Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Face Image Editing in Latent Space of Generative Adversarial Networks

Nela Petrželková

Supervisor: Ing. Jan Čech, Ph.D.
May 2021

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Petrželková Nela** Personal ID number: **474506**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Branch of study: **Computer and Information Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Face Image Editing in Latent Space of Generative Adversarial Networks

Bachelor's thesis title in Czech:

Editace obrázků obličejů v latentním prostoru generativních adversarialních sítí

Guidelines:

Propose a method for interactive editing of portrait images. A user will be able to change several attributes of a given face image, e.g. pose, expression, age, shape, etc. The method will generate an output image that is both natural and does not alter the identity of the input face. For the sake of photo-realism, use a pre-trained Generative Adversarial Network generator of face images, e.g. [1]. A challenge is to propose a method that keeps the attribute settings independent, i.e. changing a single attribute does not modify the others.

Bibliography / sources:

[1] Tero Karras, Samuli Laine, Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. In CVPR, 2019.
[2] Yujun Shen, Jinjin Gu, Xiaoou Tang, Bolei Zhou. Interpreting the Latent Space of GANs for Semantic Face Editing. ArXiv preprint, arXiv:1907.10786, 2020.
[3] Elad Richardson, Yuval Alaluf, Or Patashnik,Yotam Nitzan, Yaniv Azar, Stav Shapiro, Daniel Cohen-Or. Encoding in Style: A StyleGAN Encoder for Image-to-Image Translation. Proc. ICLR, 2021. (In review)
[4] Y. Nitzan, A. Bermano, Y. Li, and D. Cohen-Or. Disentangling in latent space by harnessing a pretrained generator. ArXiv preprint, arXiv:2005.07728, 2020.

Name and workplace of bachelor's thesis supervisor:

Ing. Jan Čech, Ph.D., Visual Recognition Group, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **08.01.2021** Deadline for bachelor thesis submission: **21.05.2021**

Assignment valid until: **30.09.2022**

Ing. Jan Čech, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

I would like to express my sincerest gratitude to my supervisor, Ing. Jan Čech, Ph.D., for his guidance, time and support he gave me in last few months. I would also like to thank my friends and family for supporting me throughout my studies.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instruction for observing the ethical principles in the preparation of university theses.

Prague, May 21, 2021

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 21. května 2021

.....

Abstract

Face image synthesis has seen tremendous progress recently due to Generative Adversarial Networks (GANs). It was observed that latent space manipulation of GANs enables to semantically edit the generated images. We introduce a novel method for editing face images via manipulation with facial landmarks and subsequent latent optimization. We describe and analyze the proposed method and compare it with the popular approach of finding linear semantic directions in the latent space. Finally, we show an algorithm, where the manipulated face image is seamlessly blended into the original photography.

Keywords: generative adversarial networks, generative modelling, face image editing

Supervisor: Ing. Jan Čech, Ph.D.

Abstrakt

Pokrok v syntéze obrázků obličejů v posledních letech postupuje velmi rychle díky generativním adversariálním sítím. Bylo zjištěno, že manipulace v latetním prostoru generativních adversarálních sítí umožňuje sémanticky upravovat obrázky. Představíme metodu pro úpravu obrázků obličejů pomocí manipulace s obličejovými body a následné optimalizace latentního kódu. Popíšeme a zanalyzujeme navrhovanou metodu a porovnáme ji s populární metodou hledání lineárních sémantických směrů v latentním prostoru. Nakonec ukážeme, jak lze upravené obličeje zakomponovat zpět do původní fotografie.

Klíčová slova: generativní adversiální sítě, generativní modelování, úprava obrázků obličejů

Překlad názvu: Editace obrázků obličejů v latentním prostoru generativních adversialních sítí

Contents

Chapter 1

Introduction

With the boom of Generative Adversarial Networks (GANs) [12] in recent years, the face image synthesis has improved significantly. Current image generation methods have achieved high visual quality and fidelity. The synthesized images look so realistic that even humans have difficulties to distinguish whether the image is real or not. The current state-of-the-art method for high-resolution image synthesis is StyleGAN [18, 19], which proposes a new generator architecture and attains remarkable quality of the generated images.

The rise of such methods brought attention to studies on the latent space of GANs, especially StyleGAN, which introduces an intermediate latent space, leading to a significant improvement of the generated results. To generate an image with StyleGAN, we input a latent code, which is a vector sampled usually from a Gaussian distribution, and the generator synthesizes an image containing a human face. However, the generated images are still random. The next goal is to invert an image of a real person, meaning that we are trying to find a latent vector, which reconstructs the target image in the best way possible, when sent through the generator. This process is called GAN inversion. Currently, there are two main approaches to inverting an image – optimizing the latent code or training an encoder to map the image to a latent code.

Finding the corresponding latent code allows to semantically meaningfully edit the images in the latent space, meaning that specific manipulations of the latent vector will cause changes in the generated image. We can change a person's age, gender, make him smile, or add glasses to his/her face. A popular technique is to find linear semantic directions in the latent space and by manipulating the latent code in these directions, edit the image [37]. We propose an optimization-based approach that utilizes the manipulation of facial landmarks. In other words, we change a person's expression in the space of facial landmarks and then optimize the latent code in such a way that the expression of the generated image matches the new expression, preserving the identity of a given person.

The aim of this work is to analyse the proposed method for editing and to

compare it with other approaches for face image editing in the latent space of generative adversarial networks. Finally, we show possible applications of facial manipulation – blending the manipulated faces into the original photos.

In Chapter 2 we will introduce the related background on the generative editing, in particular, generative adversarial networks, StyleGAN and GAN inversion process. Chapter 3 describes the editing methods used in the experiments in this work. In Chapter 4 we experimentally analyse the two mentioned approaches with different parameters and compare them to each other. Possible application is introduced in Chapter 5 and the thesis is concluded in Chapter 6.

Chapter 2

The Background and Related Work

2.1 Generative Adversarial Networks

In the last years, Generative Adversarial Networks (GANs), introduced by *Goodfellow et al.* [12], have become popular in various domains such as computer vision, natural language processing, semantic segmentation, or time series synthesis [41]. GANs have shown remarkable results especially in the computer vision field, being used for image generation [17, 30, 44], image-to-image translation [48, 15, 47], face image completion [42, 22] or image super-resolution [24, 21, 39].

A typical GAN model consists of two components: a *discriminator* and a *generator* – two convolutional neural networks [5] that are set to compete in an adversarial game against each other. This is visualized in Figure 2.1.

The generator G generates images from input noise vectors in a way that they are as similar to real images as possible, so it fools the discriminator. A noise vector $\mathbf{z} \in \mathcal{Z}$ from a given distribution $p_{\mathbf{z}}$ (typically from a Gaussian distribution uncorrelated among vector entries $\sim \mathcal{N}(0, \mathbf{I}_n)$ where n is the dimensionality of $\mathbf{z}$), is mapped to an output image $\mathbf{s} \in S$. G with parameters θ_g defines a probability distribution $p_{\mathbf{s}}$ as the distribution of the generated samples $\mathbf{s} \in \mathcal{S}$, formally

$$G(\mathbf{z}; \theta_g) : \mathcal{Z} \rightarrow \mathcal{S}. \tag{2.1}$$

The discriminator D is a binary classifier and its task is to distinguish between real and generated images. In other words, the discriminator D is trained to maximize the probability of assigning the correct label to the images it has been given, both real and generated from G, formally written as

$$D(\mathbf{x}; \theta_d) : \{\mathcal{X}, \mathcal{S}\} \rightarrow [0; 1], \tag{2.2}$$

where $\mathbf{x} \in \mathcal{X}$ is the real image with probability distribution $p_{\mathbf{x}}$.

