Face Generation and Editing With StyleGAN: A Survey

Andrew Melnik, Maksim Miasayedzenkau, Dzianis Makaravets, Dzianis Pirshtuk, Eren Akbulut, Dennis Holzmann, Tarek Renusch, Gustav Reichert, and Helge Ritter

(Survey Paper)

***Abstract*—Our goal with this survey is to provide an overview of the state of the art deep learning methods for face generation and editing using StyleGAN. The survey covers the evolution of StyleGAN, from PGGAN to StyleGAN3, and explores relevant topics such as suitable metrics for training, different latent representations, GAN inversion to latent spaces of StyleGAN, face image editing, cross-domain face stylization, face restoration, and even Deepfake applications. We aim to provide an entry point into the field for readers that have basic knowledge about the field of deep learning and are looking for an accessible introduction and overview.**

***Index Terms*—Deep learning, deepfakes, face generation, face restoration, GAN, GAN inversion, latent space, StyleGAN.**

I. Introduction

HUMANS have always been fascinated with faces. It is how we recognize people, it is the main feature we watch out for when interacting with other people. This is reflected in the fact that we have a specialized region in our brain solely dedicated to the detection of face patterns and their subtle changes [14], [15].

There are almost 8 billion people alive today and yet we can discern all of them by their face. Attempting to delineate the great diversity of faces, [16] proposes 26 facial features as relevant for the description of faces (including many shape and size features, along with features for color and texture). Providing just three value levels for each of these features already results in $3^{26} > 2.5 \cdot 10^{12}$ unique faces, which is approximately 300 times greater than the current global population. Thus, there is still a vast space for unique identification of individuals using facial features.

Manuscript received 22 April 2023; revised 7 December 2023; accepted 19 December 2023. Date of publication 15 January 2024; date of current version 3 April 2024. Recommended for acceptance by T. Hassner. *(Corresponding author: Andrew Melnik.)*

Andrew Melnik, Eren Akbulut, Dennis Holzmann, Tarek Renusch, Gustav Reichert, and Helge Ritter are with Bielefeld Univeristy, 33615 Bielefeld, Germany (e-mail: andrew.melnik.papers@gmail.com; eakbulut@techfak.uni-bielefeld.de; dholzmann@techfak.uni-bielefeld.de; trenusch@techfak.uni-bielefeld.de; gustav.reichert@uni-bielefeld.de; helge@techfak.uni-bielefeld.de).

Maksim Miasayedzenkau, Dzianis Makaravets, and Dzianis Pirshtuk are with Banuba, 220000 Minsk, Belarus (e-mail: miasoedenkov@gmail.com; denismak123@gmail.com; pirshtuk@gmail.com).

Awesome list: https://github.com/ndrwmlnk/awesome-face-generation-and-editing

Digital Object Identifier 10.1109/TPAMI.2024.3350004

Our goal with this survey is to provide an overview of the state of the art deep learning technologies for face generation and editing. We particularly focus on GAN-based architectures that have culminated in the StyleGAN approaches. These methods enable the generation of high-quality facial images and provide versatile tools for semantics editing while preserving the image's photographic quality. The StyleGAN architecture merits the attention of a broader readership because it provides an ecosystem for a variety of applications and is used as the basis for a large number of research works. For a condensed visual synopsis, see Fig. 1 and Section I-B.

A. Survey Overview

The plan of the paper is as follows: Section I-B attempts to give an impression of the richness of applications of StyleGAN-based image processing methods. Section II will explain Generative Adversarial Networks (GANs), delving specifically into the StyleGAN architectures for the generation of face images. Training such architectures needs suitable metrics that capture image similarity at different levels, which will be the topic of Section III. Section IV will discuss the different latent representations that form the basis of the controllable image editing. Section V focuses on finding the latent representation of a given image. This prepares the ground for the methods reviewed in Section VI to edit face images, and in Section VII for cross domain face stylization. In Section VIII we look at some major approaches connected with face restoration and producing deepfakes in Section IX. In Section X we provide a concise overview of alternatives to StyleGAN for face generation and editing methods. The last Section XI concludes with a short summary and outlook.

B. Synopsis of StyleGAN Applications

Synthetic Face Generation: The StyleGAN [1], [17] architecture can generate faces which do not exist (see Fig. 1(a)). Applications of StyleGAN include the generation of unique pieces of art, including NFT collections [2], [18] (see Fig. 1(b)). There have been even some StyleGAN-based techniques developed to generate a child's facial image using parental face images as input [19] [20], [21]. *Style mixing* technique [22] allows creation of facial images that share features of several source images (see Fig. 1(c)) by combining StyleGAN internal representations of source images (see Fig. 3 and Section VII).

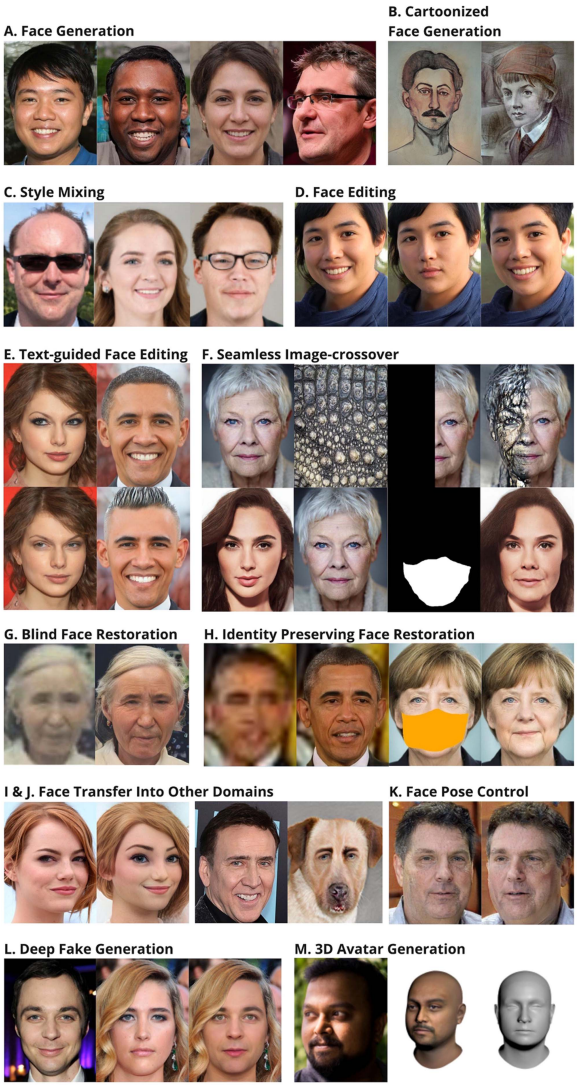


Fig. 1. Synopsis of StyleGAN Applications. (a) Faces generated using StyleGAN2 [1]. (b) NFT collection [2] generated using StyleGAN2 [1] trained on MetFaces dataset [3]. (c) Style mixing (see Fig. 3). (d) From left to right: the source image, smile removed, gender changed [1]. (e) Image editing with StyleCLIP [4] using text prompts. Upper row: original images, lower row: edited images using text prompts – Emma Stone (left), Mohawk Style (right). (f) Seamless image-crossover [5]. Left: sources image, right: resulting image. (g) Blind face restoration [6]. Left: degraded image, right: enhanced image. (h) Identity preserving face restoration and in-painting [7] [8]. (i) and (j) Transferring faces into other domains while preserving the identity. Left: real image, right: cartoon like output [9] [10]. (k) Control semantic parameters, such as face pose using StyleGAN component. Left: source image, right: pose changed [11]. (l) Deepfake generation [12]. (m) Automated 3D avatar generation using StyleGAN component. From left to right: source image, avatar, face model [13].

Editing Facial Features: While preserving a high image quality [23], the StyleGAN architecture [1], [24] allows to alter many features like age, hair, smile, etc. (see Fig. 1(d) and more details in Section VI for details) [25]. The StyleGAN architecture can be used as a component for text-driven editing of images. StyleCLIP [4], [26] offers a manipulation of facial features using text prompts alone (see Fig. 1 (3) and Section VI-B1). The *Image2StyleGAN++* framework [5] provides application examples of image editing using scribbles, inpainting, or crossover (see Fig. 1(f)).

Facial Image Recovery: StyleGAN has been demonstrated to be able to restore face images with regard to degradation such as low resolution, noise, colorization of old photos, and even missing parts (see Section VIII). For example, GFP-GAN [6] or other works [27], [28] uses a pretrained StyleGAN2 [1] as a component of the face restoration architecture (see Fig. 1(g)). However using such general facial priors for restoration of images can end up in identity loss of faces. MyStyle [7] tackles the problem of target identity, while recovering lost information (see Section VIII-C and Fig. 1(h))

Stylization of Faces: Transferring face images in another domain such as sketches [29], [30], [31], [32] [33] or cartoonization [34], [35], [36] while preserving the person identity is a big challenge. The goal of stylization approaches is to satisfy both identity preservation and perceptual features of a certain style (see Fig. 1(i) and 1(j) for example applications and Section VII for technical details).

Deepfake generation can be summed up as a face-swapping operation that is not being recognizable by human viewers (see Fig. (1l)). Deepfakes are used by artists, social media platforms, in film, game, fashion, and entertainment industry [37], [38], [39], [40]. Refer to Section IX for a comprehensive exploration of how StyleGAN functionality can be harnessed for deepfake applications: face reenactment [41], [42], swapping [12], and transfer [43].

3D Facial Avatar Generation by transferring a 2D human face image to a 3D avatar, while preserving identity [13] (see Fig. 1(m)). Such 3D avatars can be useful e.g., in gaming, real time video filters, etc.

Hands-on Applications and Mobile Networks: Face generation and editing technologies are popular for social networks, messengers, mobile and photo apps. In many cases, privacy issues motivate the localization of processing from cloud servers to mobile devices. Mobile architecture networks [44] are constrained to being fast while consuming only little memory on the order of a few megabytes (MB) instead of several gigabytes. Running architectures that include the StyleGAN model can be computationally demanding. Thus, in many cases the StyleGAN architecture can be used to generate datasets of paired examples for supervised learning of a number of mobile encoder-decoder networks (1-10 MB), one per each discrete editing feature like adding glasses, smile, makeup, etc [45], [46].

C. Training Datasets

The majority of the approaches discussed in this survey used the following datasets for training: Flickr-Faces-HQ (FFHQ) [17], CelebFaces Attributes Dataset (CelebA) [47], and CelebA-HQ [48].

FFHQ is a dataset of 70,000 high-quality PNG images at 1024×1024 resolution, featuring diverse human faces with variation in age, ethnicity, and accessories such as eyeglasses and hats. Originally designed for GAN benchmarking, the images were obtained from Flickr and were automatically aligned and cropped using dlib [49], inheriting the website's biases.

CelebA is a face attributes dataset containing over 200,000 celebrity images at 178×218 resolution, each annotated with 40 attributes. The images have diverse backgrounds and poses and come with rich annotation, including 10,177 identities, 202,599

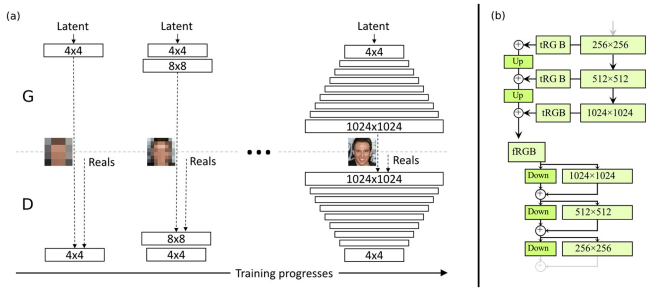


Fig. 2. (a) StyleGAN grows progressively while training, (b) StyleGAN2 does not but uses output skips and residual connections. Images from [48], [1].

face images, 40 binary attribute annotations, and 5 landmark locations (eyes, nose, mouth) per image. **CelebA-HQ** is a high quality version of CelebA dataset with 30,000 images at 1024×1024 resolution.

II. StyleGAN Architectures for Generation of Faces

A. Generative Adversarial Networks

To generate images of faces, Generative Adversarial Networks (GANs) [50] have proven to be a highly suitable architecture. A GAN offers a way to learn a mapping that transforms a known, usually simple (e.g., Gaussian) distribution into a more complex target distribution that represents a given domain of patterns, for example, images of human faces. This mapping or generative model can be learned using a sufficiently large training set of samples from the desired target distribution. Once the model has been learned, new samples can be generated by simply feeding the model with random inputs from the simple distribution that was used during training.

B. Progressive Growing GANs (PGGAN)

While vanilla GANs are able to generate images of reasonable quality, they suffer from limited controllability and unstable training [51]. To overcome these problems, Karras et al. [48] introduced a training strategy in which the neural network progressively grows more layers during training (see Fig. 2(a)). With layers at increasing depth, image resolution increases as well. Starting with low-resolution images of 4×4 up to a resolution of 1024×1024, firstly coarse structures and later fine details are learned. This makes training more stable, because it splits the task into simpler sub-tasks. Additionally, the training time benefits from this approach, because most of the iterations are done at lower resolutions and thus in a network with a smaller number of layers.

C. Stylegan

StyleGAN [17] adapts the progressive training strategy from the PGGAN [48] and the generator architecture is re-designed motivated by the style transfer ideas [22]. As a result, it offers a high degree of flexibility to mix image styles at different levels of its generator architecture.

StyleGAN no longer passes the random sample z (often referred to as *latent code* for being "decoded" by the generator) directly into the generator's input layer. Instead, StyleGAN starts generating images from a learned constant $(4 \times 4 \times 512)$, and the latent code $z \in \mathcal{Z}$ is fed into the network along a different route (see Fig. 4). First, $z \in \mathcal{Z}$ is mapped through a deep network of fully connected layers into an intermediate latent space $w \in \mathcal{W}$. The benefit of this $\mathcal{Z}$ to $\mathcal{W}$ transformation is that the intermediate space $\mathcal{W}$ does not need to follow the Gaussian distribution of the training data (however, $\mathcal{Z}$ does). Thus, latent space $\mathcal{W}$ can be disentangled which is a desirable property because it means that features in the generated images can be controlled independently of each other, and this is one of the most important features of StyleGAN, as it opens up great scope for working with images in latent space. Subsequently, for each generator layer separately, $w \in \mathcal{W}$ is converted using affine transformation (fully connected layer without activation function) into a vector of style parameters that are used to shift and scale the activity pattern in the feature maps of the respective convolutional layer. This affine transformation of feature maps is called an adaptive instance normalization (AdaIN) [22].

