SN Computer Science (2023) 4:820
https://doi.org/10.1007/s42979-023-02272-4

ORIGINAL RESEARCH

Keep It Simple: Evaluating Local Search-Based Latent Space Editing

Andreas Meißner[1,2] · **Andreas Fröhlich**[1] · **Michaela Geierhos**[2]

Received: 3 May 2023 / Accepted: 23 August 2023

Abstract
Semantic image editing allows users to selectively change entire image attributes in a controlled manner with just a few clicks. Most approaches use a generative adversarial network (GAN) for this task to learn an appropriate latent space representation and attribute-specific transformations. Attribute entanglement has been a limiting factor for previous approaches to attribute manipulation. However, more recent approaches have made significant improvements in this regard using separate networks for attribute extraction. Iterative optimization algorithms based on backpropagation can be used to find attribute vectors with minimal entanglement, but this requires large amounts of GPU memory, can lead to training instability, and requires differentiable models. To circumvent these issues, we present a local search-based approach to latent space editing that achieves comparable performance to existing algorithms while avoiding the aforementioned drawbacks. We also introduce a new evaluation metric that is easier to interpret than previous metrics.

Keywords Semantic image editing · Latent space editing · StyleGAN · Evaluation metrics

Introduction

Semantic image editing allows users to selectively change entire image attributes in a controlled manner with just a few clicks, such as changing the age of a person in a portrait, the weather in a landscape image, or the color of certain objects in different scenes. Examples of face manipulation are shown in Fig. 1. Semantic image editing is a valuable tool for many real-world applications, including photo enhancement, artistic visualization, targeted data augmentation, and image animation. In most cases, the goal is to change specific attributes of the image while preserving its overall content and all other attributes.

There are two main categories of state-of-the-art methods for semantic image editing that rely on generative adversarial networks (GANs) [1]. The first group, known as image-to-image translation methods, uses GANs to map images from one domain to another [2–8]. However, these approaches are limited in that they restrict attribute changes to predefined factors rather than allowing unrestricted adjustments. The second group of methods, called latent space editing methods, involves using a GAN that generates images and searches for directions in its latent space to enable continuous semantic image editing. However, early GAN models were not optimized for a disentangled latent space, so modifying one attribute often led to unintended changes in other attributes [9]. Recent style-based approaches [9–11] have made significant improvements in disentangling attributes, allowing specific features to be targeted during image editing.

Latent space editing methods are further divided into two groups: supervised and unsupervised approaches. Unsupervised approaches do not rely on a labeled dataset or a regressor to specify the attribute to be manipulated [12, 13]. In contrast, supervised approaches require a labeled dataset or

This article is part of the topical collection "Advances on Computational Intelligence 2022" guest edited by Joaquim Filipe, Kevin Warwick, Janusz Kacprzyk, Thomas Bäck, Bas van Stein, Christian Wagner, Jonathan Garibaldi, H. K. Lam, Marie Cottrell and Faiyaz Doctor.

✉ Michaela Geierhos
michaela.geierhos@unibw.de

Andreas Meißner
andreas.meissner@zitis.bund.de

Andreas Fröhlich
andreas.froehlich@zitis.bund.de

[1] Big Data, ZITiS, Zamdorfer Straße 88, 81677 Munich, Bavaria, Germany

[2] Research Institute CODE, University of the Bundeswehr Munich, Werner-Heisenberg-Weg 39, 85577 Neubiberg, Bavaria, Germany

Published online: 21 October 2023

a regressor [14–18]. Although supervised methods have the advantage of defining the desired attribute for manipulation, the attribute vectors computed with earlier methods are often entangled with other attributes. Recent approaches such as StyleCLIP [16], StyleMC [17], and Enjoy Your Editing [18] aim to solve this problem by defining a loss function based on a deep learning model and iteratively optimizing a latent vector for the desired attributes using gradient descent. However, these approaches have the drawback that they require a significant amount of GPU memory for backpropagation, which can make them impractical even for high-end GPUs such as the Tesla-V100. For example, when trying to apply the approach introduced for Enjoy Your Editing [18] to the best model for image quality for StyleGAN3 [11], known as "stylegan3-r-ffhqu-1024x1024.pkl", a batch size of 1 requires a massive 39 GB of GPU memory, which is too much even for high-end GPUs like Tesla-V100. Moreover, only differentiable models can be used to compute the gradients, which limits the use of black-box models and increases the implementation overhead for models originally developed in other frameworks. We have also observed some instability issues when using smaller batch sizes.

Contributions: In our previous work [19], we proposed an iterative latent space editing method based on local search that achieves results comparable to those of Enjoy Your Editing [18] in terms of identity and attribute preservation while having a similar runtime. However, our approach requires significantly less GPU memory (only 12 GB for "stylegan3-r-ffhqu-1024×1024.pkl" as opposed to the 39 GB required by their approach), making it compatible with a wider range of GPUs. In addition, our method solves the problem of numerical instability and does not require a differentiable regressor. Our loss function is simpler and uses fewer hyperparameters. While sticking to the evaluation metric introduced by Zhuang et al. [18], we discussed some of its limitations and proposed a small extension of the metric that allows for better comparisons of different approaches. This extended version of our paper includes all previous results to be self-contained, but also extends our previous work [19] in two important directions: As a first new contribution, we evaluate our local search-based approach in StyleSpace and show that it indeed generalizes to different latent spaces. While pointing out possible difficulties in the transition to more general latent space representations, we also provide a solution that comes with a deeper understanding of the corresponding latent space structure. As a second new contribution, we further investigate the limitations of the original evaluation metric and propose further modifications, resulting in an easier-to-interpret evaluation metric.

Related Work

Semantic image editing has a long history in computer vision, computer graphics, and machine learning. In recent years, GANs have received particular attention for enabling efficient image manipulation through image-to-image translation or latent space editing.

Generative Adversarial Networks

GANs [1] have achieved impressive results in image generation in recent years [9, 20–22]. However, image generation is not the only application. Image inpainting [23, 24], super resolution [25, 26], data augmentation [27], and 3D object generation [28] are other areas of research.

A GAN typically consists of two modules: a generator and a discriminator. While the generator learns to generate fake samples based on a random distribution as input, the discriminator learns to distinguish between real and fake samples. In this way, a generator learns to reproduce the distribution of the training samples but does not provide control over the category or semantic attributes of the generated samples. By providing the generator with labels for each training sample, a conditional GAN can learn to generate samples based on the class—but this requires a labeled dataset [29].

With large-scale GAN models such as BigGAN [21] and StyleGAN [9], the generation of photorealistic images has become a reality in recent years. BigGAN [21] is a comprehensive GAN model trained on ImageNet [30] that supports image generation in multiple categories due to its conditional architecture. StyleGAN [9] is another popular GAN model in which the generator maps the random sampling distribution to an intermediate latent space, using a fully connected network (often referred to as a *mapping network*). In this approach, the intermediate latent space is not tied to the random distribution of the input, resulting in an

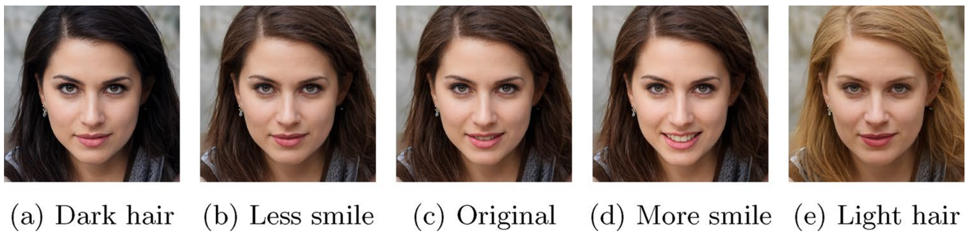


Fig. 1 Examples of semantic image editing related to manipulating facial attributes

automatically learned, unsupervised separation of high-level attributes.

Image-to-Image Translation

Image-to-image translation allows to transform one image domain into another, such as creating a drawing from a selfie [31, 32]. For example, pix2pix [33] learns this task in a supervised manner using cGANs [29]. It combines an adversarial loss with an L^1 loss to not only fool the discriminator but also to be close to the ground truth in the L^1 sense. The main drawback is that it requires paired data samples.

To circumvent the problem of obtaining paired data, unpaired image-to-image translation frameworks have been proposed [7, 31, 34]. CycleGAN [7] preserves key attributes between the input and the translated image using a cycle consistency loss.

However, all these methods are only able to learn the relationships between two different domains simultaneously. As a result, these approaches have limited scalability when processing multiple domains and cannot interpolate between the two domains.

Latent Space Editing

There have been many attempts to use the latent space of a pre-trained generator for image manipulation [35–37]. Some methods learn to perform end-to-end image manipulation by training a network that encodes a given image into a latent representation of the manipulated image [38–40].

Other techniques attempt to find latent paths in such a way that traversing them will result in the desired manipulation. These are divided into two classes:

(i) Supervised methods use either image annotations to find meaningful latent paths [15], or a pre-trained model that classifies image attributes [16–18]. The latter also allows for iterative optimization.
(ii) Unsupervised methods find reasonable directions without supervision, but require manual annotation for each direction afterwards [12, 13, 41].

In particular, the intermediate latent spaces in StyleGAN architectures [9–11] have been shown to facilitate many disentangled and meaningful image manipulations.

Many approaches perform image manipulations in the W space [12, 13, 15, 18], the more disentangled intermediate latent space generated directly by StyleGAN's mapping network [9]. The $W+$ space is an extension of the W space, where a different latent vector w is fed to each generator layer. While $W+$ was originally used to mix styles from different sources [9], it can also be used for semantic image editing [16]. StyleSpace S, the space spanned by the channel-wise style parameters, has also been proposed for latent space editing and was found to be even more disentangled than W and $W+$ [42]. StyleSpace is also used by StyleCLIP [16] and StyleMC [17].

Evaluation Metrics

The evaluation of semantic image editing approaches is typically based on qualitative assessments through sample results presented in publications. However, some publications use quantitative metrics to provide a fairer comparison. In recent studies [43, 44], a pre-trained attribute classifier was used to quantify changes in the target attribute, while the Frechet Inception Distance (FID) [45] and Kernel Inception Distance (KID) [46] were used to measure unwanted attribute changes. In another publication [47], the quality of the attribute change was evaluated based on the match between a given semantic label map and the generated output. Image similarity was also measured using FID and Learned Perceptual Image Patch Similarity (LPIPS) [48]. In addition, other works [49, 50] used standard image reconstruction measures such as mean absolute error, structural similarity index measure, and LPIPS to evaluate image manipulation. The use of semantic label maps as proposed metrics often assumes localized target attributes, which may not be appropriate for our approach, especially when a change in the target attribute affects multiple areas in the image. Furthermore, metrics such as FID or KID score may give more weight to changes in unwanted attributes such as the background of the image, rather than changes in the identity of a person, where even small variations in facial keypoints can have a significant impact on human perception. Therefore, the evaluation metric proposed by [18], which uses a pre-trained regressor to quantify changes in unwanted attributes and an InceptionResnet pre-trained on the VGGFace2-dataset [51] to measure changes in identity, is more appropriate for our approach.

Local Search-Based Latent Space Editing

We propose an iterative approach for controllable semantic image editing via latent space navigation in GANs called *LS-StyleEdit*. The process starts with a pre-trained generator G that receives as input a latent vector from a latent space. Given a target attribute, we try to find a vector in the latent space that, by adding it to the original latent vector, allows the target attribute to be changed while leaving other attributes intact.

Our approach for discovering an attribute-specific latent vector consists of two pre-trained networks G and R, and a local search component. While G and R are used to evaluate a given latent vector, the local search component provides an

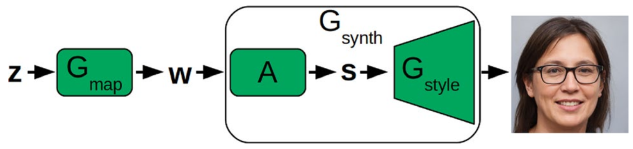


Fig. 2 Structure of StyleGAN2, which maps a normally distributed input $z \in Z$ to the intermediate latent spaces W and S

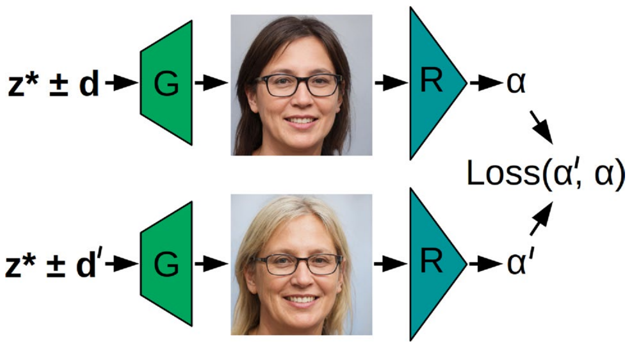


Fig. 3 Illustration of the steps that are performed in a single iteration of our local search-based optimization LS-StyleEdit

iterative framework for optimization by navigating through the latent space.