The objective of a GAN is to learn the generator distribution p_s to be as close as possible to the distribution $p_{\mathbf{x}}$ of real data, i.e., to transform

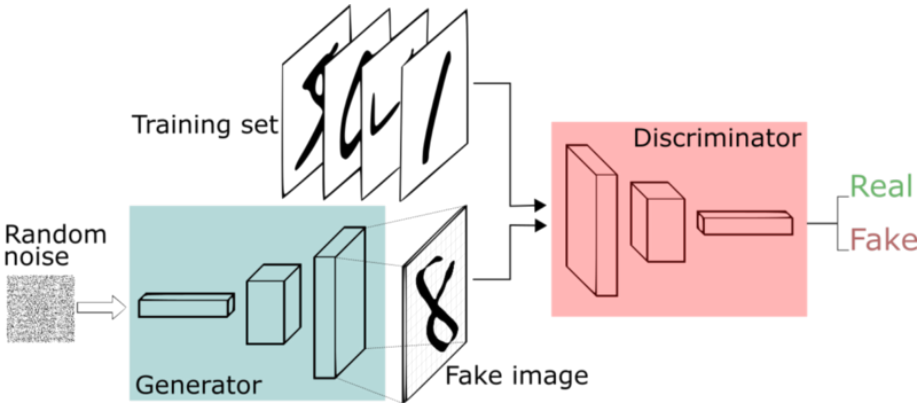


Figure 2.1: A typical architecture of a generative adversarial network. In the blue box is the generator G that gets a random vector as an input and generates a synthetic image. The task of the discriminator D, given an image as an input, is to distinguish whether it is real or synthesized image. The figure is adopted from [38].

the latent vectors to manifold that resembles the real data in the best way possible. This way the generator G would fool the discriminator D, minimizing $\log(1-D(G(\mathbf{z})))$. At the same time, we want the decisions of the discriminator D to be correct by maximizing $\log(1 - D(G(\mathbf{z})))$. From that follows that D and G play the the following two-player minimax game with joint loss function $V(G, D)$:

$$\min_G \max_D V(G, D) = \mathbb{E}_{\mathbf{x}\sim p_\mathbf{x}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_\mathbf{z}}[1 - \log D(G(\mathbf{z}))]. \tag{2.3}$$

Even though GANs achieved great success in image generation, the training process was intricate. Because G and D update their costs independently of each other, updating the gradient of both models at the same time does not guarantee a convergence. Another problem is the vanishing gradient that can appear particularly at the beginning of the training. Before G learns some distribution resembling of the real data distribution, it generates random images, making decisions of D easy. If the discriminator D always classifies the image correctly, the loss function will have a value of zero and the gradient will be zero-valued too. The next challenge in training GANs is avoiding *mode collapse*. Mode collapse is a phenomenon that happens when the generator is able to trick the discriminator, but it generates images with low variety and is not able to represent the real data distribution.

The mentioned problems have been addresed [36] by introducing feature matching, minibatch discrimination, historical averaging, and more techniques. Stability of learning was improved by *Wasserstein GANs* [3], as well as removing mode collapse.

2.1.1 Conditional GANs

Until now, we talked about unconditional generative adversarial networks. The conditional GAN [25], or shortly cGAN, is a type of GAN that involves the conditional generation of images and enables to perform targeted image generation. The target can be a class label $\mathbf{y}$, if available, allowing to generate images of a given type. This changes the GAN objective to

$$\min_G \max_D V(G,D) = \mathbb{E}_{\mathbf{x}\sim p_\mathbf{x}}[\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z}\sim p_\mathbf{z}}[1 - \log D(G(\mathbf{z}|\mathbf{y})|\mathbf{y})]. \quad (2.4)$$

2.2 StyleGAN

In March 2019 *Karras et al.* introduced StyleGAN [18], an alternative generator architecture, that was a great step forward in the field of data-driven unconditional generative image modeling. The new architecture leads to an automatically learned separation between high-level features such as pose or identity when trained on human faces and stochastic variation such as hair structure or freckles, which enables interpolation operations and scale-specific mixing in the image space.

The objective function and the discriminator stay the same as in usual unconditional GAN, only the architecture of the generator changes. The new style-based generator allows to embed the latent code $\mathbf{z} \in \mathcal{Z}$ to an intermediate latent space $\mathcal{W}$ via a non-linear *mapping network*. This is beneficial because the intermediate space $\mathcal{W}$, unlike $\mathcal{Z}$, does not have to follow any probability density, which leads to disentanglement of features.

2.2.1 The Style-Based Generator

The StyleGAN generator differs from a traditional generator in its architecture notably, as shown in Figure 2.2. It consists of a mapping network f and a synthesis network g. A latent code $\mathbf{z} \in \mathcal{Z}$ is first mapped to $\mathbf{w} \in \mathcal{W}$ by the mapping network $f : \mathcal{Z} \to \mathcal{W}$, both vectors are 512-dimensional. The mapping network consists of 8-layer multilayer perceptron [32]. Learned affine transformations then transform $\mathbf{w}$ to styles that control adaptive instance normalization (AdaIN) [14]. Noise vectors are injected to the outputs of each convolution to generate stochastic details in the images.

This architecture allows to control the image synthesis via scale-specific modifications to the styles. The mapping network combined with the affine transformations are the means for drawing samples from each style from a learned distribution, and the style-based architecture enables to generate an image composed from those styles.

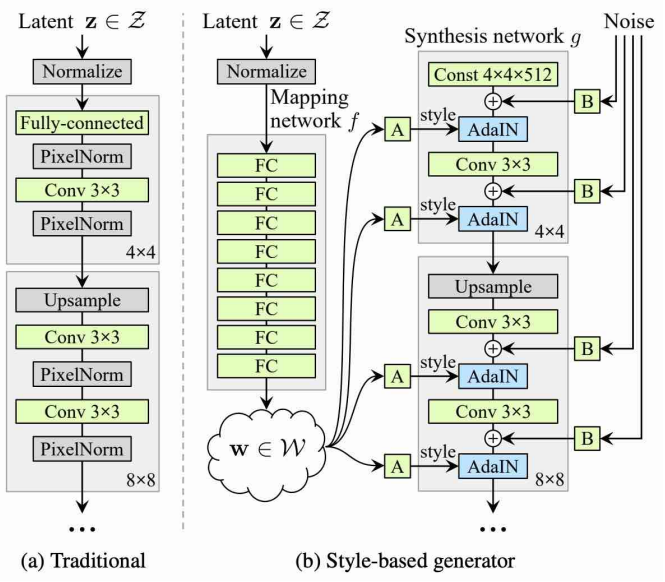


Figure 2.2: Comparison of the traditional generator on the left, where latent code $\mathbf{z}$ is passed directly to the input layer, to the style-based generator. In the style-based generator the latent vector is first mapped to intermediate latent code $\mathbf{w} \in \mathcal{W}$. An affine transformation A is applied to $\mathbf{w}$ and the result is then forwarded to an convolutional layer with AdaIN. This is done for each convolutional layer separately. Latent code $\mathbf{w}$ is the same, but the affine transformations are different for each convolutional layer. After each convolution and before non-linearity, Gaussian noise is injected to provide additional diversity of the results. The image is adopted from [18].

2.2.2 StyleGAN2

StyleGAN2 [19] is an improvement of StyleGAN. It addresses problems with blob-shaped artifacts in some of the images with redesigned normalization in AdaIN and some of the StyleGAN2 configurations were trained using progressive growing [17], which leads to the network first focusing on the low-resolution features and then slowly shifts its attention to finer details. We use StyleGAN2 in this work, since it attains the state-of-the-art visual quality on high resolution images with minimum artifacts. For humans, the generated images are hard to distinguish from the real ones. Examples of synthesized images by StyleGAN2 can be found on `https://thispersondoesnotexist.com/`.

2.3 Encoding Images into the Latent Space of GANs

The inversion of images into latent codes, also called *GAN inversion*, is a problem that has been a subject of research for some time in both computer vision and machine learning areas in many works [2, 8, 10]. The goal of the GAN inversion is to find a latent vector $\mathbf{w}$ that reconstructs a given image $\mathbf{x}$

in the most accurate way when sent through the generator.

There are two main ways in which the inversion can be performed – optimizing a random initial latent code to match a given image or encoding a given image with an encoder. The latter approach was superior in terms of quality for a long time, since it produced better results than the latent code optimization methods [8, 23].

There were also some hybrid methods introduced [45, 4, 13] which combine both mentioned approaches and a recent framework Pixel2Style2Pixel [31] with an innovative encoder architecture that directly generates a series of style vectors which are fed into a pretrained StyleGAN generator.

2.3.1 Optimizing the Latent Code

In this work, the inversion is implemented as a gradient-based optimization. The optimization starts from an initial latent code. During this procedure, the generator weights are fixed and only the latent code $\mathbf{w}$ is optimized. The direction of the gradient descent is determined by the loss function $\mathcal{L}$, formally written as

$$\mathbf{w}^{i+1} = \mathbf{w}^{i} - \alpha \nabla_{\mathbf{w}} \mathcal{L}, \tag{2.5}$$

where α denotes the learning rate and $\nabla_{\mathbf{w}} \mathcal{L}$ is the gradient of the loss w.r.t. vector $\mathbf{w}$. The loss usually consists of pixel-wise loss and perceptual loss [16], depending on the specific framework. For example, style loss [11] in Image2StyleGAN [1]. The loss function used in this work will be described in the next chapter.

Abdal *et al.* [1] proposed a StyleGAN inversion improvement, where the initial latent code $\mathbf{z}$ is mapped to an extended intermediate latent space $\mathcal{W}^{+} \subseteq \mathbb{R}^{k \times 512}$, which is a concatenation of k different 512-dimensional $\mathbf{w}$, where k is the number of style inputs. For instance, StyleGAN synthesis network of synthesizing images at a resolution 1024×1024 has 18 convolutional blocks – we can inject 18 different latent vectors $\mathbf{w}$ to each convolutional layer. This approach yielded great reconstruction fidelity and it introduced new possibilities for various image editing applications [1]. After proposing to optimize the latent code in the extended $\mathcal{W}^{+}$ latent space, the quality of the reconstructed images rapidly increased.

