HyperStyle: StyleGAN Inversion with HyperNetworks for Real Image Editing

Yuval Alaluf* Omer Tov* Ron Mokady Rinon Gal Amit Bermano

Blavatnik School of Computer Science, Tel Aviv University

Abstract

The inversion of real images into StyleGAN's latent space is a well-studied problem. Nevertheless, applying existing approaches to real-world scenarios remains an open challenge, due to an inherent trade-off between reconstruction and editability: latent space regions which can accurately represent real images typically suffer from degraded semantic control. Recent work proposes to mitigate this trade-off by fine-tuning the generator to add the target image to well-behaved, editable regions of the latent space. While promising, this fine-tuning scheme is impractical for prevalent use as it requires a lengthy training phase for each new image. In this work, we introduce this approach into the realm of encoder-based inversion. We propose HyperStyle, a hypernetwork that learns to modulate StyleGAN's weights to faithfully express a given image in editable regions of the latent space. A naive modulation approach would require training a hypernetwork with over three billion parameters. Through careful network design, we reduce this to be in line with existing encoders. HyperStyle yields reconstructions comparable to those of optimization techniques with the near real-time inference capabilities of encoders. Lastly, we demonstrate HyperStyle's effectiveness on several applications beyond the inversion task, including the editing of out-of-domain images which were never seen during training. Code is available on our project page: https://yuval-alaluf.github.io/hyperstyle/.

1. Introduction

Generative Adversarial Networks (GANs) [20], and in particular StyleGAN [32–35] have become the gold standard for image synthesis. Thanks to their semantically rich latent representations, many works have utilized these models to facilitate diverse and expressive editing through latent space manipulations [4, 6, 9, 12, 24, 38, 44, 48, 56]. Yet, a significant challenge in adopting these approaches for real-world applications is the ability to edit *real* images. For editing a real photo, one must first find its corresponding latent representation via a process commonly referred to as GAN inversion [74]. While the inversion process is a well-studied problem, it remains an open challenge.

Recent works [2, 61, 73, 75] have demonstrated the existence of a *distortion-editability* trade-off: one may invert an image into well-behaved [75] regions of StyleGAN's latent space and attain good editability. However, these regions are typically less expressive, resulting in reconstructions that are less faithful to the original image. Recently, Roich *et al*. [54] showed that one may side-step this trade-off by considering a different approach to inversion. Rather than searching for a latent code that most accurately reconstructs the input image, they fine-tune the generator in order to insert a target identity into well-behaved regions of the latent space. In doing so, they demonstrate state-of-the-art reconstructions while retaining a high level of editability. Yet, this approach relies on a costly per-image optimization of the generator, requiring up to a minute per image.

A similar *time-accuracy* trade-off can be observed in classical inversion approaches. On one end of the spectrum, latent vector optimization approaches [1, 2, 14, 35, 40]

Figure 1. Given a desired input image, our hypernetworks *learn* to modulate a pre-trained StyleGAN network to achieve accurate image reconstructions in editable regions of the latent space. Doing so enables one to effectively apply techniques such as StyleCLIP [48] and InterFaceGAN [56] for editing real images.

*Denotes equal contribution

achieve impressive reconstructions, but are impractical at scale, requiring several minutes per image. On the other end, encoder-based approaches leverage rich datasets to learn a mapping from images to their latent representations. These approaches operate in a fraction of a second but are typically less faithful in their reconstructions.

In this work, we aim to bring the generator-tuning technique of Roich *et al*. [54] to the realm of interactive applications by adapting it to an encoder-based approach. We do so by introducing a hypernetwork [23] that *learns* to refine the generator weights with respect to a given input image. The hypernetwork is composed of a lightweight feature extractor (e.g., ResNet [25]) and a set of refinement blocks, one for each of StyleGAN's convolutional layers. Each refinement block is tasked with predicting offsets for the weights of the convolutional filters of its corresponding layer. A major challenge in designing such a network is the number of parameters comprising each convolutional block that must be refined. Naïvely predicting an offset for each parameter would require a hypernetwork with over three billion parameters. We explore several avenues for reducing this complexity: sharing offsets between parameters, sharing network weights between different hypernetwork layers, and an approach inspired by depthwise-convolutions [26]. Lastly, we observe that reconstructions can be further improved through an iterative refinement scheme [5] which gradually predicts the desired offsets over a small number of forward passes through the hypernetwork. By doing so, our approach, HyperStyle, essentially learns to "optimize" the generator in an efficient manner.

The relation between HyperStyle and existing generator-tuning approaches can be viewed as similar to the relation between encoders and optimization inversion schemes. Just as encoders find a *desired latent code* via a learned network, our hypernetwork efficiently finds a *desired generator* with no image-specific optimization.

We demonstrate that HyperStyle achieves a significant improvement over current encoders. Our reconstructions even rival those of optimization schemes, while being several orders of magnitude faster. We additionally show that HyperStyle preserves the appealing structure and semantics of the original latent space, allowing one to leverage off-the-shelf editing techniques on the resulting inversions, see Fig. 1. Finally, we show that HyperStyle generalizes well to out-of-domain images, such as paintings and animations, even when unobserved during the training of the hypernetwork itself. This hints that the hypernetwork does not only learn to correct specific flawed attributes, but rather learns to refine the generator in a more general sense.

2. Background and Related Work

Hypernetworks Introduced by Ha *et al*. [23], hypernetworks are neural networks tasked with predicting the weights of a primary network. By training a hypernetwork over a large data collection, the primary network's weights are adjusted with respect to specific inputs, yielding a more expressive model. Hypernetworks have been applied to a wide range of applications including semantic segmentation [45], 3D modeling [41, 58], neural architecture search [71], and continual learning [62], among others.