In practice, we used StyleGAN2 for our experiments in Sects. "Evaluation Challenges" and "Towards a New Evaluation Metric"—however, our overall approach is generic and not limited to this specific choice.

As discussed in Sect. "Related Work", StyleGAN architectures have several latent spaces that can be used to modify an attribute. In addition to the original input space Z, three different intermediate latent spaces W, $W+$, or S can be used: The Z space is normally distributed, but attributes are more entangled. The W space has less entanglement and, therefore, allows better control over a target attribute. It has been shown that the $W+$ space and the StyleSpace S are even less entangled [42].

Since the W space is often used to explore the latent space in StyleGAN, we decided to also use the W space, for an initial implementation in our previous work [19] to provide a fair comparison between our local search-based approach and existing methods. However, our approach is not restricted to W, but can operate on any latent space. While the W space has 512 parameters, the S space has 9088, which allows us to evaluate the scalability of our approach. Therefore, we also evaluate our approach in the S space as part of this extended work. As a consequence, we also present a more general algorithmic description of our approach that is independent of a specific latent space.

StyleGAN2's generator network consists of two consecutive parts: a mapping network G_{map} and a synthesis network G_{synth}. The input to G_{map} is a normally distributed latent vector z from the original latent space Z, which is then mapped to a new latent vector w in the intermediate latent space W. $W+$ is created by stacking the W vector 18 times and follows the same distribution as W. Subsequently, affine transformations A are applied to map the $W+$ space to the StyleSpace S, which has a different distribution than W. G_{style} is then used to generate an image using s as input. Figure 2 shows the aforementioned architecture of StyleGAN2.

In our approach, R is a regressor network pre-trained on the CelebA dataset [52] and estimates 40 attributes for the image. Similar to Enjoy Your Editing [18], our approach is iterative and R is used to directly compute a loss function in each iteration. To facilitate comparison, we decided to use their regressor model [18] to optimize the attribute vectors. The same regressor was also used for the evaluation in Sect. "Previous Evaluation Metric". The vector to be optimized, d, controls the attribute change in the image. Adding or subtracting d will increase or decrease the attribute in a given image—this is evaluated by the loss function. Figure 3 illustrates a single iteration within this optimization framework. Since our goal is to present a versatile approach that can be used with any latent space, we will use the notation Z^* as a non-specific latent space, such as Z, W, or S. Similarly, we will use z^* to refer to its elements.

The main novelty of our approach is the use of a local search component to optimize d. In contrast, most other iterative approaches [16–18] use backpropagation for their optimization. While backpropagation is a powerful tool for many deep learning applications, its performance comes at the cost of high memory consumption and computational cost. Compared to other applications, such as training the weights of a deep learning network, our task is less complex, requiring only the optimization of the attribute vector; the weights of G and R remain unchanged. Local search provides a simple but efficient framework for this type of optimization task. Starting from an initial point in a search space, local search algorithms iteratively move to "better" points according to an objective function using heuristics. While local search is mainly applied to computationally intensive optimization problems in discrete search spaces, there are also methods for real-valued search spaces. In particular, our local search component is based on the concept of random optimization [53] and is described in Algorithm 1.

Algorithm 1 Local search of LS-StyleEdit

```
    Input: sampleRadius $r$, maxLength $L$
 1: $d \leftarrow \vec{0}$
 2: for $i = 0, ..., \max$ do
 3:     $z^* \leftarrow \Theta_{Z^*}$
 4:     $\pm \leftarrow rand\{+, -\}$
 5:     $\alpha \leftarrow R(G(z^* \pm d))$
 6:     $d' \leftarrow d + r \cdot \Sigma_{Z^*} \odot \mathcal{N}(\vec{0}, I)$
 7:     if $l_{Z^*}(d') > L$ then
 8:         $d' \leftarrow d' \cdot L / l_{Z^*}(d')$
 9:     end if
10:     $\alpha' \leftarrow R(G(z^* \pm d'))$
11:     if $\pm\alpha < \pm\alpha'$ then
12:         $d \leftarrow d'$
13:     end if
14: end for
```

As hyperparameters, our algorithm requires a sample radius r and a maximum length L, both of which restrict the choice of our attribute vector d.

We initialize d to be a null vector before entering the main loop (see line 1). In each iteration, we first sample a new latent vector, which is the origin for the current local search step (see line 3). Since the different latent spaces follow different distributions, we use Θ_{Z^*} to denote a general sampling function that returns a random latent vector according to the corresponding distribution of Z^*. While the latent vectors in Z are normally distributed with standard deviation $\sigma = 1$ in all dimensions, the distribution of the latent vectors in W and S is unknown. In practice, the intermediate latent spaces are not sampled directly, but the latent vectors for W and S are obtained by first sampling from Z and applying the corresponding mapping functions G_{map} and $A \circ G_{map}$, respectively. Since G_{map} and A are implemented using complex networks, the distributions of Z, W, and S are expected to differ significantly. We then sample a sign ($+$ or $-$, each with probability 0.5) to decide whether to evaluate the attribute vector in terms of its ability to increase or decrease the target attribute in the current iteration (see line 4). A manipulated latent vector is obtained by adding or subtracting d depending on the chosen sign; G is used to generate the corresponding manipulated image, and R provides a value α to estimate the degree to which the target attribute is present (see line 5). The actual local search space is then sampled by adding a non-uniformly scaled noise vector to d, resulting in a new candidate attribute vector d' (see line 6). This differs from our original implementation [19] in that we scale each dimension of the normally sampled vector not only by the scalar parameter r, but also individually by performing an element-wise multiplication $\odot$ with Σ_{Z^*} to account for the different distributions of the latent spaces. In particular, we use Σ_{Z^*} to denote the vector of standard deviations of the corresponding dimensions of Z^*. As the next step in our algorithm, we compute the length of the new candidate attribute vector d' using l_{Z^*}. Similar to sampling the noise vector, this part extends our original implementation [19] by taking into account the distribution of the latent space Z^*. To facilitate this, we define l_{Z^*} to represent the element-wise partition by Σ_{Z^*} followed by the application of the L^2 norm. If this length exceeds the previously defined maximum length of L, d' is scaled down accordingly to avoid reaching too sparsely sampled parts of the latent space (see lines 7–9).

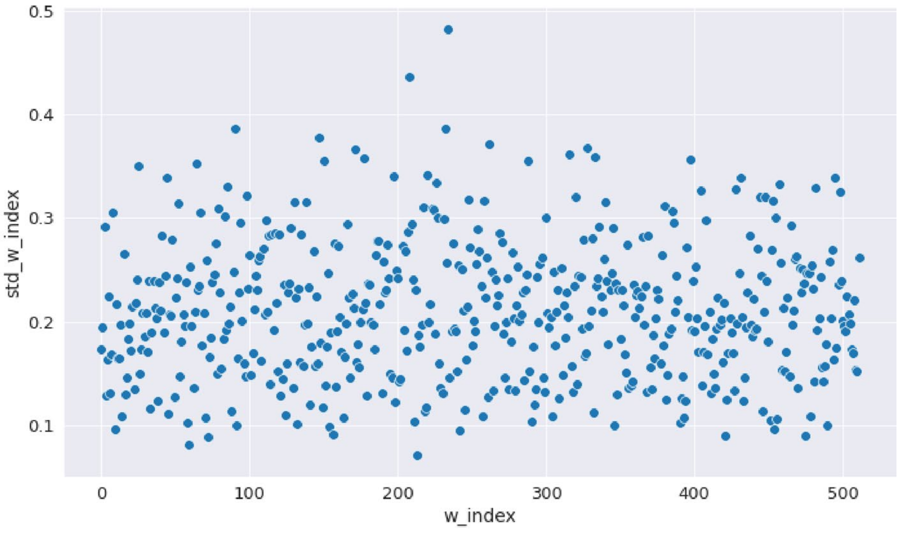
To better understand the underlying intuition of scaling the vectors as part of our refined algorithm, the standard deviations in W and S can be observed in Fig. 4. In particular, Fig. 4a shows that the standard deviation for each element is no longer uniformly 1 as in Z space, but instead ranges from 0.07 to 0.48. This difference is even more pronounced in StyleSpace, as shown in Fig. 4b, where the standard deviation of individual elements ranges from 0.03 up to 8.04. In the toRGB layers, the data points near 0, e.g., s_index=512–1024, have comparatively small variations, while the convolutional layers have a much higher variation than the latent vector in W space.

In our previous work [19], we implicitly assumed a standard deviation of 1 for all dimensions when modifying the target attribute vector d within a local search step and when measuring its length, even though d was applied in W space. While this approach was effective—though probably not optimal—for W due to the relatively small gap in standard deviations between dimensions, we realized that the strong non-uniformity of S posed a problem for the algorithm in its original shape. As a result, we adopted our generalized approach for accounting for the difference in standard deviations over the individual elements of the latent vectors by scaling the noise values and measuring the lengths accordingly.

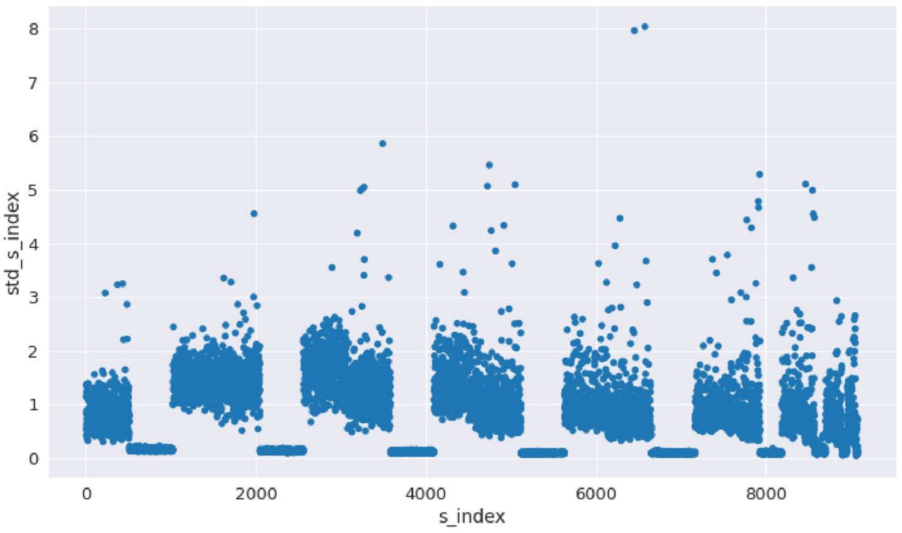
In the same way as α was determined for d, a new value α' is now calculated for d' using G and R (see line 10). d' is considered better than d if (i) $\alpha' > \alpha$ and the attribute vectors were evaluated according to their positive direction, or (ii) $\alpha' < \alpha$ and the attribute vectors were evaluated according to their negative direction. If this is the case, d is updated to the value of d' (see lines 11–13).

We decided to keep our optimization criterion as simple as possible. We used only α and α' to evaluate the attribute vectors d and d', respectively. Therefore, our objective function can be computed using only a regressor loss. In contrast, StyleCLIP [16] uses a loss term based on the CLIP model [54], L^2 distance between latent vectors, and an identity loss based on a pre-trained ArcFace model [55].

Fig. 4 Comparison of the standard deviations of the latent vector dimensions in the *W* space and in the StyleSpace *S*



(a) Standard deviation of each feature in W space over 1,000 images.



(b) Standard deviation of each feature in S space over 1,000 images.

Similarly, Enjoy Your Editing [18] uses a regressor loss, a content loss based on a VGG model [56], and an additional discriminator loss. The discriminator loss is intended to measure the quality of the generated images. Since StyleCLIP [16] has no visible artifacts and contains no discriminator loss, we assume that the latter is not necessary to produce realistic images. We also expect that content loss, identity loss, and L^2 distance mainly limit the maximum length of the attribute vector during optimization. This leads to the hypothesis that a vector of predefined length, which is then optimized to modify a target attribute as much as possible, will automatically preserve the remaining attributes and the identity of the person due to the disentanglement properties of the underlying latent space. The length of the attribute vector can be interpreted as a hyperparameter. Since StyleGAN2 produces high-quality images near the center of the input distribution, a sufficiently small length limits the amount of artifacts. Both, StyleCLIP [16] and Enjoy Your Editing [18], use three hyperparameters in their respective loss functions, which requires careful balancing.

