The Devil is in the Details: StyleFeatureEditor for Detail-Rich StyleGAN Inversion and High Quality Image Editing

Denis Bobkov[1] Vadim Titov[2] Aibek Alanov[1,2] Dmitry Vetrov[3]
[1]HSE University [2]AIRI [3]Constructor University, Bremen
dnbobkov@edu.hse.ru, titov@2a2i.org, aalanov@hse.ru, dvetrov@constructor.university

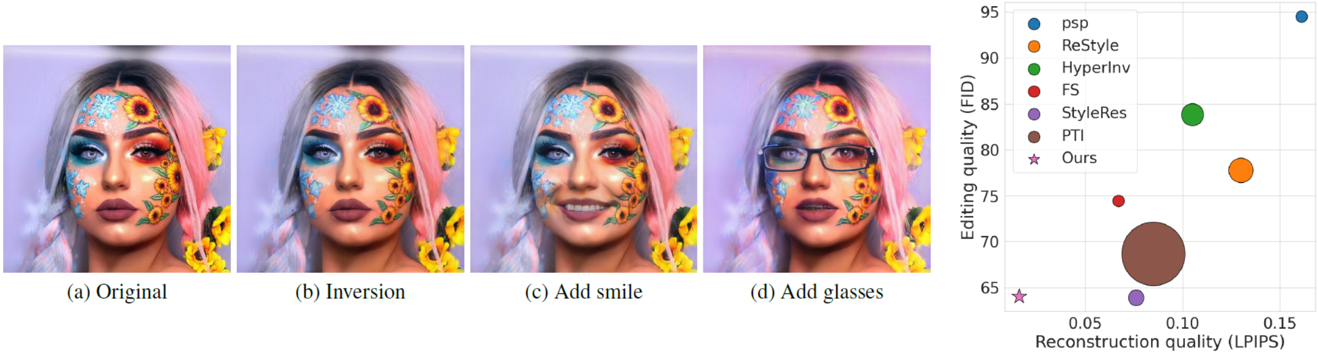
(a) Original (b) Inversion (c) Add smile (d) Add glasses

Figure 1. **Editing examples and graphical comparison for StyleFeatureEditor.** *Our approach takes a real image, inverts it to the StyleGAN latent space, edits the found latents, and synthesises the edited image. On the left, we present examples of our approach, while on the right, we display a comparison with previous approaches. To evaluate inversion quality, we used LPIPS↓. Additionally, to compare the editing capabilities, we compute FID↓ for 3 editing directions (see 4.3) and average them with coefficients equal to the average FID per editing direction. The size of markers indicates the inference time of the method, with larger markers indicating a higher time. StyleFeatureEditor capable of reconstructing even finer image details and preserving them during editing.*

Abstract

The task of manipulating real image attributes through StyleGAN inversion has been extensively researched. This process involves searching latent variables from a well-trained StyleGAN generator that can synthesize a real image, modifying these latent variables, and then synthesizing an image with the desired edits. A balance must be struck between the quality of the reconstruction and the ability to edit. Earlier studies utilized the low-dimensional W-space for latent search, which facilitated effective editing but struggled with reconstructing intricate details. More recent research has turned to the high-dimensional feature space F, which successfully inverses the input image but loses much of the detail during editing. In this paper, we introduce StyleFeatureEditor – a novel method that enables editing in both w-latents and F-latents. This technique not only allows for the reconstruction of finer image details but also ensures their preservation during editing. We also present a new training pipeline specifically designed to train our model to accurately edit F-latents. Our method is compared with state-of-the-art encoding approaches, demonstrating that our model excels in terms of reconstruction quality and is capable of editing even challenging out-of-domain examples.

1. Introduction

In recent years, GANs [15] have achieved impressive results in image generation, which has led to their use in a wide variety of computer vision tasks. One of the most successful models is StyleGAN [21–24], which not only has a high quality of generation, but also a rich semantic latent space. In this space, we can control different semantic attributes of the generated images by changing their latent code [2]. However, to apply this editing technique to real images, we must be able to find their internal representation in the StyleGAN latent space. This problem is called GAN inversion [47], and although it is well studied and many approaches have been proposed [2, 5, 12, 28, 32, 38, 44, 46], it is still an open problem to develop a method that simultaneously satisfies three requirements: high-quality reconstruction, good editability, and fast inference. Our work is dedicated to the development of such a method.

Existing GAN inversion approaches can be divided into two groups: optimization-based and encoder-based. Optimization-based methods [1, 2] learn a latent representation for each input image that best reconstructs that image. This results in good inversion quality, but such overfitted latent codes may deviate from the original distribution of the latent space, resulting in poor editing. While there are approaches that improve the quality of editing by finetuning the generator itself for a given image [33], this does not address the main drawback of such methods, which is that the inversion is too long, making them impractical to use in real-time applications. In contrast, more practical encoder-based methods [32, 38] allow us to obtain a latent representation of the input image in a single network pass. However, with these approaches, it is more difficult to achieve both high quality and good editability at the same time. This is the so-called distortion-editability trade-off [38]. Inversion quality and editability are directly related to the dimensionality of the latent space in which we encode the input image. In low-dimensional W and W^+ spaces, we will get low reconstruction quality but high editability, because the low dimensionality of the latent code is a good regularizer that keeps it in the StyleGAN manifold. If we train the encoder to predict in the high-dimensional StyleGAN feature space $\mathcal{F}_k$, this will significantly increase the quality of the reconstructions at the expense of degraded editability, since in such a space it is easier to overfit to a particular image and escape the region in the latent space where semantic transformations work. Methods working in $\mathcal{F}_k$ [28, 40, 44] try to challenge this problem by using additional transformations over the tensor F_k, but it is not completely solved. In particular, the editability problem is amplified when one increases the dimensionality of the $\mathcal{F}_k$ feature space by taking them from earlier layers to improve the quality of reconstructions.

