	-2.95	**-4.99**	-4.17
	2	-5.36	-3.50	-3.83	-5.89	**-9.83**	-8.39
	3	-7.96	-5.14	-5.70	-8.88	**-15.06**	-12.93

Table 2. **Quantitative evaluation of editability.** We measure the amount of age change for age edit and yaw angle change (in degree) for pose edit when applying the same editing magnitude γ on each method. The **best** and runner-up values are marked in bold and underline, respectively.

Qualitative Results. We show the qualitative results for editing in Figure 5. As our reconstruction is generally more robust than other encoder-based methods (e4e, ReStyle), it also allows better editing results. As can be seen, our work can perform reasonable edits while still preserving faithfully non-editing attributes, e.g., background. In comparison to SG2 $\mathcal{W}^+$, our method produces significant editing effects

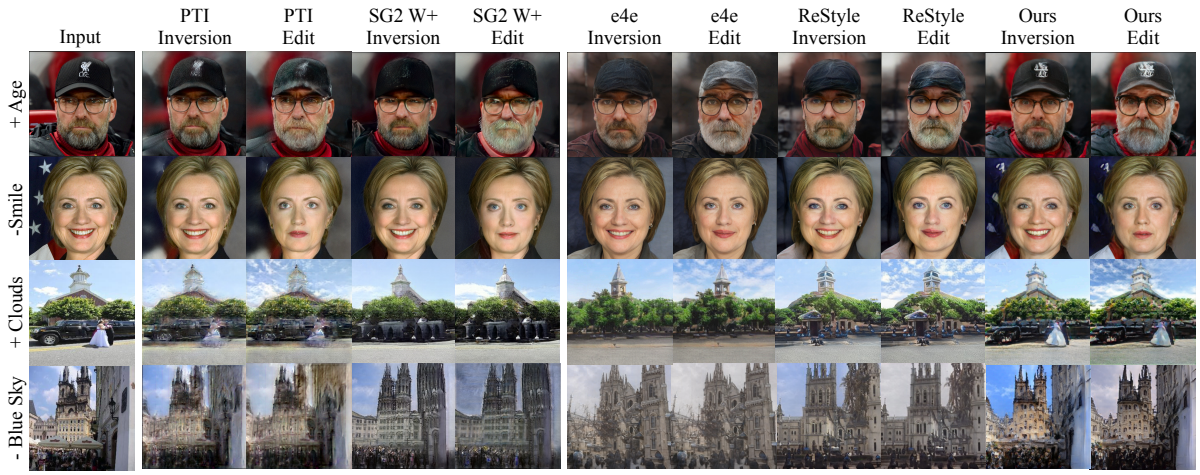

Figure 5. **Qualitative editing comparison** of our method with existing StyleGAN inversion works. Editing directions on human facial domain are obtained from InterFaceGAN [41], while those for church domain are from GANSpace [16]. More visual examples can be found in supplementary material.

with fewer artifacts, e.g., beard. The reason is that our work uses the $\mathcal{W}$ space, which has excellent editing ability, while SG2 $\mathcal{W}^+$ uses the extension $\mathcal{W}^+$ space, which suffers from a lower editability. Since PTI uses latent code on $\mathcal{W}$ space to edit, so as same as our method, on the human facial domain, they can do editing operations quite well. However, in the church domain, as PTI suffers from the distortion-perception trade-off, the reconstructed images are not realistic, leading to later editing of those images is not impressive.

4.4. User survey

We further conduct a user survey to evaluate the reconstruction and editing ability of our method through human perceptual assessment. In this survey, we only compare our method with other state-of-the-art encoder-based inversion ones including e4e [44], and ReStyle [3]. We skip comparing with pSp [37] since Tov et al. [44] pointed out that the latent code inverted by pSp is not good for editing, and they proposed e4e to fix this weakness. The setup of this user survey can be found in the supplementary. Figure 6 demonstrated our human evaluation results. As can be seen, our method outperforms e4e and ReStyle by a large margin on both reconstruction and editing quality. Notably, our method gets favored in 71.8% of reconstruction tests, 2.5 times e4e and ReStyle combined.