Latent Space Manipulation A widely explored application for generative models is their use for the editing of real images. Considerable effort has gone into leveraging StyleGAN [34, 35] for such tasks, owing to its highly-disentangled latent spaces. Many methods have been proposed for finding semantic latent directions using varying levels of supervision. These range from full-supervision in the form of semantic labels [3, 16, 19, 56] and facial priors [59,60] to unsupervised approaches [24,57,63,64]. Others have explored self-supervised approaches [**?**,**?**], the mixing of latent codes to produce local edits [11, 13, 29], and the use of contrastive language-image (CLIP) models [52] to achieve new editing capabilities [18, 48, 69]. Applying these methods to real images requires one to first perform an accurate inversion of the given image.

GAN Inversion GAN inversion [74] is the process of obtaining a latent code that can be passed to the generator to reconstruct a given image. Generally, inversion methods either directly optimize the latent vector to minimize the error for a given image [1, 2, 7, 14, 21, 40, 70, 74, 75], train an encoder over a large number of samples to learn a mapping from an image to its latent representation [5,22,22,30, 36,43,49,50,53,61,65], or use a hybrid approach combining both [73,74]. Among encoder-based methods, Alaluf *et al*. [5] iteratively refine the predicted latent code through a small number of forward passes through the network. Our work adopts this idea and applies it to the generator weight offsets predicted by the hypernetwork. Finally, in a concurrent work, Dinh *et al*. [17] also explore the use of hypernetworks for achieving higher fidelity inversions.

Distortion-Editability Typically, latent traversal and inversion methods concern themselves with one of two spaces: $\mathcal{W}$, obtained via StyleGAN's mapping network and $\mathcal{W}+$, where each layer of the generator is assigned a different latent code $w_i \in \mathcal{W}$. Images inverted into $\mathcal{W}$ show a high degree of editability: they can be modified through latent space traversal with minimal corruption. However, $\mathcal{W}$ offers poor expressiveness, limiting the range of images that can be faithfully reconstructed. Therefore, many prior works invert into the extended $\mathcal{W}+$ space, achieving reduced distortion at the cost of inferior editability. Tov *et al*. [61] suggest balancing the two by designing an encoder that predicts codes in $\mathcal{W}+$ residing close to $\mathcal{W}$. Others have explored similar ideas for optimization [75].

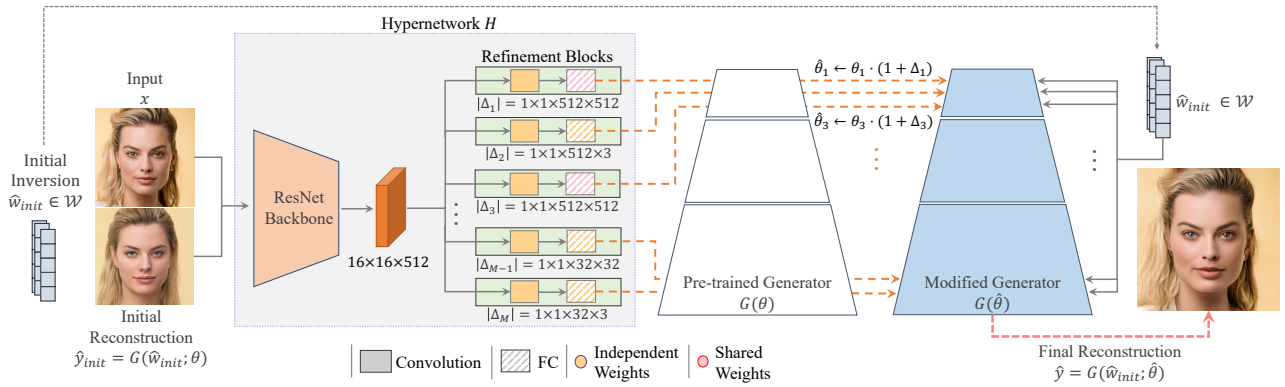

Figure 2. The HyperStyle scheme. Given an image x, we begin with an initial, approximate latent code $\hat{w}_{init} \in \mathcal{W}$ with a corresponding reconstruction $\hat{y}_{init} = G(\hat{w}_{init}; \theta)$ obtained using a pre-trained generator G with weights θ. Given inputs x and $\hat{y}_{init}$, our hypernetwork H predicts a set of offsets Δ_ℓ used to modulate G's weights at various input layers ℓ. This results in a modified generator G parameterized by new weights $\hat{\theta}$, shown in blue. To predict the desired offsets for the given image, we incorporate multiple Refinement Blocks, one for each generator layer we wish to modify. The final reconstruction $\hat{y} = G(\hat{w}_{init}; \hat{\theta})$ is then synthesized using the modified generator.

Generator Tuning To leverage the visual quality of a pre-trained generator, most works avoid altering the generator weights when performing the inversion. Nonetheless, some works have explored performing a per-image tuning of the generator to obtain more accurate inversions. Pan *et al*. [47] invert BigGAN [8] by randomly sampling noise vectors, selecting the one that best matches the real image, and optimizing it simultaneously with the generator weights in a progressive manner. Roich *et al*. [54] and Hussien *et al*. [28] invert images into a pre-trained GAN by first recovering a latent code which approximately reconstructs the target image and then fine-tuning the generator weights for improve image-specific details. Bau *et al*. [7] explored the use of a neural network to predict feature modulations to improve GAN inversion. However, the aforementioned works require a lengthy optimization for every input, typically requiring minutes per image. As such, these methods are often inapplicable to real-world scenarios at scale. In contrast, we train a hypernetwork over a large set of images, resulting in a *single* network used to refine the generator for any given image. Importantly, this is achieved in near real-time and is more suitable for interactive settings.

3. Method

3.1. Preliminaries

When solving the GAN inversion task, our goal is to identify a latent code that minimizes the reconstruction distortion with respect to a given target image x:

$$\hat{w} = \arg\min_{w} \mathcal{L}\left(x, G\left(w; \theta\right)\right), \tag{1}$$

where $G(w; \theta)$ is the image produced by a *pre-trained* generator G parameterized by weights θ, over the latent w. $\mathcal{L}$ is the loss objective, usually L_2 or LPIPS [72]. Solving Eq. (1) via optimization typically requires several minutes per image. To reduce inference times, an encoder E can be trained over a large set of images $\{x^i\}_{i=1}^N$ to minimize:

$$\sum_{i=1}^{N} \mathcal{L}\left(x^i, G\left(E\left(x^i\right); \theta\right)\right). \tag{2}$$

This results in a fast inference procedure $\hat{w} = E(x)$. A latent manipulation f can then be applied over the inverted code $\hat{w}$ to obtain an edited image $G(f(\hat{w}); \theta)$.