The drawback of this approach is that it requires a long time to converge. Inverting one image usually takes around two minutes, depending on the hardware setup and optimization hyperparameters.

2.3.2 Using Encoder to Find the Latent Code

In other works [29, 46], a learning-based method is used. An encoder is trained to return latent codes corresponding to given images. The training process is conducted in a way that a set of random latent codes is generated and they are sent through the generator to obtain the corresponding images. In this manner, a training set is created. A neural network, where images are the inputs and the latent codes are the outputs of this network, is trained by minimizing pixel-wise loss and LPIPS loss, which will be described in more detail in the next chapter.

The encoder-based approach performs well in terms of time – inversion of one image takes a fraction of a second. That is the time to evaluate the encoder. On the other hand, the reconstruction quality is usually inferior to the optimization approach [31].

Chapter 3

Editing Faces in Latent Space of GANs

Face image editing is a long-lasting problem in graphics, art and entertainment industry. The process of editing used to be a manual task, requiring tedious manual editing with some professional software. This changed with the rise of popularity of generative adversarial networks and the study on the latent space of GANs. New methods of editing were introduced and they enable to make face edits, which are not easily done with graphics software, such as head rotation, aging, change of gender, change of facial expression, and more.

The goal in editing in the latent space of GANs is to manipulate the latent code in a way that it would change a desired attribute in the photo. That attribute can be age, gender, expression, rotation, and so on. The next requirement of the transformation is that only the desired attribute is edited and the rest stays preserved.

In this chapter, we introduce two main approaches for face image editing that are used in this work. Firstly, we propose latent code optimization with manipulated landmark positions and the second is manipulation via linear directions that correspond to changes in a given attribute.

3.1 Latent Code Optimization with Facial Landmarks Manipulation

In this method, we exploit the optimization-based approach for image inversion and modify the objective function in a way that the image is being edited already during the inversion process, namely with landmark loss. We use a pretrained StyleGAN2 [19] generator with fixed weights and optimize only the latent code.

We start from a zero-valued latent code and perform gradient descent with respect to our objective function that consists of a weighted sum of four objectives which are introduced in the following section.

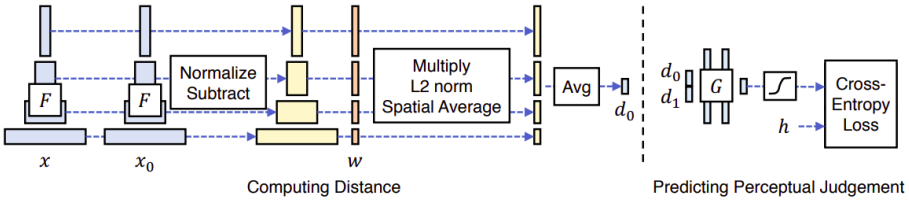


Figure 3.1: To compute LPIPS loss (d_0) between two images x, x_0, given a network F, deep encodings are first computed, then the activation are normalized in the channel dimension and are scaled by vector w. $\mathcal{L}_2$ distance is the taken and averaged across spatial dimension and across all layers. A small network G is trained to predict perceptual judgement h from distance pair (d_0, d_1). $h \in \{0, 1\}$ are humans' decisions on which distorted image out of two is closer to the original one. The image is taken from [43].

3.1.1 Loss functions

Pixel-wise ($\mathcal{L}_2$) Loss

Pixel-wise loss, also called $\mathcal{L}_2$ loss, measures the difference in individual pixel intensities between the source and the generated image. It is defined as

$$\mathcal{L}_2(\mathbf{x}, \mathbf{w}) = ||\mathbf{x} - G(\mathbf{w})||_2, \tag{3.1}$$

where $\mathbf{x}$ denotes the input image and $G(\mathbf{w})$ is the generated image – a latent code $\mathbf{w}$ sent through generator G.

Perceptual Loss

The downside of the pixel-wise function is that it does not reflect the perceptual change that humans see. For example, blurring the image does not change the value of $\mathcal{L}_2$ much, but the perceptual change for a human can be significant. Perceptual loss aims to capture the similarity of two images, comparing high level differences like content and style discrepancies. This problem is challenging because the judgement of similarity may not actually constitute a distance metric [40]. This has been addressed in [43], which introduced **Learned Perceptual Image Patch Similarity** (LPIPS) loss. The authors created a dataset containing 484k human judgements and computed the distance between the reference and the generated images. The process is denoted in Figure 3.1. A pretrained neural network is used to calculate the perceptual distance

$$\mathcal{L}_{LPIPS}(\mathbf{x}, \mathbf{w}) = ||P(\mathbf{x}) - P(G(\mathbf{w}))||_2, \tag{3.2}$$

where P denotes the perceptual feature extractor and G is the generator.

Identity Loss

One of the major challenges of GAN inversion task is the ability to preserve identity between the source and the target image. For higher quality

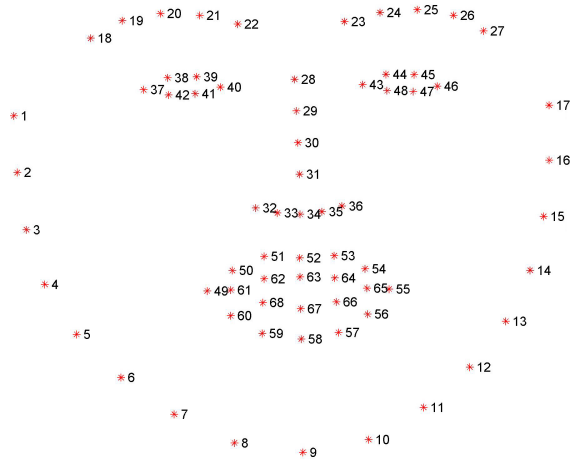


Figure 3.2: The 68 annotated landmarks on human face. Image is taken from [35]

results, we should integrate this objective into the loss function, since the previously mentioned losses are less sensitive to the preservation of facial identity.

We adopted the identity loss introduced in Pixel2Style2Pixel [31]

$$\mathcal{L}_{ID}(\mathbf{x}, \mathbf{w}) = 1 - \langle R(\mathbf{x}), R(G(\mathbf{w}))\rangle, \tag{3.3}$$

where R states for the unit-length normalized response of the pretrained face recognition network, ArcFace [9]. The source image and the generated image are cropped around the face and resized to 112×112 before being fed into the ArcFace network [9].

In Chapter 4, Figure 4.8, we show how utilizing identity loss affects the generated output on five chosen images.

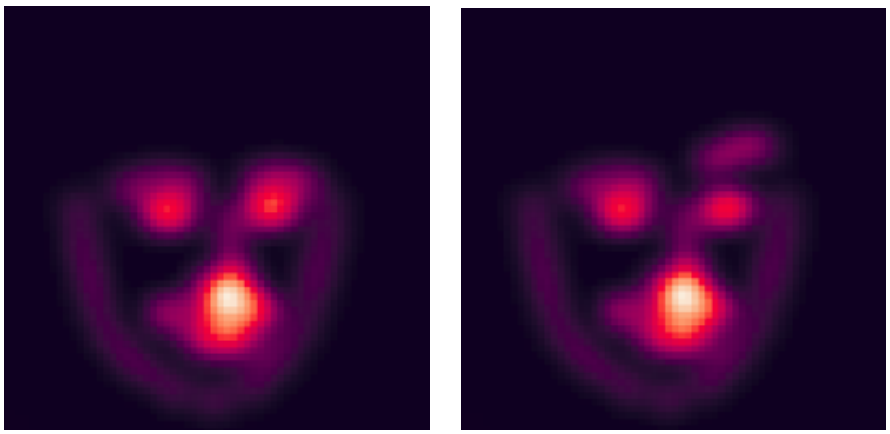
Landmark Loss

The aforementioned objectives, however, still do not edit the face, they just serve the GAN inversion process, so the the source image is reconstructed well. The editing part itself is forced by the landmark loss.

Facial landmarks are a set of points on a human's face with specific coordinates, which localize distinct parts of the human face, such as mouth, eyes, nose, jaw, and eyebrows, as shown in Figure 3.2. By the landmarks relative position, we can estimate a person's expression, head rotation, and so on.

For obtaining the landmark loss, we adopted the landmark localization framework [7], which uses the state-of-the-art Hour-Glass [27] convolutional neural network with hierarchical, parallel and multi-scale blocks [6]. The net has been trained on the dataset *300W-LP-2D*, which consists of 61,225 synthetically generated images with annotated landmarks [49].