This layer-wise feeding of style parameters $w \in \mathcal{W}$ allows style-mixing by feeding code parameters w_A and w_B of two sources A and B respectively into different layer subsets of the StyleGAN generator (see Fig. 3). If a code is injected into early layers it affects rough features (e.g., shape of a face) while injection into later layers correspond to finer details (e.g., skin color), so latent codes enable modifications at different granularities. Finally, to provide stochastic detail that would have to be learned otherwise, pixel-wise noise is injected after each convolution. This allows the network locally stochastic placing of fine structure, such as pores, hairs, or freckles. All these architectural innovations allow it to outperform the previous ProgressiveGAN.

D. StyleGAN2

StyleGAN was a major breakthrough towards the generation of high-resolution face images that looked very natural. Yet, the generated images tended to contain minor, but systematic artifacts, such as blobs or droplets. A careful analysis of this phenomenon enabled the development of an updated version, StyleGAN2 [1].

Its key change was a simplification and reorganization of the layer normalization, which in the original StyleGAN was recognized as destroying information in the relative activation strengths of the different feature maps within a layer. This was avoided by replacing the former AdaIN operations [22] by a direct rescaling of the convolutional weights, again based on the the style parameter output associated with $w \in \mathcal{W}$, followed by a normalization by the standard deviation over the scaled weights. Additionally, the noise and bias now became added outside the style block and removed from the initially learned constant input.

Another problem was that in the images created by StyleGAN some details like eye or teeth orientation seemed to be either stuck in place or jumping between positions instead of moving smoothly. This was attributed to the generator needing to produce output images at each resolution, which forces it to generate maximal frequency details. To overcome this problem, StyleGAN2 no longer trains models using progressive growing, but sums the output from different resolutions together and

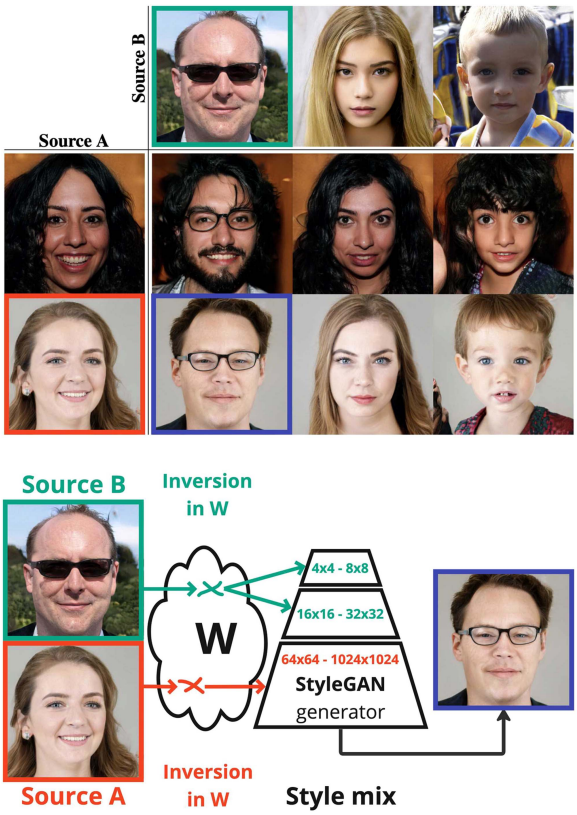


Fig. 3. Style mixing in StyleGAN [17]. Top panel shows examples of style mixing, bottom panel illustrates the style mixing pipeline in StyleGAN - latent representations of two images can be used in different levels of the generator.

utilizes skip connections (see Fig. 2(b)). The size of the model itself was also increased through the number of feature maps in the layers responsible for the highest resolutions.

Furthermore, a new regularization loss was introduced to control the perceptual path length (PPL), which quantifies the smoothness of the mapping from a latent space to the output image by measuring average LPIPS (see Section III-C) between generated images under small perturbations in latent space.

E. StyleGAN3

While StyleGAN2 abolished the above artifacts and further increased image quality by a number of measures, a remaining problem was that, when making an animation footage for a specific person by manipulating its latent representation, for example rotating head or adding smile, fine details, for example hair texture, appeared to be stuck to specific image coordinates instead of properly co-moving with the object surfaces to which they were attached. This problem is demonstrated on the animation footages that can be found here [52]. It turned out that StyleGAN2 had a strong propensity to fix a feature to image coordinates whenever any information about such coordinates was available to the network.

Major sources of such information turned out to be image borders and aliasing patterns from discretization on a pixel grid. The first problem was solved by sufficient zero padding around the image in the generator [53]. The discretization problem was solved by reformulating all operations for a continuous image , which then could be used as an equivalent representation. Correct maintenance of this equivalence required a careful enclosing of the application of the non-linearity between upsampling and downsampling operations designed to filter away any spectral out-of-band contents otherwise introduced by the non-linearity. Further changes introduced in the StyleGAN3 architecture [24] included an optimized reduction of the number of layers and simplifications, including retracting some regularizations that were introduced in StyleGAN2 but incompatible with a strict enforcement of translation invariance.

Transitional and rotational equivariance is achieved by replacing the learned $4 \times 4 \times 512$ constant used in StyleGAN2 with randomly generated and information-wise equivalent Fourier features with dimensionality $36 \times 36 \times 1024$ as the first layer. To prevent leakage of absolute image coordinates into the internal representations, a fixed-size margin around the target canvas, that is cropped after each layer, is introduced to replace the previously used padding. StyleGAN3 also uses lower cutoff frequencies for filters during up-sampling and downsampling operations to eliminate aliasing artifacts. In addition, rotational equivariance is achieved by replacing 3×3 convolutions with 1×1 convolutions, and replacing Cartesian downsampling filters with radially symmetric ones for all layers except the last two.

III. Measuring Similarity of Faces

This chapter delves into several loss functions that are applicable in evaluating the generated images and addresses the challenge of computational measures for human perception of facial images.

A. Adversarial Loss

Adversarial loss is fundamental to constructing generative adversarial models [50]. The generator network G learns to create samples from a given distribution, and the discriminator network D learns to determine if a sample is from the real data distribution or not. Given an image $G(z)$ generated from z in a known distribution p_Z, and $D(x)$ being the probability of x being drawn from the given data distribution p_{data}, the training objective of D is to discriminate real images from generated ones. In other words, it needs to maximize $D(x)$ when x is sampled from p_{data}. However, when $x = G(z)$ is produced by the generator, the generator wants $D(G(z))$ to be maximized instead.

Switching to $\log(1 - D(G(z))$ reverses the optimization direction for both D and G, allowing to express the adversarial training of D and G as the optimization of a two-player minimax game with value function $V(G, D)$ [50]:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (1)$$

Although the minimax loss is commonly used to train GANs, it can lead to instability in the training process. To address this, other loss functions have been proposed, such as the Wasserstein GAN loss [54], Hinge loss [55], and non-saturating loss [50],

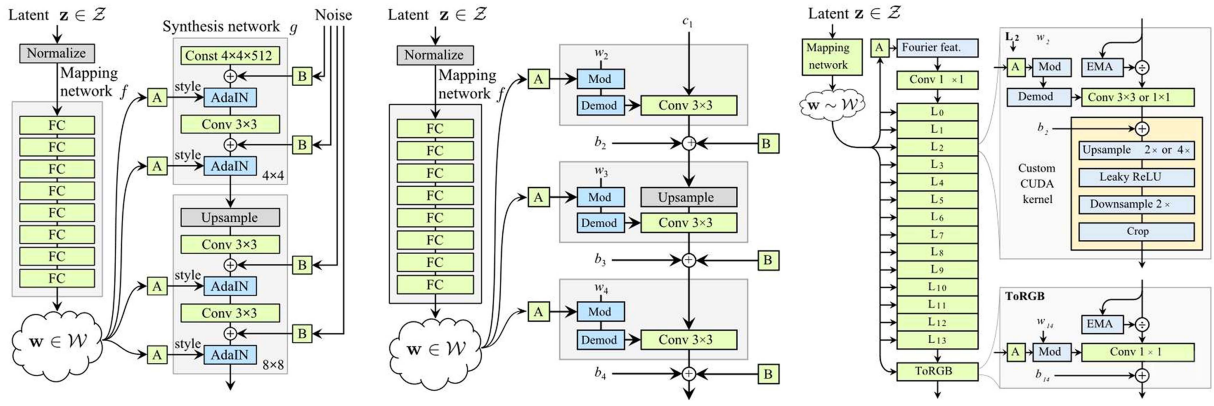


Fig. 4. Architectures of StyleGAN generators: (a) StyleGAN [17], (b) StyleGAN2 [1], (c) StyleGAN3 [24].

which have been shown to improve the stability of GANs training in various scenarios.

B. L_2 Loss

The L_2 loss measures pixel-wise similarity between two images x and $\hat{x}$. It is defined as $L_2(x, \hat{x}) = \sum_i (x_i - \hat{x}_i)^2$. While this is a fairly simple way to compute the similarity between two images, it suffers from the drawback that such pixel-wise comparison is insensitive to the local context of image points. This can make the L_2 distance very different from human image similarity judgements [56]. For example, for an image with a white horizontal line on a black background. If we shift the horizontal line just by one pixel down, the L_2 distance will be large, but the two images might be indistinguishable to a human.

C. LPIPS Loss

It is believed that CNN features provide a more abstract representation of an image. Consequently, the distances in this representation space more accurately capture the distinctions that matter in distinguishing images from one another, while being less susceptible to low-level perturbations that are unrelated to the image's content [57]. Thus, one approach is to transform images into a CNN feature space where point-wise distances more accurately reflect human similarity judgments, and then use the L_2 measure there. This is the underlying idea of the Learned Perceptual Image Patch Similarity (LPIPS) [58] measure, using CNNs trained for visual recognition tasks such as VGG [59] or AlexNet [60] to transform the images. The resulting activations in CNN layers represent increasingly abstract image features, allowing the similarity measure between an image pair $x, \hat{x}$ as a weighted sum of L_2 distances between the corresponding feature map activations $y^l, \hat{y}^l$ with shape (H_l, W_l, C_l) in the CNN layers l (see (2)). The weighting coefficients, represented by c_l, are used to scale the activations channelwise and fine-tune the metric to match human similarity judgments as closely as possible, in addition to the DNN training. The LPIPS metric thus takes the form:

$$d(x, \hat{x}) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \|c_l \odot (y^l_{hw} - \hat{y}^l_{hw})\|_2^2 \qquad (2)$$

D. Identity Preservation Loss

The losses mentioned earlier measure the overall dissimilarity between two images. However, even a minor variation in facial characteristics can lead to individuals being perceived as visually distinct (see Fig. 8 upper row vs. lower row). This, in turn, prompts the need for similarity measures designed to cater specifically to facial recognition, fine-tuned to accurately determine whether two faces correspond to the same person or not, while disregarding extraneous factors like facial position or expression.

To achieve this, models have been trained to enforce higher embedding similarity for intra-class face samples and larger embedding distances for cross-class samples, resulting in suitable identity preservation loss functions. The ArcFace model [61] is a prominent example of such a model, designed specifically for face recognition tasks. The embeddings of the same face produced by this model will be close to each other, but far from the embeddings of other faces. The loss for comparing two faces using this model is computed as $\mathcal{L}_{id} = 1 - \langle R(x), R(y) \rangle$, where R is the ArcFace model which produces embeddings, x, y are the face images and $\langle \cdot, \cdot \rangle$ is the cosine similarity.

E. Fréchet Inception Distance FID

The Fréchet Inception Distance (FID) [62] is a commonly used metric for evaluating how well the distribution of images generated by a generator matches the distribution of images used in training. FID involves embedding each image from the two distributions into a 2048-dimensional vector using InceptionV3 [63], and then comparing the two distributions of embedding vectors using Wasserstein-2 distance.

The FID approach is similar to LPIPS (see Section III-C), because it compares higher-level features rather than RGB pixel information, both of which approximate relevant features within the human visual system. A low FID score is a good indicator that the generator is producing images that are similar to the training images.

FID is mostly used for measuring generator performance, not as a loss function, because it is computationally expensive. It requires a large number of images to compute covariance between all pairs of images, and passing gradients through InceptionV3 for every image. However, computing FID once to

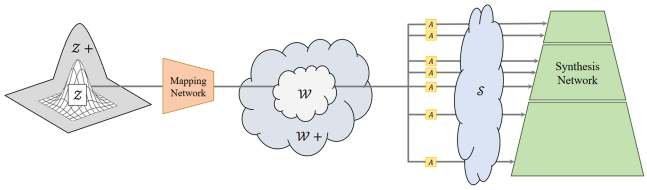
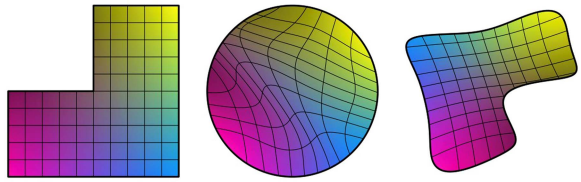


Fig. 5. Latent spaces in StyleGAN. Image from [65].



(a) Distribution of features in training set (b) Mapping from $\mathcal{Z}$ to features (c) Mapping from $\mathcal{W}$ to features

Fig. 6. $\mathcal{W}$ space (c) demonstrates semantic axes that facilitate and generalize editing operations on different faces. To illustrate, consider the *facial-beard axis* within the $\mathcal{W}$ space; when we select a point in $\mathcal{W}$ representing a face and shift it along the *facial-beard axis*, the result is the appearance of facial beard on that face. In contrast, the $\mathcal{Z}$ space (b) exhibits distinct *facial-beard curves* for different faces, making it less suitable for semantic editing operations across various faces. Additionally, it is important to consider that in the training dataset (a) there are examples of both male and female faces without beards, but the majority of faces with beards are male. See Section IV-B. Image from [17].

measure the perceptual quality of generated images by a trained generator is feasible and widely used.