Evaluation Challenges

Since Zhuang et al. [18] proposed the method that is most similar to our approach, we initially decided to use their work as a baseline for evaluating the performance of LS-StyleEdit. In this section, we present the evaluation setup as used in our previous work [19] and point out the challenges that we faced in trying to facilitate a fair comparison.

Reimplementation of Enjoy Your Editing

Unfortunately, we encountered some strange behavior while testing their StyleGAN2 implementation.[1] We observed some sporadic runtime errors due to a compatibility issue between CUDNN and the NVIDIA driver version, as well as significant variations in the output results. When using the same input multiple times with constant noise, the output images sometimes differed. While most output images were nearly identical, average pixel differences of up to 9.06 were occasionally observed in a 0–255 image. This pixel difference resulted in prediction differences of up to 14.4% from the regressor, which severely limited our ability to consistently reproduce the results. As a result, we reimplemented their approach using NVIDIA's official StyleGAN2-ADA-PyTorch implementation.[2] While the StyleGAN2 implementation of Enjoy Your Editing generates images with a size of 256x256 pixels, we used StyleGAN's FFHQ model, which provides a resolution of 1024x1024 pixels, since most applications use the best possible image quality.

For all experiments in our previous work [19], which are also presented in Table 2 and Sect. "Quantifying Instabilities of Enjoy Your Editing" of this extended paper, we used the settings $r = 3 \cdot 10^{-4}$ and $L = 0.8$ for our algorithm. As already mentioned in Sect. "Local Search-based Latent Space Editing", in our initial implementation we did not take into account the difference in standard deviations between latent vector dimensions, i.e., instead of Σ_{Z^*} we used I for sampling possible attribute vectors d during local search (cf. Algorithm 1). As suggested by Zhuang et al. [18], we used the regressor loss coefficient $\lambda_1 = 10$, the content loss coefficient $\lambda_2 = 0.05$, and the discriminator loss coefficient $\lambda_3 = 0.05$ for our reimplementation of Enjoy Your Editing. For optimization, an Adam optimizer with a learning rate of 10^{-4} was used.

Both, the initial implementation of our local search-based algorithm and the reimplementation of Enjoy Your Editing, are available in our GitHub repository.[3] It also contains the evaluation scripts used in our experiments.

Quantifying Instabilities of Enjoy Your Editing

In the original implementation of Enjoy Your Editing [18], StyleGAN2 images have a resolution of 256x256 pixels, which allows the use of larger batch sizes compared to 1024x1024 models. Larger models, such as those used by StyleGAN3 [11], require even more GPU memory, further limiting the viable batch size. To investigate the impact of using smaller batch sizes on training stability, we ran our reimplementation of Enjoy your Editing for 20,000 iterations with 10 different random seeds and checked how often numerical instabilities (i.e., NaN values in the attribute vector) occurred.

1. In the first experiment, we performed 10 runs for StyleGAN3, using its largest model "stylegan3-r-ffhqu-1024x1024.pkl" with a batch size of 1 and a learning rate of 10^{-4}. All 10 runs ended up with numerical instabilities.
2. In the second experiment, we investigated the influence of batch size on the stability of Enjoy Your Editing. Since "stylegan3-r-ffhqu-1024×1024.pkl" requires 39 GB of GPU memory at a batch size of 1, we decided to use StyleGAN2's 1024×1024-ffhq-model—which we used in all subsequent experiments—to test larger batch sizes. For a batch size of 1 and a learning rate of 10^{-4}, 7/10 runs ended in numerical instability. For batch sizes of 2, 4, and 8, 2/10 runs also ended in numerical instability. Thus, while training stability improved with batch sizes larger than 1, numerical instabilities were still observed for a batch size of 8. Since instabilities occurred with both StyleGAN2 and StyleGAN3, this suggests that the instability problem is not a model-specific effect, but is caused by the underlying approach.
3. In the third experiment, we examined the influence of the learning rate. While 7/10 runs ended in numerical instability at a learning rate of 10^{-4}, only 4/10 runs did so at a learning rate of 10^{-5}.

We traced the cause of the numerical instabilities to the regressor loss, which uses a binary cross-entropy (BCE) function:

$$L_{reg} = \mathbb{E}[-\hat{\alpha}' \log \alpha' - (1 - \hat{\alpha}') \log (1 - \alpha')]. \tag{1}$$

If α' is close to 0 and 1, the terms $\log(\alpha')$ and $\log(1 - \alpha')$ take on very large values, respectively. These terms often cannot be compensated by $\hat{\alpha}'$ and $(1 - \hat{\alpha}')$. This high loss leads to large gradients, which can be traced back to the output layer of StyleGAN2, where the first NaN values appear. We observed that switching from BCE to a mean squared error (MSE) function seems to be a possible way to avoid these instabilities. When using an MSE-based loss, no NaN values occurred in our experiments and the visual quality of the edited images remained the same. However, this was just a first impression and we did not perform a full experimental evaluation using MSE-based loss, as this was outside the scope of our work. While inspecting the GitHub implementation of Enjoy Your Editing, we found some differences

[1] https://github.com/KelestZ/Latent2im.
[2] https://github.com/NVlabs/stylegan2-ada-pytorch.
[3] https://github.com/meissnerA/LocalSearchLSpaceE.

Table 1 Influence of the vector length on the evaluation metric for the target attribute "Smiling" (taken from [19])

Smiling	Attribute preservation			Identity preservation			Buckets		
	(0, 0.3]	(0.3, 0.6]	(0.6, 0.9]	(0, 0.3]	(0.3, 0.6]	(0.6, 0.9]	(0, 0.3]	(0.3, 0.6]	(0.6, 0.9]
$\|\|d\|\|$	0.0268 ±0.0671	0.0669 ±0.1336	0.0980 ±0.1866	0.9990 ±0.0022	0.9976 ±0.0032	0.9964 ±0.0039	5442	1370	2309
$d/5$	0.0114 ±0.0333	0.0405 ±0.0968	0.0599 ±0.1396	0.9998 ±0.0005	0.9995 ±0.0008	0.9993 ±0.0007	9563	410	27

The rows show results for the vector computed by our approach and a scaled version of it. In terms of preservation metrics, the short vector performs better than the original one. However, the manipulation of target attributes is reduced

compared to the pseudocode provided in their paper [18]. In particular, one difference relates to the sampling of a random value ϵ, which is then used to compute α' for their BCE loss. While we decided to base our reimplementation on their official paper, it is possible that using the sampling distribution from their GitHub implementation would also reduce instabilities. However, both of these possible solutions highlight the well-known fact that backpropagation is sensitive to the careful choice of many hyperparameters, such as the loss function and learning rate. Moreover, even without numerical instabilities, backpropagation-based approaches still have the disadvantage of requiring differentiable models and large amounts of GPU memory. Local search can provide a simple framework to circumvent these difficulties in the context of latent space editing.

Previous Evaluation Metric

To evaluate the attribute values, we originally used the evaluation metric by Zhuang et al. [18] to show that our local search-based approach can compete with other methods on an already established evaluation metric and not just by customizing a new evaluation metric to our advantage [19]. We generated 1,000 original images, produced 10,000 edited images with different levels of editing, and calculated the difference in the target attribute between the original images and their respective edited images. Depending on the degree of change in the target attribute, an image pair is stored in one of the three buckets ((0, 0.3], (0.3, 0.6], or (0.6, 0.9]). Two different metrics are calculated for each bucket:

(i) Identity preservation is computed using the popular image identity recognition model VGGFace2, which is pre-trained on the VGGface2 dataset [57]. When VGGFace2 is applied to a face image, it outputs a feature vector. Identity preservation is the cosine similarity between the face feature vector of the original image and the edited image averaged over all image pairs.

(ii) The attribute preservation metric is computed by averaging over a set of facial attributes estimated by a regressor network. In particular, we initially used the same pre-trained regressor network that was used to estimate the target attribute [52]. We computed the 40 attribute predictions for all original images and all edited images. Ideally, editing only changes the target attribute and all other attributes remain the same. Therefore, the average change in all attributes except the target attribute is used. The attribute preservation metric is the average attribute difference across all image pairs.

Unfortunately, we encountered a problem with the described metric: Short attribute vectors tend to give significantly better results than longer ones. This is not surprising, since the length of the attribute vector directly affects the distance between the latent vectors for the original image and the manipulated image. For example, using a null vector does not change the image at all. As a result, a null vector achieves perfect identity and attribute preservation scores. A similar effect can be observed for any sufficiently short non-zero vector.

In Table 1, both preservation metrics are computed for an attribute vector found by our approach and a downscaled version of it. The downscaled version appears to perform better than the original attribute vector when no additional criterion is used for evaluation. In practice, a good attribute vector must preserve the image content while changing the target attribute as much as possible, both at the same time. This trade-off is strongly influenced by the length of the vector. In particular, downscaling not only improves the preservation metrics, but also reduces the manipulation of the target attribute. As a result, evaluating only the preservation component turns out to be insufficient. To address this shortcoming, we also include the bucket distribution in our evaluation. The bucket distribution is an indication of the degree of change with respect to the target attribute.

Table 2 Comparison (taken from [19]) of attribute preservation (a lower score is better) and identity preservation (a higher score is better) for the algorithm by [15] (Shen), our reimplementation of Enjoy your Editing (Zhuang), and our local search-based approach (with batch size=1 and batch size=8) after scaling the vectors to cause the same degree of target attribute change

	Attribute Preservation			Identity Preservation			Buckets			d
	(0, 0.3]	(0.3, 0.6]	(0.6, 0.9]	(0, 0.3]	(0.3, 0.6]	(0.6, 0.9]	(0, 0.3]	(0.3, 0.6]	(0.6, 0.9]	
Smiling										
Shen	0.0264 ±0.0659	0.0657 ±0.1319	0.0977 ±0.1875	0.9988 ±0.0027	0.9974 ±0.0034	0.9956 ±0.0047	5429	1330	2307	1.31
Zhuang	0.0300 ±0.0739	0.0718 ±0.1393	0.1020 ±0.1887	0.9991 ±0.0020	0.9979 ±0.0026	0.9966 ±0.0038	5416	1418	2320	1.32
Ours bs=1	0.0268 ±0.0671	0.0669 ±0.1336	0.0980 ±0.1866	0.9990 ±0.0022	0.9976 ±0.0032	0.9964 ±0.0039	5442	1370	2309	1.40
Ours bs=8	0.0252 ±0.0628	0.0641 ±0.1300	0.0958 ±0.1855	0.9990 ±0.0022	0.9976 ±0.0034	0.9963 ±0.0039	5426	1338	2315	1.30
Hair color										
Shen	0.0429 ±0.1008	0.0789 ±0.1372	0.0988 ±0.1744	0.9851 ±0.0229	0.9542 ±0.0346	0.9370 ±0.0430	5428	1122	1520	2.44
Zhuang	0.0399 ±0.0967	0.0745 ±0.1317	0.0936 ±0.1700	0.9869 ±0.0197	0.9543 ±0.0347	0.9357 ±0.0431	5395	1279	1689	2.01
Ours bs=1	0.0447 ±0.1003	0.0880 ±0.1461	0.1093 ±0.1829	0.9814 ±0.0283	0.9409 ±0.0449	0.9201 ±0.0531	5380	1134	1447	3.20
Ours bs=8	0.0452 ±0.1047	0.0842 ±0.1479	0.1030 ±0.1800	0.9849 ±0.0233	0.9538 ±0.0358	0.9396 ±0.0412	5345	1129	1536	2.79

While the bucket distribution is similar after scaling, the resulting length of the attribute vectors can be different. The first four rows show metrics for the “Smiling” attribute, the last four rows show metrics for the “Hair color” attribute

While highlighting an important aspect, the bucket distribution does not automatically allow for a direct ranking of different algorithms. Due to the strong negative correlation between target attribute change and preservation metrics, approaches tend to be better at one or the other. A naive attempt to overcome this limitation could be to normalize the attribute vectors before evaluation. Unfortunately, this turns out to be insufficient. Even small variations in an algorithm, e.g., a different random seed or a different batch size, lead to different latent vectors. As part of our evaluation, we found that different attribute vectors require different lengths for the same degree of attribute editing. This is not surprising since W does not follow a known distribution. To solve this problem, we proposed to scale attribute vectors so that they change the target attribute by the same amount.