The network returns a set of heatmaps, one heatmap for each landmark, in our case it is 68. To obtain the landmark coordinates, we simply choose the argmax and map it to the image coordinates (heatmap resolution is 64×64, while the images have usually size 256×256 or 1024×1024, so we have to scale the positions appropriately). However, we cannot use argmax when doing gradient descent because it is not a differentiable function. For that reason, we work only with the heatmaps and use the landmarks themselves just for visualization purposes.



Original heatmaps Manipulated heatmaps

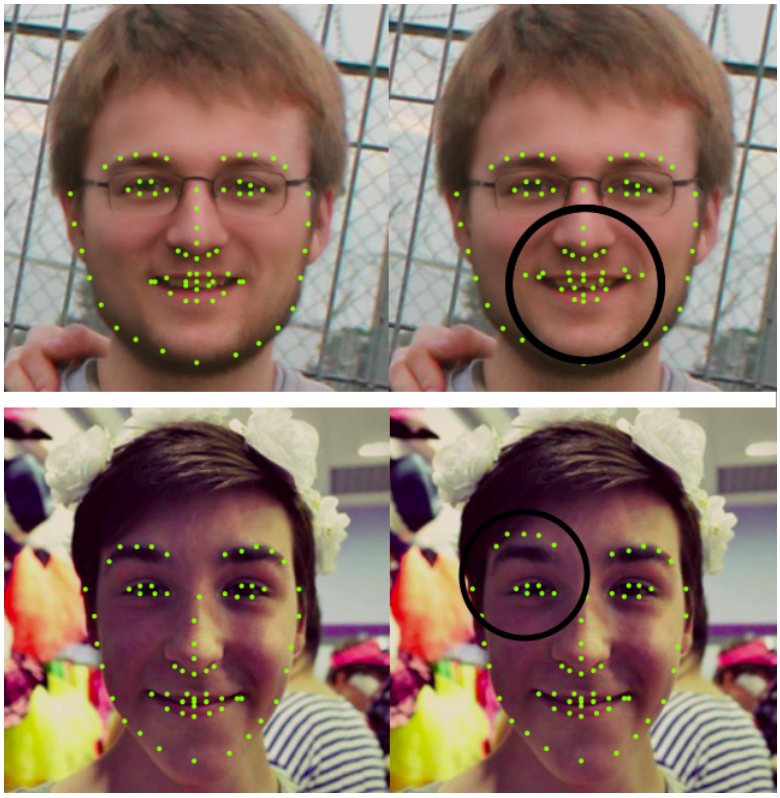
Figure 3.3: Visualization of heatmaps before and after manipulation for raising right eyebrow. The images show the sum of all 68 heatmaps.

3.1.2 Landmark Manipulation

The landmark manipulation is done in a way that the whole heatmap is shifted. The area, where no values are after the shift, is filled with zeros. Thus, for example, for raising the right eyebrow, heatmaps with indices 23–27 (as seen in Figure 3.2) are shifted upwards by given constants that can be scaled by some factor, based on how much we want the eyebrow to raise. This is shown in Figure 3.3. Similar process applies for smiling, when the heatmap corresponding to the left mouth corner is shifted to the left side and upward, and the right one is shifted to the right side and upwards. In this manner, we obtain the target heatmaps. Other example is shown in Figure 3.4.

Given that, the objective of this loss function is minimizing the mean squared error between the target heatmaps h and the heatmaps of the generated image. $F(\cdot)$ denotes the landmark localization network producing a bank of heatmaps.

$$\mathcal{L}_{LM}(h, \mathbf{w}) = ||h - F(G(\mathbf{w}))||_2. \tag{3.4}$$



Original After manipulation

Figure 3.4: Example of landmark relocation. In the left column are original landmarks detected on the face and in the right column are the target landmark. Manipulations "smile" (scale 1.5) and "raising left eyebrow" (scale 1), respectively, are applied to the heatmaps of landmarks.

3.1.3 Latent Code Optimization

The overall objective function is then a weighted sum of the previously mentioned four loss functions

$$\mathcal{L} = \lambda_1 \mathcal{L}_2 + \lambda_2 \mathcal{L}_{LPIPS} + \lambda_3 \mathcal{L}_{ID} + \lambda_4 \mathcal{L}_{LM}, \tag{3.5}$$

where $\lambda_1, \ldots, \lambda_4$ denote the weights of particular objectives. Our goal is to minimize the value of the loss function $\mathcal{L}$, formally written as:

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \tag{3.6}$$

and we do that with the gradient descent optimization method:

$$\begin{aligned} \mathbf{w}^0 &= \mathbf{w}^{\text{init}}, \\ \mathbf{w}^{t+1} &= \mathbf{w}^t + \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}^t). \end{aligned} \tag{3.7}$$

We find the gradient $\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w})$ by backpropagation through the computational graph with PyTorch [28] `backward` function. To compute the gradient,

a computational graph is created and is differentiated using the chain rule. More precisely, the direction of the latent code update is found by Adam [20] optimization method. Adam uses estimations of the first and second moments of the gradient to adapt the learning rate for each weight of the net. Moment of a random variable is defined as the expected value of that variable to the power on n:

$$m_n = \mathbb{E}[X^n], \tag{3.8}$$

The first moment is mean and the second moment is uncentered variance. To estimate the moment, Adam utilizes exponentially moving averages:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla\mathcal{L}_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)\nabla\mathcal{L}_t^2 \end{aligned} \tag{3.9}$$

m_t and v_t are estimate of the first and second moment, respectively, and β_1 and β_2 are the moment decay rates set to 0.9 and 0.999, respectively. It has been observed that m_t and v_t are biased towards zero, especially during the initial steps, therefore a bias correction is performed

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t}. \end{aligned} \tag{3.10}$$

We optimize the latent code w in the following way:

$$\mathbf{w}^t = \mathbf{w}^{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t, \tag{3.11}$$

where α denotes the learning rate and ϵ is term added to prevent division by zero, usually set to 10^{-8}. The algorithm is shown in Algorithm 1.

3.1.4 Advantages and Disadvantages

The main downside of this approach is that the process of editing is slow, taking usually several minutes, depending on the hyperparameters and hardware. On the other hand, it enables us to do transformations which are not strongly represented among the training samples of the generator. Moreover, this method preserves the identity of the person and it does not affect the other attributes.

3.2 Linear Manipulation in the Latent Space

The latent spaces of GAN models often have semantically meaningful directions that represent specific facial attributes in the image, such as pose, age, smile, gender, and many more. Moving in these directions corresponds to image transformations that change the target attribute and does not change much other information on the input face [37].

Algorithm 1 Latent Code Optimization

Input: Load source image $\mathbf{x}$ and generator G

Obtain target heatmaps h by manipulation with heatmaps from $\mathbf{x}$

$\mathbf{w} := \overrightarrow{0}, m_0 := 0, v_0 := 0$ ▷ Initialization

$\beta_1 := 0.9, \beta_2 := 0.999, \alpha := 0.1, \epsilon := 10^{-8}$

for $t = 1, \ldots, num\_iter$ **do**

  Generate image $\mathbf{x}'$ from latent code $\mathbf{w}$ by $\mathbf{x}' = G(\mathbf{w})$

  Compute pixel-wise loss $\mathcal{L}_2(\mathbf{x}, \mathbf{w})$ ▷ Get the loss values

  Compute perceptual LPIPS loss $\mathcal{L}_{LPIPS}(\mathbf{x}, \mathbf{w})$

  Compute identity loss $\mathcal{L}_{ID}(\mathbf{x}, \mathbf{w})$

  Compute landmark loss $\mathcal{L}_{LM}(h, \mathbf{w})$

  Get overall loss $\mathcal{L} = \lambda_1\mathcal{L}_2 + \lambda_2\mathcal{L}_{LPIPS} + \lambda_3\mathcal{L}_{ID} + \lambda_4\mathcal{L}_{LM}$

  Obtain $\nabla_{\mathbf{w}}\mathcal{L}_t$ by calling `backward` function on the loss $\mathcal{L}$

  $m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_{\mathbf{w}}\mathcal{L}_t$ ▷ Compute the gradient direction

  $v_t = \beta_2 v_{t-1} + (1-\beta_2)\nabla_{\mathbf{w}}\mathcal{L}_t^2$

  $\hat{m}_t = \dfrac{m_t}{1-\beta_1^t}$ ▷ Bias correction

  $\hat{v}_t = \dfrac{v_t}{1-\beta_2^t}$

  $\mathbf{w}^t = \mathbf{w}^{t-1} - \dfrac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t$ ▷ Update the latent vector $\mathbf{w}$

end for

Return: Edited image $\mathbf{x}' = G(\mathbf{w}^t)$

3.2.1 Semantics in the Latent Space

The generator of a given GAN model can be expressed as a deterministic function $g : \mathcal{Z} \to \mathcal{X}$ where $\mathcal{X}$ denotes the image space and $\mathcal{Z} \subseteq \mathbb{R}^{512}$ stands for the *512*-dimensional latent space, for which Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_{512})$ is commonly used [18, 26].