In this paper, we propose a framework that allows us to train an encoder in a high-dimensional $\mathcal{F}_k$ space that simultaneously achieves both excellent reconstruction quality and good editability. The main idea of our approach is to divide the training of our encoder into two phases. In the first, we train an encoder that predicts a latent code in $\mathcal{F}_k$ space with high resolution, which allows us to reconstruct images with high quality, but at the same time significantly reduces editability. To recover the editability , we introduce a second phase of training : we propose to train a new Feature Editor module that task is to modify the feature tensor F_k to obtain the target editing in image generation. The main difficulty in training this module is that we do not have a training data, where for each image there would be its edited versions. Therefore, we proposed to automatically generate such data using an encoder operating in W^+ space. That is, as training samples for Feature Editor, we take reconstructions of real images using a standard encoder with low inversion quality, but with good editability. And on this data we train the Feature Editor, which predicts F_k' for the feature tensor F_k of the input image, from which its edited version should be generated.

Thus, thanks to the proposed two-phase encoder learning framework, we are able to train an inversion model that has both high reconstruction quality, significantly better than the current state-of-the-art, and good editability. We conducted extensive experiments, demonstrating a significant improvement over state-of-the-art methods in the inversion task, and comparable results in the image editing. In particular, we have significantly improved the reconstruction metrics in terms of LPIPS and L_2 by more than a factor of 4 compared to StyleRes[28], while the running time is equivalent to conventional encoder-based methods.

2. Related Work

Latent Space Manipulation. With the development of StyleGAN models [21–24], they started to be actively used for the task of image editing. Many methods have shown that by changing the latent code of an image in the latent space of StyleGAN, it is possible to change the semantic attributes of the image [2]. There are methods that find such directions using supervised approaches utilized attribute labelled samples or pre-trained classifiers [3, 14, 35, 37]. Unsupervised methods do not use any kind of labelling [9, 16, 34, 39], instead they, for example, perform PCA either in StyleGAN's feature space [16] or find directions in the weight space [9]. Other methods use a self-supervised learning approach [19, 30, 36]. And there are methods that utilize language-image models [31] to find desired edits guided by text [13, 27, 43]. To apply all these methods to real images, it is necessary first to encode images in StyleGAN's latent space.

GAN Inversion. The task of GAN inversion [47] is to find the latent code for a real image, from which it can be generated by StyleGAN and the result has to be perceptually equal to the input image and can be edited by changing this latent code. Existing GAN inversion methods can be divided into two types: optimization-based methods [1, 2, 7, 8, 10, 33, 46, 48] and encoder-based methods [4–6, 12, 18, 26, 28, 29, 32, 38, 40, 44, 46].

Optimization-based methods. Optimization methods find latent code by optimizing directly over the reconstruction losses. The first approaches [1, 2, 10, 46] performed optimization in Z/W/W^+ spaces. To improve the quality of the reconstruction, later methods proposed to optimize additionally in the StyleGAN feature space [48]. Since the latent code can escape from the StyleGAN manifold during the optimization process and thus negatively affect the editability, it has been proposed to additionally fine-tune the generator weights for each image [33]. Although high reconstruction quality and good editability can be achieved with these approaches, the optimization process is too long, requiring up to several minutes for each image, which is not applicable for real-time interactive editing.

Encoder-based methods. Encoder-based methods allow learning the mapping from the space of real images to the StyleGAN latent space in one or more passes through the neural network. Basically, these methods differ in the latent spaces they encode to. The first methods trained the mapping for the simplest Z, W, W^+ spaces [29, 32, 38, 46],

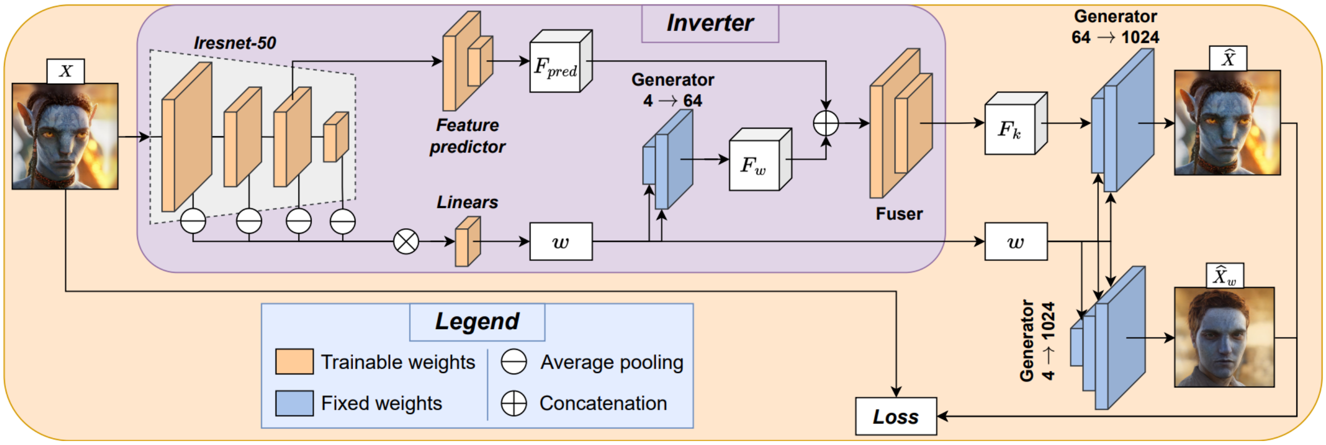

Figure 2. **The Inverter training pipeline.** Input image X is passed to Feature-Style-like backbone that predicts $w \in W^+$ and $F_{pred} \in \mathcal{F}_k$. Then $F_w = G(w_{0:k})$ is synthesized and passed with F_{pred} to the Fuser that predicts F_k. Inversion $\widehat{X} = G(F_k, w_{k+1:N})$ is generated. Additional reconstruction $\widehat{X}_w = G(w_{0:N})$ is synthesized from w-latents only. Loss is calculated for pairs $(X, \widehat{X})$ and $(X, \widehat{X}_w)$.

which gave good editability but low reconstruction quality. Next methods were proposed that additionally predicted changes in the generator weights using a hypernetwork to better reconstruct the input image [5, 12]. This increased the quality of the reconstruction without sacrificing editability. There are also methods that propose to use multiple passes over the encoder to refine the details of the image during reconstruction [4, 5]. The most successful methods train encoders for StyleGAN's feature space $\mathcal{F}_k$ [28, 40, 44]. Such methods achieve the highest reconstruction quality among encoder-based methods, and are comparable to optimization-based methods. The main remaining problem is poor editability, since in such a high-dimensional latent space it is very easy to overfit the image and go out of the natural StyleGAN manifold.