IV. Latent Spaces of StyleGAN

In the original GAN architectures, the latent code was found to be highly entangled and difficult to use for controlling the output image features [64].

The first key idea of StyleGAN architectures [1], [17], [24] is to introduces more than one innate latent space ($\mathcal{Z}, \mathcal{W}, \mathcal{S}$; see Fig. 5), thereby allowing to learn intermediate latent representations with properties better tailored to the semantic structure of the image space (see Fig. 6). Moreover, to increase the expressive power of StyleGAN, it is common to work with extensions of these spaces ($\mathcal{Z}+, \mathcal{W}+$; see Fig. 5). Here, we review the commonly used spaces and describe the differences between them.

The second key idea in StyleGAN architectures [1], [17], [24] is to inject latent codes into every layer of the generator pipeline, not just at the beginning. This allows StyleGANs to offer very flexible control modulation of the activities passing through their respective layers of the generation of images at different resolutions (see Fig. 4(a)–(c)).

A. $\mathcal{Z}$ and $\mathcal{Z}+$ Spaces

In the StyleGAN architectures, 512-dimensional samples from an isotropic normal distribution with unit variance and zero mean provide the random inputs $z \in \mathcal{Z}$ at the root of the entire generation pipeline. $\mathcal{Z}+$ space implies sequential mapping of 18 $z \in \mathcal{Z}$ vectors into 18 corresponding $w \in \mathcal{W}$ vectors (relevant for StyleGAN and StyleGAN2 architectures with 18 layers).

B. $\mathcal{W}$ Space

Latent codes from $\mathcal{Z}$ are transformed to latent codes in the 512-dimensional $\mathcal{W}$ space of StyleGAN through the mapping network (see Fig. 5). This transformation, which must be learned during training, allows to distort the simple $\mathcal{Z}$-distribution into a distribution $\mathcal{W}$ of the same dimensionality of style parameters. In this $\mathcal{W}$ space meaningful editing operations on images can become realizable by simple axis-parallel movements of a point [17]. The eight fully connected mapping layers of the StyleGAN's mapping network can provide the adaptivity to unfold the disc-shaped $\mathcal{Z}$ space into a space $\mathcal{W}$ whose shape is much closer to the required feature distribution (Fig. 6(c), right). Fig. 6 illustrates a (toy) situation where the feature distribution of to-be-generated images excludes a combination of two features, leading to a distribution where one quadrant of all possible combinations is absent (Fig. 6(a), left). To create such a distribution from the disc-shaped input distribution $\mathcal{Z}$ (Fig. 6(b), middle) requires a very non-linear mapping.

C. $\mathcal{W}+$ Space

Usually, a 512-dimensional vector $w \in \mathcal{W}$ is used 18 times as the style input to 18 layers of the StyleGAN2 generator. This suggests that each of these 18 can be individually modified for fine-tuning of a generated image. This extends latent space into 18 copies of $\mathcal{W}$ ($d = 18 \times 512$) and is denoted by $\mathcal{W}+$ (see Fig. 5). This larger space is able to provide a different latent code for each layer of the StyleGAN generator (e.g., 18 for a StyleGAN2 generator with a 1024×1024 output resolution).

Since the StyleGAN architecture is trained using $\mathcal{W}$ space, images sampled from $\mathcal{W}+$ do not necessarily have realistic perceptual quality. This can allow to generate entirely novel patterns that are still face-like (e.g. "aliens"). However, it may also lead to patterns with useless structure or level of quality. As the distribution of $\mathcal{W}$ cannot be explicitly modeled, keeping the latent code in within a range that corresponds to semantically and quality-wise useful patterns is a challenging task. To learn more about trade-offs between $\mathcal{W}$ and $\mathcal{W}+$ spaces see Section V-B3.

D. $\mathcal{S}$ Space

In a further step of StyleGAN processing, the latent code $w \in \mathcal{W}$ of an image is further transformed to $s \in \mathcal{S}$ vectors for each layer of the StyleGAN generators [66]. The details of these transformations differ slightly between the versions, but a shared commonality is a mapping to a parameter vector of style parameters $\mathcal{S}$ that parametrizes a set of affine transformations (one for each layer) that either normalize the activity pattern in a layer (in the case of StyleGAN), or that directly define a two-step scaling of the weights of a layer (mod/demod operations of StyleGAN2 and StyleGAN3). While the activity normalization in StyleGAN requires a specification of two parameters (bias and scaling) for each feature map, StyleGAN2/3 get by with a mere scaling (single parameter) for the feature map scalings in the mapping layers. For the sake of brevity, we focus on the case of StyleGAN2 [1] in the following discussion, which is very representative of the major ideas behind the style mapping.

At the style input of every layer of the StyleGAN2 generator is an independent single-layer perceptron denoted by A (affine transformation). This network maps $w \in \mathcal{W}$ vector into a new vector $s \in \mathcal{S}$ vector that provides for each of the layer's weight kernel a separate scalar scaling parameter. The size of the vector $s \in \mathcal{S}$ equals the number of channels in all layers of the StyleGAN2 generator. For example, in StyleGAN2 $\dim(\mathcal{W}) = 512$ and $\dim(\mathcal{S}) = 9088$ [66]. This modulated kernel is then applied to the layer $L-1$ of the StyleGAN2 generator to produce the activation of the channel in the layer L of StyleGAN2. In [66] Wu et al. proposed to name this latent space of coefficients s StyleSpace $\mathcal{S}$. Analysis from Wu et al. indicates, that the $\mathcal{S}$ space is more semantically disentangled than previous latent spaces of StyleGAN2. The usecases of S space for face editing is described in Sections VI-A3 and VI-A4.

V. Inversion to Latent Spaces of StyleGAN

This chapter will review recently developed solutions that map an image to a suitable StyleGAN latent code. It is important to know the latent representation of an image of a face in StyleGAN space in order to manipulate that image or combine it with the style of some other image of a face. Since a StyleGAN model has several latent spaces (see Fig. 5), this task can come in different variants, depending on the choice of the latent space for which a representation is sought. Furthermore, we shall see that these different choices may entail different properties, e.g. how the image will change under manipulations of its latent code, but also how well the inversion can be steered to faithfully capture the important visual features of the given real image.

After a brief overview over the major groups of inversion methods in general in Section V-A we turn in Section V-B to the important issue of *evaluating* the inversion for distortion, perceptual quality, and editability, and how to control a suitable trade-off between these properties. Achieving an approximate resemblance is relatively easy (see Fig. 8), but resemblance in fine detail is very important for the perception of faces. Then, in Section V-C, we delve into specific methods for obtaining inversion encoders that can solve the inversion task: *pixel2style2pixel* (pSp) [67], *encoder4editing* (e4e) [68] and *ReStyle* [69]. Finally, Section V-D describes techniques for improving inversion quality by selective tuning of StyleGAN generator weights: *Pivotal Tuning* [70] introduces optimization-based fine-tuning of StyleGAN; *MyStyle* [7] extends fine-tuning to hundreds of portrait images of a given person; *HyperStyle* [71] introduces encoder based prediction of fine-tuning weights of the StyleGAN generator.

A. Major Groups of Inversion Methods

Inversion methods can typically be divided into three major groups of methods: *gradient-based optimization* of the latent code (Section V-A1 and Fig. 7(a)), direct *encoder-based mapping* onto the latent code (Section V-A2 and Fig. 7(b)), and *fine-tuning* of weights of the StyleGAN generator (Section V-A3 and Fig. 7(c) and (d)).

1) Gradient-Based Optimization of the Latent Code: Gradient-based optimization methods (see Fig. 7(a)) [1], [5] [72] directly optimize the latent vector using gradients from the loss between the real image and the generated one. Such methods

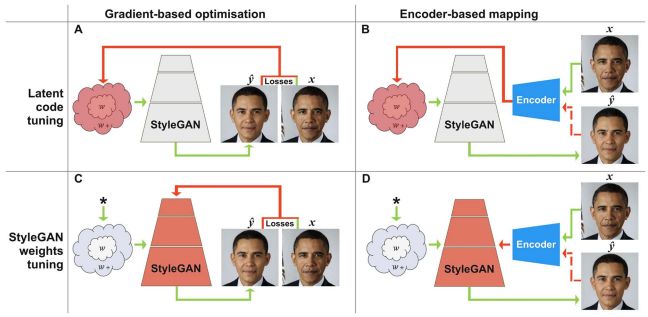


Fig. 7. Inversion methods: target image x, reconstruction image $\hat{y}$, initial inversion $*$. Typical losses are L_2, LPIPS [58], ArcFace loss [61].

Fig. 8. Obama [73] and Actress [74] before (upper row) and after Inversion (lower row). This inversion was performed using StyleGAN2-ada [3].

can find a latent representation of the original image with a reasonable similarity (see Fig. 8). However, there still exist three main drawbacks [5]:

- Optimizing the procedure is a time-consuming task that usually takes several minutes on a modern GPU.
- The random initialization choice can significantly impact the final reconstruction image.
- The found latent point in $\mathcal{W}$ or $\mathcal{W}+$ spaces through inversion optimization steps is less stable while editing of the generated image than latent points obtained by sampling from $\mathcal{Z}$ space and generating latent points in $\mathcal{W}$ or $\mathcal{W}+$ spaces through the mapping network (from $\mathcal{Z}$ to $\mathcal{W}$ space). The mapping network generates points in the distribution of the training dataset, while inversion optimization steps can move the latent point away from the distribution of the training dataset (see Section V-B1).

2) Encoder-Based Mapping to the Latent Code: Alternatively, encoder-based methods for finding the latent code (see Fig. 7(b)) [67], [68] train an encoder network over a large number of samples to directly map from the RGB image space into a latent space of StyleGAN. Once trained, the encoding can be done in the fraction of a second needed to process through the CNN encoder. Latent points obtained from the encoder network are more suitable for editing [68] by moving the point in the

latent space, as the encoder network is trained to generate points inside the distribution of the training dataset of StyleGAN. However, training such an encoder is not trivial and the image generated from the obtained latent code may lose the identity of the original face (see Fig. 8).

Conventional image- and feature-level losses (i.e. MSE and Perceptual losses [58], [75]) between the input image and the reconstructed image, may not be enough to guide the training of the encoder network. Wei et al. [76] proposed the method of training the encoder in cooperation with an optimization-based iterator. One more possibility for getting a better inversion quality is to compute the loss based on SSIM [77], Identity loss [61] and LPIPS [58]) to train an encoder that maps RGB into $\mathcal{W}$ space. An additional option is encoding into the $\mathcal{W}+$ latent space that provides finer control over the generator, utilizing regularization methods while training of such RGB to $\mathcal{W}+$ encoder [68].

3) Fine-Tuning of the StyleGAN Generator: When an image of a face includes something outside the training distribution e.g., a tattoo (see Fig. 12), it is difficult to find a good inversion, as there is no such a point in the latent space of StyleGAN that allows for reconstruction of such details. In this case, a possible solution is to fine-tune the StyleGAN generator weights themselves, using the target image or a set of target images [70]. This motivates a set of methods that operate directly on the generator weights to improve the inversion quality of a given image.

Before starting such fine-tuning of the StyleGAN generator, the target image must be first inverted into StyleGAN's latent space (see Fig. 7(c) and (d)) to the best possible reconstruction match (see Fig. 7(a) and (b)) using the previously discussed approaches. Then the StyleGAN generator model can be fine-tuned using loss-functions applied to the reconstructed and target images (see Fig. 7(c) and Chapter $V-D1$).

Fine-tuning the StyleGAN generator by gradient-based optimization (Fig. 7(c)) for each new image requires a couple of minutes of computation. This makes such methods difficult to apply in practice. By analogy with encoder-based methods for predicting the inversion latent vector (see Fig. 7(b)), we can, here too, seek an encoder whose output is used to modulate the weights of the StyleGAN generator (Fig. 7(c)). An example of such an approach is *Hypernetworks* [71]. The created *Hypernetwork* is trained to scale channel-weights of kernels of selected layers of the StyleGAN generator to match the target image and image generated by StyleGAN from the latent code (see Fig. 7(d) and Section V-D3).

B. Evaluating GAN Inversions

1) Reconstruction Quality Vs. Editability in StyleGAN: In addition to preserving similarity to the original image, the central motivation of the inversion step is to facilitate further latent editing operations. There exist a variety of points in the latent space that result in similar images to the original one, some of these points are more suitable for latent editing than others [68], [78], [79]. A successful encoding of a real image into a latent space should enable decent editability via the latent code.

2) Distortion, Perceptual Quality, Editability: Following the above observations, inversion methods and reconstruction quality should be evaluated based on several components: **distortion**, **perceptual quality** and **editability**. Simply put, **distortion** is a dissimilarity (e.g. MSE, SSIM, LPIPS) between the original and reconstructed images [56]. Distortion alone, however, does not capture the quality of the reconstruction. **Perceptual quality** measures how realistic the reconstructed images are (e.g. adversarial discriminator), with no relation to any reference image [56]. **Editability** stands for maintaining high perceptual quality of the image generated from the *edited* latent code [68]. The work [56] proved that there exists an explicit trade-off between distortion and perceptual quality. Therefore, the distortion, perceptual quality and edibility of the reconstructed images must be evaluated to provide a complete evaluation of the inversion method.

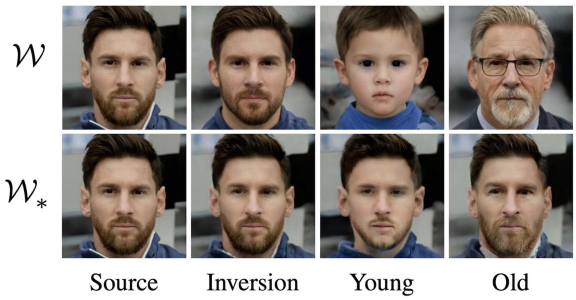


Fig. 9. Editability gap in $\mathcal{W}_*$ space. In paper [68] the space of vectors in $\mathcal{W}$ that are out of the mapping network manifold was denoted as $\mathcal{W}_*$.