It has been observed that when linearly interpolating two latent vectors $\mathbf{z_1}$ and $\mathbf{z_2}$, the corresponding generated images change continuously. That means that the semantics contained in the image also change gradually [18, 30]. That implies that for any binary attribute such as, e.g. *old* $\leftrightarrow$ *young*, *male* $\leftrightarrow$ *female*, there exists a separation boundary in the form of a linear hyperplane that separates the data into two groups. This has been empirically verified by training linear classifiers with linear boundaries achieving over 95% accuracy in [37].

We can define an oriented distance between a hyperplane and latent code $\mathbf{z}$ as

$$\mathrm{d}(\mathbf{n}, \mathbf{z}) = \mathbf{n}^T\mathbf{z} \tag{3.12}$$

where $\mathbf{n}$ is the unit length normal vector of the hyperplane. It has been shown [37], that the corresponding semantic varies the most when we move the latent code in the direction of a normal vector of the hyperplane. The

distance from the separating hyperplane determines how "strong" a given semantic is. For example, having the gender attribute, a male face with a beard will be further from the separating hyperplane than a boy without a beard and with long lashes, which is an attribute usually typical for women.

We describe the process in the latent space $\mathcal{Z}$, but the same applies for latent spaces $\mathcal{W}$ and $\mathcal{W}^+$ as well.

3.2.2 Discovering Semantic Directions

One of the common approaches of finding the separation boundary is to do it in a supervised manner. First, we generate n (usually several thousands) of examples of random latent codes $\mathbf{z} \in \mathcal{Z}$ and measure how strong is the attribute we are interested in, we call it a semantic score s. We create a training set $\mathcal{T} = \{(\mathbf{z}_1, s_1), (\mathbf{z}_2, s_2), \ldots, (\mathbf{z}_n, s_n)\}$. Score s can be either 1 or 0 for binary attributes such as gender or $s \in \mathbb{R}$ for continuous attributes (e.g., age, yaw).

Binary Attributes

To find the boundary between male and female attribute or some other binary attribute, we need a pretrained network or other classifier for classifying a person's gender. In this case, we map s to $y \in \{-1; 1\}$. After we create a dataset of labeled images, we can use a linear classifier, such as SVM (Support Vector Machines), to find the separating hyperplane. We find a hyperplane with parameters $(\mathbf{n}, b)$, for which holds

$$y = \text{sign}(\mathbf{n}^T\mathbf{z} + b) \tag{3.13}$$

and at the same time, we want to maximize margin $m = 2 \min_{\mathbf{z}\in\mathcal{T}} \text{d}(\mathbf{z})$ where $\text{d}(\mathbf{z})$ is the distance of $\mathbf{z}$ from the separating hyperplane. The optimal hyperplane parameters are then defined as

$$\begin{aligned} (\mathbf{n}^*, b^*) &= \text{argmin}_{(\mathbf{n},b)} \frac{1}{2}||\mathbf{n}||^2 \\ \text{subject to: } & y(\mathbf{n}^T\mathbf{z}) \geq 1, \forall(\mathbf{z}, y) \in \mathcal{T} \end{aligned} \tag{3.14}$$

and were found by an SVM solver in our experiments. See Figure 3.5 for illustration.

Continuous Attributes

For continuous attributes, the semantic score s is a real number, therefore we need to use a regression approach in this case. In this work, we adopted a facial landmark detector, where for each image we estimated the distance between an eye and an eyebrow to measure whether the person has raised eyebrows. Or we can measure the relative position of the mouth to other

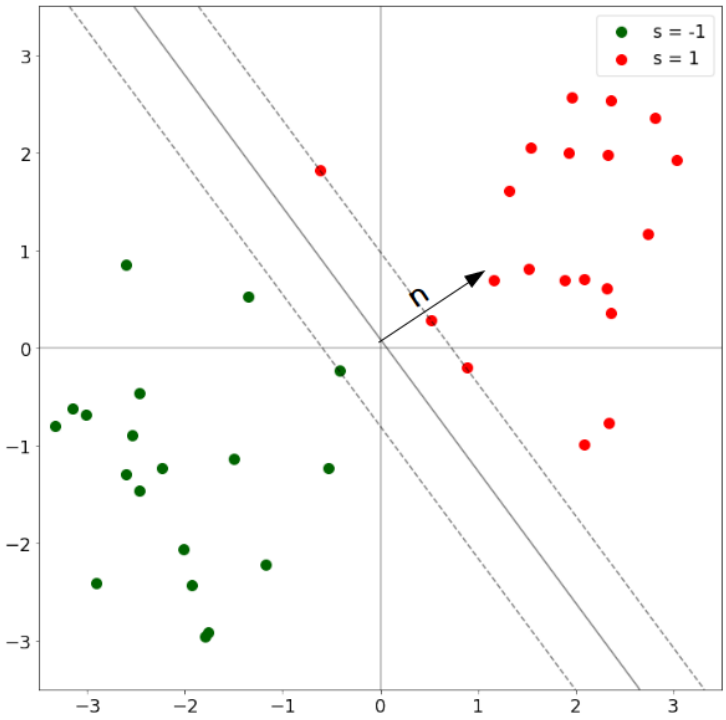


Figure 3.5: Finding a semantic direction for a binary attribute. Illustration in 2D – normally, the latent space is 512-dimensional. Given latent codes with binary score s, we find a maximum margin linear classifier (SVM). The hyperplane normal vector $\mathbf{n}$ is the semantic direction.

facial landmarks to estimate whether a person is smiling. This is our score s. Having a set of latent codes $\{\mathbf{z}_1, \ldots, \mathbf{z}_n\} \in \mathbb{R}^{512}$ and set of corresponding semantics scores $\{s_1, \ldots, s_n\} \in \mathbb{R}$, we model the attribute predictor as a linear function

$$\hat{s} = \mathbf{n}^T \mathbf{z}. \tag{3.15}$$

To find $\mathbf{n}$, we are minimizing the following function:

$$\min_{\mathbf{n}} \sum_{i=1}^{n} (\mathbf{z}_i \mathbf{n} - s_i)^2 = \min_{\mathbf{n}} ||\mathbf{Z}\mathbf{n} - \mathbf{S}||^2, \tag{3.16}$$

where $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_n]^T$ and $\mathbf{S} = [s_1, \ldots, s_n]^T$. The optimum is found by

$$\mathbf{n}^* = \mathbf{Z}^+ s, \tag{3.17}$$

where $\mathbf{Z}^+$ is the Moore-Penrose pseudoinverse of the matrix with latent codes. Then $\mathbf{n}^*$ is the direction of maximum change of a given semantic attribute. This is visualized in Figure 3.6.

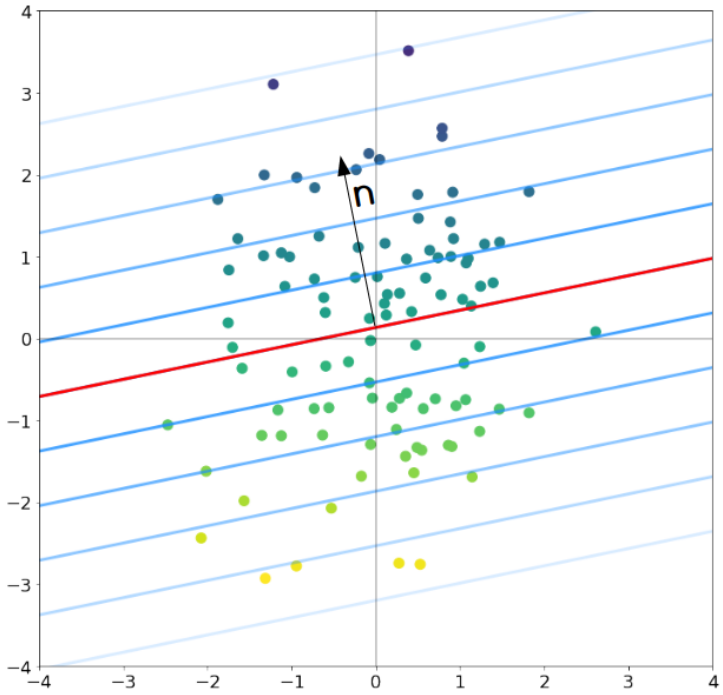


Figure 3.6: Finding a semantic direction for a continuous attribute. Illustration in 2D – normally, the latent space is 512-dimensional. The points are latent codes $\mathbf{z} \in \mathcal{R}^{512}$ with semantic score $s \in \mathcal{R}$ symbolized by the point color. We find the semantic direction using linear regression.