In our paper, we propose a framework that preserves the editability of an encoder trained in the StyleGAN's feature space $\mathcal{F}_k$, and achieves phenomenal reconstruction quality.

3. Method

3.1. Overview

The goal of StyleGAN inversion methods is to find an internal representation of the input image in the StyleGAN latent space that contains as much information and detail as possible about the image itself, and at the same time allows editing it. This internal representation can be searched in different StyleGAN latent spaces, which have different properties. We can distinguish two main latent spaces that are considered in the StyleGAN inversion task, namely W^+ and $\mathcal{F}_k$. W^+ is the concatenation of N vectors $w_1, \ldots,$ w_N, which are fed into each of the N convolutional layers of StyleGAN. $\mathcal{F}_k$ is feature space, which is the combination of the W^+ space and the space of the feature outputs of the k-th convolutional layer of the StyleGAN.

It is known that the representation of an image in W^+ space preserves few details, but allows good editing. In $\mathcal{F}_k$ space the situation is the opposite – we can almost perfectly reconstruct the original image, but this representation is difficult to edit. The latest most advanced encoders Feature-Style [44] and StyleRes [28] work in $\mathcal{F}_k$ space, and to solve the editing problem, they offer their own techniques to transform the $F_k \in \mathcal{F}_k$ feature tensor during editing. But these techniques do not solve the problem completely. And it is exacerbated if the resolution of the F_k feature tensor is increased. In this case, the quality of reconstructions improves significantly, but the editability completely vanishes.

In our work, we propose a way to edit F_k feature tensor that preserves high quality of the reconstruction with good editability. The basic idea is to train an additional module called Feature Editor, which will transform the feature tensor F_k in the right way for each edit. But to train Feature Editor, we will need a special training dataset, where for each image we need to have its edited versions. It is clear that it is very difficult and expensive to manually build such a dataset. Therefore, we generated this dataset using an encoder that operates in W^+ space. That is, for each real image from the dataset, we find its reconstruction in W^+ space, get its edited version, and use these two images to train our Feature Editor module. This approach allowed us to significantly improve the quality of edits, even for high resolutions of F_k . Further, we give more details about the architecture of StyleFeatureEditor and the training process.

3.2. Architecture

In this section, we describe StyleFeatureEditor, which consists of two parts: Inverter I and Feature Editor H. The task of Inverter is to extract reconstruction features from the input image, while Feature Editor should transform these features according to the information about the desired edit.

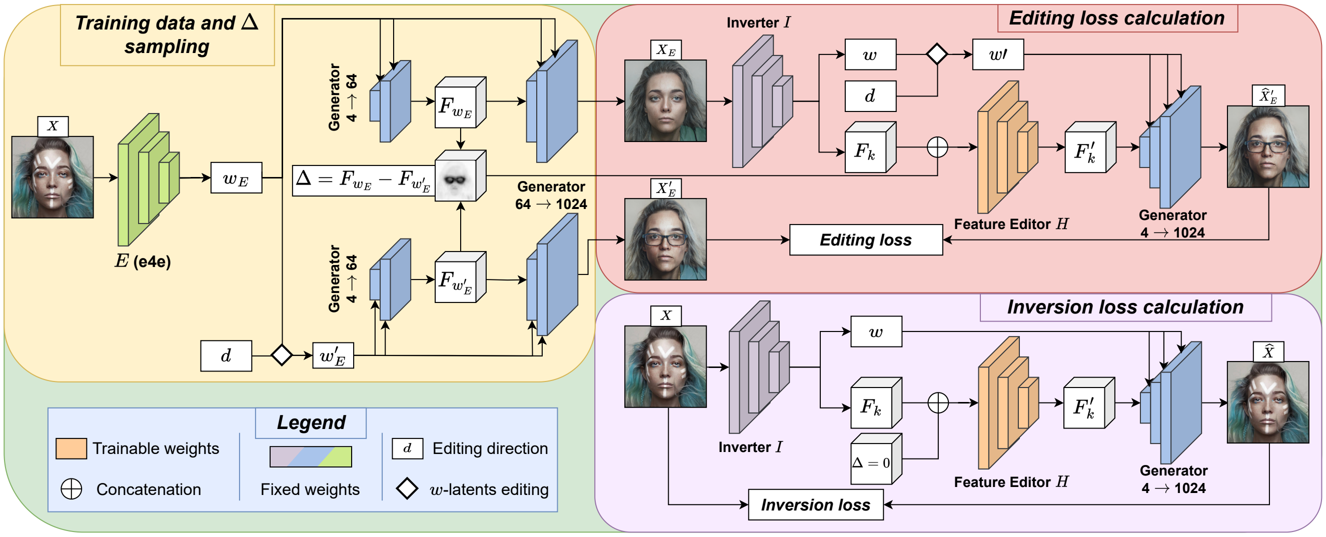

Figure 3. **The Feature Editor training pipeline and inference.** To obtain **editing loss**, one need to synthesize training samples: X_E – training input, and X'_E – training target. The pre-trained encoder E takes the real image X and predicts $w_E \in W^+$. Edited direction $d \in \mathcal{D}$ is randomly sampled, after which w_E is edited to $w'_E = w_E + d$. Image X_E and intermediate features F_{w_E} are synthesized from w_E, while X'_E and $F_{w'_E}$ are synthesized from w'_E via generator G. X_E is used as input and passed to frozen Inverter I that predicts F_k and w that is edited to w' according sampled d. Then Δ is calculated, and Feature Editor H edits F_k according Δ. The edited reconstruction $\widehat{X}'_E$ is synthesized from F'_k and $w'_{k+1:N}$. **Editing loss** is calculated between X'_E and $\widehat{X}'_E$. To obtain the **inversion loss**, the real image X is passed to I that predicts w and F_k, F_k is edited to F'_k by H with $\Delta = 0$. The inversion $\widehat{X}$ is synthesized from F'_k and $w_{k+1:N}$. The Inversion loss is calculated between X and $\widehat{X}$. **Inference pipeline** is the same as synthesizing $\widehat{X}'_E$ but with the assumption that I takes real image X instead of X_E.