Recently, Roich *et al*. [54] propose injecting new identities into the well-behaved regions of StyleGAN's latent space. Given a target image, they use an optimization process to find an initial latent $\hat{w}_{init} \in \mathcal{W}$ leading to an approximate reconstruction. This is followed by a fine-tuning session where the generator weights are adjusted so that the same latent better reconstructs the specific image:

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(x, G(\hat{w}_{init}; \theta)), \tag{3}$$

where $\hat{\theta}$ represents the new generator weights. The final reconstruction is obtained by utilizing the initial inversion and altered weights: $\hat{y} = G(\hat{w}_{init}; \hat{\theta})$.

3.2. Overview

Our method HyperStyle aims to perform the identity-injection operation by efficiently providing modified weights for the generator, as illustrated in Fig. 2. We begin with an image x, a generator G parameterized by weights θ, and an initial inverted latent code $\hat{w}_{init} \in \mathcal{W}$. Using these weights and $\hat{w}_{init}$, we generate the initial reconstructed image $\hat{y}_{init} = G(\hat{w}_{init}; \theta)$. To obtain such a latent code we employ an off-the-shelf encoder [61].

Our goal is to predict a new set of weights $\hat{\theta}$ that minimizes the objective defined in Eq. (3). To this end, we present our hypernetwork H, tasked with predicting these weights. To assist the hypernetwork in inferring the desired modifications, we pass as input both the target image x and the initial, approximate image reconstruction $\hat{y}_{init}$.

The predicted weights are thus given by: $\hat{\theta} = H\left(\hat{y}_{init}, x\right)$. We train H over a large collection of images with the goal of minimizing the distortion of the reconstructions:

$$\sum_{i=1}^{N} \mathcal{L}\left(x^i, G\left(\hat{w}^i_{init}; H\left(\hat{y}^i_{init}, x^i\right)\right)\right). \tag{4}$$

Given the hypernetwork predictions, the final reconstruction can be obtained as $\hat{y} = G(\hat{w}_{init}; \hat{\theta})$.

Owing to the reconstruction-editability trade-off outlined in Sec. 2, the initial latent code should reside within the well-behaved (i.e., editable) regions of StyleGAN's latent space. To this end, we employ a pre-trained e4e encoder [61] into $\mathcal{W}$ that is kept fixed throughout the training of the hypernetwork. As shall be shown, by tuning around such a code, one can apply the same editing techniques as used with the original generator.

In practice, rather than directly predicting the new generator weights, our hypernetwork predicts a set of offsets with respect to the original weights. In addition, we follow ReStyle [5] and perform a small number of passes (e.g., 5) through the hypernetwork to gradually refine the predicted weight offsets, resulting in higher-fidelity inversions.

In a sense, one may view HyperStyle as *learning to optimize* the generator, but doing so in an efficient manner. Moreover, by learning to modify the generator, HyperStyle is given more freedom to determine how to best project an image into the generator, even when out of domain. This is in contrast to standard encoders which are restricted to encoding into existing latent spaces.

3.3. Designing the HyperNetwork

The StyleGAN generator contains approximately 30M parameters. On one hand, we wish our hypernetworks to be expressive, allowing us to control these parameters for enhancing the reconstruction. On the other hand, control over too many parameters would result in an inapplicable network requiring significant resources for training. Therefore, the design of the hypernetwork is challenging, requiring a delicate balance between expressive power and the number of trainable parameters involved.

We denote the weights of the ℓ-th convolutional layer of StyleGAN by $\theta_\ell = \{\theta^{i,j}_\ell\}^{C^{out}_\ell, C^{in}_\ell}_{i,j=0}$ where $\theta^{i,j}_\ell$ denotes the weights of the j-th channel in the i-th filter. Here, C^{out}_ℓ represents the total number of filters, each with C^{in}_ℓ channels. Let M be the total number of layers. The generator weights are then denoted as $\{\theta_\ell\}^M_{\ell=1}$. Our hypernetwork produces offsets Δ_ℓ for each modified layer ℓ. These offsets are then multiplied by the corresponding layer weights θ_ℓ and added to the original weights in a channel-wise fashion:

$$\hat{\theta}^{i,j}_\ell := \theta^{i,j}_\ell \cdot (1 + \Delta^{i,j}_\ell), \tag{5}$$

where $\Delta^{i,j}_\ell$ is the scalar applied to the j-th channel of the i-th filter. Learning an offset per channel reduces the number of hypernetwork parameters by 88% compared to predicting an offset for each generator parameter (see Tab. 1). Later experiments verify that this does not harm expressiveness.

HyperStyle Trainable Parameters

Delta-Per Channel	Shared Refinement	Number of Parameters
		3.07B
	✓	1.40B
✓		367M
✓	✓	332M
pSp [53] / e4e [61]: 267M		ReStyle [4]: 205M

Table 1. Our final hypernetwork configuration, consisting of an offset predicted per channel and Shared Refinement blocks reduces the number of parameters by 89% compared to a naïve network design. We compare this to the size of existing encoders.

To process the input images, we incorporate a ResNet34 [25] backbone that receives a 6-channel input $\left(x^i,\ y^i_{init}\right)$ and outputs a $16 \times 16 \times 512$ feature map. This shared backbone is then followed by a set of *Refinement Blocks*, each producing the modulation of a single generator layer. Consider layer ℓ with parameters θ_ℓ of size $k_\ell \times k_\ell \times C^{in}_\ell \times C^{out}_\ell$ where k_ℓ is the kernel size. The corresponding Refinement Block receives the feature map extracted by the backbone and outputs an offset of size $1 \times 1 \times C^{in}_\ell \times C^{out}_\ell$. The offset is then replicated to match the $k_\ell \times k_\ell$ kernel dimension of θ_ℓ. Finally, the new weights of layer ℓ are updated using Eq. (5). The Refinement Block is illustrated in Fig. 3.