3) Distortion Vs. Perceptual Quality & Editability Trade-Off: All possible output vectors of the StyleGAN mapping network constitute its latent space $\mathcal{W}$, where the input vectors of the mapping network are normally distributed vectors in $\mathcal{Z}$ space. Latent vectors generated as the result of an inversion method may, however, not necessarily lie within this manifold of the StyleGAN mapping network $\mathcal{W}$. In [68] and Fig. 9 the set of such vectors is denoted as $\mathcal{W}_*$. Moreover, manipulation via the inverted image code can be performed not only in $\mathcal{W}$ or $\mathcal{W}_*$ latent spaces of StyleGAN, but also in $\mathcal{W}+$ space, where vectors for 18 layers of StyleGAN2 are independent but each 512-dim style vector belongs to $\mathcal{W}$ space. In $\mathcal{W}$ space, reconstructions have the highest distortion, but good editability and perceptual quality. In $\mathcal{W}_*$ space (see Fig. 9), reconstructions have the lowest distortion, but the worst editability and perceptual quality after editing. In $\mathcal{W}+$ space, reconstructions have low distortion, good editability and perceptual quality.

C. Inversion Encoders

Inversion encoders solve the GAN inversion task in the form of a direct mapping from an image onto a latent code. Ideally, the concatenation of the GAN and the Inversion Encoder should produce the identity mapping. An early version of this idea came up in BiGAN [80], where the forward and backward mappings are developed simultaneously. However, with regard to StyleGAN inversion, the inverted part of the GAN is not the entire path from the image to its z-input, but to one (or several) of its intermediate latent spaces (see Chapter IV).

1) Psp Encoder: Image-to-StyleGAN: The idea behind the *pixel2style2pixel* (pSp) inversion encoder into $\mathcal{W}+$ space [67] is based on the fact that different layers of the StyleGAN architecture correspond to different levels of generated detail (coarse,

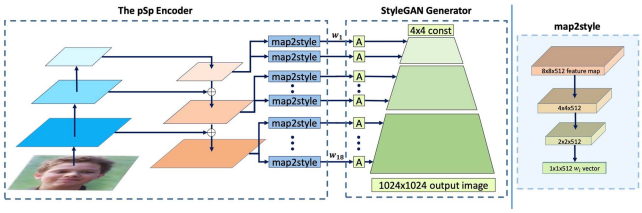


Fig. 10. pSp encoder architecture [67].

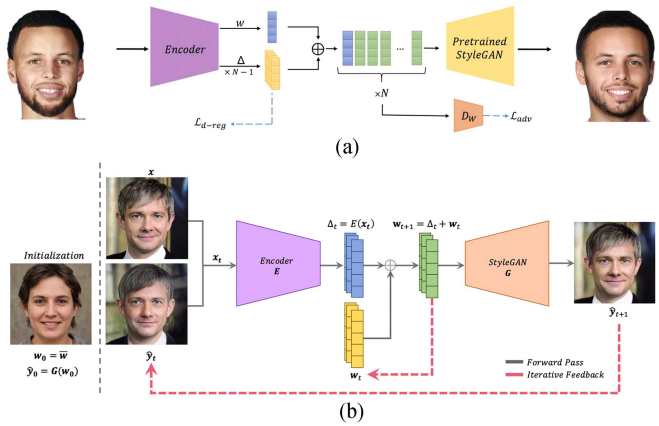


Fig. 11. (a) e4e encoder architecture with *progressive* training of encoder sequence by combining regularization, adversarial and distortion losses (not depicted) to improved control of editability-distortion trade-off [68]. (b) *ReStyle* iterative inversion scheme [69].

medium, and fine). Similarly, different layers of CNN-encoder feature maps also correspond to different levels of detail (coarse, medium, and fine). In the pSp encoder (see Fig. 10) feature maps are first extracted using a standard feature pyramid over a ResNet backbone. For each of the 18 target styles, a small mapping network is trained to extract the learned styles from the corresponding feature map, where styles (0-2) are generated from the small feature map, (3-6) from the medium feature map, and (7-18) from the largest feature map. The mapping networks, *map2style*, are small fully convolutional networks, each of which generates a 512 Δ_i vector that are added to $\overline{w}$, where $\overline{w}$ is the mean of the distribution of the *mapping network* output vectors when sampling in Z space [67]. The resulting 18×512 vectors $\overline{w} + \Delta_i$ in the $W+$ space are fed into the StyleGAN, starting from its matching affine transformation, A (see Fig. 10) [67]. The pSp encoder is trained using the L_2, LPIPS [58], Identity [61] and additional regularization loss functions. For the training of the encoders the weights of the StyleGAN generator are frozen, leaving as the adapted components only the ResNet backbone, the upsampling layers, and the *map2style* mapping networks.

2) e4e Encoder for StyleGAN Image Manipulation: The authors of *encoder4editing* or simply *e4e* (Fig. 11(a)) [68] approach took the above described pSp architecture [67] (Fig. 10) as their starting point and proposed a novel encoding scheme. The encoder's input is an image and the encoder output is a tensor of shape 18×512 in the $W+$ space, whose first 1×512 values specifies the StyleGAN2's w vector, and its remaining 17×512 values represent the Δ_i vectors for the last 17 layers of the StyleGAN2 generator. Now, instead of training all components of this tensor simultaneously, the authors propose a *progressive* learning scheme that they structure in the following way: first, learning starts with setting all 17×512 $\Delta_i = 0$, thus training only the first 1×512 values for matching StyleGAN2 w vector. Subsequently, the mapping networks sequentially unfreeze learning of their Δi outputs for the higher layers $i = 2 \ldots 18$ of the StyleGAN2, in ascending order. This scheme allows the encoder to first learn a coarse reconstruction, to which it then learns to add increasingly finer details.

Additionally, they propose a refined regularization scheme for the 17×512 Δ_i vectors to keep $w + \Delta_i$ in the space $\mathcal{W}$ to achieve a better editability, due to the individual adjustment of the style vector w for each layer of generator, along with improved perceptual quality. This regularization scheme employs three parts: a L_2 regularization loss is used to minimize the joint variance of all Δ_i ($\mathcal{L}_{d-reg}$ in Fig. 11(a)). A second part is a loss derived from a latent discriminator ($\mathcal{L}_{adv}$ in Fig. 11(a)) which is trained in an adversarial manner to discriminate between latent codes from the "true" $\mathcal{W}$ (obtained by feeding samples from $\mathcal{Z}$ to the StyleGAN encoder), and the encoder's learned latent codes. Finally, to make the Δ_i also to contribute to distortion reduction, additional distortion losses (L_2, LPIPS [58], and Identity preservation ArcFace [61]) are added.

3) ReStyle: Iterative Inversion Refinement: The *ReStyle* [69] method differs from typical encoder-based inversion methods, which infer the inverted latent code of the input using a single forward pass, by adding an iterative inversion mechanism with an additional feedback input. (see Fig. 11(b)). The encoder is fed with the output of the previous iteration along with the original input image, using several forward passes. This approach enables the encoder to focus on the relevant regions while leveraging the knowledge acquired in earlier iterations.

This method begins with an initial reconstruction $\hat{x_0} = G(w_0)$ of a source image x, generated from an initial representation w_0. To predict a sequence $w_t, t = 1..N$ of image style codes, this method performs $N > 1$ steps. The final inversion $w = w_N$ and its corresponding reconstruction $\hat{x} = G(w)$ are the final results. At each step t, the encoder E receives an input $x_t = (x, \hat{x}_t)$ consisting of the original image x, paired with its reconstruction $\hat{x}_t$ from the most recent time step, to compute a refined new residual code $\Delta_t = E(x_t)$, which is added to the inversion code of the source image as $w_{t+1} = \Delta_t + w_t$. This new latent w_{t+1} is passed through the StyleGAN generator again to update image reconstruction $\hat{x}_{t+1} = G(w_{t+1})$, which is used in the next iteration.

The *ReStyle* approach demonstrates better L_2, LPIPS [58], and Identity [61]) metrics than encoder-based approaches [67], [68] delivering high-quality reconstructions while maintaining fast inference times.

D. Fine-Tuning of the StyleGAN Generator

Images of real faces often contain various unique details such as tattoos, scars, fashion elements or light. It is challenging to apply identity-preserved editing to such out-of-domain face images even with the previous methods. Inversion of such face images may lead to poor results far away from the generator's domain, because the nearest image in this domain may not have all these details. As a result they will be lost after the editing process (see Fig. 12).

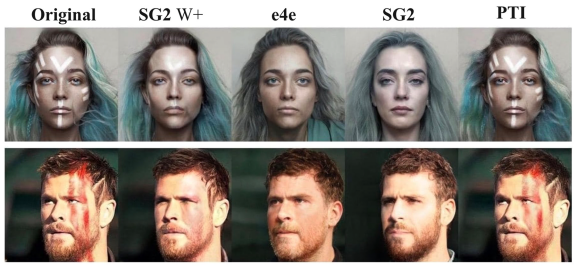


Fig. 12. Comparison of *Pivotal Tuning* [70] and other inversion methods (see [1] [5], [70] and Section V-C2)).

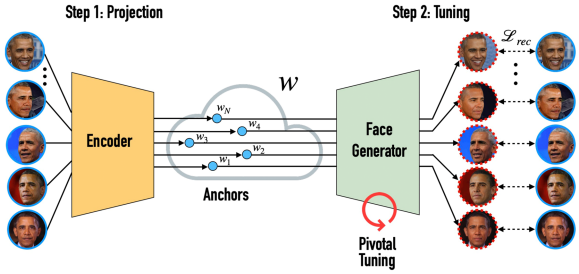


Fig. 13. *MyStyle* fine-tuning scheme of the StyleGAN generator [7] using *pivotal tuning* around a large number of *anchors* given by a collection of views of the face of a single person.

1) Latent-Based Editing of Real Images: To better cope with the inversion problem of unique face details (e.g., tattoos) the *Pivotal Tuning* [70] method adds a tuning step for the StyleGAN generator. The method inverts the source image x to style code $w_p \in \mathcal{W}$ in the native latent space of StyleGAN, which will be called as the *pivot* code. Let $x^p = G(w_p; \theta)$ be the image generated from the *pivot* code w_p when using the generator with weights θ. Then, the tuning step consists of adapting the weights between the *pivot* code and the output to bring x^p closer to x using LPIPS and L_2 losses. The mapping network layers between z and w remain frozen. However, it is important to constrain the changes of the tuning such that they only affect the reconstruction mapping within a neighborhood of the *pivot* code w_p. To achieve this, they introduce a suitably designed regularization term, implemented in the form of an iterative process. At each iteration, a z value is sampled from the standard normal distribution and the StyleGAN mapping network produces corresponding style code w_z. The difference $w_z - w_p$ and w_p gives a direction away from the *pivot* point whose length is scaled to the value of a coefficient α. The resulting end point w_r of this scaled vector, attached to the *pivot* point w_p (see (3)) is a code in a certain neighborhood of the *pivot* w_p:

$$w_r = w_p + \alpha \frac{w_z - w_p}{\|w_z - w_p\|_2}. \tag{3}$$

It contributes a regularization step for the fine-tuned StyleGAN generator by adapting the weights θ^* of the latter towards minimizing the distance between the image pair (x_r, x_r^*) obtained from w_r with original generator $(x_r = G(w_r, \theta))$ and the tuned generator $(x_r^* = G(w_r, \theta^*))$, using LPIPS and L_2 loss functions. This neighborhood-restricted tuning process ends up altering appearance features that represent mostly face identity, without interfering with editing capabilities.

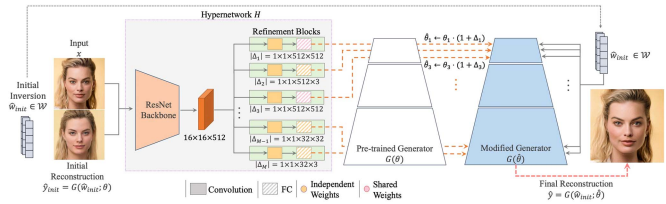


Fig. 14. HyperStyle method illustration [71].

2) MyStyle: A Personalized Generative Prior: The MyStyle architecture [7], [81] (see Fig. 13) extends the idea of *pivotal* fine-tuning from a single to hundreds of *pivots*, all taken to be portrait images (called *anchors*) of a given person. The authors propose to collect about 100 *anchor* examples that form a *personalized region* in the latent space $\mathcal{W}$ of the StyleGAN2 generator and fine-tune the StyleGAN2 generator on these examples. This allows it to use the StyleGAN2 generator as backbone for super-resolution (see Section VIII-C), inpainting or semantic editing tasks for a given person. This *anchor* space was described by Generalized Barycentric coordinates, which allows us to detect when a latent point is inside the convex region of known *anchors* in the latent space, and when a latent point is outside this region. For semantic editing task is necessary to preserve personality. Existing directions in the latent space are learned for the whole domain of faces and are not personalized. Because of this, some directions may lie outside the *anchor* space and any small step in this direction will lead to degradation of the quality of the edited image. To solve this problem we need to project the direction on the *anchor* space and perform editing in it [7].

3) HyperStyle: StyleGAN Inversion With HyperNetworks for Real Image Editing: The *HyperStyle* [71] approach proposes to combine several ideas of the previous discussed approaches of *ReStyle* [69] (iterative improvement with inversion encoder), *Pivotal Tuning* [70] (fine-tuning of StyleGAN generator weights for a specific image of a face), and *e4e* [68] (initial inversion prediction of $w \in \mathcal{W}$) for fine-tuning the pre-trained StyleGAN generator on the fly.

Key idea is to learn to predict coefficients for a channel-wise scaling of weights of selected layers of the pre-trained StyleGAN generator (see Fig. 14) to personalize it for a given image of a face. In this way *HyperStyle* is computationally much lighter direct mapping for fine-tuning coefficients of the StyleGAN generator, than the gradient based fine-tuning approaches like *Pivotal Tuning* [70]. The required scaling of channel weights is parameterized as $\hat{\theta}_l^{i,j} := \theta_l^{i,j}(1 + \Delta_l^{i,j})$, where $\theta_l^{i,j}$ denotes the weights of the j-th channel in the i-th filter in the StyleGAN generator's l-th layer and $\Delta_l^{i,j}$ is the output of the Refinement Blocks (see Fig. 14). For a better trade-off between network expressiveness and feasibility of learning, the *HyperStyle* model is trained to predict scaling coefficients only for kernels of selected StyleGAN generator layers. As the initial inversion captures coarse details, only layers that are responsible for medium and fine details may be selected. The *HyperStyle* architecture with iterative refinement of StyleGAN weights (depicted in Fig. 14) consists of a *ResNet* backbone that receives a source image with its initial reconstruction and Refinement Blocks of convolutions and fully-connected layer. Training is guided by an image-space

reconstruction objective through pixel-wise L_2, LPIPS [58] and ArcFace [61] losses.