3.2.3 Face Editing with Latent Directions

After obtaining the desired latent direction, we simply add the scaled direction to our latent code $\mathbf{z}$, so the new latent code $\mathbf{z}_{new}$ is obtained as follows:

$$\mathbf{z}_{new} = \mathbf{z} + \beta\mathbf{n} \tag{3.18}$$

where $\mathbf{n}$ is the semantic direction of the hyperplane and β is the scalar representing the change in a given direction. Its magnitude affects how strong the effect on the photo will be and its sign determines the side of the space divided by the hyperplane, so the direction of change of the effect. We demonstrate this in Figure 3.7.

3.2.4 Disentanglement Studies

Attribute entanglement is often observed problem with the approach of linear manipulation. That means that when we manipulate one attribute, some others change too. It happens, e.g., when we make people look older. Glasses appear on their faces, because many old people in the training set had glasses, therefore the two attributes correlate with each other. It projects the distribution of the training samples. This has been addressed in [37].

The paper proposes to use to do conditional manipulation by forcing the separation boundaries to be orthogonal to each other to mitigate the problem.

3.2.5 Advantages and Disadvantages

Compared with the optimization approach of editing the latent code, the linear manipulation is much faster. If we have already found the linear direction, it takes just a fraction of a second to generate the image, which is the time of the pass through the generator. Another advantage of this approach is that we can do edits that are not possible to do with landmark manipulation, such as change of age or gender. On the other hand, it does not work well for semantic directions that are not represented in the training set. Another downside is that sometimes the manipulation of the semantics can change a person's identity.

-1 -0.7 -0.3 0 0.3 0.7 1

Raising eyebrows

-0.7 -0.3 -0.1 0 0.1 0.3 0.7

Smile

-400 -200 -100 0 100 200 400

Yaw rotation

Figure 3.7: Effects of β scaling factor applied on the three different latent directions. The images were generated from random latent vectors, a semantic direction was found by linear regression, then it was scaled be β and added to the latent vector.

Chapter 4

Experiments

In this chapter, we will introduce the experiment environment, then show the results of experiments comparing different weights in loss function objectives, generating images in latent space $\mathcal{W}$ in extended latent space $\mathcal{W}+$, show how the progressive landmark objective affects the result, the effect of identity loss, how the learning rate changes the optimization and finally we compare the latent vector optimization and the latent direction manipulation methods.

4.1 Environment and Implementation Details

The implementation builds upon the pretrained StyleGAN2 [19] generator and its PyTorch implementation [33] that was trained on *FFHQ dataset* [18] with 550 000 iterations on 256×256 images. We also work with landmark detection framework from *Adrian Bulat* [7]. All experiments using the method of optimizing the latent code were done in 200 optimization steps and we used Adam optimizer with 0.1 learning rate, and β_1, β_2 and ϵ were set to 0.9, 0.999 and 10^{-8}, respectively, if not stated otherwise. Other parameters for generating images are specified in each experiment separately.

To find the latent directions in Experiment 4.7, we generated 10000 random images with the generator and detected the landmark positions on each of them. Then we calculated eyebrow-to-eye distance and distance between the corners of the mouth, and by linear regression we found the direction of the largest change of the given semantic attribute, in this case raising eyebrows and smile, respectively.

4.2 Effect of Different Weight Setting in Optimization Objective on Generated Images

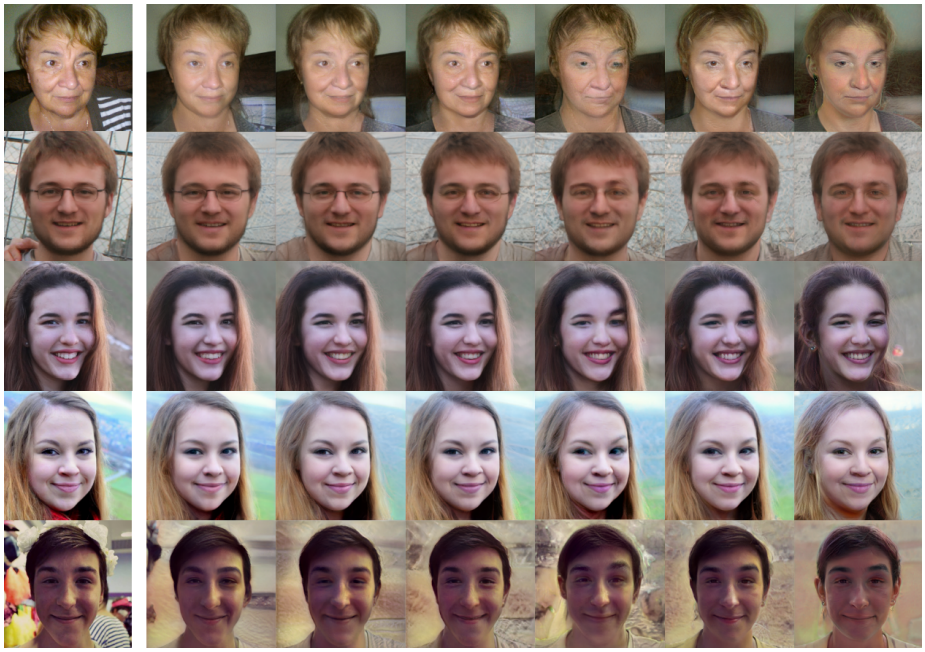
As explained in Section 3.1.1, our optimization objective consists of a weighted sum of four loss functions – landmark loss, identity loss, pixel-wise loss, and perceptual loss. The goal of the following experiment was to compare the effect of different weight settings on the output image.

Weights						
Landmark	0	0	0	1	1	2
Identity	0	2	3	2	3	2
L2	8	8	8	4	8	4
Perceptual	8	4	8	8	8	4

Figure 4.1: Effect of different loss weights in the loss function and their results when optimization is performed in $\mathcal{W}$ latent space. The landmark manipulation was raising right eyebrow.

In the first column are the original photos. In the next three columns, we show the inversion without the landmark manipulation. In the latter three columns, the landmark manipulation was raising the right eyebrow. The experiment was conducted both in $\mathcal{W}$ (Figure 4.1) and extended $\mathcal{W}+$ (Figure 4.2) space to show the difference between the two approaches.

In the rightmost column, we can observe that when the landmark loss weight is set too high, the desired transformation, in this case raising the right eyebrow, is well visible, but the people in the generated images do not resemble the people in the original photos. The results are better for optimizing in the $\mathcal{W}^+$ latent space, but still the best combination seems to be 1, 2, 4, 8 for landmark, identity, L2 and perceptual loss, respectively, therefore we will use this setting for most of the upcoming experiments.



Landmark	0	0	0	1	1	2
Identity	0	2	3	2	3	2
L2	8	8	8	4	8	4
Perceptual	8	4	8	8	8	4

Weights

Figure 4.2: Effect of different loss weights in the loss function and their results when optimization is performed in $\mathcal{W}+$ latent space. The landmark manipulation was raising right eyebrow.

4.3 Optimization Process in $\mathcal{W}$ and $\mathcal{W}^+$

Figures 4.3 and 4.4 show the process of optimizing the latent code for a given image. The optimization always starts from the same mean latent vector, in our case it is a zero vector. Then the latent vector changes with respect to the direction opposite to the gradient of the loss function, as described in 3.1. The generated images were created with the following parameters: landmark manipulation was raising the right eyebrow, and the weights for partial loss functions were following landmark loss 1, identity loss 2, L2 loss 4, perceptual loss 8.

We can observe that the optimization in the extended latent space $\mathcal{W}+$ leads to results that are more similar to the original photo, and we can see the effect of the moved landmarks more clearly. On the other hand, on some of the images, the background changed more and does not look very realistic.

Figure 4.3: Progress of latent optimization in latent space $\mathcal{W}$ for each image. Transformation of raising right eyebrow was applied.

4.4 Progressive Landmark Manipulation

In the following experiment, we wanted to show what effect will have progressive changing of the target landmarks. We show the results of editing when the target landmark positions are set before the optimization and are not changed during it. We compare it with progressively changing target landmark positions with a smaller displacement after every 40 optimization steps in total 200 steps. For example, if the manipulation would be shifting all landmarks corresponding to the left eyebrow by 10 pixels upwards, in the progressive landmark manipulation, they are shifted by 2 pixels every 40 iterations. This is visualized in Figure 4.5.

In this experiment, landmark loss was set to 1, ID loss to 2, pixel-wise loss to 8, and perceptual loss to 8. The optimization had 200 steps and in the progressive setting, landmarks were moved every 40 iterations. In Figure 4.7 we show how the loss values of particular objectives developed during the optimization process.

Figure 4.4: Progress of latent optimization in latent space $\mathcal{W}+$ for each image. Transformation of raising right eyebrow was applied.