Inverter. I consists of Feature-Style-like Encoder I_{fse} and an additional module I_{fus} called Fuser. I_{fse} consists of Iresnet50 backbone, Feature predictor and Linear layers (see Fig. 2). First, the input image X is passed to the backbone, which predicts 4 intermediate features, pools them to the same dimensionality, concatenates them, and maps to $w \in W^+$ by Linear layers. The third intermediate feature is also passed to Feature predictor that predicts $F_{pred} \in \mathcal{F}_k$:

$$(w, F_{pred}) = I_{fse}(X). \tag{1}$$

Despite good inversion quality, Feature-Style Encoder fails to reconstruct fine details of the image, thus we increased the predicted feature tensor from $F_{pred} \in \mathcal{F}_5$ to $F_{pred} \in \mathcal{F}_9$ that increases its dimensionality from $\mathbb{R}^{512\times16\times16}$ to $\mathbb{R}^{512\times64\times64}$ respectively.

To take into account the impact of $w_{0:k}$ we additionally synthesize output of the k-th generator layer $F_w = G(w_{0:k})$ via the StyleGAN2 generator G. F_w then fused with predicted F_{pred} by an additional module I_{fus}, which predicts $F_k \in \mathcal{F}_k$:

$$F_k = I_{fus}(F_{pred}, F_w) \tag{2}$$

Thus, I takes input image X and predicts w and F_k:

$$(w, F_k) = I(X). \tag{3}$$

after then, the reconstructed image $\widehat{X}$ is synthesized from F_k and $w_{k+1}, \dots, w_N$:

$$\widehat{X} = G(F_k, w_{k+1:N}). \tag{4}$$

It is also possible to synthesize image $\widehat{X}_w = G(w)$ from w-latents only, which we use during training.

Feature Editor. The predicted feature tensor F_k contains much of the input image information, which allows even finer image details to be reconstructed. However, if we do not transform F_k during editing, artefacts may appear or editing may not work at all. Therefore, we propose an additional Feature Editor module H that transforms predicted F_k according to the desired edit. In order for H to understand what to change, it is necessary to have information Δ about which regions F_k need to be edited. To obtain such information, we propose to use a pre-trained encoder E in W^+ space that is capable of good editing (we use pre-trained e4e encoder [38]).

E takes input image X and predicts $w_E = E(x) \in W^+$, which is edited to $w'_E = w_E + d$ by editing direction d. The outputs of the k-th generator layer F_{w_E} and $F_{w'_E}$ are synthesized from w_E and w'_E respectively. Difference between F_{w_E} and $F_{w'_E}$ contains information about edited regions:

$$\Delta = F_{w_E} - F_{w'_E}. \tag{5}$$

After gaining Δ, H transforms F_k to F'_k according Δ:

$$F'_k = H(F_k, \Delta). \tag{6}$$

The edited image $\widehat{X}'$ is synthesized from F'_k and $w'_{k+1}, \ldots, w'_N$, where w' is edited w (see Fig. 3):

$$\widehat{X}' = G(F'_k, w'_{k+1:N}). \tag{7}$$

To sum up, the inference pipeline of StyleFeatureEditor during editing consists of predicting w and F_k (Eq. 3), editing w to $w' = w + d$, computing Δ according Eq. 5, editing F_k (Eq. 6) and synthesizing edited image (Eq. 7). Inversion assumes the same pipeline, but with $\Delta = 0$.

3.3. Training Inverter (Phase 1)

This section is related to training Inverter I to reconstruct source images. The pipeline of phase 1 is presented in Fig. 2.

The source image X is passed to I which predicts $w, F_k = I(X)$, where $w \in \mathbb{R}^{N\times512}$ and $F_k \in \mathbb{R}^{512\times64\times64}$. Then the generator G synthesizes $\widehat{X} = G(F_k, w_{k+1:N})$ – reconstruction of the input image X. The loss function $\mathcal{L}_{phase1}$ is calculated between X and $\widehat{X}$. In addition, to force information flow not only through feature space $\mathcal{F}_k$, we also calculate $\mathcal{L}_{phase1}$ for reconstruction $\widehat{X}_w = G(w)$ obtained from w-latents only.

The loss function $\mathcal{L}_{phase1}$ consists of two equal parts: the image loss $\mathcal{L}_{im}$ applied to both $(X, \widehat{X})$ and $(X, \widehat{X}_w)$, and the regularization $\mathcal{L}_{reg}$ for constraining the norm of F_k tensor. $\mathcal{L}_{im}$ is calculated as a weighted sum of per-pixel loss $\mathcal{L}_2$, perceptual LPIPS loss $\mathcal{L}_{lpips}$ [45], identity-based similarity loss (ID) $\mathcal{L}_{id}$ [32] by utilizing a pre-trained network (ArcFace [11] for the face domain and ResNet-based [38] for non-facial domains), adversarial loss $\mathcal{L}_{adv}$ by employing a pre-trained StyleGAN discriminator D which we fine-tune during training. As the regularization loss, we use $\mathcal{L}_{reg} = \|F_k\|_2$. So, the total loss $\mathcal{L}_{phase1}$ is calculated as:

$$\mathcal{L}_{im} = \mathcal{L}_2 + \lambda_{lpips}\mathcal{L}_{lpips} + \lambda_{id}\mathcal{L}_{id} + \lambda_{adv}\mathcal{L}_{adv}, \tag{8}$$

$$\mathcal{L}_{phase1} = \mathcal{L}_{im} + \lambda_{reg}\mathcal{L}_{reg}, \tag{9}$$

where $\lambda_{lpips} = 0.8, \lambda_{id} = 0.1, \lambda_{adv} = 0.01, \lambda_{reg} = 0.01$.

3.4. Training Feature Editor (Phase 2)

The goal of this phase is to train the Feature Editor H to edit F_k. The training pipeline of this phase is available in Fig. 3. In this phase, we assume that I is already trained, so we froze its weights and train only H weights.