VI. Editing Face Images With StyleGAN

The main idea about editing an image using StyleGAN is that editing is achieved through some manipulation of its latent code, thereby moving the point that represents this latent code within one of the latent spaces of StyleGAN (see Section IV). At first we do not know how movement in the latent space will affect the generated image but there are methods to learn how to navigate the latent Space to edit an image in a more controlled and semantically meaningful manner. Movement of the point in a wrong or a random direction will in the worst case lead away from the face distribution (thereby destroying the "faceness" of the image), or lead to an undesired, simultaneous change of different attributes, most likely accompanied by a loss of identity of a person on the image.

In this chapter we will focus on methods to identify directions in a latent space of StyleGAN which are correlated with desired, *semantically interpretable editing attributes*, like smile, pose, hairstyle, age, and so on. Also, we will discuss how the different latent spaces of the StyleGAN architecture differ with regard to their editing properties.

Identifying semantic directions can typically be divided into global semantic directions (see Section VI-A) and single image semantic directions (see Section VI-B). Section VI-A1 introduces a supervised approach for finding semantic directions. Section VI-A2 offers an unsupervised approach for discovering semantic directions by using PCA. Section VI-A3 describes discovering directions in StyleSpace using only few points with known attributes. Section VI-B1 introduces a text-driven approach for discovering semantic directions.

A. Global Semantic Directions

The first class of approaches is based on averaging a direction in latent space correlated with the given attribute over a number of contrastive pairs, with and without a desired attribute.

1) Identifying Semantic Directions in a $\mathcal{W}$ Space: InterFaceGAN [82] proposes a supervised method for identifying semantic directions in a StyleGAN's latent space. Suppose there is a pre-trained binary classifier for some attribute we want to edit. We make an assumption that for any binary semantic attribute there exists a hyperplane in the StyleGAN latent space serving as the separation boundary between points of positive and negative examples of that attribute. To find a suitable separating hyperplane that represents editing direction in the latent space $\mathcal{W}$ of the StyleGAN, InterFaceGAN [82] proposes to apply a linear SVM to points of positive and negative examples of the selected semantic attribute. They generate images for 500,000 randomly selected latent points in $\mathcal{W}$ and use the scores of a pre-trained ResNet binary classifier to identify a subset of 10,000 images for which the classifier reports the highest scores (attribute is present), and 10,000 images with the lowest scores (attribute is absent). Thus, to manipulate the attribute of an image, we can move the original latent point along the normal vector of the hyperplane.

When we edit several fine-grained attributes, one may affect another. To achieve a more disentangled editing it was proposed to orthogonalize a discovered set of semantic directions [82]. For example, given two hyperplanes for two semantics with normal vectors n_1, n_2, then we can find a projected direction $n = n_1 - (n_1^T n_2)n_2$ such that editing along this projected direction correlates with the first attribute, but is not affecting the second attribute.

Fig. 15. Moving along the tenth principal component in 7-8 layers changes hair color [83].

2) Discovering Interpretable Semantic Directions in $\mathcal{W}$: In contrast to the previous, supervised method, GANSpace [83] describes an unsupervised approach for discovering semantic directions. The method, which requires only a pre-trained StyleGAN generator, proposes sampling of a large number of random points in $\mathcal{Z}$ space, collecting the corresponding points in $\mathcal{W}$ space, and applying PCA to obtain a basis in $\mathcal{W}$ space. These PCA basis vectors contain the semantic directions responsible for some attributes. Within their study, using on the order of 100 basis vectors, the authors find that large-scale changes to geometric configuration seem to be limited to the first 20 principal components. For a further refined control of editing features, these PCA directions can be applied individually to each of the 18 StyleGAN2 layers. By checking the influence of the discovered PCA directions, it is possible to identify directions responsible for specific face editing attributes, for example hair color (see Fig. 15). The authors show that such a check – in their work of about 1800 combinations – can be done both automatically or manually.

3) Identifying Semantic Directions in StyleSpace ($\mathcal{S}$): The search for semantic directions can be done not only in the $\mathcal{W}$ space, but also in the more disentangled StyleSpace ($\mathcal{S}$ space, see Section IV-D) [66]. In this space, identification of semantic directions can already succeed with as few as 10 to 30 positive examples that contain the target attribute. This utilizes the idea that the differences between the mean style vector of the positive examples and the mean of the entire generated distribution of 500 k samples in $\mathcal{S}$ space reveal which StyleGAN channels are the most relevant for the target attribute [66]. Such channels can be identified, e.g., with statistical methods that determine which vector components show statistically significant deviations towards higher values. Once we know the right channel, we can change its activation along the corresponding $\mathcal{S}$ dimension so that the generated image shows an increase or decrease of the desired semantic attribute.

4) Text-Driven Discovering of Semantic Directions: It is also possible to search for a global semantic direction using combined image-text embeddings, e.g., using the CLIP model [4], [26]. For example, to find the semantic direction of the attribute *smile* in the latent space of CLIP, we can take the direction between text embedding vectors such as *face with smile* and *face*.

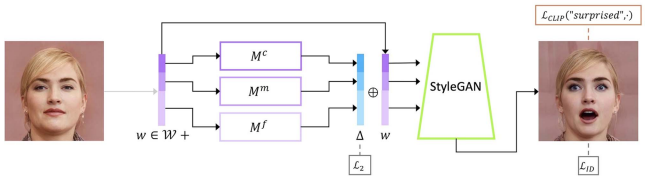


Fig. 16. *StyleCLIP* mapping network architecture [4]. For a fixed "surprised" text attribute three mapping networks M^c, M^m, and M^f for coarse, medium and fine scale are trained for images to map their latent vector w (blue bar, left) of images into a short change direction Δ that, if added to w (blue bars, right side), moves the generated output towards better satisfying (lower CLIP loss of the new image-text pair) the chosen text attribute.

Fig. 17. Examples of reference-guided BlendGAN synthesis [86].

To find the global direction of the example attribute *smile* in the StyleSpace $\mathcal{S}$ of StyleGAN for some image, the coordinate of its latent point is moved in the positive and negative directions for a selected channel. This produces a pair of generated images ($\pm\sigma$) that are fed into CLIP to measure how the perturbed direction of the chosen StyleGAN channel in $\mathcal{S}$ space is correlated with semantic direction of *smile* in the latent space of CLIP. This process is repeated for a number of images always using the same channel in $\mathcal{S}$ space, to compute the corresponding averaged direction in CLIP space. Projection of this computed average direction on the semantic direction of attribute *smile* in the latent space of CLIP provides a direct measure how much the chosen channel in $\mathcal{S}$ space affects the selected semantic direction in CLIP space. After going through all the StyleGAN2 channels, those with projection values greater than a certain threshold are selected as related to the given semantic direction in the $\mathcal{S}$ space.

B. Single Image Semantic Directions

The second class of approaches optimizes a single image and requires an assessment network over the generated image, like CLIP or a binary attribute classifier. Loss from such an assessment network produces a gradient step in latent spaces which indicates the direction to move the input point of the generator to decrease the target attribute loss.

1) Text-Driven Manipulation of StyleGAN Imagery: The *StyleCLIP* [4] offers a text-based approach to image editing (technical details are already described in VI-A4). The CLIP-model produces gradients through the CLIP image-encoder to minimize the similarity loss of text and image points in the combined latent space of CLIP, thus allowing to obtain the overall gradient that indicates the editing direction for the point in the $\mathcal{W}$ latent space of StyleGAN with regard to the textual attribute that is represented in CLIP. In addition, Identity [61] and L_2 losses on the StyleGAN generated image allow to penalize loosing the person's identity in the generated image under text-attribute driven changes. However, the disadvantage of this method is the need to conduct computationally expensive gradient based optimization steps for each image and text attribute, because each pairing of image and text attribute gets its own editing direction in a StyleGAN latent space.

As seen several times before, this problem of computationally expensive gradient based optimization can again be remedied by training an additional mapping network (Fig. 16). Each text attribute that is of interest for editing, e.g. *smile*, requires to train a separate mapping network. This *StyleCLIP* mapping network [4] moves the point in $\mathcal{W}+$ space according to the text attribute. The *StyleCLIP* mapping network employs an architecture with three prediction sub-networks (denoted as M^c, M^m and M^f in Fig. 16) for the coarse, medium and fine layers of the StyleGAN2 generator respectively. Each sub-network takes the corresponding $\mathcal{W}+$ dimensions as input and predicts Δ_w for them (see Fig. 16). The overall gradient guidance by the CLIP loss of the text-image pair is supplemented by terms to keep the result image in the vicinity of the input image (small L_2 loss) and to preserve identity (identity loss L_{id}). Authors employ the CelebA-HQ dataset [48] for training.

2) Interpretable Control: Facial Pose and Expression Change Based on FaceRig: To be able to create editable faces as flawlessly as possible, controlling semantic face parameters that are interpretable in 3D like face pose and expressions play a crucial role. StyleRig [11] by Tewari et al. offers nice results using both three-dimensional morphable face models (3DMM) in combination with StyleGAN to provide such control. (See Fig. 1(k) for StyleRig examples)

VII. Cross Domain Face Stylization

The original StyleGAN architecture allows mixing styles of two sources of information (see Fig. 3) from the same training domain, for example CelebA-HQ dataset [48]. However, what if we want to mix styles of two sources of information from different domains, for example mixing with cartoon faces (Fig. 17)? This chapter discusses approaches for style mixing from different domains by fine-tuning or merging the StyleGAN generators.

StyleAlign [84] explains why we can do cross domain face stylization using StyleGAN fine-tuning to another domain (see Section VII-A). The layer-swapping approach [85] devises a controllable domain adaptation between original and fine-tuned StyleGAN models (see Section VII-B). BlendGAN [86] proposes a style encoder for fine-tuning face StyleGAN such that its output becomes adaptable to an arbitrary style (see Section VII-C). StyleGAN-NADA [10] (see Fig. 18) proposes fine-tuning of the StyleGAN generator towards a target style domain using CLIP (see Section VII-D).

A. Fine-Tuning Generator to the Target Style Domain

Let's consider two similar style domains - source and target. We can fine-tune the StyleGAN generator pre-trained on the source domain towards the target domain. To stylize a source image we can invert it into a latent code and generate from

Fig. 18. StyleGAN-NADA [10] allows to create images that do not even exist in real life ("Nicolas Cage Dogs" using a generator trained to generate dogs and the "Nicolas Cage" text prompt).

this code using the fine-tuned generator. StyleAlign [84] work suggests that the same latent code $z \in \mathcal{Z}$ is mapped to similar codes in $\mathcal{W}$ in source and target domains, which is called *point-wise alignment*. Moreover, changing individual channels in $\mathcal{S}$ or moving in directions in $\mathcal{W}$ leads to the same semantic changes in generated images of source and target domains. This alignment can be measured by calculating the overlap between channels in $\mathcal{S}$ used for semantic editing (see Section VI-A3) in original and fine-tuned generator, which is called *semantic alignment* [84]. These alignments explain why fine-tuning based methods can be used for cross domain face stylization.

B. Layer Swapping

Since we know that fine-tuning can be applied to cross domain stylization, we need methods to better control the stylization. *Point-wise alignment* doesn't guarantee that we will preserve all the necessary features from the original face after sampling from a tuned generator. So in [85] a layer swapping technique was proposed (see Fig. 19). This technique allows to control which features we want to transfer from a target domain.

Layer swapping domain adaptation is done by combining selected layers from the original and fine-tuned generator. Now we can transfer, for example, low level features from the target domain such as face geometry or hair style (see Fig. 19(c)) or transfer only high level features (see Fig. 19(d)).

If it's not enough to swap layers, for example, when the shape variations between the photo domain and the target domain are too large to successfully capture the exaggerations and color stylization of caricatures using the layer swapping technique, we can use additional blocks [34] appended to the layers of the first StyleGAN blocks to modify coarse features.

C. Blending for Stylized Face Generation

Paper [86] suggested an alternative to the layer swapping mechanism which allows to generate images in different domains using only one model. It proposed to fine-tune a StyleGAN model on a dataset of faces in different artistic styles (see Fig. 17). A separate encoder is trained using contrastive learning to extract style latent code z from artistic images. The encoder is trained on augmented variations of stylized images of faces to produce the same latent vector for geometric augmentations of the same style image, but distinct to latent vectors of other style images. Thus, the encoder becomes insensitive for facial details, but focused on the style of an artistic face image. The latent space of this encoder is connected to the latent space $\mathcal{W}$ of StyleGAN via a second mapping network of 8 fully connected layers.

Fig. 19. [85] (a) and (b) are samples of the "Base" and "Transferred" trained models, respectively. (c)–(e) are "Interpolated" results of different combination of layers from the "Base" and "Transferred" models.

The blending procedure is similar to the vanilla StyleGAN merging of two sources of information. The resulting image is generated by taking two points in the latent space $\mathcal{Z}$. One z point that represents a face is being transforming into the latent space $\mathcal{W}$ using the original mapping network and used for some first layers of the StyleGAN model. The second z point that represents a style is obtained from the reference artistic image by the encoder. This point is being transformed into the latent space $\mathcal{W}$ using the second mapping network and used for other layers of StyleGAN.

D. CLIP-Guided Domain Adaptation of StyleGAN

Default fine-tuning and layer swapping requires a large dataset of stylized images from the same domain. There are some advanced methods for fine-tuning a generator to a target domain without large stylized datasets. StyleGAN-NADA [10] allows to fine-tune the StyleGAN generator using CLIP based gradients towards a different style domain using only text prompts, for example "zombie faces". For that, first the semantic direction of the prompt is identified in CLIP latent space by subtracting embeddings of two text prompts "zombie face" and "face". Using the CelebA-HQ dataset [48] for training, the authors demonstrate that the semantic direction in CLIP latent space can be used to fine-tune a StyleGAN model [48] towards domain defined by the text. To this end, thousands of images of faces are generated by sampling a latent space of the trained StyleGAN model. Then, according to the loss, embeddings of these images in CLIP space have to move in a parallel direction to the found semantic direction in CLIP text space. This parallel movement of thousands of points in the CLIP latent space becomes the training objective for changes of the StyleGAN model weights.