However, the target attribute change is influenced by several aspects and there is no simple measure. As a result, we initially decided to approximate the target attribute change by the number of samples with an attribute change of at most 0.3, which roughly corresponds to the samples in bucket (0, 0.3]. When implementing the scaling of the vector, we wondered what range we should use as the measure of attribute change. Values greater than 0.9 are not represented in the buckets, but an attribute vector that changes the target attribute by more than 0.9 should be considered when determining the scaling factor. Therefore, we decided to scale the vectors so that the number of samples with an attribute change of at most 0.3 is within ±1%. This means that the number of samples with an attribute change of more than 0.3 is also within ±1%. In total, we ran our reimplementation of Enjoy Your Editing with a batch size of 1 for 20,000 iterations, resulting in a bucket distribution of [5416, 1418, 2320] for the attribute “Smiling”, and scaled all latent vectors in our experiments for the same attribute so that bucket $(0, 0.3] = 5416 \pm 54$. The results of this evaluation, which correspond to the main results of our previous work [19], are shown in Table 2.

To have comparable runtimes, we also used this run as a reference, which took 4105 s on a NVIDIA Quadro GV100, and stopped each run after that time. In the original implementation of Enjoy Your Editing, d is initialized with a random distribution. However, this random initialization affects the performance of the computed attribute vector. For reasons of reproducibility, we have initialized the attribute vector in Enjoy Your Editing with a null vector. Since the loss networks use pre-trained weights, this does not negatively affect performance.

Towards a New Evaluation Metric

The primary focus of our previous work [19] was to introduce a local search-based algorithm for latent space editing. We explicitly tried to avoid any changes to any components that were not specific to our approach. In particular, we stuck to the evaluation metric defined by Zhuang et al. [18] and also used their regressor model for estimating facial attributes to ensure a fair comparison and eliminate any potential bias that might favor our method. Having shown that our approach can indeed compete with existing algorithms, we can lift the self-imposed restriction for the extensions presented here.

As the main novelty within this section, we propose a new evaluation metric for facial attribute manipulation that addresses several challenges currently present in attribute vector evaluation while aiming to be easier to interpret. Similar to the metric introduced by Zhuang et al. [18], a pre-trained regressor model will still be part of the backbone of our metric. However, since we were missing some crucial details about how their regressor was trained, we decided to train our own regressor for attribute classification. To minimize the influence of JPEG compression artifacts, we trained our model using a ResNet50 on the PNG images of the CelebA dataset [52]. We trained our model for five epochs and obtained an accuracy of 0.92. The details of our hyperparameters and preprocessing steps during training are available in our GitHub repository.[4] The regressor is used in all experiments related to the new evaluation metric. However, we made a conscious decision not to use the regressor to compute the attribute vectors. Instead, we used the regressor provided by Zhuang et al. [18]. It was deemed inappropriate to optimize the attribute vectors on the same model that they are evaluated on, since StyleMC [17] uses a CLIP model to compute the attribute values, which would give an unfair advantage to the other algorithms.

Influence of the Initial Image

Although normalizing the vector length based on one bucket is a step toward normalization, it may not be the best solution. Comparing two attribute vectors that differ significantly in the other two buckets is not a straightforward process and leaves room for interpretation as to which attribute vector might be better. To have a metric with less room for interpretation, we propose to normalize the attribute vectors so that they produce the same average change in the target attribute over a given number of images. Furthermore, instead of calculating attribute and identity preservation for each of the three buckets, we calculate the average over all samples. This approach provides a standardized attribute and identity preservation metric that allows us to compare our results using only two numbers instead of the previous six plus the bucket distribution. Depending on the specific use case, these two metrics could even be combined into a weighted sum that reflects the relative importance of preserving identity or avoiding unwanted attribute changes.

However, there is an open question that needs to be discussed: What is the appropriate amount by which the target attribute should change, on average, to allow for a fair comparison? To establish a predetermined range of values, it may be advantageous to use a sigmoid function for the target attribute change. However, because the slope of the sigmoid decreases as the value moves away from 0, we hypothesized that the same attribute vector might have different manipulation strengths depending on the source attribute value.

To test our hypothesis, we generated 10,000 random images and used our regressor to determine the attribute value of each image. We then calculated the corresponding sigmoid for each image. These images were then divided into buckets ranging from 0.0–0.1 to 0.9–1.0 based on their sigmoid values. For the purposes of this experiment, we chose "Smiling" as the target attribute. Since the generated images had a bias towards high values of smiling, the majority of the images fell into the 0.9–1.0 bucket. To ensure that the standard deviations were not biased by the different number of samples within each bucket, we randomly selected 226 samples from each bucket to match the number of samples in the smallest bucket.

For each image, we added a given attribute vector for the attribute "Smiling" and computed the difference between the initial output of the regressor and the adjusted attribute value. We also calculated the attribute change in the negative direction of the attribute vector. Based on the attribute changes, we calculated the minimum and maximum attribute change for each bucket, as well as the average attribute change and standard deviation, which are presented in Table 3. When examining the minimum and maximum values, we observed a large possible range across all buckets.

Since the same attribute vector resulted in significantly different attribute changes, our first conclusion is that an evaluation metric must average over a sufficient number of samples.

Shifting the focus to the averages reveals a second trend, where the average attribute change in the positive direction decreases steadily starting from the third row (0.7389). This can also be observed in the negative direction, where the average attribute change decreases from −0.7173 to −0.0306. Since the sigmoid function limits the output to the range from 0.0 to 1.0, it is evident that an initial attribute value of 0.7, for example, limits the attribute change to a maximum of 0.3 in the positive direction. Our second conclusion is that an evaluation metric can benefit from changing an attribute in a positive direction for initial attribute values between 0.0 and 0.5, and in a negative direction for initial attribute vectors between 0.5 and 1.0.

The third notable observation is that buckets 0.0–0.1 and 0.9–1.0 are different from the others in that they have a higher standard deviation compared to the average attribute change. Although all other buckets show a consistently decreasing average attribute change in the positive direction,

[4] https://github.com/meissnerA/LocalSearchLSpaceE2.

Table 3 Influence of the attribute prediction origin on the attribute change for the attribute "Smiling"

Attr. value	Positive direction		Negative direction	
	min *max*	*mean* *std*	*min* *max*	*mean* *std*
0.0–0.1	0.0010 0.9390	0.4404 0.3047	−0.0948 0.0030	−0.0306 0.0249
0.1–0.2	0.1834 0.8912	0.7389 0.1199	−0.1913 −0.0725	−0.1268 0.0274
0.2–0.3	0.3406 0.7940	0.6892 0.0795	−0.2871 −0.1492	−0.2234 0.0308
0.3–0.4	0.3987 0.6934	0.6110 0.0530	−0.3885 −0.1986	−0.3165 0.0358
0.4–0.5	0.2478 0.5978	0.5195 0.0482	−0.4831 −0.2887	−0.4033 0.0432
0.5–0.6	0.3253 0.4968	0.4340 0.0365	−0.5846 −0.2956	−0.4932 0.0501
0.6–0.7	0.2415 0.3967	0.3350 0.0310	−0.6823 −0.3264	−0.5739 0.0625
0.7–0.8	0.1675 0.2977	0.2418 0.0296	−0.7777 −0.3510	−0.6565 0.0756
0.8–0.9	0.0733 0.1993	0.1431 0.0309	−0.8769 −0.2817	−0.7173 0.1106
0.9–1.0	−0.0004 0.0905	0.0120 0.0213	−0.9459 0.0001	−0.2693 0.3172

The left column shows the value of the attribute prediction origin. In the second and third columns the attribute was changed in a positive direction, in the last two columns the negative attribute vector was used. The min/max columns show the minimum and maximum attribute change, mean and std show the average attribute change and standard deviation

the 0.0−0.1 bucket has a smaller value than the 0.1−0.2 bucket, and in the negative direction, the −0.2693 attribute change is smaller than the −0.7173 attribute change. We found it particularly interesting that a positive attribute manipulation resulted in a negative attribute change for the 0.9−1.0 bucket, and in the negative direction there were positive attribute changes (0.003 and 0.0001) in the first and last buckets, respectively. Therefore, we conclude that the ranges between 0.0−0.1 and 0.9−1.0 may pose a challenge for a reliable evaluation metric.

Influence of the Attribute Manipulation Strength

After observing the strange behavior of the sigmoid ranges between 0.0−0.1 and 0.9−1.0 with respect to attribute changes, we decided to investigate this behavior further. We created 1000 images and added our smile attribute vector scaled by a factor of $l\_vec\_coeff \in [-6, \ldots, 6]$. Since the sigmoid function squeezes the range of values, we used the logit (the regressor output before applying the sigmoid) of the regressor output. Figure 5a shows the regressor logit over the manipulation strength, where the blue dots represent the average of all 1000 images and the orange dots represent the regressor logit of one sample. We notice that the regressor logit continuously increases until l_vec_coeff reaches a value of about 2, and from there the regressor power slowly decreases. We conclude that the smiling attribute increases in a natural way from 0 to 2. Figure 5b shows the image that reached the maximum regressor output at l_vec_coeff=2.04. However, if the attribute vector tries to increase the smile beyond that, the logit decreases because the regressor has not seen such extreme cases in the training data. The example for $l\_vec\_coeff = 6.0$ is shown in Fig. 5c. In fact, such examples may seem unnatural to a human observer and are not present in significant numbers in the CelebA dataset.

Our observation also clarifies the reason for the negative attribute change in the positive direction of Table 1 and a positive attribute change in the negative direction. Therefore, we recommend to refrain from evaluating in these extreme areas where the regressor has not been properly trained and limit the length of the attribute vector accordingly. Even if we limit the length of the attribute vector to a reasonable amount, the direction of the attribute vector is important. If the initial random image is already smiling and we try to increase the attribute further, we might as well end up in the "unnatural region" as shown in Fig. 5c. Thus, when evaluating the influence of an attribute vector, we suggest that it is beneficial to first compute the regressor output of the initial image. If the logit value is < 0, change the attribute in the positive direction, otherwise in the negative direction.

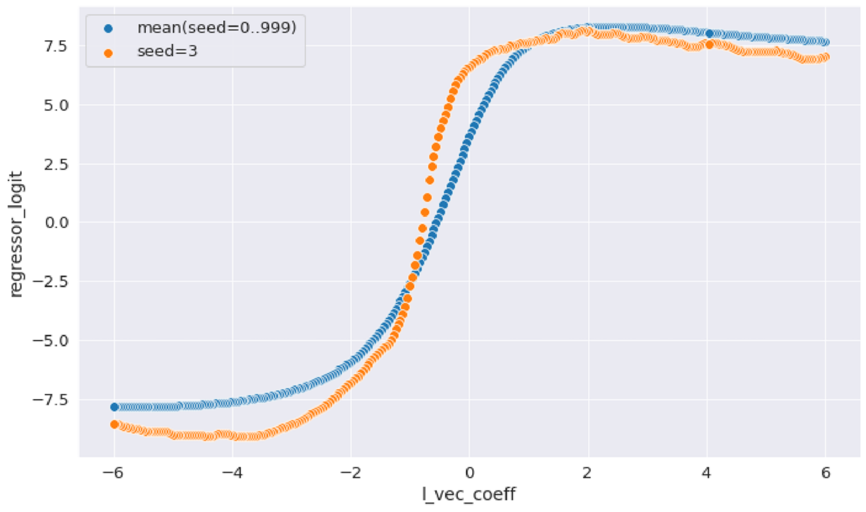
An appropriate evaluation metric must not only include the change in the target attribute, but also quantify how much any undesired features have changed. In the case of facial images, changing one attribute, such as smiling, should change as little as possible all other attributes. To achieve this, we use the attribute preservation metric introduced by Zhuang et al. [18]. Since the attributes alone are not enough to fully represent a person's identity, we additionally measure the degree of identity preservation by calculating the cosine distance between the embeddings of the original and modified images computed using VGGFace2.

As before, we created 1,000 images and added the attribute vector for smiling scaled by $l\_vec\_coeff \in [-6, \ldots, 6]$. We plotted the attribute and identity preservation in Fig. 6. Both graphs are monotonically increasing with increasing distance from 0. Although the identity preservation shows almost linear behavior, the attribute preservation does not. Therefore, we cannot rely on a fixed coefficient between attribute modification and attribute preservation, and we must evaluate different amounts of attribute modification.

