UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL DE CALCULATOARE

PROIECT DE DIPLOMĂ

Generarea fețelor umane

Andrei-Bogdan Florea

Coordonator științific:

Prof. dr. ing. Irina Georgiana Mocanu

BUCUREȘTI
2023

UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

DIPLOMA PROJECT

Human face generation

Andrei-Bogdan Florea

Thesis advisor:

Prof. dr. ing. Irina Georgiana Mocanu

BUCHAREST
2023

CONTENTS

SINOPSIS

Odată cu utilizarea pe scară largă a învățării automate, cererea pentru aplicații care valorifică potențialul acestui domeniu a crescut substanțial. Aceste tehnologii necesită o cantitate masivă de date pentru a satisface constrângerile de calitate, iar folosirea exclusivă a datelor reale are uneori rezultate limitate. Această lucrare prezintă o modalitate de generare a unor imagini cu fețe umane folosind o rețea neurală generativă, cu aplicabilitate în problema segmentării semantice a părților componente ale feței. Așadar, folosind o rețea convoluțională ca extensie a generatorului, se obțin etichete asociate imaginilor generate, cu un F1 mediu de 92.01% pentru 18 clase semantice. Pentru a controla generarea, se realizează conversia unor valori explicite de vârstă și gen către spațiul de intrare al generatorului, cu ajutorul unei rețele de tip perceptron multistrat. Imaginea produsă reflectă atributele cerute, cu o eroare medie de 4.47 ani și respectiv o acuratețe de 95.53% a genului.

ABSTRACT

With the widespread use of machine learning, the demand for applications that leverage the potential of this domain has increased substantially. Such technologies typically require massive amounts of data in order to satisfy quality constraints, and using real data exclusively sometimes leads to limited results. This paper presents a way of generating images with human faces based on a generative adversarial network, with the purpose of being later used for the task of face parts semantic segmentation. Therefore, through a convolutional neural network added as an extension of the generator, semantic segmentation labels are obtained for the generated images, with a mean F1 of 92.01% for 18 classes. To control the generation process, a multilayer perceptron network is trained to map explicit values of age and gender into the latent space of the generator. The generated images reflect the requested attributes, with a mean age error of 4.47 years and a gender accuracy of 95.53%.

1 INTRODUCTION

We begin this chapter by offering an overview of the context that has inspired this paper, along with mentioning the problem that it tries to tackle. We briefly present the objectives of our research, along with the proposed solution for reaching those goals. Finally, we touch on the results and introduce the structure of the following chapters.

1.1 Context

Data has always been essential to solving real-world problems using machine learning, and more recently, deep learning, alongside the learning algorithm used and available computational power [23]. Although most of the early development of the field was focused on real data, there have been some studies indicating the possibility of using artificial data in the 1990s to protect sensitive information in datasets [49]. However, the process of generating completely new data points has become apparent after 2014, starting with the paper on Generative Adversarial Networks proposed by Ian Goodfellow et al. [17]. Since then, synthetic data generation has become an actively researched topic, warranting papers that reflect on the progress of the domain [34, 25] and how it has had multiple notable use cases such as preserving privacy in healthcare [62], improving the performance of computer vision applications [61] or enabling the voices of digital assistants [59]. Moreover, the topic of synthetic data has attracted public attention due to recent advances in image generation [46, 47], while it has also created concerns related to the possibility of creating harmful artificial content, namely deep-fakes [60].

With this context in mind, we explore the current capabilities of human face generation and analyze ways of using it to create custom training data for building powerful computer vision applications.

1.2 Problem

Machine learning researchers or developers often require the use of extensive labeled training data. This usually implies using public data, which might not fit their problem. Because of the nature of some target domains (for example, the medical domain [12]), data may not be publicly available. Even if the public data exists and it fits the given problem, it might not cover the full distribution of possible scenarios or it might contain certain unwanted biases [36]. The alternative of acquiring real data and manually annotating it with proper labels

later used in the training procedure is time-consuming and expensive [24].

To alleviate these issues, multiple techniques related to synthetic data have been proposed, including using special software (for instance, 3D rendering engines [9]) or generative neural networks [3, 17, 63]. With regards to face synthesis specifically, using 3D models has proved to be useful in terms of the quality of annotations and control over the generation process [61], but there still exists a domain gap between the rendered models and natural images [67]. The alternative of using a Generative Adversarial Network [17] reduces the gap, but obtaining high-quality annotations and control over the generation is more difficult.