To further reduce the number of trainable parameters, we introduce a *Shared Refinement Block*, inspired by the original hypernetwork [23]. These output heads consist of independent convolutional layers used to down-sample the input feature map. They are then followed by two fully-connected layers shared across multiple generator layers, as illustrated in Fig. 3. Here, the fully-connected weights are shared across the non-toRGB layers with dimension $3 \times 3 \times 512 \times 512$, i.e., the largest generator convolutional blocks. As demonstrated in Ha *et al.* [23] this allows for information sharing between the output heads, yielding improved reconstruction quality. Detailed layouts of the Refinement Blocks are given in the supplementary materials.

Combining the Shared Refinement Blocks and per-channel predictions, our final configuration contains 2.7B fewer parameters (~89%) than a naïve hypernetwork. We summarize the total number of parameters of different hypernetwork variants in Tab. 1. We refer the reader to Sec. 4.3 where we validate our design choices and explore additional avenues for reducing the number of parameters.

Which layers are refined? The choice of which layers to refine is of great importance. It allows us to reduce the output dimension while focusing the hypernetwork on the more meaningful generator weights. Since we invert one identity at a time, any changes to the affine transformation

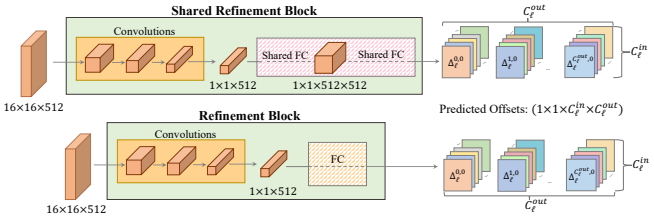

Figure 3. The Refinement Block (shown on bottom) consists of a series of down-sampling convolutions and a fully-connected layer resulting in an output of size $1 \times 1 \times C_\ell^{in} \times C_\ell^{out}$. The Shared Refinement Block (top) is introduced to further reduce the network parameters and encourage information sharing between layers.

layers can be reproduced by a respective re-scaling of the convolution weights. Moreover, we find that altering the toRGB layers harms the editing capabilities of the GAN. We hypothesize that modifying these layers mainly alters the pixel-wise texture and color [67], changes that do not translate well under global edits such as pose (see the supplementary materials for examples). Therefore, we restrict ourselves to modifying only the non-toRGB convolutions.

Lastly, we follow Karras *et al*. [35] and split the generator layers into three levels of detail — coarse, medium, fine — each controlling different aspects of the generated image. As the initial inversions tend to capture coarse details, we further restrict our hypernetwork to output offsets for the medium and fine generator layers.

3.4. Iterative Refinement

To further improve the inversion quality, we adopt the iterative refinement scheme suggested by Alaluf *et al*. [4]. This enables us to perform several passes through our hypernetwork for a single image inversion. Each added step allows the hypernetwork to gradually refine its predicted weight offsets, resulting in stronger expressive power and a more accurate inversion.

We perform T passes. For the first pass, we use the initial reconstruction $\hat{y}_0 = G(\hat{w}_{init}; \theta)$. For each refinement step $t \geq 1$, we predict a set of offsets $\Delta_t = H(\hat{y}_{t-1}, x)$ used to obtain the modified weights $\hat{\theta}_t$ and updated reconstruction $\hat{y}_t = G(\hat{w}_{init}; \hat{\theta}_t)$. The weights at step t are defined as the accumulated modulation across all previous steps:

$$\hat{\theta}_{\ell,t} := \theta \cdot (1 + \sum_{i=1}^{t} \Delta_{\ell,i}). \tag{6}$$

The number of refinement steps is set to $T = 5$ during training. Following Alaluf *et al*. [5] we compute the losses at each refinement step. Note, $\hat{w}_{init}$ remains fixed during the iterative process. The final inversion $\hat{y}$ is the reconstruction obtained at the last step.

3.5. Training Losses

Similar to encoder-based methods, our training is guided by an image-space reconstruction objective. We apply a weighted combination of the pixel-wise L_2 loss and LPIPS perceptual loss [72]. For the facial domain, we further apply an identity-based similarity loss [53] by employing a pre-trained facial recognition network [15] to preserve the facial identity. As suggested by Tov *et al*. [61], we apply a MoCo-based similarity loss for non-facial domains. The final loss objective is given by:

$$\mathcal{L}_2(x, \hat{y}) + \lambda_{\text{LPIPS}} \mathcal{L}_{\text{LPIPS}}(x, \hat{y}) + \lambda_{sim} \mathcal{L}_{sim}(x, \hat{y}). \tag{7}$$

Method	↑ ID	↑ MS-SSIM	↓ LPIPS	↓ L_2	↓ Time (s)
StyleGAN2 [35]	0.78	0.90	0.09	0.020	227.55
PTI [54]	0.85	0.92	0.09	0.015	55.715
IDInvert [73]	0.18	0.68	0.22	0.061	0.04
pSp [53]	0.56	0.76	0.17	0.034	0.106
e4e [61]	0.50	0.72	0.20	0.052	0.106
ReStyle$_{pSp}$ [5]	0.66	0.79	0.13	0.030	0.366
ReStyle$_{e4e}$ [5]	0.52	0.74	0.19	0.041	0.366
HyperStyle	0.76	0.84	0.09	0.019	1.234

Table 2. Quantitative reconstruction results on the human facial domain measured over the CelebA-HQ [31, 42] test set.