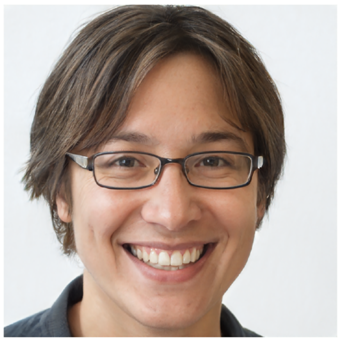
New Evaluation Metric

As previously stated, an attribute vector should only be scaled to remain within the range where the regressor has

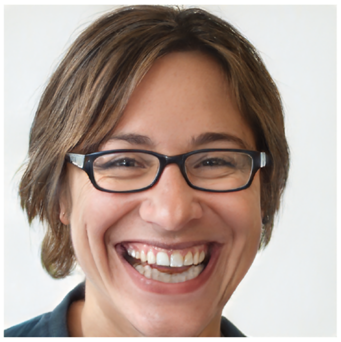
Fig. 5 Impact of overmodulating an attribute in areas where the regressor was not trained



(a) Regressor logit (output before applying the sigmoid function) over the scaling of the latent vector. *l_vec_coeff* is sampled from [-6,. . . ,6] with a step size of 0.04.

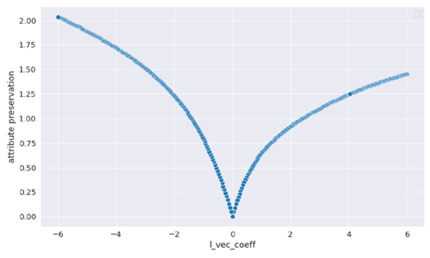


(b) Maximum output of the regressor at $l\_vec\_coeff = 2.04$.

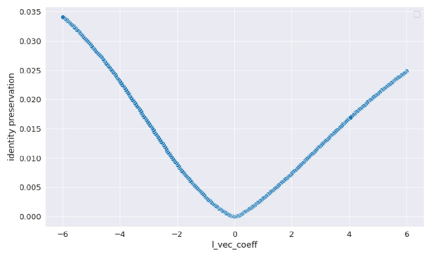


(c) Overmodulated smile at $l\_vec\_coeff = 6$.

Fig. 6 Attribute and identity preservation by scaling the latent vector for smiling. *l_vec_coeff* is sampled from [−6,...,6] with a step size of 0.04. The preservation values are averaged over 1000 images



(a) Attribute preservation. Can only be approximated linearly for small attribute changes.



(b) Identity preservation. Behaves nearly linearly over the full range of attribute changes.

been appropriately trained. The precise scaling factor may vary depending on the attribute and regressor used; however, it is reasonable to assume that a regressor is more reliable in the range of 0.1–0.9 than values close to 0 or 1. In this range, identity and attribute preservation also tend to behave more predictable. Furthermore, considering the argument presented earlier, if the prediction is less than 0.5, the attribute should be increased; otherwise, it should be decreased. Consequently, we aim to scale the attribute vector in such a way that it changes the attribute by an average of ± 0.4. Several methods can be used to rescale the attribute vector to achieve the desired attribute change, and one possible approach is presented in Algorithm 2. This algorithm iteratively tries to find a scaling factor that achieves the given target attribute change on average over *N* representative images. To account for outliers, we must iterate over a sufficient number of images, as described in the previous parts of this section. ϵ determines the acceptable difference between the desired average attribute change and the calculated average attribute change for the current scaling factor. Additionally, to avoid an infinite loop, the optimization is limited to a maximum of *max* iterations.

Algorithm 2 Get scaling factor

```
    Input: scaling_factor, d, goal_attr_change, N, stepsize
1:  scaling_direction_flag ← 0
2:  for $i = 0, ..., max$ do
3:      $\alpha\_list = []$
4:      for $random\_seed = 0, ..., N$ do
5:          $z^* \leftarrow \Theta_{Z^*}$
6:          $\alpha_{orig} \leftarrow R(G(z^*))$
7:          if $\alpha_{orig} > 0$ then
8:              $\alpha_{delta} \leftarrow \alpha_{orig} - R(G(z^* - scaling\_factor \cdot d))$
9:          else
10:             $\alpha_{delta} \leftarrow R(G(z^* + scaling\_factor \cdot d)) - \alpha_{orig}$
11:         end if
12:         $\alpha\_list.append(\alpha_{delta})$
13:     end for
14:     if $\alpha\_list.mean() - goal\_attr\_change < \epsilon$ then
15:         return scaling_factor
16:     end if
17:     if $\alpha\_list.mean() < goal\_attr\_change$ then
18:         scaling_factor ← scaling_factor + stepsize
19:         if scaling_direction_flag == $-1$ then
20:             stepsize ← stepsize/2
21:         end if
22:         scaling_direction_flag ← 1
23:     else
24:         scaling_factor ← scaling_factor − stepsize
25:         if scaling_direction_flag == 1 then
26:             stepsize ← stepsize/2
27:         end if
28:         scaling_direction_flag ← $-1$
29:     end if
30: end for
31: return scaling_factor
```

Within the main loop, we iteratively generate *N* images, calculate their attribute changes in lines 5–13, and store them in a list. If the average attribute change is less than the specified goal, the optimization is complete (lines 14–16). Otherwise, if the attribute change is too small, we increase the scaling factor by stepsize (lines 18–19); if it is too large, we decrease the scaling factor (lines 24–25). For faster convergence to the optimization minimum, we halve the stepsize each time we change the optimization direction (lines 20–23 and 26–29). Since the optimization time scales linearly with *N*, we can speed up the process by first calculating the scaling factor for a smaller *N* and using that scaling factor as a starting point for a larger *N*.

Algorithm 3 Evaluate attribute vector

```
    Input: d, attr_idx
1:  for $seed = 0, ..., max$ do
2:      $z^* \leftarrow \Theta_{Z^*}$
3:      $img_{orig} \leftarrow G(z^*)$
4:      $\alpha_{orig} \leftarrow R(img_{orig})$
5:      $vggF_{orig} \leftarrow VGGFace2(img_{orig})$
6:      if $\alpha_{orig}[\text{attr\_idx}] > 0$ then
7:          $img_d \leftarrow (G(z^* - d \cdot U(0, 1, seed))$
8:      else
9:          $img_d \leftarrow (G(z^* + d \cdot U(0, 1, seed))$
10:     end if
11:     $\alpha_d \leftarrow R(img_d)$
12:     $vggF_d \leftarrow VGGFace2(img_d)$
13:     vggF_dist[$seed$] ← cos_dist($vggF_{orig}, vggF_d$)
14:     attr_pres[$seed$] ← $\|\alpha_d[\neq attr\_idx] - \alpha_{orig}[\neq attr\_idx]\|$
15:     attr_change[$seed$] ← $\|\alpha_d[attr\_idx] - \alpha_{orig}[attr\_idx]\|$
16: end for
17: return attr_change.mean(), attr_pres.mean(), vggF_dist.mean()
```

Algorithm 3 is used to compare two attribute vectors. It computes the mean attribute change of the target attribute, the mean attribute change of all other attributes, and the mean distance between the VGGFace2 embeddings of the original images and the modified images. Since each attribute vector is normalized to a predefined attribute change in Algorithm 2, the attr_change.mean() values should be approximately the same. However, attribute preservation and attribute change do not behave linearly, as shown in Figs. 5a and 6a. Therefore, the evaluation should examine different manipulation strengths, limited to an upper bound of ± 0.4, obtained by multiplying the normalized attribute vector by a scaling factor randomly sampled from a uniform distribution in lines 7 and 9. This approach facilitates the comparison of attribute vectors from multiple methods by comparing only two values, the attr_pres.mean() and embedding_distance.mean() of each approach. Depending on the needs of a particular use case, the two metrics can be weighted differently to select the most appropriate method.

Results

We used our evaluation metric to compare several algorithms and to investigate the influence of different latent spaces in Table 4. Similar to our previous work [19], we selected the approach by Shen et al. [15], our reimplementation of Enjoy Your Editing [18], and LS-StyleEdit, as being state-of-the-art algorithms that operate on *W* space and to allow comparing results of our new metric with those in Sect. "Previous Evaluation Metric". Additionally, we evaluate the approach by Larsen et al. [14], representing an earlier approach that might be more entangled, allowing to observe how this is

Table 4 We compared our approach to three other approaches in the *W* space and to StyleMC in the *S* space using the newly introduced evaluation metric

	Z^*	Target change	Attr. dist.	VGGFace2 dist.
Smiling				
Larsen et al. [14]	Z	1.0400	0.2765	77.95e−4
Larsen et al. [14]	W	1.0378	0.2216	9.76e−4
Shen et al. [15]	W	1.0417	0.1759	8.02e−4
Zhuang et al. [18]	W	1.0476	0.1851	2.24e−4
LS-StyleEdit	W	1.0476	0.1850	9.77e−4
Kocasarı et al. [17]	S	1.0361	0.2624	48.22e−4
LS-StyleEdit	S	1.0422	0.1639	6.46e−4
Hair color				
Larsen et al. [14]	Z	0.9659	0.5292	0.0136
Larsen et al. [14]	W	0.9609	0.4538	0.0120
Shen et al. [15]	W	0.9560	0.2343	0.0124
Zhuang et al. [18]	W	0.9650	0.2226	0.0082
LS-StyleEdit	W	0.9618	0.2961	0.0147
Kocasarı et al. [17]	S	0.9432	0.1942	0.0137
LS-StyleEdit	S	0.9571	0.2380	0.0105

shown in our metric. Finally, we include StyleMC [17] to provide another algorithm, next to LS-StyleEdit, that also operates on *S* space.

We performed an evaluation of Larsen et al. [14] in both *Z* space and *W* space to demonstrate the effect of the selected latent space. As shown in Table 4, it performs significantly worse in *Z* space than in *W* space. This improvement obtained by changing the latent space provides evidence that *Z* space indeed has a larger number of attribute entanglements. Similarly, the evaluation of LS-StyleEdit shows the benefit of changing the latent space from *W* to *S*.

Moreover, it seems that the performance of different methods is also influenced by the attribute being manipulated. For instance, for the attribute “Smiling”, our approach (*S*) has the best attribute distance, while for “Hair color”, our approach (*S*) is on par with Shen et al. [15] and Zhuang et al. [18]. Since our claim was that our approach works in all latent spaces, we used the same hyperparameters $L = 20$, $r = 0.006$ and a batch size of 1 for “Smiling” and “Hair color” in both *W* space and *S* space. We computed the attribute vectors for StyleMC [17] using the queries “a photo of a face with blond hair” and “a photo of a smiling face”.

Comparing the results for the algorithm by Shen et al. [15], our reimplementation of Enjoy Your Editing [18], StyleMC [17], and our LS-StyleEdit approach shows that there is no clear winner. Considering the evaluation metric as well as the comparison of edited images in the appendix, all four approaches seem to perform on the same level. StyleMC clearly performs worst on “Smiling” but very well on “Hair color”. In contrast, the quantitative evaluation by Zhuang et al. [18] claims a significantly worse performance for the algorithm by Shen et al. [15].

While showing comparable performance of our approach, we were able to achieve those results without using a large number of hyperparameters. In particular, we do not use any hyperparameter in our objective function. As discussed in Sect. “Local Search-based Latent Space Editing”, the maximum vector length *L* takes a role similar to those of hyperparameters within the loss functions of Enjoy Your Editing [16] and StyleCLIP [16]—however, both approaches require three hyperparameters instead of just a single one. Although we define the sample radius *r* as another hyperparameter, it mainly affects the way the search space is traversed. As a result, it is more closely related to other hyperparameters, such as the learning rate during backpropagation. Both, Enjoy Your Editing [18] and StyleCLIP [16] use the Adam optimizer, which comes with further hyperparameters in addition to the existing ones.

We have studied the different latent space distributions and scale the noise used in the local search according to the standard deviation of the corresponding latent space. This allows our approach to be agnostic and applicable to all latent spaces of StyleGAN2. Our newly proposed evaluation metric calculates the average attribute and VGGFace2 distance for a predefined average target attribute change. This allows for easier comparison between different approaches since only two numbers need to be compared.

Conclusion

We propose an effective local search-based approach LS-StyleEdit to semantically manipulate images based on a given target attribute. Our method enables continuous image manipulation on par with state-of-the-art approaches, while being latent space agnostic. At the same time, it requires significantly less GPU memory than existing iterative methods based on backpropagation. Since we do not rely on backpropagation, our method is applicable to non-differentiable black-box models for both the generator and the regressor, and does not suffer from instabilities. Furthermore, our approach has fewer hyperparameters, which allows for more efficient tuning. We have also emphasized the importance of comparing vectors that have similar levels of attribute change. The amount of attribute change is not only determined by the size of the vector, but also by the initial attribute value. Moreover, even with the same attribute value, different samples may respond differently to the same attribute vector. Therefore, evaluation metrics must be averaged over a large enough number of samples. The accuracy of the attribute preservation metric depends heavily on the reliability of the regressor prediction. It has been shown that the regressor can produce unreliable predictions for extreme cases that are rare in the training set. Therefore, it is recommended that the evaluation of the metric be performed within a reasonable range. Thus, we have explored the requirements for appropriate evaluation metrics and proposed our own metric that is easier to interpret than the existing ones. A possible direction for future work could be the use of more sophisticated local search algorithms, e.g., by adopting heuristics that have proven successful in other local search domains.