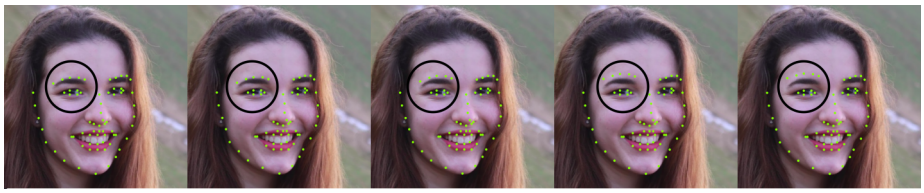


Figure 4.5: The positions of the target landmarks are changing during the optimization after every 40 steps. The manipulation is raising left eyebrow, the five landmarks corresponding to left eyebrow are relocating.

4.5 Identity Loss

In this experiment, we analyse the visual effect of the identity loss in the objective function. As its name suggests, the identity loss should preserve the identity of the person on the photo during the inversion process. In Figure 4.8 is clearly visible that it really helps to preserve identity. Without this term in the loss function, the faces on the inverted photos do not resemble the same person.

Figure 4.6: In this figure we show the difference between progressive and classical landmark optimization. We have manipulated the landmarks of the right eyebrow. In this case, the manipulation was quite strong and we see that both approaches have problems to mimic this transformation.

4.6 Learning Rate

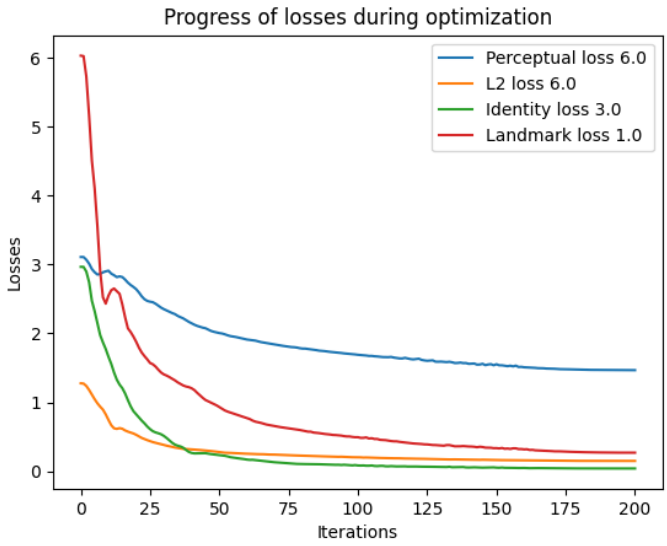
In this section, we are showing the effect of different learning rate values for the Adam optimizer, which is used to find the optimal latent vector during editing with the optimization method. Namely, the learning rate values were 0.01, 0.05, 0.1, 0.5. The weights of landmarks, identity, L2-norm, and perceptual loss items were 1, 2, 4, and 8, respectively. Optimized latent code was optimized in the extended latent space $\mathcal{W}^+$ with 200 iterations. Landmark manipulation of raising the right eyebrow was applied. We show the result in Figure 4.9.

We can observe that too high value of the learning rate creates strong artifacts and the generated image does not look like the person from the original photo and the output does not look realistic at all. The reason for this is that the local minimum of the loss function is not reachable with large steps of the gradient, therefore the optimization diverges and the result does not look similar to the target image.

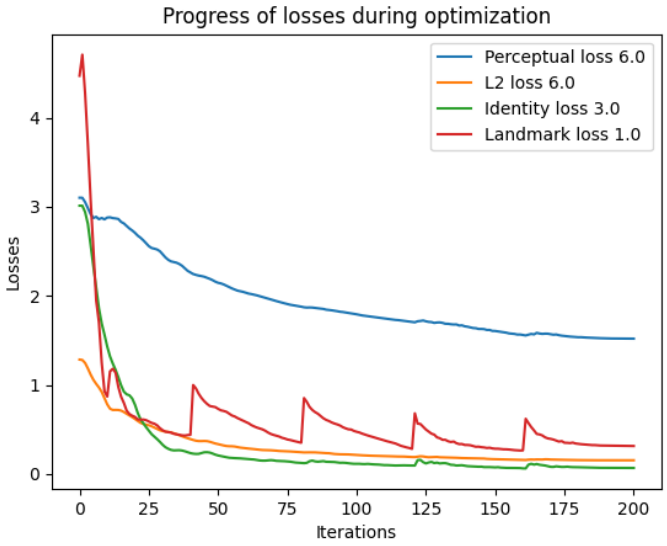
4.7 Comparison of Face Editing Approaches

In the last section of Chapter 4, we compare two approaches of face editing – linear latent manipulation and latent optimization.

First of them is linear latent manipulation. In Section 4.1, we described how we found the semantic directions. Then we just add the corresponding



(a) Fixed landmarks



(b) Changing landmarks positions during optimization

Figure 4.7: Comparison of losses with and without approach of progressive landmarks. In the legend are listed individual losses with their weights.

semantic direction $\mathbf{n}$ to a given latent vector $\mathbf{z}$:

$$\mathbf{z}_{new} = \mathbf{z} + \beta\mathbf{n}, \tag{4.1}$$

where $\mathbf{z}_{new}$ is the final latent code we are sending through the generator. The scaling factor β was set to 0.7 for raising eyebrows and 0.5 for smiling.

The latter approach is the latent optimization with manipulated landmarks, where we optimize not only with respect to the objectives that affect the similarity with the original image, but also with respect to landmark loss,

Figure 4.8: Difference in generated output with and without identity loss in the objective function. In the first row are original images, second shows the outputs without identity loss and the third one includes the identity loss in the objective function. The manipulation of raising right eyebrow was applied.

where we are minimizing the distance between the detected landmarks and target landmarks. The weights for the landmark, identity, pixel-wise, and perceptual loss were set to 1, 2, 4, and 8, respectively.

The result of this comparison is shown in Figure 4.10. We can see that the latent optimization approach does not change the overall facial expression, which cannot be said about the linear manipulation approach, which also slightly changes the other parts of the face.

Figure 4.9: Effect of different values of the learning rate in the latent optimization. Landmark manipulation of raising the right eyebrow was applied.

Figure 4.10: Comparison of face editing with linear latent manipulation and latent optimization. We chose six randomly generated images and performed the two edits independently on each other.

Chapter 5

Application: Blending Manipulated Faces into the Original Photo

From the previous photos, it can be seen that all faces in the photos are in the same position. The generator is trained on samples with a specific alignment format. The photos of the faces have to have the same width as height, the eyes of the people in the photos need to be on certain coordinates, and the eye-to-eye distance to the resolution is given. The generated images are therefore in this format too.

In this chapter, we are inserting the generated images back to the original photos that can be in any arbitrary resolution. First, we explain how the insertion is done and then we show the results.

5.1 The Process of Inserting Face to Photo

To insert the generated face back to the photo, we first locate it in the photo, then we find the facial landmarks, extract the face in the required format by the generator, we find the latent code by GAN inversion, we manipulate the code and then finally blend the manipulated image back to the original photo. This process is shown in Figure 5.1.

Face Normalization. The first step is to locate the face and crop the image accordingly. We use a facial landmark detector to obtain 68 landmarks. We do same transformations as the the authors of the *FFHQ* [18] dataset to obtain the normalized image, of resolution 256×256, where the nose should be in the center of the photo, the eyes should be in the same height, and the eye-to-eye distance is a given constant. This gives a template landmark locations. From the correspondence between detected landmarks and template landmarks, we estimate the affine transformation A. Then we use A to warp the input image into normalized image.

GAN Inversion and Image Manipulation. GAN inversion, as described Section 2.3, is performed to find the latent code. Arbitrary latent code manip-

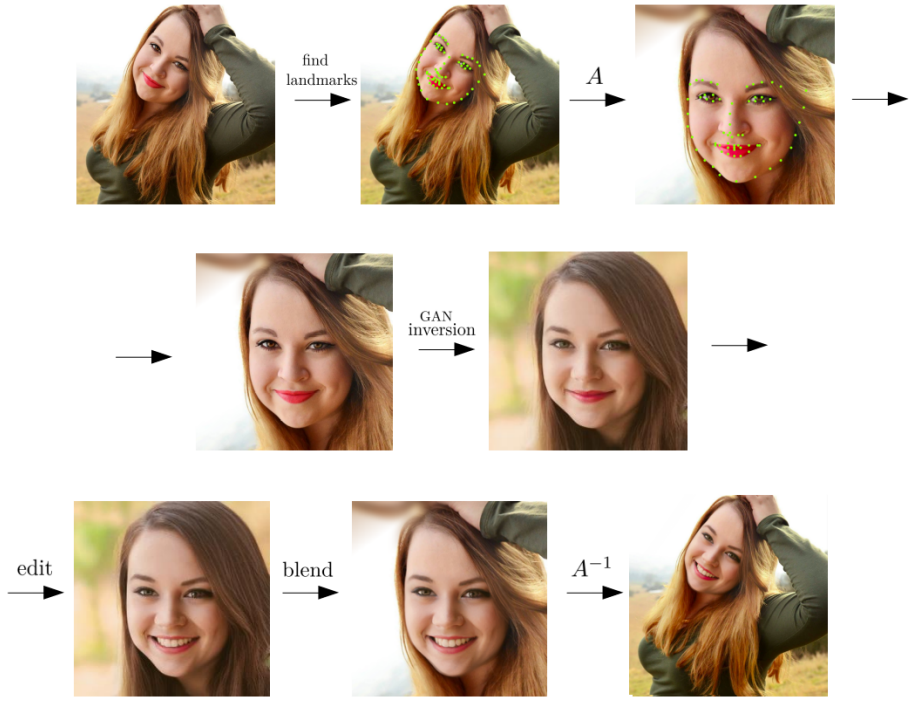


Figure 5.1: Process of editing the photo. We first detect the facial landmarks on the original photo, based on them we rotate and crop the image. Then we perform the GAN inversion, edit the face with an arbitrary latent manipulation method, blend it with the original background and insert the result back to the original photo.

ulation, as discussed in Chapter 3, is made, which gives us the manipulated image in normalized coordinates.