To solve overfitting and divergence problems, [10] proposes to fine-tune only StyleGAN layers with the most influence on the new domain and freeze weights of the rest of the layers. To determine fine-tuned layers, the approach moves these latent image points in CLIP space to the text embedding of the target domain and chooses those layers in which the latent code has changed the most. These layers are considered as sensitive to the new domain. Further only these layers are unfrozen and generator weights are fine-tuned by above mentioned method.

VIII. Face Restoration With StyleGAN

Face restoration aims at recovering high-quality (HQ) face detail from low-quality (LQ) counterparts suffering from unknown

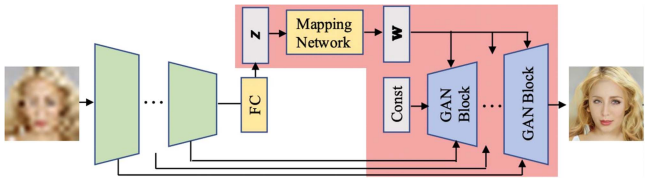


Fig. 20. GPEN architecture [27]. A CNN encoder network receives a LQ input (left) and has learned to provide z-input and via skip connections control inputs to additional input fields in the GAN generator layer hierarchy (right) to cause generation of a corresponding HQ output image.

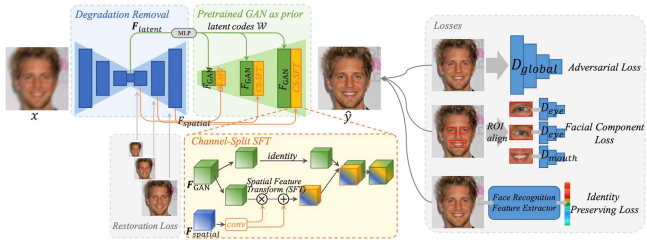


Fig. 21. GFP-GAN architecture [6].

degradation, such as low resolution, blur, compression artifacts or noise. Context information may allow to properly infer such missing detail. For example, location of hairs on a blurry face can be estimated from low resolution details of a face. Exact location of each hair is not important for human perception, but it is important to restore overall structure and texture. Thus one may expect that supervised training of neural network architectures with LQ-HQ training pairs using pixel-wise losses will not lead to high perceptual quality. Instead, pixel-wise loss functions cause over-smoothed result images, as the model tends to be the mean of high-quality faces.

However, architectures with energy-based components, like GAN discriminators, can be a good approach for reconstruction of fine detail from low resolution images and may provide perceptually plausible results. In the fully converged state, the discriminator will not be able to differentiate between a generated sample and a sample from the dataset. Ideally, the same will then also apply to human perception. Thus, we can expect GANs to be able to generate deblurred images of perceptually high-quality, but there remains the challenge of mapping a blurred input into an appropriate latent vector of StyleGAN that makes StyleGAN generate the corresponding deblurred version.

A. GAN Prior Embedded Network for Blind Face Restoration in the Wild

The GAN Prior Embedded Network (GPEN) [27] is one of the solutions for blind face restoration problem. The main idea of this approach is to embed a StyleGAN like architecture as the decoder part into an encoder-decoder architecture, thereby utilizing its strong prior for generating high-quality faces. The LQ image is fed to the encoder part, which is realized as a conventional CNN whose output serves as the latent code for the subsequent GAN decoder to generate an appropriate HQ image (see Fig. 20). To this end, the activations of the final CNN output layer are fed in place of the random vector into the GAN's latent space mapping network. Furthermore, to make efficient use of the controllability of the StyleGAN decoder, the GAN is slightly modified by padding its generator layers with additional input fields for receiving activity patterns via skip connections from more shallow layers in the encoder (see Fig. 20). They are chosen such that the feature map hierarchy of the encoder CNN and the GAN generator layers become connected at matching levels of resolution.

In the first phase the GAN network (for example the StyleGAN generator) is trained to generate high quality faces from scratch. This phase uses the FFHQ dataset. During this phase, the added channels of the GAN layers are not yet used by the encoder and the GAN generator network is encouraged to learn ignoring signals from these noise channels by just providing random noise for these inputs. Thus, in this phase the noise channels play a placeholder role for a subsequent fine-tuning of the GAN prior network. In the second phase of training, the encoder is included, now replacing the random inputs to the GAN by the activations of matching encoder layers.

For the second phase, training the whole architecture, a dataset of pairs of low quality (LQ) and high quality (HQ) images was synthesized. HQ images are downgraded to obtain LQ images using blurring, gaussian noise, downsampling and JPEG compression. A weighting of three losses was used to train the GPEN architecture: L_1, adversarial and LPIPS losses (see Section III-A and III-C) based on features from discriminator, rather than VGG network.

B. GFP-GAN: Towards Real-World Blind Face Restoration With Generative Facial Prior

Generative Facial Prior GAN (GFP-GAN) [6] is a framework for real-world blind face restoration applications (see Fig. 21). It resembles GPEN (see Section VIII-A), but employs a degradation removal module in the form of a U-Net as an encoder, along with with a number of connectivity changes that allow the usage of a pre-trained StyleGAN as a decoder without the need for a training from scratch and subsequent fine-tuning of a modified StyleGAN, as in GPEN.

To achieve this, the U-Net decoder features are fed to a channel split-spatial feature transform (CS-SFT) module that learns to modify activations in StyleGAN layers for better perceptual quality of the generated image with identity preservation. The interaction between the standard StyleGAN and the newly introduced spatial channels are shown in the CS-SFT box within Fig. 21. The SFT operation happens as follows. Spatial features from the U-Net go into a convolutional module generating an output tuple (α, β) with parameters for spatial-wise feature modulation. Parameter α is being used for element-wise scaling of GAN features, β specifies an additive shift. After this, the obtained modulated channels are concatenated with the untouched GAN channels to form the final result, as depicted in box "Channel-Split SFT" in Fig. 21.

Again a composition of losses is applied to train the entire architecture. First, motivated from the U-Net, a pixel-wise reconstruction loss L_1 between the degraded input image and ground truth version of the image at different scales (see the blue U-Net in Fig. 21) is used. Second, the restored output image $\hat{x}$ of the StyleGAN generator can be compared with the degraded input

Fig. 22. Personalized super-resolution results using MyStyle [7], [81].

image x using LPIPS, and Identity losses (see Section III). Third, since the eyes and mouth are crucial to a person's personality, a further Facial Component loss is introduced which consists of local discriminators for the left eye, right eye, mouth, and the complete face. The Region of Interest Align operator provides these discriminators with corresponding regions of the generated face.

C. MyStyle: A Personalized Generative Prior

We have described this architecture [7], [81] already in some detail in Section V-D2. Here, from the perspective of HQ face image restoration (see Fig. 22), we only remark that this architecture also applies a StyleGAN to solve the problem of super-resolution, using L_2 pixel and LPIPS losses, with a downscaling operation applied to the images before application of these losses.

D. VAEs With Codebooks - Alternative Approaches

Codeformer [87] and VQFR [88] are encoder-decoder based approaches that don't use StyleGAN infrastructure, however achieve similar blind face restoration results. One common property of these approaches is that they use a discrete latent space (VQ-VAE codebooks [89]). Also the spatial resolution of the bottleneck part and modulated skip connections have an important role in restoration-identity preservation trade-off.

IX. Deepfakes

Deepfake: is the manipulation of an image of a face that is hardly recognizeable by humans. Deepfake creation methods can be classified into the following types (Fig. 23):

Synthesis: a deepfake face is created without a face image. Section II covers the synthesis of photo-realistic fake images of faces that do not exist.

Editing: facial features are altered, added or removed (see Section VI).

Reenactment: a source face is used to control gaze, mouth or expression of a target face [42], [90], [91]. Following articles focus on mouth reenactment [92], [93], gaze reenactment [94], and pose reenactment [95].

Replacement: the identity of the source face is used to replace the identity of the target face [96], [97], [98].

Replacement (Transfer): the identity and expression of the source face is used to replace the identity and expression of the target face, which can be perceived as consecutive face swapping and face reenactment [43].

This chapter will focus on **reenactment** (Sections IX-A) and **replacement** (Section IX-B) methods. The main difference between face reenactment and face replacement is that in face replacement, the identity of the source face is transferred to the target face.

A. Face Reenactment

Face reenactment (see Fig. 23(b)) is useful in the advertisement or in the film industry [38]. Typically, if the main actor (target) has limited time or high costs, another substitute actor (source) can perform the acting and the expressions are then transferred to the main actors face, as if he or she is actually performing [99]. In this case, the main actor's face would be the target face and the substitute actor would be the source. One usage of mouth reenactment could be voice dubbing into another language for advertisements, music or video games. Gaze reenactment can be used to improve photographs by making the people (target) look into the camera. Pose reenactment has been used for face frontalization in security footage (target) to improve face recognition [100].

Facial reenactment can be realized by StyleGAN and its latent code. For instance, for transferring expression from a source image x_s to a target image x_t the authors of [41] solve the following optimization problem to find the latent code s_r of the desired result image:

$$s_r = \underset{s}{\operatorname{argmin}}[Dist_1(G(s), x_s) + Dist_2(G(s), x_t)].$$

Here, $G(\cdot)$ is the mapping function of the generator, and $Dist_1(x, x'), Dist_2(x, x')$ are suitable distance functions in image space that measure similarity of a face image pair w.r.t. expression ($Dist_1$) and appearance ($Dist_2$) of the faces.

To solve this equation the authors constrain the solution space to particular linear combinations $s_r = \alpha s_s + \beta s_t$ of the StyleGAN latent codes of the given images. Here, α and $\beta = \mathbf{1} - \alpha$ are diagonal 18×18 matrices whose diagonal elements may only attain values 0 or 1. This is equivalent to adopting for the 18 style parameters of s_r either the corresponding value of s_s or s_t, leading to a finite search space of 2^{18} possibilities for the optimization problem. From their solutions, they discover that StyleGAN layer 4 is primarily responsible for expression, while other layers (e.g., 8 and 9) primarily control hair color and hat style [41].

B. Face Replacement and Face Transfer

Face replacement and face transfer are closely related. While face replacement swaps a face into the position of a previous face, face transfer additionally takes care to make the swapped face inherit the expression of the previous face, which is usually achieved with an additional face reenactment step.

Face replacement can be established using classical computer graphics-based approaches [101] by using facial landmarks extracted for each face. The results, however, are not as good as using neural network based methods.

FakeApp [96] used an encoder-decoder NN architecture to swap faces of two different people A and B. Expressions and emotions from the target image are kept, while the facial identity is swapped. First, it collects a dataset of faces of two people, A and B, using an object detection method [102]. Secondly, it trains two auto-encoders E_A, E_B to encode and two decoders DC_A, DC_B to reconstruct the faces of A and B respectively. The key idea is that the two encoders have to share the same weights, but they keep their respective decoder weights independent. This allows the encoder to learn global features of the two faces A and B while the two decoders are trained to use this general encoding

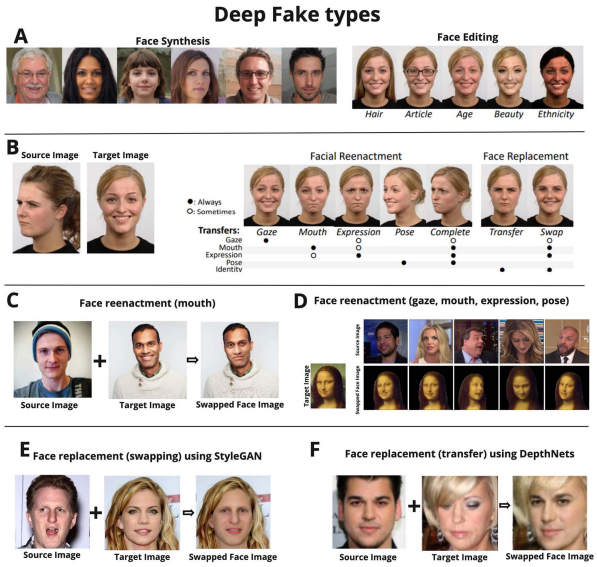


Fig. 23. (a) Face Synthesis and editing examples [38], (b) Description of the types of deepfakes [38], (c) Face Reenactment of the mouth [41], (d) complete face reenactment [42], (e) Face swapping utilizing StyleGAN [12], (f) Face Transfer using DepthNets [43].

to generate face specific details of person A or B respectively. Thirdly, to create the face swap, trained encoder E_A and decoder DC_B are used for the images of person A. This method is applicable for videos as well. The results are similar to Fig. 23(e) and (f). The downside of this approach is the large dataset of faces of both people that is needed to train the encoder-decoder networks. When using this method on videos, a big problem is the temporal coherence. As this method does not take preceding frames into account, it may produce flickering of the faces. This can be mitigated by providing context or implementing temporal coherence losses [103], [104].

The Face Transfer Module (FTM) [12] mitigates most of the problems described above. It can transfer faces without ever having seen them before. The expression of the face of the target image remains, but the facial identity is transferred from the target image (see Fig. 23(d)). It uses a trained StyleGAN model as a decoder and employs the Hierarchical Representation Face Encoder (HieRFE), [12] which is based on the ResNet50 network, to project images of faces into the improved latent space of StyleGAN with transformers [105].

Styletalk: method [106] creates lifelike animated faces from a single image of a speaker based on reference videos using 3D Morphable Models expression parameters. Another approach [43] tries to predict the 3D pose of the face and its facial landmarks from the 2D source image and predict the rotation and translation of the landmarks to map them onto the facial landmarks of the target image. For predicting the 3D poses of the facial landmarks, [43] uses an unsupervised Siamese-like network [107] consisting of convolutions, pooling layers, and densely connected layers that they named *DepthNet*. The two images pass through the network which then predicts the depth and affine transformation to map one face onto the other. A CycleGAN [108] is used to cleanly blend the new face position into the image.