1.3 Objective

In this paper, we leverage the performance of a generative network in order to obtain a framework for generating images of human faces, alongside corresponding labels, with control over certain facial attributes. With this in mind, we aim to obtain semantic segmentation labels of face parts and controllable age and gender attributes. Semantic segmentation represents assigning a certain class for all pixels of an image, more precisely, in our case, face semantic segmentation consists of annotating each pixel with a value corresponding to the element of the face it is part of. We analyze the feasibility of the solution both qualitatively and through the ease of use compared to other methods.

1.4 Proposed solution

In order to reach the previously mentioned targets, we use the StyleGAN2 [29] architecture. It is a well-established and researched generative adversarial network that was trained to produce human faces. We build upon the network in two directions that help reach our goal of creating high-quality annotations and controllable synthesis.

Firstly, with a way to encode information about the controlled attributes of the person into the latent space of the network, a face with the desired features is obtained. The latent space of a generative network refers to a multi-dimensional space, from which random vectors are extracted to serve as initialization or input to the generator network.

Secondly, in order to obtain the semantic segmentation of face parts, we update the generator network architecture with an extension that predicts per-pixel labels. This new part is trained without affecting the weights of the generator network and works by transferring the generator's knowledge of a face currently synthesized onto the new task of assigning semantic segmentation labels. This training was performed on a small set of manually annotated images produced beforehand by the generator.

The developed solution proves to be effective in achieving the proposed objectives, namely face generation with associated semantic segmentation labels. Moreover, the techniques involved

may also serve as a baseline for creating synthetic training data for other tasks, as the approach is not limited to the semantic segmentation task.

1.5 Results

As previously mentioned, the objective is twofold: to acquire semantic segmentation labels of face parts and to obtain control over face attributes.

For the problem of semantic segmentation, the evaluation consists of measuring the quality of the segmentation prediction on a set of carefully annotated images with metrics specific to this task, including an overall metric for the entire image, and separate metrics for each of the semantic classes. With this in mind, we report an overall accurate result of 92.01% mean F1 score for 18 semantic classes, on a subset of the manually annotated images specifically reserved for testing the solution.

In terms of control over attributes, the proposed solution is evaluated from the perspective of both the accuracy and diversity of the generated samples, where we achieve high precision of control (4.47 mean age error and 95.53% gender accuracy) while maintaining high sample quality and diversity.

1.6 Paper structure

In the next chapter, we list our reasons for developing the human face generation framework. We not only discuss the motivation behind synthesizing data in general but also reflect on our choice of face generation and face semantic segmentation.

Following that, chapter 3 details the existing methods. After a brief presentation of the context surrounding synthetic data, we explore a few concepts and work related to the problem at hand. Based on previous solutions, we choose a couple of directions to investigate and integrate in our framework.

In chapter 4, an overview of the proposed system is presented. This includes the overall architecture and detailed views of each component, with an emphasis on the functionality instead of the implementation.

The next chapter is strongly related to chapter 4, in that we detail the procedures involved in developing the system, including specific implementations of each component.

In chapter 6, we build an argument for the relevancy of our work, by presenting the results and comparing them to previous works.

Lastly, we reserve a chapter for drawing conclusions and mentioning further research directions.

2 MOTIVATION

In this chapter, we reflect on the key concepts that establish the motivation behind this paper. These factors include advancing research in synthetic data generation, which is vital to many current applications, creating a cost-effective alternative for businesses that require the use of massive amounts of data, and understanding the impact of artificial data on society. On the topic of research, we also discuss the motivation for tackling the subject of human face generation and face semantic segmentation and show its potential use cases.

Scientific motivation

With the development of machine learning and the strong interest shown by the research community, new ideas and technologies are bound to emerge. Recently, this has been the case with synthetic data, which has seen tremendous growth over the last few years. The emergence of artificially generated data has been fueled in large part by the advantages it offers over real data and its applicability in various sectors.

To explain further, consider using synthetic data in the context of developing a certain solution. In a simplified and controlled scenario, the development of an initial solution is accelerated [24]. Afterward, one can revert to running experiments in a real environment and solve the smaller-scale problems that may arise.

Another aspect worth mentioning is that synthetic data is capable of improving the performance of systems based on artificial intelligence in a variety of tasks, especially when mixed with real training data [53]. This approach has been shown to have practical results in tasks such as semantic segmentation of urban scenes [51], activity recognition based on sensor data [11], or facial landmark detection [61].