In all the experiments, we only consider inner face of the normalized image for the inversion. The inner face is detected with GrabCut algorithm [34]. Regions out of the inner face are masked out for the inversion loss functions. The reason is that we want to avoid reconstructing the background that may be complex for real photos. Consequently, the background (and hair) of the inverted face is different (see Figure 5.1), however, only the inner face is used for the manipulation and final processing.

Masking the Face into the Background. To seamlessly blend the manipulated face with the original photo, we use a smooth mask. We create a mask for the face area. The mask is created by *GrabCut* algorithm. We propose an initial estimate of the foreground (face) as a convex hull around the detected landmarks enlarged by 1.4 and we pass this to GrabCut. After having the mask, we smooth it with Gaussian blur, so the edges between foreground and background are not sharp. In Figure 5.2 we show how the mask is used to merge the face with the original background. The mask $\mathbf{M}$ has values in the range $[0; 1]$ and has the same resolution as the normalized images, 256×256. The result is obtained as follows:

Figure 5.2: Blending the manipulated face to the original photo using a mask. The manipulated image is multiplied element-wise with the mask **M** and original image is multiplied with inverted mask (1 - **M**). The masked images are summed up to obtain the final result.

$$\mathbf{x}_{final} = \mathbf{M} \cdot \mathbf{x}_{gen} + (\mathbf{O} - \mathbf{M}) \cdot \mathbf{x}_{orig}, \tag{5.1}$$

where we denote the the final image as $\mathbf{x}_{final}$, original image as $\mathbf{x}_{orig}$ and the manipulated image as $\mathbf{x}_{gen}$ and $\mathbf{O}$ is matrix of ones with same shape as matrix $\mathbf{M}$.

The last step is to insert the image with the generated face back to the original photo. We do that by an affine transformation A^{-1} and map the pixel coordinates with bilinear interpolation.

5.2 Example of Edited Photos

In Figure 5.3, we demonstrate an example of the entire pipeline result, i.e., an edited face blended into the original photography. Other example is shown in Figure 5.4, where multiple attributes are manipulated. We can see that the results do not suffer from any apparent artifacts from the blending.

We tried to do the same for cases where are more people in the photo. We show that in Figure 5.5. When there are more people in the photo, the attributes change for all faces in the same way. We also had to solve the problem of two faces being close to each other, therefore their close area was overlapping, which caused problems in the masking process. This was solved by editing the faces one by one. First inverting, editing and inserting the face back into the photo, then doing the same for the next image and so on. Again, the images looks realistic and the manipulation is not obvious.

(a) Original photo (b) Gender +2.7

Figure 5.3: Example of incorporating the generated edited face back to the original image.

(a) Original photo

(b) Age +2.3, eyeglasses -1.2, gender +2.1 (c) Age -0.6, gender -0.7, smile +1.6

Figure 5.4: Different settings of some of the facial attributes. The attributes that are not mentioned in the caption were not changed.

(a) Original photo

(b) Generated faces (not edited)

(c) Eyeglasses +2.5, gender -0.7

(d) Gender +1.7

(e) Smile -2

(f) Age +2.9

Figure 5.5: Examples of changing some of the facial attributes for multiple faces in the photo. When there are more people in the photo, all faces are changed the same way. The attributes that are not mentioned in the caption were not changed.

Chapter 6

Conclusion

In this work, we analyzed and compared different strategies for editing facial images with generative adversarial networks. First, we introduced the background – GANs, StyleGAN and GAN inversion, and then we described and proposed a method for face editing by optimizing the latent code to match the desired expression. We defined the expression by the positions of the facial landmarks and obtained the desired image by optimizing the latent code to minimize the objective function.

It has been shown in the Chapter 4 that our method achieves a quality similar to the current state-of-the-art methods. We elaborated on different parameter settings and analysed how it affects the generated image. We also compared our approach to the linear manipulation method. The proposed method shows better results in terms of preserving the person's identity and it allows to perform different edits of the faces. The downside of the method is that it takes much longer time. On the other hand, we can invert and edit the image at the same time and we do not need another encoder to invert the images.

We also showed that the edited faces can be blended into the original photos without any visible artifacts, so the person can see the changes directly in the photo, where can be even more than one face. This work can be extended by editing each face separately and by providing some better user interface, where a user can manipulate the facial landmarks, e.g., with mouse dragging instead of defining the changes as script parameters.

Bibliography

[1] R. Abdal, Y. Qin, and P. Wonka. Image2stylegan: How to embed images into the stylegan latent space?, 2019.

[2] R. Abdal, Y. Qin, and P. Wonka. Image2stylegan++: How to edit the embedded images?, 2020.

[3] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan, 2017.

[4] P. Baylies. stylegan-encoder. `https://github.com/pbaylies/stylegan-encoder`, 2019.

[5] Y. Bengio and Y. Lecun. Convolutional networks for images, speech, and time-series. 11 1997.

[6] A. Bulat and G. Tzimiropoulos. Binarized convolutional landmark localizers for human pose estimation and face alignment with limited resources. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[7] A. Bulat and G. Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In *International Conference on Computer Vision*, 2017.

[8] A. Creswell and A. A. Bharath. Inverting the generator of a generative adversarial network, 2016.

[9] J. Deng, J. Guo, N. Xue, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition, 2019.

[10] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks, 2016.

[11] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[12] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[13] S. Guan, Y. Tai, B. Ni, F. Zhu, F. Huang, and X. Yang. Collaborative learning for faster stylegan embedding, 2020.

[14] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017.

[15] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks, 2018.

[16] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016.

[17] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.

[18] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018.

[19] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan. *CoRR*, abs/1912.04958, 2019.

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[21] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2017.

[22] Y. Li, S. Liu, J. Yang, and M.-H. Yang. Generative face completion, 2017.

[23] Z. C. Lipton and S. Tripathi. Precise recovery of latent vectors from generative adversarial networks, 2017.

[24] D. Mahapatra and B. Bozorgtabar. Progressive generative adversarial networks for medical image super resolution. *ArXiv*, abs/1902.02144, 2019.

[25] M. Mirza and S. Osindero. Conditional generative adversarial nets, 2014.

[26] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *CoRR*, abs/1802.05957, 2018.

[27] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation, 2016.

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner,

L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[29] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez. Invertible conditional gans for image editing, 2016.

[30] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[31] E. Richardson, Y. Alaluf, O. Patashnik, Y. Nitzan, Y. Azar, S. Shapiro, and D. Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. *arXiv preprint arXiv:2008.00951*, 2020.

[32] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[33] rosinality. `https://github.com/rosinality/stylegan2-pytorch`.

[34] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23:309–314, 08 2004.

[35] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 397–403, 2013.

[36] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans, 2016.

[37] Y. Shen, C. Yang, X. Tang, and B. Zhou. Interfacegan: Interpreting the disentangled face representation learned by gans, 2020.

[38] T. Silva. `https://sthalles.github.io/intro-to-gans/`.

[39] A. Šubrtová. Face interpretation problems on low quality images. 2018.

[40] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977.

[41] Z. Wang, Q. She, and T. E. Ward. Generative adversarial networks in computer vision: A survey and taxonomy, 2020.

[42] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention, 2018.

[43] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018.

[44] J. J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. *CoRR*, abs/1609.03126, 2016.

[45] J. Zhu, Y. Shen, D. Zhao, and B. Zhou. In-domain gan inversion for real image editing, 2020.

[46] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold, 2018.

[47] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020.

[48] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman. Toward multimodal image-to-image translation, 2018.

[49] X. Zhu, X. Liu, Z. Lei, and S. Z. Li. Face alignment in full pose range: A 3d total solution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(1):78–92, Jan 2019.

Appendix A

CD Contents

- petrzelkova_thesis.pdf - the thesis in .pdf format
- Thesis_LaTeX.zip - a compressed file containing the latex source code and images