4. Experiments

Datasets and Baselines For the human facial domain we use FFHQ [34] for training and the CelebA-HQ test set [31, 42] for quantitative evaluations. On the cars domain, we use the Stanford Cars dataset [37]. Additional results on AFHQ Wild [10] are provided in the supplementary. We compare our results to the state-of-the-art encoders pSp [53], e4e [61], and ReStyle [5] applied over both pSp and e4e. A visual comparison with IDInvert [73] is provided in the supplementary materials. For a comparison with optimization techniques, we compare to PTI [54] and the latent vector optimization into $\mathcal{W}+$ from Karras *et al*. [35].

4.1. Reconstruction Quality

Qualitative Evaluation We begin with a qualitative comparison, provided in Fig. 4. While optimization techniques are typically able to achieve accurate reconstructions, they come with a high computational cost. HyperStyle offers visually comparable results with an inference time several orders of magnitude faster. Furthermore, PTI may struggle when inverting a low-resolution input (2nd row), yielding a blurred reconstruction due to its inherent design of overfitting to the target image. Our hypernetwork, meanwhile, is trained on a large image collection and is therefore less likely to re-create such resolution-based artifacts. In addition, compared to single-shot encoders (pSp and e4e), HyperStyle better captures the input identity (3rd row). When compared to the more recent ReStyle [5] encoders, HyperStyle is still able to better reconstruct finer details such as complex hairstyles (1st row) and clothing (2nd row).

Figure 4. Reconstruction quality comparison. HyperStyle achieves comparable results to optimization techniques while requiring a fraction of the inference time. Compared to encoders, our method attains superior results in terms of the preservation of identity and finer details such as hairstyles and clothing. Additional results are presented in the supplementary materials. Best viewed zoomed-in.

Quantitative Evaluation In Tab. 2, we present a quantitative evaluation focusing on the time-accuracy trade-off. Along with the inference time of each method, we report the pixel-wise L_2 distance, the LPIPS [72] distance, and the MS-SSIM [66] score between each reconstruction and source. We additionally measure identity similarity using a pre-trained facial recognition network [27]. For HyperStyle and ReStyle [5], we performed multiple iterative steps until the metric scores stopped improving or until 10 iterations were reached. For optimization, we use at most $1,500$ steps, while for PTI we perform at most 350 pivotal tuning steps.

As presented in Tab. 2, HyperStyle's performance consistently surpasses that of the encoder-based methods. Surprisingly, it even achieves results on par with the StyleGAN2 optimization [35], while being nearly 200 times faster. Overall, HyperStyle demonstrates optimization-level reconstructions achieved with encoder-like inference times.

4.2. Editability via Latent Space Manipulations

Good inversion methods should provide not only meticulous reconstructions but also *highly-editable* latent codes. We thus evaluate the editability of our produced inversions. We do so by analyzing two key aspects. One aspect of interest is the range of modifications that an inverted latent can support (e.g., how much the pose can be changed). The other is how well the identity is preserved along this range.

Qualitative Evaluation As shown in Fig. 5, our method successfully achieves realistic and meaningful edits, while being faithful to the input identity. The inversions of optimization, pSp, and *ReStyle*$_{pSp}$ reside in poorly-behaved latent regions of $\mathcal{W}+$. Therefore, their editing is less meaningful and introduces significant artifacts. For instance, in the cars domain, they struggle in making notable changes to the car color and shape. In the 4th row, these methods fail to either preserve the original identity or perform a full frontalization. On the other end of the reconstruction-editability trade-off, e4e and *ReStyle*$_{e4e}$ are more editable but cannot faithfully preserve the original identity, as demonstrated in the 3rd and 4th rows. In contrast, HyperStyle and PTI, which invert into the well-behaved $\mathcal{W}$ space, are more robust in their editing capabilities while successfully retaining the original identity. Yet, HyperStyle requires a significantly lower inference overhead to achieve these results.

Quantitative Evaluation Comparing the editability of inversion methods is challenging since applying the same editing step size to latent codes obtained with different methods results in different editing strengths. This would introduce unwanted bias to the identity similarity measure, as the less-edited images may tend to be more similar to the source. To address this, we edit using a *range* of various step sizes and plot the measured identity similarity along this range, resulting in a continuous similarity curve for each inversion method. This allows us to validate the identity preservation with respect to a fixed editing magnitude, as well as examine the range of edits supported. Ideally, an inversion method should achieve high identity similarity across a wide range of editing strengths. We measure the editing magnitude using trait-specific classifiers (HopeNet [55] for pose and the classifier from Lin *et al*. [39] for smile extent). As before, identity similarity is measured using the CurricularFace method [27].

As can be seen in Fig. 6, HyperStyle consistently outperforms other encoder-based methods in terms of identity preservation while supporting an equal or greater editing range. Compared to optimization-based techniques, HyperStyle achieves similar identity preservation and editing range yet does so substantially faster.