For this purpose it is necessary to have a dataset consisting of pairs (X, X'), where X' is the edited version of the image X, but it is difficult to collect such data manually. Therefore, we propose to use a pre-trained encoder E in W^+ space suitable for editing to generate such data. E takes input image X and predicts w_E, it is edited with specified editing direction d to $w'_E = w_E + d$, after which images X_E and X'_E are synthesized from w_E and w'_E respectively.

During this phase, we fix a set of 13 editing directions $\mathcal{D}$ used in training (more details in Appendix 7). The pipeline of training H using synthetic data is:

1. Pass X to E to obtain w_E and $w'_E = w_E + d$ for the editing direction d randomly sampled from $\mathcal{D}$.
2. Synthesize images X_E, X'_E and feature tensors $F_{w_E}, F_{w'_E}$ from w_E and w'_E respectively.
3. Calculate $\Delta = F_{w_E} - F_{w'_E}$.
4. Compute $(w, F_k) = I(X_E)$.
5. Obtain the edited tensor $F'_k = H(F_k, \Delta)$.
6. Synthesize $\widehat{X}'_E = G(F'_k, w'_{k+1:N})$ – the edited reconstruction.
7. Calculate the loss between $\widehat{X}'_E$ and X'_E.

However, if H is trained only on synthetic images, the reconstruction quality for real images may degrade. To solve this problem, we propose to train H not only on editing, but also on the classical inversion task. The training pipeline is the same, but for inversion we use a real image X as input and assume $\Delta = 0$. X is passed to I, which predicts F_k and w (Eq. 3), $\Delta = 0$ and F_k goes to the Feature editor which predicts F'_k and reconstruction $\widehat{X}$ is synthesised assuming $w' = w$ (Eq. 7). The loss is calculated between X and its reconstruction $\widehat{X}$.

For this phase we used $\mathcal{L}_2$, $\mathcal{L}_{lpips}$ and $\mathcal{L}_{id}$ for both inversion and editing tasks with coefficients from phase 1. For inversion task we additionally use adversarial loss $\mathcal{L}_{adv}$:

$$\mathcal{L}_{edit} = \mathcal{L}_2 + \lambda_{lpips}\mathcal{L}_{lpips} + \lambda_{id}\mathcal{L}_{id}, \tag{10}$$

$$\mathcal{L}_{inv} = \mathcal{L}_2 + \lambda_{lpips}\mathcal{L}_{lpips} + \lambda_{id}\mathcal{L}_{id} + \lambda_{adv}\mathcal{L}_{adv}. \tag{11}$$

The general loss $\mathcal{L}_{phase2}$ for phase 2 is calculated as:

$$\mathcal{L}_{phase2} = \mathcal{L}_{edit}(X'_0, \widehat{X}'_0) + \mathcal{L}_{inv}(X, \widehat{X}) \tag{12}$$

During training we fixed a set of 13 editing directions $\mathcal{D}$, however SFE is capable of generalising to new directions without any retraining. Furthermore, $\mathcal{D}$ can be restricted while SFE's editing abilities remain good on both: seen and unseen directions (see Ablation Study 4.4, Appendix 10). This can be explained by the fact that Δ (which contains almost all editing information) of even one direction will be very different for different images. Therefore, during training, H does not learn specific directions, but generalizes to gather information from Δ. Since Δ depends only on the edited w-latents obtained from E (e4e), our method is able to apply any editing applicable to E (e4e).

More training and architecture detail available in the Appendix 7, 8.

4. Experiments

4.1. Experiment set-up

In our experiments for face domain, we used FFHQ [21] image dataset for training and official test part of Celeba

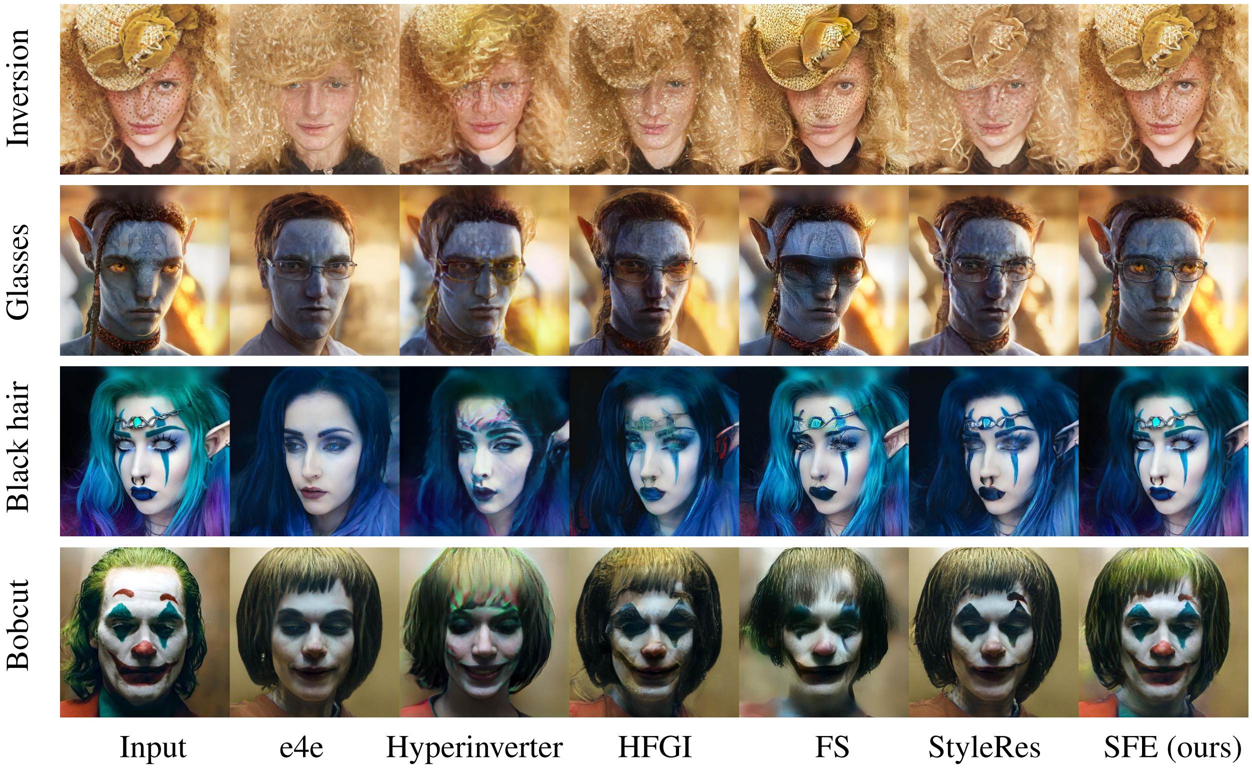

Figure 4. Visual comparison of our method with previous encoder-based approaches on face domain. Row 1 represents the inversion, row 2 – the addition of glasses, row 3 – the darkening of the hair colour, row 4 – the changing of the hairstyle.