Implications for businesses

From the perspective of a business that develops machine learning technologies, the success of a solution relies in a major part on the data the algorithm or neural network is trained on. As relevant data sources, businesses can generally employ the use of public or proprietary data, both of which can help achieve a reasonable performance.

Data from external sources can be unreliable, whether it is not available due to privacy concerns [12], is unfit for the problem at hand (i.e. the training labels do not correspond to the requirements), or does not cover all of the scenarios in which the solution is expected

to perform (i.e. training a solution on images with front-facing people, but expecting it to perform under different head orientations).

Proprietary training data that was designed for the given task usually offers a great boost of performance to a neural model. However, the acquisition process is expensive and laborious [40]. Moreover, once acquisitions are made, one can return to fix a previous mistake only by relabeling the existing data or by conducting new acquisitions.

Therefore, what motivates the use of synthetic data is its accessibility. Generating artificial data is a process that requires technical expertise, but the results are high-quality, scalable, and flexible. Synthetic data also severely diminishes potential privacy-related issues that may arise when using real data [6], a topic that we will elaborate on in the following paragraphs.

Social impact

With the increasing volumes of data collected by businesses, the issue of digital privacy has recently gained attention from the public, resulting in several legislative changes, most notably the General Data Protection Regulation (GDPR) [15].

What motivates the use of synthetic data here as an alternative to real data is cases where privacy issues arise. There have been studies, such as the one conducted by John M. Abowd and Lars Vilhuber [2], that measure to what extent an artificial distribution of data resembles the original one, while still complying with an imposed level of privacy.

In the case of human faces specifically, privacy aspects have also been researched, with Fadi Boutros et al. [6] showing that one can obtain a high-performance face recognition system by using a privacy-preserving dataset of synthetic faces, where identities from the original dataset were not present in the generated dataset. Therefore, it is clear that privacy is an important aspect to consider in the problem of generated faces, which motivates us to produce synthetic data to be used for training in the downstream task of face parts semantic segmentation.

Human face generation and semantic segmentation

Face synthesis is an interesting and actively researched topic [28, 29, 30], in many ways serving as a benchmark for measuring the quality of image generation methods. Among its use cases, we can name content creation, video game character modeling, semantic face editing [54, 55] and, notably, improving the performance of deep neural networks in tasks that require processing human faces [61].

In addition, the task of semantic segmentation of face parts (or face parsing) has several direct applications, such as background removal or blurring in video conferencing [26], face beautification, or serves as an auxiliary tool for other applications, such as gaze tracking [8] or person re-identification [70].

3 EXISTING METHODS

In this chapter, we present the research ideas that form the foundation of this paper. We start by offering a general context for the problem of data generation, with an accent on artificial human faces. Afterward, we introduce a few concepts our work is based on. We then proceed with details on work related to this paper, which can be divided into three parts. Firstly, we give an overview of the most notable image synthesis methods. Following that, we analyze ways of obtaining annotations for data generated by such image generators, in our case - semantic segmentation labels. Later, we study a few techniques that achieve the goal of controlling certain attributes of the generated image. Ultimately, we offer detailed reasoning for the choices made in developing our solution.

3.1 Data generation context

Data synthesis is a broad topic and sub-domain of artificial intelligence with important ramifications. Synthetic data refers to data obtained through the use of computer software or algorithms, rather than through capturing aspects of the real world, as is the case of a photograph or sensory data, for instance. It is important to make the distinction between data synthesis, which creates completely new data points, from data augmentation, which consists of modifying already-existing real data in some way, such as applying geometric transformations (scaling, rotation, cropping, perspective changes) or color transformations (adding noise, sharpness or color jitter) to an image or changing the pitch of an audio sequence.

As also stated in the previous chapters, synthetic data can bring certain advantages compared to real data. It has been extensively studied because of its various applications, from overcoming data scarcity to improving the performance of artificial intelligence systems or protecting the rights to privacy.

However, its disadvantages are noteworthy as well. The harmful use of synthetic data through deepfakes has generated concerns. "Deepfake" [60] is the term used for artificially generated content that aims to replace a real person's identity, whether through video or audio, generally with malicious intent. Even though there have been several research papers that focus on detecting such malicious content [18, 16], it is crucial to understand and control the risks associated with synthetic data generation technologies.