Appendix A: Attribute Manipulation Examples

See Figs. 7, 8, 9, 10, 11, 12.

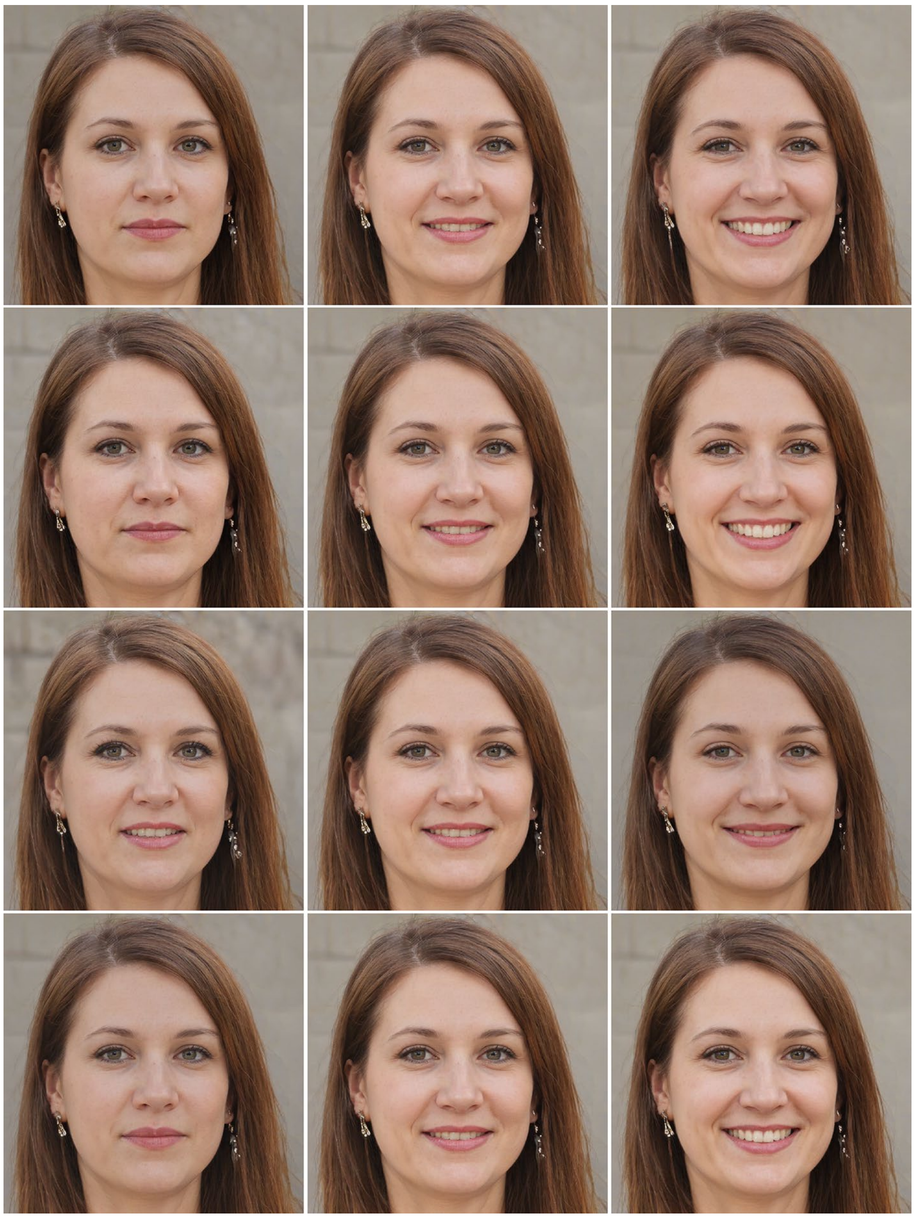


Fig. 7 Comparison of Smiling: Shen et al. (first row), Zhuang et al. (second row), Kocasarı et al. (third row), and our approach (fourth row) for image seed=0. Left column: less smiling, middle column: original image, right column: more smiling. This figure is taken from [19]

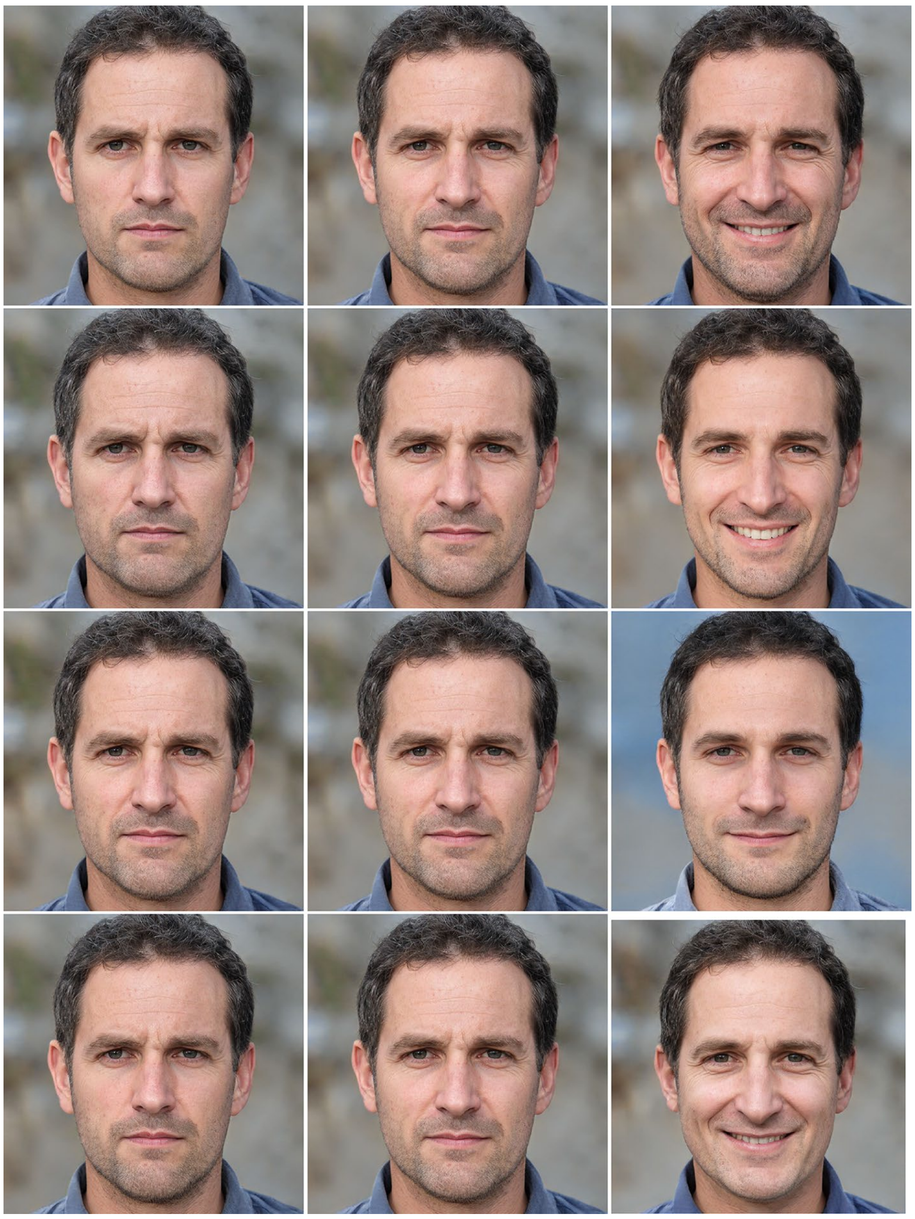


Fig. 8 Comparison of Smiling: Shen et al. (first row), Zhuang et al. (second row), Kocasarı et al. (third row), and our approach (fourth row) for image seed=1. Left column: less smiling, middle column: original image, right column: more smiling. This figure is taken from [19]

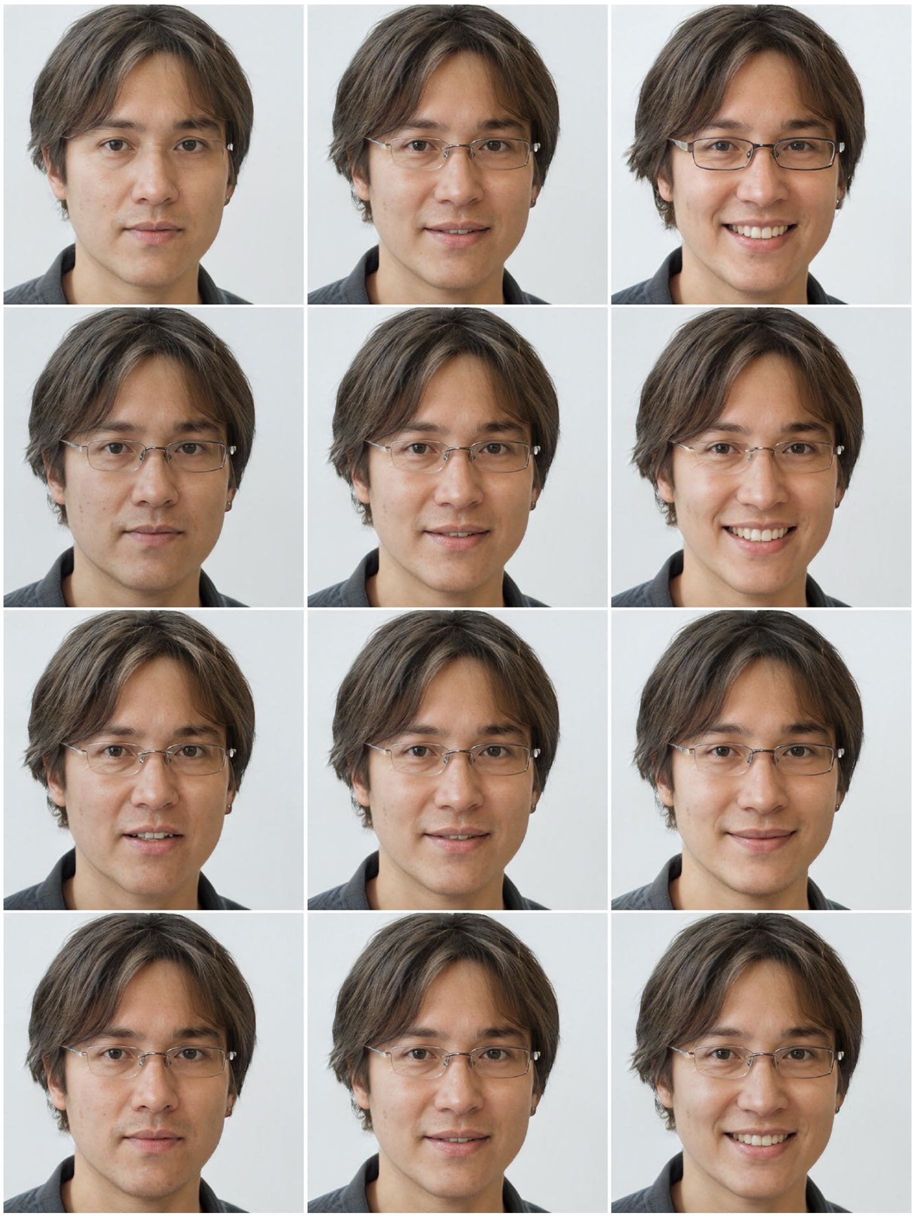


Fig. 9 Comparison of Smiling: Shen et al. (first row), Zhuang et al. (second row), Kocasarı et al. (third row), and our approach (fourth row) for image seed=2. Left column: less smiling, middle column: original image, right column: more smiling. This figure is taken from [19]