HQ dataset [20] for inference. For the car domain, we used train part of Stanford Cars [25] for training and test part for evaluation. For test editings we used InterfaceGAN[35] and Stylespace[42] for both face and car domains, StyleClip[27] and GANSpace[16] for face domain. To extract Δ and sample images for editing loss calculation during training phase 2, we used pre-trained e4e [38] as E. For the inversion calculation, we used our full pipeline including both I and H, assuming $\Delta = 0$ as in Fig. 3.

We compare our method with state-of-the-art encoder approaches such as e4e[38], psp[32], StyleTransformer[18], ReStyle[4], PaddingInverter[6], HyperInverter[12], Hyperstyle[5], HFGI[40], Feature-Style[44], StyleRes[28] and optimisation-based PTI[33]. We used author's original checkpoints, but in car domain, some of them are not public. We train Feature-Style on Stanford Cars by using authors code and omitting models without official checkpoints.

4.2. Qualitative evaluation

To demonstrate the performance of our method, in Figure 4 we compare it with previous approaches on several hard out-of-domain examples. Our approach not only reconstructs more detail than previous ones, but also preserves it during editing. For example, in the first row, our method accurately reconstructs woman's hat while others smooth it out. In the second row, our method preserves the yellow eye colour while editing the eye zone. In rows 3 and 4, it is evident that our approach is better at reconstructing difficult make-up and preserving the colours of the source image.

Additionally, in Figure 5 we show comparison of our method on car domain. In the first row, our method even manages to reconstruct the original shape of a car when the others do not. Moving on to the second row, our method most accurately reconstructs the outline and white lines of the original car, while FS Encoder distorts them. Apart from our approach in the third row, FS Encoder is the only one that can reconstruct the shadow on the car, but it fails in changing car colour.

4.3. Quantitative evaluation

To evaluate the effectiveness of the inversion technique, two key aspects can be examined. Firstly, the accuracy of the inversion, which refers to the degree to which the method is able to reconstruct the details of the original image. Second, the editability – how well the inverted image can be edited.The comparison in both aspects on CelebA-HQ dataset is presented in Table 1.

To measure quality of the inversion details, we used LPIPS [45], L_2 and MS-SSIM [41]. Additionally, we de-

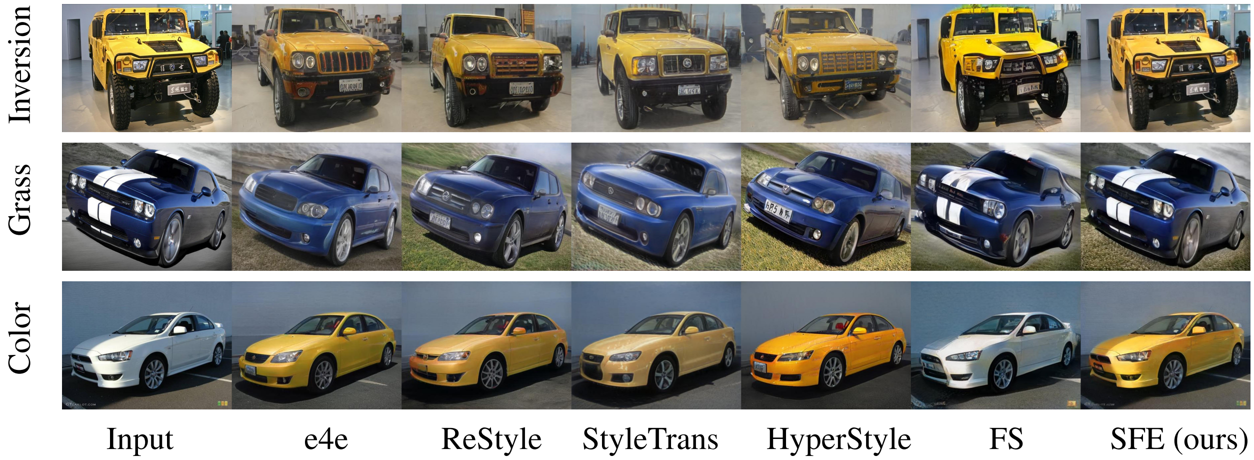

Figure 5. Additional visual comparison of our method with previous encoder-based approaches in the car domain. Row 1 represents the inversion, Row 2 – the addition of grass, Row 3 – the change in car colour.

Table 1. Quantitative comparison results for inversion quality and editing abilities on face domain. To measure inversion we report LPIPS, L_2, MS-SSIM and FID calculated on Celeba HQ test set. To measure editing abilities, we used FID as described in 4.3.We also measured the time required to edit a single image on a single TeslaV100.