X. Alternative Face Generation and Editing Approaches

A. 3D-Consistent Generative Adversarial Networks

NeRF [109] based approaches exhibit consistency of an object while altering position and camera orientation. Development of successive StyleGAN architectures goes along the improving of such consistency along different camera poses. Thus, a straightforward idea is to substitute the StyleGAN generator with a NeRF-like architecture with disentangled camera pose representation and integrate the idea of the mapping network and W latent space [110]. While the high resolution of StyleGAN (1024×1024) can be computationally expensive for training a NeRF architecture that would need to compute a batch of 1 M rays gradient updates, [110] proposed to generate an only (64×64) image/feature space with NeRF and upscale it to 1024×1024 using a generator modulated by W. Another performance improvement is proposed by [111] where the encoding of positional information is proposed to be represented by a Tri-plane Decoder, thus reducing NeRF to learn direction information for every position. To conclude incorporating of the NeRF components allows to achieve the required consistency of face generation for different poses and directions, while building on solutions accumulated through the development of the StyleGAN architectures.

B. Diffusion Models

Deepfake: diffusion models [112] [113], [114] have emerged as a result of advancements in diffusion-based methods. They are designed to generate realistic *Deepfake* videos with audio information used as a conditioning signal. The person's identity is preserved in the generated videos, conditioned by a single image of a face or a full video, and the movements of their lips and other facial features are synchronized with the audio. The U-Net-based model serves as the diffusion component.

Face editing: with diffusion models can also be conditioned by text instruction. In this way the *InstructPix2Pix* model [115] can edit images, including faces. To train such text-conditioned editing model, a dataset of triplets containing the original image, the edited image, and the text editing instruction, was generated [115]. Similarly, *ControlNet* [116] allows to control the generation of images (including faces) from two sides: on the one hand the generated image originates from a simplified representation of a given image (Canny Edge, Semantic Segmentation or Normal Map), and on the other hand it must fit a certain text prompt. The text prompt can be changed, thus allowing image editing. In papers [117] [118], pre-trained diffusion models are used to perform multi-modal guided face generation or editing.

XI. Conclusion

StyleGAN architectures [1], [17], [24] have been extensively researched as deep learning models for face generation and editing, making them the most studied in this domain. They are providing a versatile editing control over the generation process of high-quality images, which makes StyleGAN architectures particularly valuable.

Existing limitations of StyleGAN, and emerging competing techniques, like diffusion models, result in the following trends

and expectations for future research in the area of face generation and editing with deep neural networks:

- *Foundation Models:* Pre-trained StyleGAN models can be considered as *Foundation Models* and require further research on fast fine-tuning techniques, like neural network adapters for integration with other architectures [119] [120], [121], scaling StyleGAN to large diverse datasets [122] or new interactive image manipulation methods [123].
- *Mobile Applications:* Generative models on-mobile-device are an open research topic with a lot of challenges. A StyleGAN model is too computationally demanding for mobile devices or applications that require a high frame rate for videos. One common solution is to employ StyleGAN to produce a synthetic dataset for a specific single feature, like pairs of face with and without sunglasses. Then, this synthetic dataset is utilized to train a tiny, single-function-image-processing network for mobile applications (see Section I-B). Another solution is to distill a downscaled version of StyleGAN architecture from the original StyleGAN [124]. For example, *BlazeStyleGAN* [124] has around 16 times fewer parameters and around 16 times fewer FLOPs needed to generate a 1024^2 image than the StyleGAN architecture. As a result, *BlazeStyleGAN* is able to generate high quality images on mobile devices. However, further research is needed to find out how latent space works in compact models compared to the original model.
- *Diffusion Models:* are emerging competing approaches for face generation [125], [126] [127], editing [112] [113], [114], enhancement [128] [129], [130] and temporal consistency [131]. However, the evaluation step of such models is even more computationally demanding. Additionally, the control over the image generation and editing is so far not as well optimized as those of StyleGAN architectures. In addressing the aforementioned challenges, this work [132] presents an accelerated solution specifically tailored for the task of local text-driven editing of images.
- *Text2Face:* Recent works [133] [134] have shown that StyleGAN-based networks can be comparable in quality to diffusion models in Text2Image task, while having better performance. However, the quality of the generated images is still inferior to such strong works as Imagen [135] or DALLE-2 [136], so an open question is to improve the generation of images from text using GANs.
- *Neural Rendering:* The 3D consistency of a generated face when viewed from different angles is another crucial aspect of face perception. NeRF [109] is a technique that has shown impressive results for 3D scenes with faces [137], and there have been efforts to combine StyleGAN and NeRF [110] or to create 3D-GANs [138] [139], [140] [141] and solve inversion task for them [142] [143].
- *Temporal Consistency:* A number of works in this survey demonstrate nice style transfer results. However, when applied to a video sequence of frames, inconsistency of identity of a person, lighting conditions, and features flickering at different frames become obvious. Thus, it is necessary to apply different flicker suppression losses [144]. Also good direction of development is to work with unaligned faces as in StyleGANEX [145], which allows you to edit all the details of a video frame (hair, clothes, environment) in a compatible manner.

We expect to see a multitude of forthcoming papers in the coming years focusing on the topics covered in this survey, but with an emphasis on applying them to a blend of StyleGAN, Diffusion, and NeRF techniques.

ACKNOWLEDGMENT

The authors thank the North Rhine-Westphalia Ministry of Culture and Science for supporting this work within the KI-Starter project.

REFERENCES

[1] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of styleGAN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8110–8119.
[2] "GAN folks," 2021. [Online]. Available: https://opensea.io/collection/ganfolk
[3] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training generative adversarial networks with limited data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 12104–12114.
[4] O. Patashnik, Z. Wu, E. Shechtman, D. Cohen-Or, and D. Lischinski, "StyleCLIP: Text-driven manipulation of styleGAN imagery," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 2065–2074.
[5] R. Abdal, Y. Qin, and P. Wonka, "Image2StyleGAN : How to edit the embedded images?," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8296–8305.
[6] X. Wang, Y. Li, H. Zhang, and Y. Shan, "Towards real-world blind face restoration with generative facial prior," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 9164–9174.
[7] Y. Nitzan et al., "Mystyle: A personalized generative prior," 2022, *arXiv:2203.17272*.
[8] "My style examples," 2022. [Online]. Available: https://mystyle-personalized-prior.github.io/
[9] "Toonify.photos," 2020. [Online]. Available: https://toonify.photos/
[10] R. Gal, O. Patashnik, H. Maron, G. Chechik, and D. Cohen-Or, "StyleGAN-NADA: Clip-guided domain adaptation of image generators," *ACM Trans. Graph.*, vol. 41, 2022, Art. no. 141.
[11] A. Tewari et al., "StyleRig: Rigging StyleGAN for 3D control over portrait images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, Art. no. 6141–6150.
[12] Y. Zhu, Q. Li, J. Wang, C. Xu, and Z. Sun, "One shot face swapping on megapixels," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 4832–4842.
[13] H. Luo et al., "Normalized avatar synthesis using styleGAN and perceptual refinement," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11657–11667.
[14] J. Parvizi et al., "Electrical stimulation of human fusiform face-selective regions distorts face perception," *J. Neurosci.*, vol. 32, no. 43, pp. 14915–14920, 2012.
[15] A. Melnik et al., "Systems, subjects, sessions: To what extent do these factors influence EEG data?," *Front. Hum. Neurosci.*, vol. 11, 2017, Art. no. 150.
[16] "Facial features report," Accessed: Sep. 27, 2023. [Online]. Available: https://www.cryobank.com/_resources/pdf/sampleinformation/facialfeaturesample.pdf,
[17] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. pattern Recognit.*, 2019, pp. 4396–4405.
[18] "GAN folks medium," 2021. [Online]. Available: https://towardsdatascience.com/ganfolk-using-ai-to-create-portraits-of-fictional-people-to-sell-as-nfts-6e24f5214ed1
[19] H. Li, X. Hou, Z. Huang, and L. Shen, "StyleGene: Crossover and mutation of region-level facial genes for kinship face synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 20960–20969.
[20] C.-H. Lin, H.-C. Chen, L.-C. Cheng, S.-C. Hsu, J.-C. Chen, and C.-Y. Wang, "StyleDNA: A high-fidelity age and gender aware kinship face synthesizer," in *Proc. IEEE 16th Int. Conf. Autom. Face Gesture Recognit.*, 2021, pp. 1–8.

[21] X. Qin, X. Tan, and S. Chen, "Tri-subject kinship verification: Understanding the core of a family," *IEEE Trans. Multimedia*, vol. 17, no. 10, pp. 1855–1867, Oct. 2015.

[22] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1510–1519.

[23] Y. Alaluf et al., "Third time's the charm? Image and video editing with StyleGAN3," 2022, *arXiv:2201.13433*.

[24] T. Karras et al., "Alias-free generative adversarial networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 852–863.

[25] A. Melnik, E. Akbulut, J. Sheikh, K. Loos, M. Buettner, and T. Lenze, "Faces: Ai blitz XIII solutions," 2022, *arXiv:2204.01081*.

[26] A. Radford et al., "Learning transferable visual models from natural language supervision," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8748–8763.

[27] T. Yang, P. Ren, X. Xie, and L. Zhang, "GAN prior embedded network for blind face restoration in the wild," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 672–681.

[28] Y. Poirier-Ginter and J.-F. Lalonde, "Robust unsupervised styleGAN image restoration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 22292–22301.

[29] M. Zhang, N. Wang, Y. Li, and X. Gao, "Neural probabilistic graphical model for face sketch synthesis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2623–2637, Jul. 2020.

[30] M. Zhang, N. Wang, Y. Li, and Gao, "Deep latent low-rank representation for face sketch synthesis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 10, Oct. 2019.

[31] M. Zhang, N. Wang, Y. Li, and X. Gao, "Bionic face sketch generator," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2701–2714, Jun. 2020.

[32] M. Zhang, R. Wang, X. Gao, J. Li, and D. Tao, "Dual-transfer face sketch–photo synthesis," *IEEE Trans. Image Process.*, vol. 28, no. 2, pp. 642–657, Feb. 2019.

[33] M. Zhang, Y. Li, N. Wang, Y. Chi, and X. Gao, "Cascaded face sketch synthesis under various illuminations," *IEEE Trans. Image Process.*, vol. 29, pp. 1507–1521, 2020.

[34] W. Jang, G. Ju, Y. Jung, J. Yang, X. Tong, and S. Lee, "StylecariGAN: Caricature generation via styleGAN feature map modulation," *ACM Trans. Graph.*, vol. 40, no. 4, pp. 1–16, 2021.

[35] H. Wang, G. Lin, S. C. Hoi, and C. Miao, "3D cartoon face generation with controllable expressions from a single GAN image," 2022, *arXiv:2207.14425*.

[36] J. Back, "Fine-tuning StyleGAN2 for cartoon face generation," 2021, *arXiv:2106.12445*.

[37] T. T. Nguyen et al., "Deep learning for deepfakes creation and detection: A survey," *Comput. Vis. Image Understanding*, vol. 223, 2022, Art. no. 103525.

[38] Y. Mirsky and W. Lee, "The creation and detection of deepfakes: A survey," *ACM Comput. Surv.*, vol. 54, no. 1, pp. 1–41, 2021.

[39] "D. nostalgia," Accessed: Sep. 18, 2022. [Online]. Available: https://www.myheritage.com/deep-nostalgia

[40] L. Verdoliva, "Media forensics and deepFakes: An overview," *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 5, pp. 910–932, Aug. 2020.

[41] C. Yang and S.-N. Lim, "Unconstrained facial expression transfer using style-based generator," 2019, *arXiv: 1912.06253*.

[42] E. Burkov, I. Pasechnik, A. Grigorev, and V. Lempitsky, "Neural head reenactment with latent pose descriptors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 13783–13792.

[43] J. R. A. Moniz, C. Beckham, S. Rajotte, S. Honari, and C. Pal, "Unsupervised depth estimation, 3D face rotation and replacement," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 9759–9769.

[44] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv: 1704.04861*.

[45] M. Li, J. Lin, Y. Ding, Z. Liu, J.-Y. Zhu, and S. Han, "GAN compression: Efficient architectures for interactive conditional GANs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 5283–5293.

[46] Y. Viazovetskyi, V. Ivashkin, and E. Kashin, "StyleGAN2 distillation for feed-forward image manipulation," in *Proc. Eur. Conf. Comput. vis.*, 2020, pp. 170–186.

[47] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 3730–3738.

[48] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *Proc. Int. Conf. Learn. Representations*, 2018.

[49] D. E. King, "Dlib-ml: A machine learning toolkit," *J. Mach. Learn. Res.*, vol. 10, pp. 1755–1758, 2009.

[50] I. Goodfellow et al., "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[51] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, "On convergence and stability of GANs," 2017, *arXiv: 1705.07215*.

[52] "Alias-free generative adversarial networks (styleGAN3)," 2021. [Online]. Available: https://nvlabs.github.io/stylegan3/

[53] R. Xu, X. Wang, K. Chen, B. Zhou, and C. C. Loy, "Positional encoding as spatial inductive bias in GANs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 13569–13578.

[54] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 214–223.

[55] J. H. Lim and J. C. Ye, "Geometric GAN," 2017, *arXiv: 1705.02894*.

[56] Y. Blau and T. Michaeli, "The perception-distortion tradeoff," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 2018, pp. 6228–6237.

[57] A. Dosovitskiy and T. Brox, "Generating images with perceptual similarity metrics based on deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016.

[58] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 586–595.

[59] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[60] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[61] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. pattern Recognit.*, 2019, pp. 4690–4699.

[62] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017.

[63] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 2016, pp. 2818–2826.

[64] J. Zhu, D. Zhao, B. Zhang, and B. Zhou, "Disentangled inference for GANs with latently invertible autoencoder," *Int. J. Comput. Vis.*, vol. 130, pp. 1259–127, 2022.

[65] A. H. Bermano et al., "State-of-the-art in the architecture, methods and applications of styleGAN," in *Computer Graphics Forum*. Hoboken, NJ, USA: Wiley, 2022, pp. 591–611.