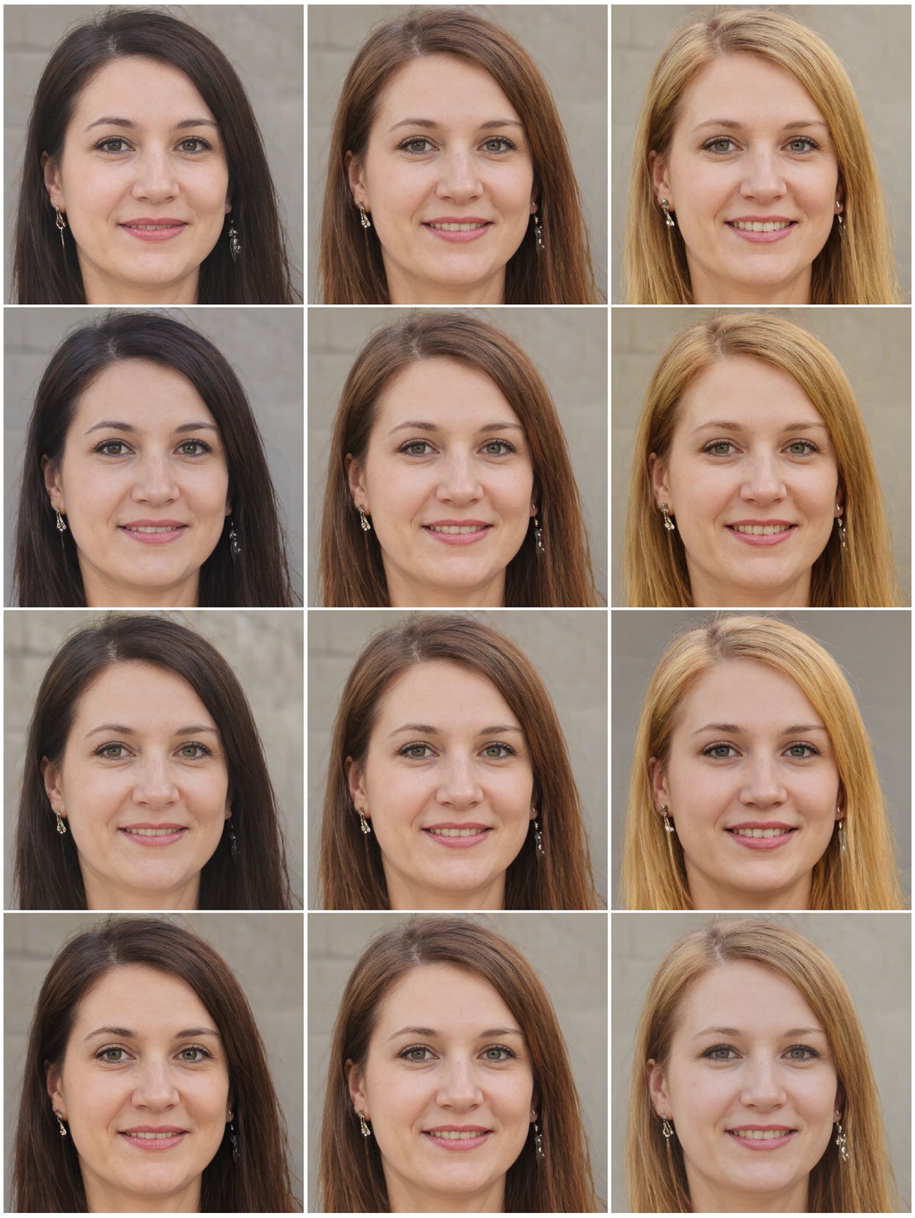


Fig. 10 Comparison of hair color: Shen et al. (first row), Zhuang et al. (second row), Kocasarı et al. (third row), and our approach (fourth row) for image seed=0. Left column: darker hair, middle column: original image, right column: lighter hair. This figure is taken from [19]

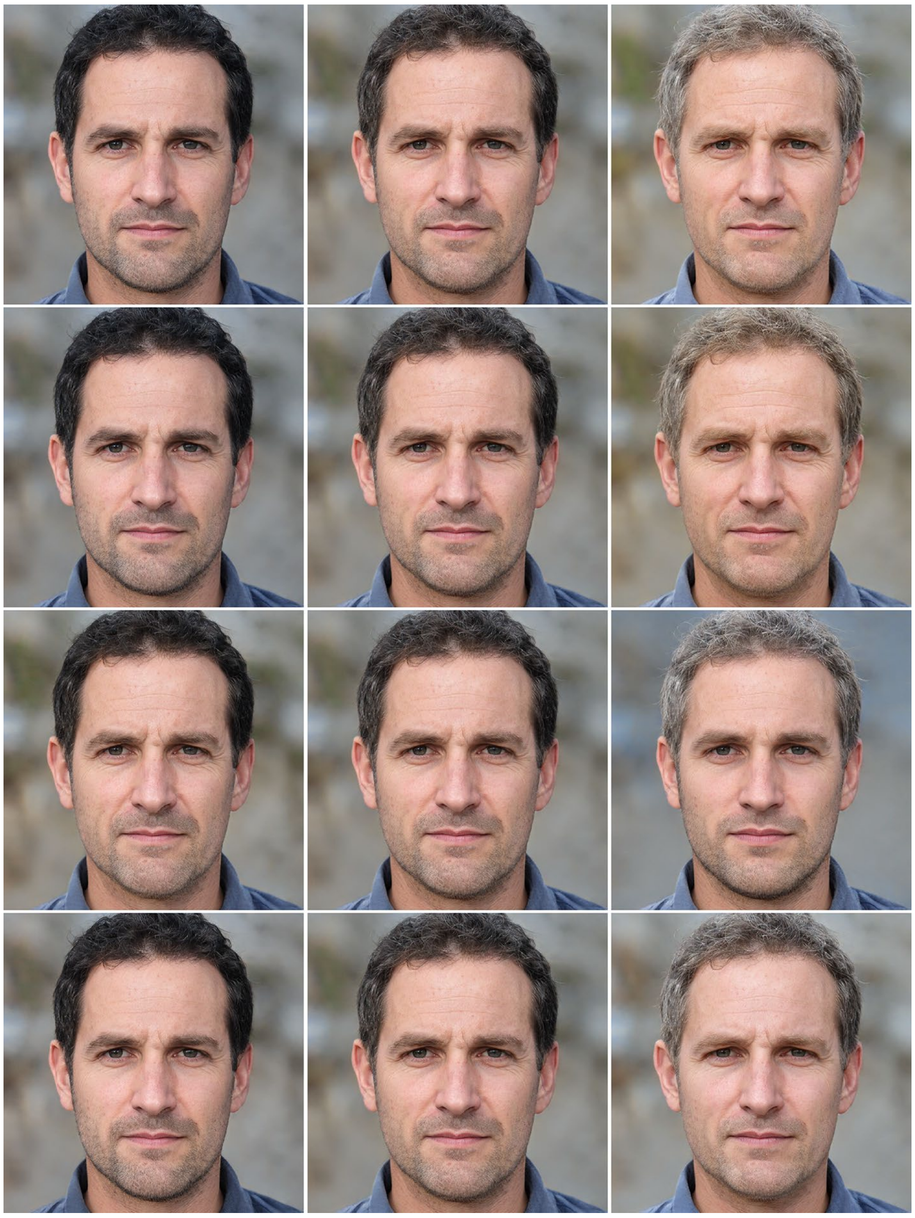


Fig. 11 Comparison of hair color: Shen et al. (first row), Zhuang et al. (second row), Kocasarı et al. (third row), and our approach (fourth row) for image seed=1. Left column: darker hair, middle column: original image, right column: lighter hair. This figure is taken from [19]

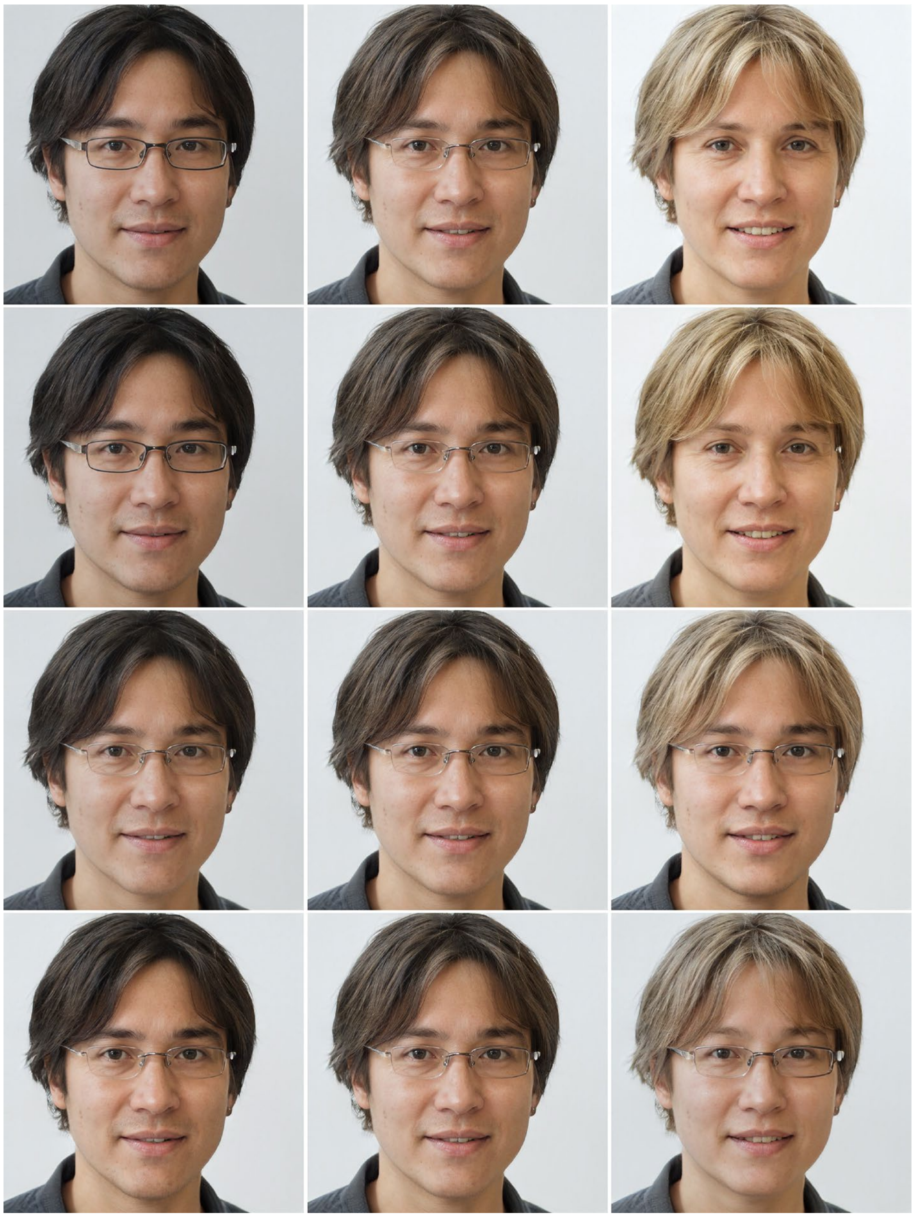


Fig. 12 Comparison of hair color: Shen et al. (first row), Zhuang et al. (second row), Kocasarı et al. (third row), and our approach (fourth row) for image seed=2. Left column: darker hair, middle column: original image, right column: lighter hair. This figure is taken from [19]