While notable early studies have been conducted on obtaining and using synthetic data in different systems, mostly for privacy-related issues [49], significant progress in the field was made together with the evolution of computer vision and the emergence of image generation

neural architectures and learning techniques. On one hand, this evolution has been powered by persistent improvements in hardware performance. On the other hand, accessible and efficient frameworks such as PyTorch [41] and TensorFlow [1] have enabled the rapid development of deep learning technologies.

An interesting application of synthetic data is the problem of human face generation. Having recently improved to remarkable quality, synthetic faces play a key role in boosting the performance of computer vision systems or modeling video game characters. What demonstrates this recent surge in image quality are experiments such as the one conducted by Sophie J. Nightingale and Hany Farid [39], where they show that participants are unable to distinguish between real or synthetic faces, having only a 48.2% accuracy in this task.

3.2 Concepts used

Before exploring the following sections on related work and our chosen solutions, we must first introduce a few fundamental notions. Initially, we present the details of Generative Adversarial Networks. Secondly, we establish how they were adapted for the task of image generation, through the use of Convolutional Neural Networks. Lastly, we elaborate on the metrics used in the image generation task.

3.2.1 Generative Adversarial Networks

One of the breakthroughs in the context of data generation has been the use of the Generative Adversarial Networks (GAN) framework for training, proposed by Goodfellow et al. [17]. They model the problem of data synthesis as a minimax game between two networks, named the generator G and the discriminator D. The purpose of the discriminator is to distinguish between real samples and fake samples produced by the generator. The role of the generator is to create samples that follow the distribution of real data. The two networks are trained simultaneously until they converge to a point where the discriminator is unable to fulfill his task.

During the training process, the generator learns a function $G(z)$ that maps input noise vectors z onto the distribution of the data that is to be generated. At the same time, the discriminator learns to assign the probability that a data point x comes from the real distribution rather than the one learned by the generator, and we will refer to that probability as $D(x)$.

The generator is trained to maximize the chance that samples produced are labeled as real by the discriminator, by minimizing the equation:

$$log(1 - D(G(z))) \tag{1}$$

Simultaneously, the discriminator is trained by maximizing:

$$log(D(x)) + log(1 - D(G(z))) \tag{2}$$

where x is some data from the real distribution and z is the input noise vector of the generator.

Since their proposal, GANs have been extensively studied and improved. To access their effectiveness in the context of the highly-dimensional image generation problem, Convolutional Neural Networks have been introduced as a basis for the generator and discriminator models, instead of the multilayer perceptrons proposed in the initial paper.

3.2.2 Convolutional Neural Networks

The soaring popularity of the computer vision domain among research groups in the 2010s has in large part been attributed to the performances that convolutional neural networks (CNNs) can achieve in computer vision tasks, including image generation. They power the analysis and representation of more complex and higher-dimensional problems, while maintaining a reduced set of learnable parameters and satisfying memory constraints.

The overview of a typical CNN used for image classification is illustrated in Figure 1, where the input image is passed through an image encoder that is composed of convolutional and pooling layers. In order to classify the input image, the feature map is flattened and passed through a feedforward network that in the end predicts probabilities that the input image belongs to each class.

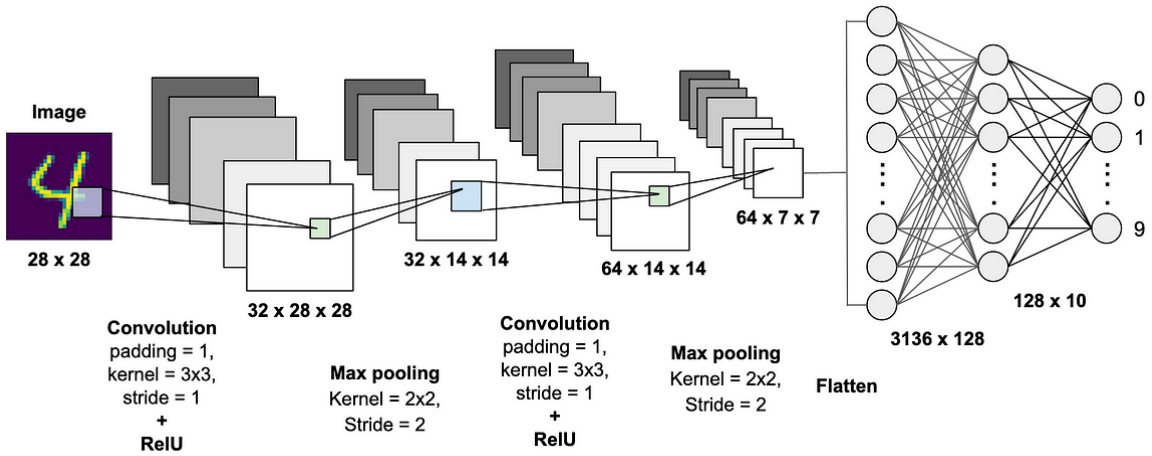


Figure 1: Typical Convolutional Neural Network [42]