	Inversion quality				**Editing quality**			
Model	LPIPS ↓	L2 ↓	FID ↓	MS-SSIM ↑	Smile (-)	Glasses (+)	Old (+)	Time (s)
e4e[38]	0.199	0.047	28.971	0.625	51.245	119.437	68.463	0.034
pSp[32]	0.161	0.034	25.163	0.651	46.220	105.740	67.505	0.034
StyleTransformer[18]	0.158	0.034	22.811	0.656	32.936	81.031	67.250	0.032
ReStyle[4]	0.130	0.028	20.664	0.669	36.365	87.410	56.025	0.138
Padding Inverter[6]	0.124	0.023	25.753	0.672	42.305	98.719	62.283	0.034
HyperInverter[12]	0.105	0.024	16.822	0.673	41.201	93.723	65.282	0.105
HyperStyle[5]	0.098	0.022	20.725	0.700	34.578	86.764	49.267	0.275
HFGI[40]	0.117	0.021	15.692	0.721	27.151	77.213	51.489	0.072
Feature-Style[44]	0.067	0.012	10.861	0.758	26.034	85.686	56.050	0.038
StyleRes[28]	0.076	0.013	8.505	0.797	24.465	**73.089**	43.698	0.063
PTI[33]	0.085	0.008	14.466	0.781	28.302	78.058	44.856	124
SFE (ours)	**0.019**	**0.002**	**3.535**	**0.922**	**24.388**	73.098	**41.677**	0.070

termined realism of the synthesized images by measuring distance between distributions of real and inverted images using FID [17]. Our method outperformed all previous approaches. The most notable difference was seen in LPIPS and L_2, indicating that our method is capable of extremely fine detail reconstruction. We also tested our method in the domain of cars on the Stanford Cars dataset presented in Table 2, which confirms the results described above.

It is challenging to accurately estimate the quality of the editing numerically in the absence of target images. To perform these calculations, we use the technique proposed in [28]. We determine the attribute to be edited, then, based on the Celeba HQ markup, we divide the test dataset into images A with and B without this attribute. Next, we apply the method to B to add this attribute and synthesize B'. The FID between B and B' demonstrates the effectiveness of the technique for editing this attribute. We provide experiments with 3 attributes: removing smile, adding glasses and increasing age.

The results show that our method not only inverts well, but is also comparable to the current state-of-the-art StyleRes in terms of editing capabilities. Furthermore, our method requires only 0.066 seconds to edit a single image on the TeslaV100, far outperforming optimisation-based

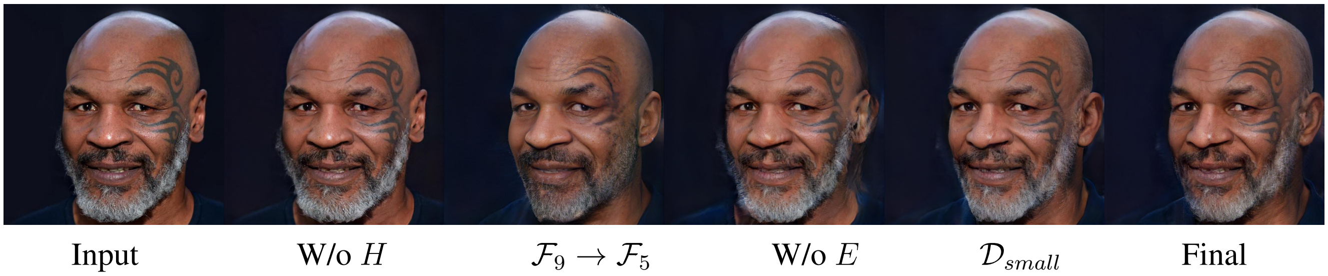

Figure 6. Ablation study. Visual representation of outputs of different ablations (described in 4.4) during pose rotation.

Table 2. Additional quantitative comparisons on the Stanford Cars dataset. We do not provide a calculation of editing ability, as the test set does not have the required markup.

Model	LPIPS ↓	L2 ↓	FID ↓
e4e[38]	0.325	0.122	13.397
ReStyle[4]	0.306	0.102	13.008
StyleTransformer[18]	0.276	0.092	10.644
HyperStyle[5]	0.287	0.080	8.044
Feature-Style[44]	0.147	0.045	7.180
SFE (ours)	**0.039**	**0.004**	**4.035**

Table 3. Ablation study. Quantitative comparison of different ablations (described in 4.4). We calculated all metrics on the test part of Celeba HQ. To measure editing quality, we calculated FID as described in 4.3.

	Inversion			**Editing**	
Model	LPIPS ↓	L2 ↓	FID ↓	Smile (-)	Old (+)
Final model	0.019	0.0017	3.535	**24.388**	**41.677**
W/o H	**0.016**	**0.0013**	**2.975**	28.149	54.621
W/o Fuser	0.023	0.0019	4.239	26.410	42.121
W/o inv loss	0.037	0.0027	5.101	26.179	42.361
W/o E (e4e)	0.021	0.0024	3.829	24.941	44.398
$F_9 \rightarrow F_5$	0.064	0.0089	7.915	25.933	43.317
$\mathcal{D}_{small}$	0.021	0.0019	3.842	24.548	42.317

PTI and matching previous encoder-based methods in terms of inference speed.

4.4. Ablation Study

To ensure the importance of each component in the proposed pipeline, we conducted several ablation experiments. We present the quantitative results of these experiments in Table 3 and visual representations in Figure 6.

First, we tried to discard H and use only I as in training phase 1. Despite a small increase in the inversion metrics, the edits stopped working, proving the significance of H. We also tried an architecture without Fuser I_{fus} (which refers to the case where $F_k = F_{pred}$) and an experiment where the inversion loss is omitted during the second training phase. Both of these experiments resulted in a drop in reconstruction quality that is difficult to detect at low resolution and only visible at high resolution. The fourth experiment was related to omitting E and predicting features for Δ from w obtained from I. The predicted w is much less editable than w_E from e4e, leading to artefacts during editing (Figure 6) and showing that E should be well editable.

We also attempted to train our pipeline with a lower predicted feature dimensionality. We reduced the predicted F_k from $k = 9$ to $k = 5$, which is the dimensionality of the Feature Space Encoder. Despite the significant decrease in inversion quality, this approach is still capable of good editing, unlike Feature-Style. During the last ablation, we reduced the number of editing directions in $\mathcal{D}$ from 13 to 6 in the second training phase. The reduced $\mathcal{D}_{small}$ consists of Age, Afro, Angry, Face Roundness, Bowlcut Hairstyle and Blonde Hair. Despite a slight decrease in metrics, our method is still able to edit directions that were not used during training, as shown in Figure 6.

5. Conclusion

In this paper, we have demonstrated StyleFeatureEditor – a novel approach to image editing via StyleGAN inversion and introduced a new technique for training it. Even for challenging out-of-domain images, we have achieved a reconstruction quality that makes it almost impossible to tell the difference between the real and synthetic images with the naked eye. Thanks to Feature Editor, our method is not only able to reconstruct finer facial details, but also preserves most of them during editing.