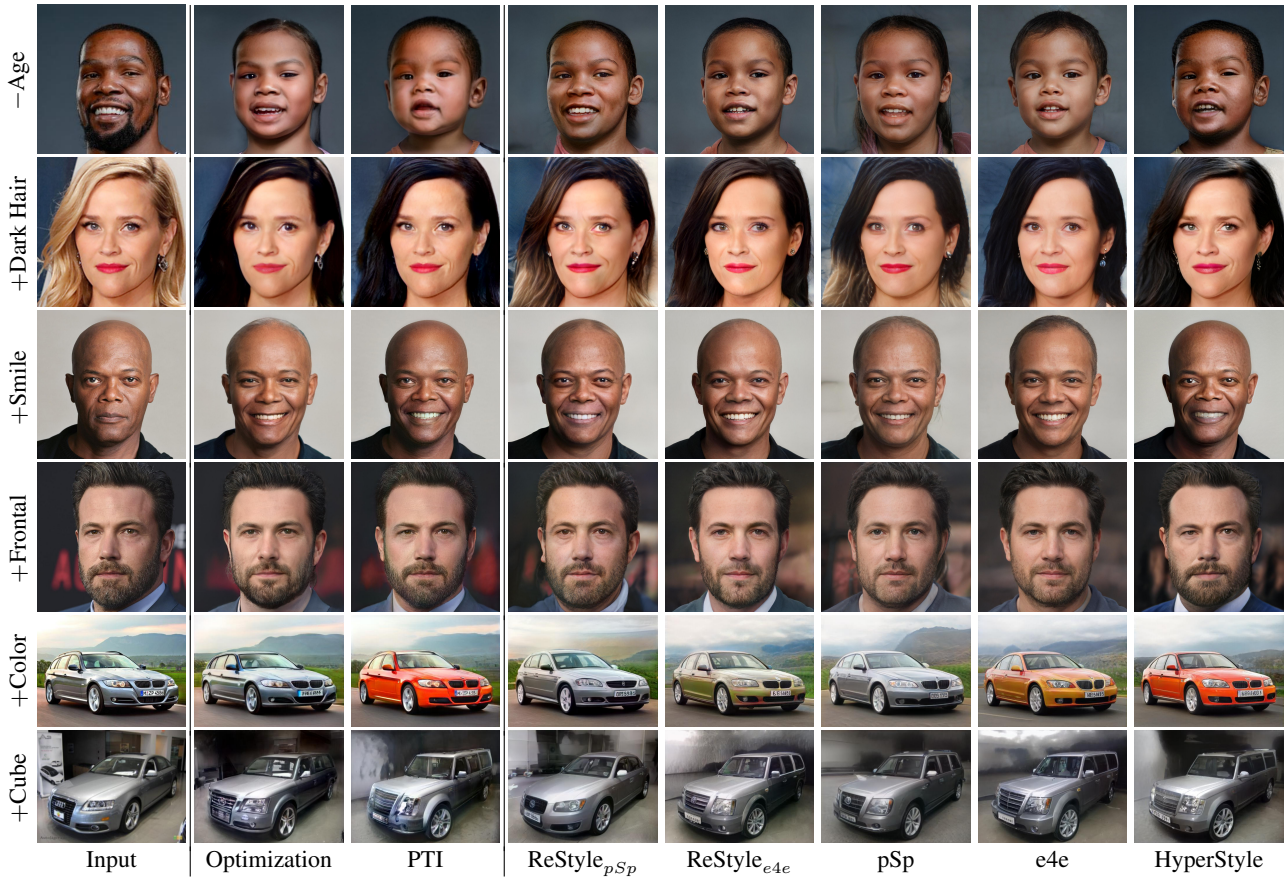

Figure 5. Editing quality comparison. We perform various edits [24, 48, 56] over latent codes obtained by each inversion method. HyperStyle achieves both realistic, faithful edits and a high level of identity preservation across the different edits. Best viewed zoomed-in.

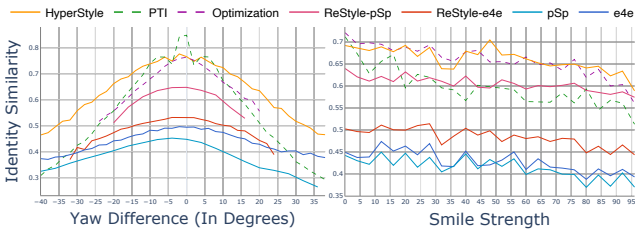

Figure 6. Quantitative editing metrics. For each method, we compute the identity similarity between the original and edited images as a function of editing magnitude.

These results highlight the appealing nature of HyperStyle. With respect to other encoders, HyperStyle achieves superior reconstruction quality while providing strong editability and fast inference. Additionally, compared to optimization techniques, HyperStyle achieves comparable reconstruction and editability at a fraction of the time, making it more suitable for real-world use at scale. This places HyperStyle favorably on both the reconstruction-editability and the time-accuracy trade-off curves.

4.3. Ablation Study

We now validate the design choices described in Sec. 3. Results are summarized in Tab. 3. First, we investigate the choice of layers refined by the hypernetwork. We observe that training only the medium and fine non-toRGB layers achieves comparable performance, a slimmer network, and faster inference. Notably, we also find that altering toRGB layers may harm editability. Second, we find the iterative scheme to be more accurate with fewer artifacts. Finally, we validate the effectiveness of the Shared Refinement Block and the information sharing it provides. Visual comparisons of all ablations can be found in the supplementary materials.

Separable Convolutions Our final configuration uses shared offsets for each convolutional kernel. An important question is whether this constrains the network too strongly. To answer this, we design an alternative refinement head, inspired by separable convolutions [26]. Rather than predicting offsets for an entire $k \times k \times C^{in} \times C^{out}$ filter in one step, we decompose it into two slimmer predictions: $k \times k \times C^{in} \times 1$ and $k \times k \times 1 \times C^{out}$. The final offset block is then given by their product. This allows us to predict an offset for every parameter of the kernel, potentially increasing the network's expressiveness. We observe (Tab. 3) that the increased flexibility of predicting an offset per parameter does not improve reconstruction, indicating that simpler, per-channel predictions are sufficient.

Method	Layers	Iters	↑ ID	↓ LPIPS	↓ L_2	↓ Time
No Iterative Refinement	C,M,F,R	1	0.68	0.10	0.02	0.17
	C,M,F	1	0.67	0.10	0.02	0.16
	M,F	1	0.66	0.11	0.021	0.15
HyperStyle	**M,F**	**10**	0.76	0.09	0.019	1.23
HyperStyle + Coarse	C,M,F	10	0.74	0.10	0.02	1.54
HyperStyle w/o Shared Refinement	M,F	10	0.68	0.12	0.022	1.36
Separable Convs.	M,F	10	0.71	0.10	0.019	1.28

Table 3. Ablation study. We validate the hypernetwork components and design choices: the importance of different layers — coarse (C), medium (M), fine (F), and toRGB (R) — as well as the iterative refinement scheme and Shared Refinement. We also explore separable convolutions as an alternative refinement head.