The conventional convolutional neural architecture, initially used for other vision tasks, was altered to better model the problem of image generation. Alec Radford et al. [45] introduced the deep convolutional generative adversarial network (DCGAN). The architecture of the developed generator is displayed in Figure 2. The main change to the conventional convolutional architecture consists of removing any pooling layers, which are replaced by upsampling layers

in the form of transposed convolutions (deconvolution) for the generator, and strided convolutions for the discriminator. This type of layer facilitates the learning of a custom upsampling operation, required for mapping the lower dimensional input latent vector to the final image space.

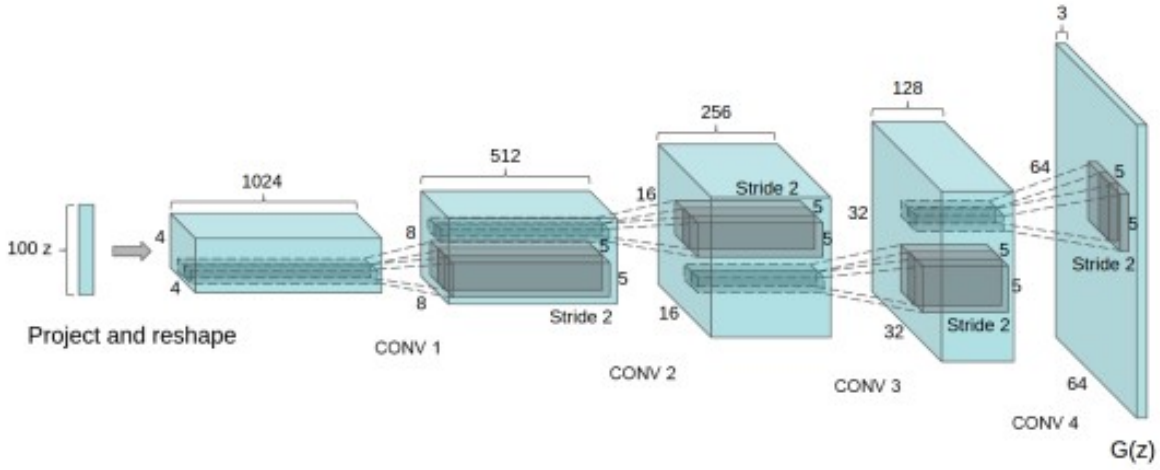


Figure 2: DCGAN generator architecture [45]

3.2.3 Image generation evaluation metrics

Properly assessing the performance of image synthesis models remains an active discussion among the research community. The most popular metrics currently used for this task include the Inception Score (IS) [52] and the Fréchet Inception Distance (FID) [20].

Both of them rely on the concepts of fidelity and diversity when comparing the distribution of the generated data to the real distribution. Fidelity measures how close the generated samples resemble the real ones, while diversity represents the amount of the real data distribution that is covered by the generated data.

We mention the results of generator networks in the following section using FID and use it to guide the choices for our system. The FID score aims to be an improvement over the Inception Score, and relies on comparing the distributions of the real and fake (generated) feature maps at the deepest layer of the Inception v3 model [58]. Assuming the real feature maps follow a Gaussian distribution with mean and covariance matrix (μ_X, Σ_X) and the generated ones follow a Gaussian distribution with mean and covariance matrix (μ_Y, Σ_Y)), the FID score is computed as the Fréchet distance [14] between those distributions:

$$d^2 = |\mu_X - \mu_Y|^2 + tr(\Sigma_X + \Sigma_Y - 2(\Sigma_X \Sigma_Y)^{1/2}) \tag{3}$$

3.3 Related Work

The purpose of this section is to establish the connections between this paper and other works that tackle the issue of face generation. We analyze all of the technologies in detail, mentioning their advantages and limitations.