4.5. Ablation Study

Is Phase II required? Figure 7 and the row (2) in Table 3 demonstrate the importance of our phase II in improving the reconstruction quality. As can be seen, without this phase, the reconstruction quality is significantly dropped, illustrated by all quantitative metrics. In Figure 7, we can see that the reconstructed images lose various details such as hat, shadowing, background, makeup, and more. In contrast,

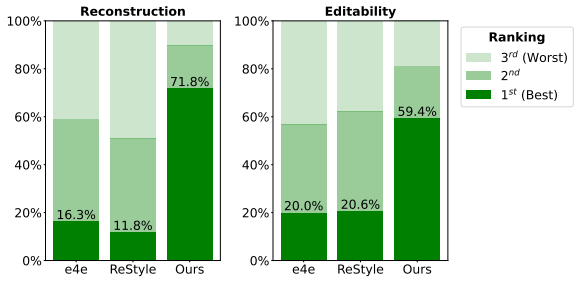

Figure 6. **User study results.** We reported the percentage of times testers rank the method at 1^{st} (best), 2^{nd}, and 3^{rd} (worst) based on two criteria, which are reconstruction and editing quality. As can be seen, our method outperforms e4e and ReStyle with a large gap.

No.	Method	L2 (↓)	LPIPS (↓)	ID (↑)	PSNR (↑)
(1)	Ours (full)	**0.0243**	**0.1054**	**0.6013**	**21.4417**
(2)	(1) w/o Phase II	0.0539	0.2145	0.3983	18.6522
(3)	(1) w/o $\hat{x}_w$ feat.	0.0313	0.1264	0.5412	20.4963
(4)	(1) w/o layer-wise	0.0287	0.1173	0.5768	20.7693
(5)	$D = 32$	0.0378	0.1492	0.5251	19.8097
(6)	$D = 64$	0.0308	0.1285	0.5571	20.5361
(7)	$D = 128$	0.0253	0.1103	0.5939	21.2115
(8)	$D = 256$	**0.0243**	**0.1054**	**0.6013**	**21.4417**

Table 3. **Ablation study** on human faces.

with the refinement from phase II, our method can recover the input image faithfully.

Should we fuse the appearance codes? We test the effectiveness of our design for extracting and using the appearance code in Phase II. Firstly, we compare our full version, which uses the appearance code h in the layer-wise design ($h \in \mathbb{R}^{L \times 1024 \times 8 \times 8}$), with the code-sharing version ($h \in \mathbb{R}^{1024 \times 8 \times 8}$). Secondly, we test the effectiveness of using both the features from the input image x and the initial image $\hat{x}_w$ in computing appearance code h. We compare

Figure 7. Visual examples demonstrating the effectiveness of our proposed second phase. Best viewed in zoom.

our method with the version using only the features from the input image x. The results of these experiments are shown in rows (3) and (4) in Table 3, respectively. As can be seen, our full version significantly surpasses the simplified ones.

Different choices of hyperparameter D in hypernetwork. We empirically find that the hidden dimension D in our hypernetworks's design has a remarkable effect on the model performance. Rows (5), (6), (7), (8) in Table 3 investigates different choices of D, proving that $D = 256$ is best option.

Which layers in the generator are updated? We also visualize the residual weights predicted by the hypernetworks to analyze which StyleGAN layers change the most when applying Phase II. Specifically, we choose to visualize the mean absolute amount of weight change between the parameters of each layer and compare this value with other layers. This statistic is calculated from the residual weights predicted by our method on $2,824$ images in the CelebA-HQ test set. Figure 8 visualizes this analysis. From this figure, we draw two following insights. First, the convolution layers in main generator blocks contribute more significantly than the convolution layers in *torgb* blocks. Second, the weight change for the last resolution (1024×1024) is significantly larger than the ones for other resolutions. As we expected, on human faces, since the first phase reconstructed images quite well overall, therefore in phase II, they should mainly focus on refining the fine-grained details. This argument is supported well by insight 2.

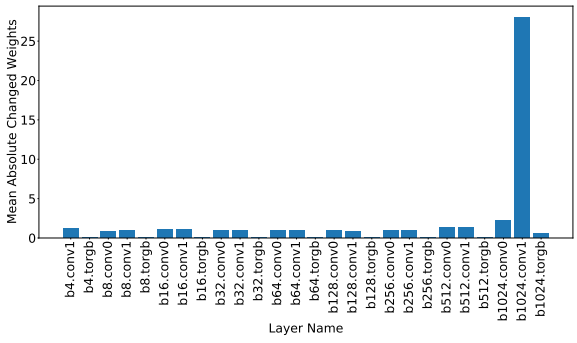

Figure 8. Visualizing the statistic of residual weights.