4.4. Additional Applications

Domain Adaptation Many works [18,46,51,68] have explored fine-tuning a pre-trained StyleGAN towards semantically similar domains. This process maintains a correspondence between semantic attributes in the two latent spaces, allowing translation between domains [68]. Yet, some features, such as facial hair or hair color, may be lost during this translation. To address this, we use HyperStyle trained on the source generator to modify the fine-tuned target generator. Namely, given an input image, we can take the weight offsets predicted with respect to the source generator and apply them to the target generator. The image in the new domain is then obtained by passing the image's original latent code to the modified target generator.

Fig. 7 shows examples of applying weight offsets over various fine-tuned generators. As shown, when no offsets are applied, important details are lost. However, HyperStyle leads to more faithful translations preserving identity without harming the target style. Importantly, the translations are attained with no domain-specific hypernetwork training.

Editing Out-of-Domain Images To this point, we have discussed handling images from the same domain as used for training. If our hypernetwork has indeed learned to generalize, it should not be sensitive to the domain of the input. As may be expected, standard encoders cannot handle out-of-domain images well (see Fig. 8 for e4e and the supplementary materials for others). By adjusting the pre-trained generator towards a given out-of-domain input, HyperStyle enables editing diverse images, without explicitly training a new generator on their domain. This points to improved expressiveness and generalization. It seems the hypernetwork does not just fix poorly reconstructed attributes but learns to adapt the generator in a more general sense. We find these results to be a promising direction for manipulating out-of-domain images without having to train new generators or perform lengthy per-image tuning.

Figure 7. Weight offsets predicted by HyperStyle trained on FFHQ are also applicable for modifying fine-tuned generators (e.g., Toonify [51] and StyleGAN-NADA [18]). Our refinement leads to improved identity preservation while retaining target style.

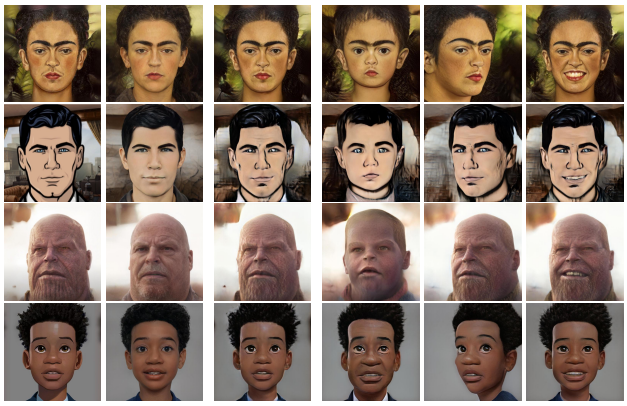

Figure 8. Trained only on real images, our method successfully generalizes to challenging styles not observed during training, even without generator fine-tuning.

5. Conclusions

We introduced HyperStyle, a novel approach for StyleGAN inversion. We leverage recent advancements in hypernetworks to achieve optimization-level reconstructions at encoder-like inference times. In a sense, HyperStyle *learns* to efficiently optimize the generator for a given target image. Doing so mitigates the reconstruction-editability tradeoff and enables the effective use of existing editing techniques on a wide range of inputs. In addition, HyperStyle generalizes surprisingly well, even to out-of-domain images neither the hypernetwork nor the generator have seen during training. Looking forward, further broadening generalization away from the training domain is highly desirable. This includes robustness to unaligned images and unstructured domains. The former may potentially be addressed through StyleGAN3 [33] while the latter would probably warrant training on a richer set of images. In summary, we believe this approach to be an essential step towards interactive and semantic in-the-wild image editing and may open the door for many intriguing real-world scenarios.

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE international conference on computer vision*, pages 4432–4441, 2019. 1, 2

[2] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8296–8305, 2020. 1, 2

[3] Rameen Abdal, Peihao Zhu, Niloy Mitra, and Peter Wonka. Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows, 2020. 2

[4] Yuval Alaluf, Or Patashnik, and Daniel Cohen-Or. Only a matter of style: Age transformation using a style-based regression model. *ACM Trans. Graph.*, 40(4), 2021. 1, 4, 5

[5] Yuval Alaluf, Or Patashnik, and Daniel Cohen-Or. Restyle: A residual-based stylegan encoder via iterative refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021. 2, 4, 5, 6

[6] David Bau, Alex Andonian, Audrey Cui, YeonHwan Park, Ali Jahanian, Aude Oliva, and Antonio Torralba. Paint by word, 2021. 1

[7] David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. Semantic photo manipulation with a generative image prior. 38(4), 2019. 2, 3

[8] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 3

[9] Shu-Yu Chen, Feng-Lin Liu, Yu-Kun Lai, Paul L. Rosin, Chunpeng Li, Hongbo Fu, and Lin Gao. Deepfaceediting: Deep face generation and editing with disentangled geometry and appearance control. *ACM Trans. Graph.*, 40:90:1–90:15, 2021. 1

[10] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains, 2020. 5

[11] Min Jin Chong, Wen-Sheng Chu, Abhishek Kumar, and David Forsyth. Retrieve in style: Unsupervised facial feature transfer and retrieval, 2021. 2

[12] Edo Collins, R. Bala, B. Price, and S. Süsstrunk. Editing in style: Uncovering the local semantics of gans. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5770–5779, 2020. 1

[13] Edo Collins, Raja Bala, Bob Price, and Sabine Susstrunk. Editing in style: Uncovering the local semantics of gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5771–5780, 2020. 2

[14] Antonia Creswell and Anil Anthony Bharath. Inverting the generator of a generative adversarial network. *IEEE transactions on neural networks and learning systems*, 30(7):1967–1974, 2018. 1, 2

[15] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition, 2019. 5

[16] Emily Denton, Ben Hutchinson, Margaret Mitchell, and Timnit Gebru. Detecting bias with generative counterfactual face attribute augmentation. *arXiv preprint arXiv:1906.06439*, 2019. 2

[17] Tan M. Dinh, Anh Tuan Tran, Rang Nguyen, and Binh-Son Hua. Hyperinverter: Improving stylegan inversion via hypernetwork. *arXiv preprint arXiv:2112.00719*, 2021. 2