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 27. Curran Associates, Inc., Morehouse Lane Red Hook NY, United States 2014. https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf. Accessed 13 Oct 2023
2. Choi Y, Choi M, Kim M, Ha J-W, Kim S, Choo J. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018.
3. Choi Y, Uh Y, Yoo J, Ha J-W. Stargan v2: Diverse image synthesis for multiple domains. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2020.
4. Isola P, Zhu J-Y, Zhou T, Efros AA. Image-to-image translation with conditional adversarial networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017;5967–5976. https://doi.org/10.1109/CVPR.2017.632
5. Lee H-Y, Tseng H-Y, Mao Q, Huang J-B, Lu Y-D, Singh MK, Yang M-H. Drit++: diverse image-to-image translation viadisentangled representations. Int J Comput Vis. 2020;128:2402–17.
6. Wu P-W, Lin Y-J, Chang C-H, Chang EY, Liao S-W. Relgan: Multi-domain image-to-image translation via relative attributes. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019;5913–5921.
7. Zhu J-Y, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: 2017 IEEE International Conference on Computer Vision (ICCV), 2017;2242–2251. https://doi.org/10.1109/ICCV.2017.244
8. Zhu J-Y, Zhang R, Pathak D, Darrell T, Efros AA, Wang O, Shechtman E. Toward multimodal image-to-image translation. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., Morehouse Lane Red Hook NY, United States 2017. https://proceedings.neurips.cc/paper/2017/file/819f46e52c25763a55cc642422644317-Paper.pdf. Accessed 13 Oct 2023
9. Karras T, Laine S, Aila T. A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2019.
10. Karras T, Laine S, Aittala M, Hellsten J, Lehtinen J, Aila T. Analyzing and improving the image quality of stylegan. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, pp. 8107–8116. Computer Vision Foundation / IEEE, 29 Chase Rd Scarsdale, NY 10583, USA 2020. https://doi.org/10.1109/CVPR42600.2020.00813
11. Karras T, Aittala M, Laine S, Härkönen E, Hellsten J, Lehtinen J, Aila T. Alias-free generative adversarial networks. In: Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems 2021. https://openreview.net/forum?id=Owggnutk6lE. Accessed 13 Oct 2023
12. Voynov A, Babenko A. Unsupervised discovery of interpretable directions in the GAN latent space. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, 9786–9796. PMLR, 29 Great Smith Street, Westminster, London 2020. https://proceedings.mlr.press/v119/voynov20a.html. Accessed 13 Oct 2023
13. Härkönen E, Hertzmann A, Lehtinen J, Paris S. Ganspace: Discovering interpretable gan controls. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, 9841–9850. Curran Associates, Inc, Morehouse Lane Red Hook NY, United States 2020. https://proceedings.neurips.cc/paper/2020/file/6fe43269967adbb64ec6149852b5cc3e-Paper.pdf. Accessed 13 Oct 2023
14. Larsen ABL, Sønderby SK, Larochelle H, Winther O. Autoencoding beyond pixels using a learned similarity metric. In: Balcan, M.F., Weinberger, K.Q. (eds.) Proceedings of The 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 48, pp. 1558–1566. PMLR, New York, USA 2016. https://proceedings.mlr.press/v48/larsen16.html. Accessed 13 Oct 2023
15. Shen Y, Gu J, Tang X, Zhou B. Interpreting the latent space of gans for semantic face editing. In: CVPR 2020.
16. Patashnik O, Wu Z, Shechtman E, Cohen-Or D, Lischinski D. Styleclip: Text-driven manipulation of stylegan imagery. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021;2085–2094
17. Kocasarı U, Dirik A, Tiftikci M, Yanardag P. Stylemc: Multi-channel based fast text-guided image generation and manipulation. 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2022;3441–3450
18. Zhuang P, Koyejo OO, Schwing A. Enjoy your editing: Controllable GANs for image editing via latent space navigation. In: International Conference on Learning Representations 2021. https://openreview.net/forum?id=HOFxeCutxZR. Accessed 13 Oct 2023
19. Meißner A, Fröhlich A, Geierhos M. Keep it simple: Local search-based latent space editing. In: Bäck, T., van Stein, B., Wagner, C., Garibaldi, J.M., Lam, H.K., Cottrell, M., Doctor, F., Filipe, J., Warwick, K., Kacprzyk, J. (eds.) Proceedings of the 14th International Joint Conference on Computational Intelligence, IJCCI 2022, Valletta, Malta, October 24-26, 2022, pp. 273–283. SCITEPRESS, Av. D. Manuel I, 27A - 2 Dir, Setubal, Portugal 2022. https://doi.org/10.5220/0011524700003332
20. Radford A, Metz L, Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks. In: Bengio, Y., LeCun, Y. (eds.) 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico,

May 2-4, 2016, Conference Track Proceedings 2016. arXiv:1511.06434

21. Brock A, Donahue J, Simonyan K. Large scale GAN training for high fidelity natural image synthesis. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019 2019. https://openreview.net/forum?id=B1xsqj09Fm. Accessed 13 Oct 2023
22. Karras T, Aila T, Laine S, Lehtinen J. Progressive growing of GANs for improved quality, stability, and variation. In: International Conference on Learning Representations 2018. https://openreview.net/forum?id=Hk99zCeAb. Accessed 13 Oct 2023
23. Yu J, Lin Z, Yang J, Shen X, Lu X, Huang TS. Generative image inpainting with contextual attention. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 5505–5514;2018. https://doi.org/10.1109/CVPR.2018.00577
24. Demir U, Ünal GB. Patch-based image inpainting with generative adversarial networks. CoRR **abs/1803.07422** 2018 arXiv:1803.07422
25. Ledig C, Theis L, Huszar F, Caballero J, Cunningham A, Acosta A, Aitken AP, Tejani A, Totz J, Wang Z, Shi W. Photo-realistic single image super-resolution using a generative adversarial network. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pp. 105–114. IEEE Computer Society, 2001 L Street N.W., Suite 700 Washington, USA 2017. https://doi.org/10.1109/CVPR.2017.19.
26. Wang X, Yu K, Wu S, Gu J, Liu Y, Dong C, Qiao Y, Loy CC. Esrgan: Enhanced super-resolution generative adversarial networks. In: Leal-Taixé L, Roth S, editors. Computer Vision - ECCV 2018 Workshops. Cham: Springer; 2019. p. 63–79.
27. dos Santos Tanaka FHK, Aranha C. Data augmentation using gans. CoRR **abs/1904.09135** 2019 arXiv:1904.09135
28. Gadelha M, Maji S, Wang R. 3d shape induction from 2d views of multiple objects. In: 2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017, pp. 402–411. IEEE Computer Society, 2001 L Street N.W., Suite 700 Washington, USA 2017. https://doi.org/10.1109/3DV.2017.00053
29. Mirza M, Osindero S. Conditional generative adversarial nets. CoRR **abs/1411.1784** 2014 arXiv:1411.1784
30. Deng J, Dong W, Socher R, Li L, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA, pp. 248–255. IEEE Computer Society, 2001 L Street N.W., Suite 700 Washington, USA 2009. https://doi.org/10.1109/CVPR.2009.5206848
31. Kim T, Cha M, Kim H, Lee JK, Kim J. Learning to discover cross-domain relations with generative adversarial networks. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Proceedings of Machine Learning Research, vol. 70, 1857–1865. PMLR, 29 Great Smith Street, Westminster, London 2017. http://proceedings.mlr.press/v70/kim17a.html. Accessed 13 Oct 2023
32. Zhu J, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017, pp. 2242–2251. IEEE Computer Society, 2001 L Street N.W., Suite 700 Washington, USA 2017. https://doi.org/10.1109/ICCV.2017.244
33. Isola P, Zhu J, Zhou T, Efros AA. Image-to-image translation with conditional adversarial networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, 5967–5976. IEEE Computer Society, 2001 L Street N.W., Suite 700 Washington, USA 2017. https://doi.org/10.1109/CVPR.2017.632.
34. Liu M-Y, Breuel T, Kautz J. Unsupervised image-to-image translation networks. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., Morehouse Lane Red Hook NY, United States 2017. https://proceedings.neurips.cc/paper_files/paper/2017/file/dc6a6489640ca02b0d42dabeb8e46bb7-Paper.pdf. Accessed 13 Oct 2023
35. Collins E, Bala R, Price B, Süsstrunk S. Editing in style: Uncovering the local semantics of gans. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, pp. 5770–5779. Computer Vision Foundation / IEEE, 29 Chase Rd Scarsdale, NY 10583, USA 2020. https://doi.org/10.1109/CVPR42600.2020.00581
36. Tov O, Alaluf Y, Nitzan Y, Patashnik O, Cohen-Or D. Designing an encoder for stylegan image manipulation. ACM Trans Graph. 2021. https://doi.org/10.1145/3450626.3459838.
37. Zhang Y, Wu Z, Wu Z, Meng D. Resilient observer-based event-triggered control for cyber-physical systems under asynchronous denial-of-service attacks. Sci China Inf Sci. 2022. https://doi.org/10.1007/s11432-020-3190-2.
38. Nitzan Y, Bermano A, Li Y, Cohen-Or D. Face identity disentanglement via latent space mapping. ACM Trans Graph. 2020;39(6):225–122514. https://doi.org/10.1145/3414685.3417826.
39. Richardson E, Alaluf Y, Patashnik O, Nitzan Y, Azar Y, Shapiro S, Cohen-Or D. Encoding in style: A stylegan encoder for image-to-image translation. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, Virtual, June 19-25, 2021, pp. 2287–2296. Computer Vision Foundation / IEEE, 29 Chase Rd Scarsdale, NY 10583, USA 2021. https://openaccess.thecvf.com/content/CVPR2021/html/Richardson_Encoding_in_Style_A_StyleGAN_Encoder_for_Image-to-Image_Translation_CVPR_2021_paper.html. Accessed 13 Oct 2023
40. Alaluf Y, Patashnik O, Cohen-Or D. Only a matter of style: age transformation using a style-based regression model. ACM Trans Graph. 2021;40(4):45–14512. https://doi.org/10.1145/3450626.3459805.
41. Shen Y, Zhou B. Closed-form factorization of latent semantics in gans. In: CVPR 2021.
42. Wu Z, Lischinski D, Shechtman E. Stylespace analysis: Disentangled controls for stylegan image generation. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, Virtual, June 19-25, 2021, 12863–12872. Computer Vision Foundation / IEEE, 29 Chase Rd Scarsdale, NY 10583, USA 2021. https://openaccess.thecvf.com/content/CVPR2021/html/Wu_StyleSpace_Analysis_Disentangled_Controls_for_StyleGAN_Image_Generation_CVPR_2021_paper.html. Accessed 13 Oct 2023
43. Ling H, Kreis K, Li D, Kim SW, Torralba A, Fidler S. Editgan: High-precision semantic image editing. In: Advances in Neural Information Processing Systems (NeurIPS) 2021.
44. Lee C-H, Liu Z, Lingyun,W, Luo P. Maskgan: Towards diverse and interactive facial image manipulation, 2020;5548–5557. https://doi.org/10.1109/CVPR42600.2020.00559
45. Heusel M, Ramsauer H, Unterthiner T, Nessler B, Hochreiter S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., Morehouse Lane Red Hook NY, United States 2017. https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf. Accessed 13 Oct 2023
46. Bińkowski M, Sutherland DJ, Arbel M, Gretton A. Demystifying MMD GANs. In: International Conference on Learning

Representations 2018. https://openreview.net/forum?id=r1lUOzWCW. Accessed 13 Oct 2023

47. Luo W, Yang S, Wang H, Long B, Zhang W. Context-consistent semantic image editing with style-preserved modulation. In: Avidan, S., Brostow, G.J., Cissé, M., Farinella, G.M., Hassner, T. (eds.) Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XVII. Lecture Notes in Computer Science, vol. 13677, pp. 561–578. Springer, 11 West 42nd Street, Manhattan, New York, USA 2022. https://doi.org/10.1007/978-3-031-19790-1_34
48. Zhang R, Isola P, Efros AA, Shechtman E, Wang O. The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018.
49. Zhang Z, He H, Plummer BA, Liao Z, Wang H. Semantic Image Manipulation with Background-guided Internal Learning 2023. https://openreview.net/forum?id=1z9VTrxCgf. Accessed 13 Oct 2023
50. Dhamo H, Farshad A, Laina I, Navab N, Hager GD, Tombari F, Rupprecht C. Semantic image manipulation using scene graphs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020;5213–5222
51. Github implementation of vgg_face2. https://github.com/ox-vgg/vgg_face2. Accessed 04 Apr 2023
52. Liu Z, Luo P, Wang X, Tang X. Deep learning face attributes in the wild. 2015 IEEE International Conference on Computer Vision (ICCV), 2015;3730–3738
53. Matyas J. Random optimization. Autom Remote Contr. 1965;26(2):246–53.
54. Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J, Krueger G, Sutskever I. Learning transferable visual models from natural language supervision. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event. Proceedings of Machine Learning Research, vol. 139, pp. 8748–8763. PMLR, 29 Great Smith Street, Westminster, London 2021. http://proceedings.mlr.press/v139/radford21a.html. Accessed 13 Oct 2023
55. Deng J, Guo J, Xue N, Zafeiriou S. Arcface: Additive angular margin loss for deep face recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, 4690–4699. Computer Vision Foundation / IEEE, 29 Chase Rd Scarsdale, NY 10583, USA 2019. https://doi.org/10.1109/CVPR.2019.00482
56. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings 2015. arxiv:1409.1556
57. Cao Q, Shen L, Xie W, Parkhi OM, Zisserman A. Vggface2: A dataset for recognising faces across pose and age. In: 13th IEEE International Conference on Automatic Face & Gesture Recognition, FG 2018, Xi'an, China, May 15-19, 2018, pp. 67–74. IEEE Computer Society, 2001 L Street N.W., Suite 700 Washington, USA 2018. https://doi.org/10.1109/FG.2018.00020