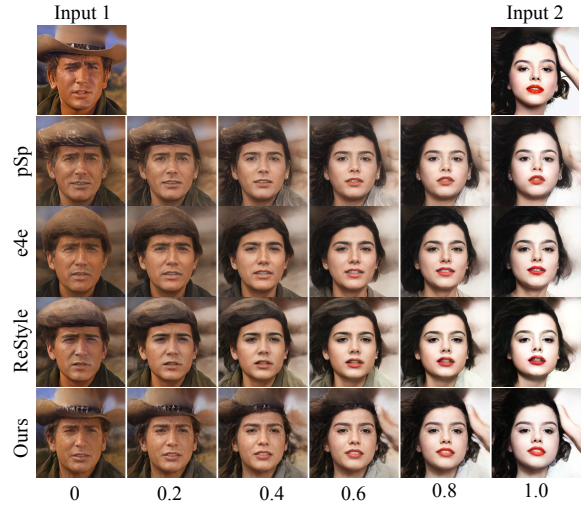

Figure 9. Real-world image interpolation results.

4.6. Application: Real-world Image Interpolation

In this section, we present a new approach to interpolate between two real-world images. Given two input images, a common-used process is first to find the corresponding latent codes via GAN inversion, compute the linear interpolated latent codes, and pass it through the GAN model to get the interpolated images. Motivated by our two-phase inversion mechanism, we propose a new interpolation approach by *interpolating both latent codes and generator weights* instead of using only latent codes as the common existing pipeline. Figure 9 shows some qualitative results comparing our method with the ones interpolating only latent codes. As can be seen, our method not only reconstructed input images with correctly fine-grained details (e.g., hat, background, and hand) but also provided the smooth interpolated images.

5. Conclusion

This paper presented a fully encoder-based method for solving the StyleGAN inversion problem using a two-phase approach and the hypernetworks to refine the generator's weights. As a result, our method has high fidelity reconstruction, excellent editability while running almost real-time.

Despite the promising results, our work is not without limitations. As an encoder-based method, it generates high-quality images but appears not completely better than approaches based on per-image optimization such as SG2 $\mathcal{W}^+$, PTI in terms of metrics. However, we also include some cases in Figure 4 to show that both SG2 $\mathcal{W}^+$ and PTI can yield inaccurate reconstruction. We advocate further development of encoder-based methods to reduce such a performance gap since encoder-based methods (including ours) run faster at inference, which makes them more suitable for interactive and video applications. Specially, our work can be further improved by applying iterative refinement [3], multi-layer identity loss [49], or extending to StyleGAN3 [26].

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *ICCV*, 2019. 1, 2, 5, 6

[2] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018. 4

[3] Yuval Alaluf, Or Patashnik, and Daniel Cohen-Or. Restyle: A residual-based stylegan encoder via iterative refinement. In *ICCV*, 2021. 1, 5, 6, 7, 8

[4] Yuval Alaluf, Omer Tov, Ron Mokady, Rinon Gal, and Amit H. Bermano. Hyperstyle: Stylegan inversion with hypernetworks for real image editing, 2021. 3

[5] David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. Semantic photo manipulation with a generative image prior. *TOG*, 2019. 2

[6] David Bau, Jun-Yan Zhu, Jonas Wulff, William Peebles, Hendrik Strobelt, Bolei Zhou, and Antonio Torralba. Seeing what a gan cannot generate. In *ICCV*, 2019. 2

[7] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 2020. 1

[8] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018. 5

[9] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 4

[10] Antonia Creswell and Anil Anthony Bharath. Inverting the generator of a generative adversarial network. *Transactions on Neural Networks and Learning Systems*, 2018. 1, 2

[11] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, 2019. 4

[12] Lore Goetschalckx, Alex Andonian, Aude Oliva, and Phillip Isola. Ganalyze: Toward visual definitions of cognitive image properties. In *ICCV*, 2019. 2

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NeurIPS*, 2014. 1, 4

[14] Shanyan Guan, Ying Tai, Bingbing Ni, Feida Zhu, Feiyue Huang, and Xiaokang Yang. Collaborative learning for faster stylegan embedding. *arXiv preprint arXiv:2007.01758*, 2020. 2

[15] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 3, 4

[16] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In *NeurIPS*, 2020. 1, 2, 7