[18] Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators, 2021. 2, 8

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

[20] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press. 1

[21] Jinjin Gu, Yujun Shen, and Bolei Zhou. Image processing using multi-code gan prior, 2020. 2

[22] 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

[23] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 2, 4

[24] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. *arXiv preprint arXiv:2004.02546*, 2020. 1, 2, 7

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 2, 4

[26] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2, 7

[27] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5901–5910, 2020. 6

[28] Shady Abu Hussein, Tom Tirer, and Raja Giryes. Image-adaptive gan based reconstruction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3121–3129, 2020. 3

[29] Omer Kafri, Or Patashnik, Yuval Alaluf, and Daniel Cohen-Or. Stylefusion: A generative model for disentangling spatial segments, 2021. 2

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

[31] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 5

[32] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data, 2020. 1

[33] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *CoRR*, abs/2106.12423, 2021. 1, 8

[34] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 1, 2, 5

[35] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020. 1, 2, 5, 6

[36] Hyunsu Kim, Yunjey Choi, Junho Kim, Sungjoo Yoo, and Youngjung Uh. Exploiting spatial dimensions of latent in gan for real-time image editing, 2021. 2

[37] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 5

[38] Kathleen M. Lewis, Srivatsan Varadharajan, and Ira Kemelmacher-Shlizerman. Tryongan: body-aware try-on via layered interpolation. *ACM Trans. Graph.*, 40:115:1–115:10, 2021. 1

[39] Ji Lin, Richard Zhang, Frieder Ganz, Song Han, and Jun-Yan Zhu. Anycost gans for interactive image synthesis and editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14986–14996, 2021. 6

[40] Zachary C Lipton and Subarna Tripathi. Precise recovery of latent vectors from generative adversarial networks. *arXiv preprint arXiv:1702.04782*, 2017. 1, 2

[41] Gidi Littwin and Lior Wolf. Deep meta functionals for shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1824–1833, 2019. 2

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

[43] Junyu Luo, Yong Xu, Chenwei Tang, and Jiancheng Lv. Learning inverse mapping by autoencoder based generative adversarial nets. In *International Conference on Neural Information Processing*, pages 207–216. Springer, 2017. 2

[44] Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. Pulse: Self-supervised photo upsampling via latent space exploration of generative models. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 2437–2445, 2020. 1

[45] Yuval Nirkin, Lior Wolf, and Tal Hassner. Hyperseg: Patch-wise hypernetwork for real-time semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4061–4070, 2021. 2

[46] Utkarsh Ojha, Yijun Li, Jingwan Lu, Alexei A. Efros, Yong Jae Lee, Eli Shechtman, and Richard Zhang. Few-shot image generation via cross-domain correspondence, 2021. 8

[47] Xingang Pan, Xiaohang Zhan, Bo Dai, Dahua Lin, Chen Change Loy, and Ping Luo. Exploiting deep generative prior for versatile image restoration and manipulation. In *European Conference on Computer Vision*, pages 262–277. Springer, 2020. 3

[48] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery, 2021. 1, 2, 7

[49] Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M. Álvarez. Invertible conditional gans for image editing, 2016. 2

[50] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14104–14113, 2020. 2

[51] Justin N M Pinkney and Doron Adler. Resolution Dependant GAN Interpolation for Controllable Image Synthesis Between Domains. *arXiv preprint arXiv:2010.05334*, 2020. 8

[52] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 2

[53] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2, 4, 5

[54] 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

[55] Nataniel Ruiz, Eunji Chong, and James M. Rehg. Fine-grained head pose estimation without keypoints. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018. 6

[56] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9243–9252, 2020. 1, 2, 7

[57] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. *arXiv preprint arXiv:2007.06600*, 2020. 2

[58] Przemysław Spurek, Artur Kasymov, Marcin Mazur, Diana Janik, Sławomir Tadeja, Łukasz Struski, Jacek Tabor, and Tomasz Trzciński. Hyperpocket: Generative point cloud completion. *arXiv preprint arXiv:2102.05973*, 2021. 2

[59] Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhöfer, and Christian Theobalt. Stylerig: Rigging stylegan for 3d control over portrait images. *arXiv preprint arXiv:2004.00121*, 2020. 2

[60] Ayush Tewari, Mohamed Elgharib, Mallikarjun B R., Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhöfer, and Christian Theobalt. Pie: Portrait image embedding for semantic control, 2020. 2

[61] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation, 2021. 1, 2, 3, 4, 5

[62] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019. 2

[63] Andrey Voynov and Artem Babenko. Unsupervised discovery of interpretable directions in the gan latent space. *arXiv preprint arXiv:2002.03754*, 2020. 2

[64] Binxu Wang and Carlos R Ponce. A geometric analysis of deep generative image models and its applications. In *International Conference on Learning Representations*, 2021. 2

[65] Tengfei Wang, Yong Zhang, Yanbo Fan, Jue Wang, and Qifeng Chen. High-fidelity gan inversion for image attribute editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[66] 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*, volume 2, pages 1398–1402. Ieee, 2003. 6

[67] Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12863–12872, 2021. 5

[68] Zongze Wu, Yotam Nitzan, Eli Shechtman, and Dani Lischinski. Stylealign: Analysis and applications of aligned stylegan models, 2021. 8

[69] Weihao Xia, Yujiu Yang, Jing-Hao Xue, and Baoyuan Wu. Tedigan: Text-guided diverse face image generation and manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[70] Raymond A. Yeh, Chen Chen, Teck Yian Lim, Alexander G. Schwing, Mark Hasegawa-Johnson, and Minh N. Do. Semantic image inpainting with deep generative models, 2017. 2

[71] Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. *arXiv preprint arXiv:1810.05749*, 2018. 2

[72] 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. 3, 5, 6

[73] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. *arXiv preprint arXiv:2004.00049*, 2020. 1, 2, 5

[74] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European conference on computer vision*, pages 597–613. Springer, 2016. 1, 2

[75] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. Improved stylegan embedding: Where are the good latents?, 2020. 1, 2