[66] Z. Wu, D. Lischinski, and E. Shechtman, "Stylespace analysis: Disentangled controls for styleGAN image generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 12858–12867.

[67] E. Richardson et al., "Encoding in style: A styleGAN encoder for image-to-image translation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 2287–2296.

[68] O. Tov, Y. Alaluf, Y. Nitzan, O. Patashnik, and D. Cohen-Or, "Designing an encoder for styleGAN image manipulation," *ACM Trans. Graph.*, vol. 40, pp. 1–14, 2021.

[69] Y. Alaluf, O. Patashnik, and D. Cohen-Or, "Restyle: A residual-based styleGAN encoder via iterative refinement," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 6711–6720.

[70] D. Roich, R. Mokady, A. H. Bermano, and D. Cohen-Or, "Pivotal tuning for latent-based editing of real images," *ACM Trans. Graph.*, vol. 42, no. 1, pp. 1–13, 2022.

[71] Y. Alaluf, O. Tov, R. Mokady, R. Gal, and A. Bermano, "HyperStyle: StyleGAN inversion with hypernetworks for real image editing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 18 511–18 521.

[72] Z. C. Lipton and S. Tripathi, "Precise recovery of latent vectors from generative adversarial networks," 2017, *arXiv: 1702.04782*.

[73] S. Garfield, "Official portrait of president-elect barack obama," 2009, published under CC BY 2.0.

[74] V. S. Photography, "Actors headshots female white late twenties," 2016, published under CC BY 2.0.

[75] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 694–711.

[76] S. Guan, Y. Tai, B. Ni, F. Zhu, F. Huang, and X. Yang, "Collaborative learning for faster styleGAN embedding," 2020, *arXiv: 2007.01758*.

[77] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[78] T. Wei et al., "E2Style: Improve the efficiency and effectiveness of styleGAN inversion," *IEEE Trans. Image Process.*, vol. 31, pp. 3267–3280, 2022.

[79] J. Zhu, Y. Shen, D. Zhao, and B. Zhou, "In-domain GAN inversion for real image editing," in *Proc. 16th Eur. Conf. Comput. Vis.*, Glasgow, U.K., 2020, pp. 592–608.

[80] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," 2016, *arXiv:1605.09782*.

[81] A. Melnik and M. Miasoedenkov, "MyStyle: A personalized generative prior," 2022. [Online]. Available: https://youtu.be/8YVSIDcFjaI

[82] Y. Shen, J. Gu, X. Tang, and B. Zhou, "Interpreting the latent space of GANs for semantic face editing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9240–9249.

[83] E. Härkönen, A. Hertzmann, J. Lehtinen, and S. Paris, "Ganspac: Discovering interpretable GAN controls," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 9841–9850.

[84] Z. Wu, Y. Nitzan, E. Shechtman, and D. Lischinski, "StyleAlign: Analysis and applications of aligned styleGAN models," in *Proc. Int. Conf. Learn. Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=Qg2vi4ZbHM9

[85] J. N. Pinkney and D. Adler, "Resolution dependent GAN interpolation for controllable image synthesis between domains," arXiv preprint 2020, *arXiv: 2010.05334*.

[86] M. Liu, Q. Li, Z. Qin, G. Zhang, P. Wan, and W. Zheng, "Blendgan: Implicitly GAN blending for arbitrary stylized face generation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 29710–29722.

[87] S. Zhou, K. C. Chan, C. Li, and C. C. Loy, "Towards robust blind face restoration with codebook lookup transformer," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 30599–30611.

[88] Y. Gu et al., "VQFR: Blind face restoration with vector-quantized dictionary and parallel decoder," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 126–143.

[89] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," 2017, *arXiv: 1711.00937*.

[90] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2Face: Real-time face capture and reenactment of RGB videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2387–2395.

[91] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, "MoCoGAN: Decomposing motion and content for video generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1526–1535.

[92] Y. Song, J. Zhu, D. Li, A. Wang, and H. Qi, "Talking face generation by conditional recurrent adversarial network," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 919–925.

[93] L. Yu, J. Yu, and Q. Ling, "Mining audio, text and visual information for talking face generation," in *Proc. IEEE Int. Conf. Data Mining*, 2019, pp. 787–795.

[94] D. Kononenko, Y. Ganin, D. Sungatullina, and V. Lempitsky, "Photorealistic monocular gaze redirection using machine learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 11, pp. 2696–2710, Nov. 2018.

[95] L. Tran, X. Yin, and X. Liu, "Representation learning by rotating your faces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, pp. 3007–3021, Dec. 2019.

[96] "Fakeapp," [Online]. Available: https://www.malavida.com/en/soft/fakeapp/

[97] J. Zhang, X. Zeng, Y. Pan, Y. Liu, Y. Ding, and C. Fan, "FaceSwapNet: Landmark guided many-to-many face reenactment, 2019, *arXiv:1905.11805*.

[98] L. Li, J. Bao, H. Yang, D. Chen, and F. Wen, "Advancing high fidelity identity swapping for forgery detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 5073–5082.

[99] "How deepfake technology can change the movie industry," Accessed: Dec. 7, 2023. [Online]. Available: https://screenrant.com/movies-deepfake-technology-change-hollywood-how/

[100] Y. Liu and J. Chen, "Unsupervised face frontalization using disentangled representation-learning CycleGAN," *Comput. Vis. Image Understanding*, vol. 222, 2022, Art. no. 103526.

[101] Y. Zhang, L. Zheng, and V. L. L. Thing, "Automated face swapping and its detection," in *Proc. IEEE 2nd Int. Conf. Signal Image Process.*, 2017, pp. 15–19.

[102] C. Limberg, A. Melnik, A. Harter, and H. J. Ritter, "Yolo-you only look 10647 times," 2022, *arXiv:2201.06159*.

[103] K. Vougioukas, S. Petridis, and M. Pantic, "End-to-end speech-driven facial animation with temporal GANs," in *Proc. Brit. Mach. Vis. Conf.*, 2018.

[104] T.-C. Wang et al., "Video-to-video synthesis," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018.

[105] H. Li, J. Liu, Y. Bai, H. Wang, and K. Mueller, "Transforming the latent space of styleGAN for real face editing," 2022, *arXiv:2105.14230*.

[106] Y. Ma et al., "Styletalk: One-shot talking head generation with controllable speaking styles," 2023, *arXiv:2301.01081*.

[107] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in *Proc. 6th Int. Conf. Neural Inf. Process. Syst.*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 737–744.

[108] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2017, *arXiv: 1703.10593*.

[109] B. Mildenhall, P.P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Commun. ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[110] J. Gu, L. Liu, P. Wang, and C. Theobalt, "StyleNeRF: A style-based 3D-aware generator for high-resolution image synthesis," 2021, *arXiv:2110.08985*.

[111] R. Or-El, X. Luo, M. Shan, E. Shechtman, J. J. Park, and I. Kemelmacher-Shlizerman, "StyleSDF: High-resolution 3D-consistent image and geometry generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 13493–13503.

[112] D. Bigioi, S. Basak, H. Jordan, R. McDonnell, and P. Corcoran, "Speech driven video editing via an audio-conditioned diffusion model," 2023, *arXiv:2301.04474*.

[113] S. Shen et al., "DiffTalk: Crafting diffusion models for generalized talking head synthesis," 2023, *arXiv:2301.03786*.

[114] M. Stypułkowski, K. Vougioukas, S. He, M. Zieba, S. Petridis, and M. Pantic, "Diffused heads: Diffusion models beat GANs on talking-face generation," 2023, *arXiv:2301.03396*.

[115] T. Brooks, A. Holynski, and A. A. Efros, "Instructpix2pix: Learning to follow image editing instructions," 2022, *arXiv:2211.09800*.

[116] L. Zhang and M. Agrawala, "Adding conditional control to text-to-image diffusion models," 2023, *arXiv:2302.05543*.

[117] Z. Huang, K. C. Chan, Y. Jiang, and Z. Liu, "Collaborative diffusion for multi-modal face generation and editing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 6080–6090.

[118] N. G. Nair, W. G. C. Bandara, and V. M. Patel, "Unite and conquer: Plug & play multi-modal synthesis using diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 6070–6079.

[119] R. Zhang et al., "Llama-adapter: Efficient fine-tuning of language models with zero-init attention," 2023, *arXiv:2303.16199*.

[120] A. Sevastopolsky, Y. Malkov, N. Durasov, L. Verdoliva, and M. Nießner, "How to boost face recognition with StyleGAN?," in *Proc. Int. Conf. Comput. Vis.*, 2023, pp. 20924–20934.

[121] M. Wu, H. Zhu, L. Huang, Y. Zhuang, Y. Lu, and X. Cao, "High-fidelity 3D face generation from natural language descriptions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 4521–4530.

[122] A. Sauer, K. Schwarz, and A. Geiger, "StyleGAN-XL: Scaling styleGAN to large diverse datasets," in *Proc. ACM SIGGRAPH 2022 Conf. Proc.*, 2022, Art. no. 49.

[123] X. Pan, A. Tewari, T. Leimkühler, L. Liu, A. Meka, and C. Theobalt, "Drag your GAN: Interactive point-based manipulation on the generative image manifold," in *Proc. ACM SIGGRAPH 2023 Conf.*, 2023, pp. 1–11.

[124] H. Jia et al., "BlazeStyleGAN: A real-time on-device styleGAN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 4689–4693.

[125] P. Melzi et al., "GANDiffFace: Controllable generation of synthetic datasets for face recognition with realistic variations," 2023, *arXiv:2305.19962*.

[126] M. Kansy et al., "Controllable inversion of black-box face-recognition models via diffusion," 2023, *arXiv:2303.13006*.

[127] M. Kim, F. Liu, A. Jain, and X. Liu, "DCFace: Synthetic face generation with dual condition diffusion model," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit.*, 2023, pp. 3078–3087.

[128] J. Wang, Z. Yue, S. Zhou, K. C. Chan, and C. C. Loy, "Exploiting diffusion prior for real-world image super-resolution," arXiv preprint *arXiv:2305.07015*, 2023.

[129] Z. Liang, C. Li, S. Zhou, R. Feng, and C. C. Loy, "Iterative prompt learning for unsupervised backlit image enhancement," 2023, *arXiv:2303.17569*.

[130] Z. Wang et al., "DR2: Diffusion-based robust degradation remover for blind face restoration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 1704–1713.

[131] G. Kim, H. Shim, H. Kim, Y. Choi, J. Kim, and E. Yang, "Diffusion video autoencoders: Toward temporally consistent face video editing via disentangled video encoding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 6091–6100.
[132] O. Avrahami, O. Fried, and D. Lischinski, "Blended latent diffusion," *ACM Trans. Graph.*, vol. 42, no. 4, pp. 1–11, 2023.
[133] A. Sauer, T. Karras, S. Laine, A. Geiger, and T. Aila, "StyleGAN-T: Unlocking the power of GANs for fast large-scale text-to-image synthesis," 2023, *arXiv:2301.09515*.
[134] M. Kang et al., "Scaling up GANs for text-to-image synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 10124–10134.
[135] C. Saharia et al., "Photorealistic text-to-image diffusion models with deep language understanding," 2022, *arXiv:2205.11487*.
[136] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," 2022, *arXiv:2204.06125*.
[137] M. Zheng, H. Zhang, H. Yang, and D. Huang, "NeuFace: Realistic 3D neural face rendering from multi-view images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 16868–16 877.
[138] E. R. Chan et al., "Efficient geometry-aware 3D generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 16102–16112.
[139] R. Abdal et al., "3DAvatarGAN: Bridging domains for personalized editable avatars," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 4552–4562.
[140] J. Sun et al., "Next3D: Generative neural texture rasterization for 3D-aware head avatars," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 20991–21002.
[141] Z. Ma, X. Zhu, G.-J. Qi, Z. Lei, and L. Zhang, "OTAvatar: One-shot talking face avatar with controllable tri-plane rendering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 16901–16910.
[142] Y. Xu, Z. Shu, C. Smith, J.-B. Huang, and S. W. Oh, "In-n-out: Face video inversion and editing with volumetric decomposition," arXiv preprint 2023, *arXiv:2302.04871*.
[143] F. Yin et al., "3D GAN inversion with facial symmetry prior," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 342–351.
[144] S. Yang, L. Jiang, Z. Liu, and C. C. Loy, "Vtoonify: Controllable high-resolution portrait video style transfer," *ACM Trans. Graph.*, vol. 41, no. 6, pp. 1–15, 2022.
[145] S. Yang, L. Jiang, Z. Liu, and C. C. Loy, "StyleGANEX: StyleGAN-based manipulation beyond cropped aligned faces," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2023.

Andrew Melnik received the BSc and MSc degrees in applied mathematics and physics, and the PhD degree in cognitive science. Currently, he is as a postdoctoral researcher with the University of Bielefeld, Germany.

Maksim Miasayedzenkau is a research engineer with the Banuba Face Editing Department. He specializes in developing efficient and stable algorithms tailored for mobile applications. His research primarily revolves around applications like virtual makeup try-on, 3D-face reconstruction, and facial color analysis.

Dzianis Makaravets is a research engineer with Banuba. His research focuses on efficient algorithms on mobile phones for such applications as blind face restoration, face editing and facial landmarks detection.

Dzianis Pirshtuk is a principal research engineer with the Computer Vision Department, Banuba (https://www.banuba.com). Focused on real time algorithms of segmentation, facial tracking, reconstruction and editing. Proposed technologies are commercialized in Banuba Face AR and Video Editor SDKs for background replacement, creation of augmented reality visual effects, virtual try-on and other use cases.

Eren Akbulut, biography not available at the time of publication.

Dennis Holzmann received the BS degree in cognitive informatics, in 2021. He is currently working toward the master's degree in intelligent systems with Bielefeld University.

Tarek Renusch received the BSc degree in computer science from Bielefeld University. He is currently working toward the master's degree with Bielefeld University focusing on intelligent systems.

Gustav Reichert, biography not available at the time of publication.

Helge Ritter studied physics and mathematics with the Universities of Bayreuth, Heidelberg and Munich, and the PhD degree in physics from the Technical University of Munich, in 1988. Since 1990, he has been professor with the University of Bielefeld, Germany.