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4

[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *NeurIPS*, 2017. 5

[19] Yuge Huang, Yuhan Wang, Ying Tai, Xiaoming Liu, Pengcheng Shen, Shaoxin Li, Jilin Li, and Feiyue Huang. Curricularface: adaptive curriculum learning loss for deep face recognition. In *CVPR*, 2020. 5

[20] Feng ju Chang, Anh Tran, Tal Hassner, Iacopo Masi, Ram Nevatia, and Gérard Medioni. FacePoseNet: Making a case for landmark-free face alignment. In *7th IEEE International Workshop on Analysis and Modeling of Faces and Gestures, ICCV Workshops*, 2017. 6

[21] Kyoungkook Kang, Seongtae Kim, and Sunghyun Cho. Gan inversion for out-of-range images with geometric transformations. In *ICCV*, 2021. 1, 2

[22] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018. 1, 5

[23] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 1, 5

[24] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *NeurIPS*, 2020.

[25] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *CVPR*, 2020. 1, 2, 5

[26] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *NeurIPS*, 2021. 1, 8

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 2012. 4

[29] Gidi Littwin and Lior Wolf. Deep meta functionals for shape representation. In *ICCV*, 2019. 3

[30] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *ICLR*, 2019. 5

[31] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015. 5

[32] Fangchang Ma, Ulas Ayaz, and Sertac Karaman. Invertibility of convolutional generative networks from partial measurements. 2019. 1, 2

[33] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *ICML*, 2018. 5

[34] Yuval Nirkin, Lior Wolf, and Tal Hassner. Hyperseg: Patch-wise hypernetwork for real-time semantic segmentation. In *CVPR*, 2021. 3

[35] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *CVPR*, 2020. 1, 2

[36] Xuanchi Ren, Tao Yang, Yuwang Wang, and Wenjun Zeng. Learning disentangled representation by exploiting pretrained generative models: A contrastive learning view. In *ICLR*, 2022. 2

[37] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *CVPR*, 2021. 1, 2, 5, 6, 7

[38] Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *arXiv preprint arXiv:2106.05744*, 2021. 1, 2, 3, 5, 6

[39] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks. *IJCV*, 2018. 6

[40] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. In *CVPR*, 2021. 1, 2

[41] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *CVPR*, 2020. 1, 2, 6, 7

[42] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *NeurIPS*, 2020. 3

[43] Ayush Tewari, Mohamed Elgharib, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhöfer, and Christian Theobalt. Pie: Portrait image embedding for semantic control. *TOG*, 2020. 2, 3

[44] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation. *TOG*, 2021. 1, 2, 3, 5, 6, 7

[45] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. Continual learning with hypernetworks. In *ICLR*, 2020. 3

[46] Andrey Voynov and Artem Babenko. Unsupervised discovery of interpretable directions in the gan latent space. In *ICML*, 2020. 2

[47] Binxu Wang and Carlos R Ponce. A geometric analysis of deep generative image models and its applications. In *ICLR*, 2020. 2

[48] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers*, 2003. 5

[49] Tianyi Wei, Dongdong Chen, Wenbo Zhou, Jing Liao, Weiming Zhang, Lu Yuan, Gang Hua, and Nenghai Yu. A simple baseline for stylegan inversion. *arXiv preprint arXiv:2104.07661*, 2021. 2, 8

[50] Ceyuan Yang, Yujun Shen, and Bolei Zhou. Semantic hierarchy emerges in deep generative representations for scene synthesis. *IJCV*, 2020. 1

[51] Huiting Yang, Liangyu Chai, Qiang Wen, Shuang Zhao, Zixun Sun, and Shengfeng He. Discovering interpretable latent space directions of gans beyond binary attributes. In *CVPR*, 2021. 2

[52] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 5

[53] Oğuz Kaan Yüksel, Enis Simsar, Ezgi Gülperi Er, and Pinar Yanardag. Latentclr: A contrastive learning approach for unsupervised discovery of interpretable directions. In *ICCV*, 2021. 2

[54] Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. In *ICLR*, 2019. 3

[55] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. In *NeurIPS*, 2019. 5

[56] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 4, 5

[57] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *ECCV*, 2020. 2

[58] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016. 1, 2

[59] Peihao Zhu, Rameen Abdal, Yipeng Qin, John Femiani, and Peter Wonka. Improved stylegan embedding: Where are the good latents? *arXiv preprint arXiv:2012.09036*, 2020. 2, 3, 6