6. Acknowledgments

The analysis of related work in sections 1 and 2 were obtained by Aibek Alanov with the support of the grant for research centres in the field of AI provided by the Analytical Center for the Government of the Russian Federation (ACRF) in accordance with the agreement on the provision of subsidies (identifier of the agreement 000000D730321P5Q0002) and the agreement with HSE University No. 70-2021-00139. This research was supported in part through computational resources of HPC facilities at HSE University.

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF 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 J Mitra, and Peter Wonka. Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows. *ACM Transactions on Graphics (ToG)*, 40(3):1–21, 2021. 2

[4] 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*, pages 6711–6720, 2021. 2, 3, 6, 7, 8

[5] Yuval Alaluf, Omer Tov, Ron Mokady, Rinon Gal, and Amit Bermano. Hyperstyle: Stylegan inversion with hypernetworks for real image editing. In *Proceedings of the IEEE/CVF conference on computer Vision and pattern recognition*, pages 18511–18521, 2022. 1, 3, 6, 7, 8

[6] Qingyan Bai, Yinghao Xu, Jiapeng Zhu, Weihao Xia, Yujiu Yang, and Yujun Shen. High-fidelity gan inversion with padding space. In *European Conference on Computer Vision*, pages 36–53. Springer, 2022. 2, 6, 7

[7] Anand Bhattad, Viraj Shah, Derek Hoiem, and DA Forsyth. Make it so: Steering stylegan for any image inversion and editing. *arXiv preprint arXiv:2304.14403*, 2023. 2

[8] Pu Cao, Lu Yang, Dongxu Liu, Zhiwei Liu, Shan Li, and Qing Song. What decreases editing capability? domain-specific hybrid refinement for improved gan inversion. *arXiv preprint arXiv:2301.12141*, 2023. 2

[9] Anton Cherepkov, Andrey Voynov, and Artem Babenko. Navigating the gan parameter space for semantic image editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3671–3680, 2021. 2

[10] 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. 2

[11] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4690–4699, 2019. 5

[12] Tan M Dinh, Anh Tuan Tran, Rang Nguyen, and Binh-Son Hua. Hyperinverter: Improving stylegan inversion via hypernetwork. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11389–11398, 2022. 1, 2, 3, 6, 7

[13] Rinon Gal, Or Patashnik, Haggai Maron, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022. 2

[14] Lore Goetschalckx, Alex Andonian, Aude Oliva, and Phillip Isola. Ganalyze: Toward visual definitions of cognitive image properties. In *Proceedings of the ieee/cvf international conference on computer vision*, pages 5744–5753, 2019. 2

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 1

[16] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. *Advances in neural information processing systems*, 33:9841–9850, 2020. 2, 6, 1

[17] 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. In *Neural Information Processing Systems*, 2017. 7

[18] Xueqi Hu, Qiusheng Huang, Zhengyi Shi, Siyuan Li, Changxin Gao, Li Sun, and Qingli Li. Style transformer for image inversion and editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11337–11346, 2022. 2, 6, 7, 8

[19] Ali Jahanian, Lucy Chai, and Phillip Isola. On the" steerability" of generative adversarial networks. *arXiv preprint arXiv:1907.07171*, 2019. 2

[20] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018. 6

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

[22] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33:12104–12114, 2020.

[23] 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.

[24] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34, 2021. 1, 2

[25] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013. 6

[26] Hongyu Liu, Yibing Song, and Qifeng Chen. Delving stylegan inversion for image editing: A foundation latent space viewpoint. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10072–10082, 2023. 2

[27] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF Inter-*

national Conference on Computer Vision, pages 2085–2094, 2021. 2, 6, 1

[28] Hamza Pehlivan, Yusuf Dalva, and Aysegul Dundar. Styleres: Transforming the residuals for real image editing with stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1828–1837, 2023. 1, 2, 3, 6, 7

[29] 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

[30] Antoine Plumerault, Hervé Le Borgne, and Céline Hudelot. Controlling generative models with continuous factors of variations. *arXiv preprint arXiv:2001.10238*, 2020. 2

[31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 2

[32] 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*, pages 2287–2296, 2021. 1, 2, 5, 6, 7

[33] Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *ACM Transactions on graphics (TOG)*, 42(1):1–13, 2022. 1, 2, 6, 7

[34] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1532–1540, 2021. 2

[35] 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. 2, 6, 1

[36] Nurit Spingarn-Eliezer, Ron Banner, and Tomer Michaeli. Gan" steerability" without optimization. *arXiv preprint arXiv:2012.05328*, 2020. 2

[37] Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhofer, and Christian Theobalt. Stylerig: Rigging stylegan for 3d control over portrait images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6142–6151, 2020. 2

[38] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation. *ACM Transactions on Graphics (TOG)*, 40(4): 1–14, 2021. 1, 2, 4, 5, 6, 7, 8

[39] Andrey Voynov and Artem Babenko. Unsupervised discovery of interpretable directions in the gan latent space. In *International conference on machine learning*, pages 9786–9796. PMLR, 2020. 2

[40] 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*, pages 11379–11388, 2022. 2, 3, 6, 7

[41] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, pages 1398–1402 Vol.2, 2003. 6

[42] Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12858–12867, 2020. 6, 1

[43] Weihao Xia, Yujiu Yang, Jing-Hao Xue, and Baoyuan Wu. Tedigan: Text-guided diverse face image generation and manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2256–2265, 2021. 2

[44] Xu Yao, Alasdair Newson, Yann Gousseau, and Pierre Hellier. Feature-style encoder for style-based gan inversion. *arXiv preprint arXiv:2202.02183*, 2022. 1, 2, 3, 6, 7, 8

[45] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 5, 6

[46] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *European conference on computer vision*, pages 592–608. Springer, 2020. 1, 2

[47] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pages 597–613. Springer, 2016. 1, 2

[48] Peihao Zhu, Rameen Abdal, John Femiani, and Peter Wonka. Barbershop: Gan-based image compositing using segmentation masks. *arXiv preprint arXiv:2106.01505*, 2021. 2