Since the purpose of our system is to generate training data for a downstream task (in the case of this paper - face parts semantic segmentation), we follow three research directions:

- face generation, in order to identify the best-performing methods
- obtaining training labels associated with images, requiring semantic segmentation labels in particular
- controllable synthesis, with the purpose of enabling the generation in underrepresented cases

3.3.1 Face generation

The task of face synthesis has had some early solving attempts, but researchers struggled with the high-dimensionality nature of this problem.

One of the first solutions came through the use of a 3D morphable model (3DMM) [5]. It enables the possibility of estimating 3D faces from single images, based on minimizing a series of parameters that form the representation of a face: shape, texture, pose, scale, illumination, etc. After obtaining the initial estimation, the face can be manipulated in the parametric space. Other works have developed their own representation of the 3D morphable model, most notably the Basel Face Model (BFM) [43] and FLAME [33]. However, although they can freely modify the face in the parametric space, one shortcoming of all of these methods is that they rely on preexisting UV texture maps of faces for the initial rendering. That is, they do not generate the original face. Most recently, 3DMMs have been incorporated into other GAN-based neural models [35] that synthesize realistic faces based on the 3DMM parameter priors.

Another early use of face synthesis has been in face recognition, which consists of identifying or verifying one's identity based on their face. To help solve this problem, face synthesis has two distinct use cases. Firstly, Hassner et al. [19] create a framework that is based on face frontalization, where they obtain a frontal view of a query person in order to simplify the challenging problem of recognizing faces regardless of head orientation. On the other hand, Peng et al. [44] take the opposite approach. They solve the problem of recognition under any viewpoint by synthesizing pose variations of faces used for training. Again, both of these methods prove useful for the task of face recognition, particularly because they are based on a preexisting input face. What motivates us, though, is the ability to generate new faces.

As stated in the previous section, a significant step in solving the problem of unconstrained face

generation has been the introduction of Generative Adversarial Networks. After their initial proposal by Goodfellow et al. [17], several efforts have been made to improve the general framework, whether through the use of different adversarial learning objectives [66, 4] or by employing architectural changes to the networks of the generator and discriminator, including the unanimous usage of CNNs.

To start with, Radford et al. [45] proposed some architectural changes for CNNs in order to generate images with a resolution of 64x64, including the replacement of all pooling layers with transposed convolutions, the use of batch normalization in both the generator and the discriminator in order to facilitate training and the removal of the fully connected layers. Although the results are far from the ones achievable later on, these changes paved the way for the following work on image generation.

Karras et al. [27] proposed ProGAN (Progressive Growing GAN) for obtaining images at high resolution and quality. They solve training instability by gradually increasing the depths of the generator and discriminator, adding new layers that increase the resolution, in the end producing high-resolution 1024x1024 face images. The model is later evaluated to a Fréchet Inception Distance (FID) of 8.04, after the introduction of the FlickrFacesHQ dataset.

Further, Karras et al. [28] introduce a new generator architecture named StyleGAN that builds upon their previous work on ProGAN in different ways, with the purpose of learning a more disentangled and interpretable latent space. Disentanglement refers to the notion of separability on the semantic level between different image features, such as high-level ones (identity, pose) or lower-leveled ones (hair, illumination, background). One addition to the ProGAN methodology is the use of a mapping network consisting of eight fully-connected layers that transform a random latent vector into a style vector. Based on notions from the literature on style transfer, they add the style vector to each block of the generator through an operation called adaptive instance normalization (AdaIN). It consists of standardizing the feature map x_i of a certain block and then scaling and biasing the result with the style vector y:

$$AdaIN(x_i, y) = y_{s,i}\frac{x_i - \mu(x_i))}{\sigma(x_i))} + y_{b,i} \tag{4}$$

Another particularity of the StyleGAN architecture is the noise injection. Before each AdaIN operation, random Gaussian noise is added to the feature map, resulting in an increased level of variation of the generated samples.

With increased interest from the research community, the same researchers later introduced StyleGAN2 [29], an updated version of StyleGAN with an increase in image quality. Instead of injecting the style vectors in the architecture through the use of the AdaIN operation, they are added through weight modulation and demodulation, with an impact on reducing artifacts in the generated images. The injection of the style vector s is embedded directly into the weights w of the convolutional layer in each style block, as seen in the following equation:

$$w'_{i,j,k} = \frac{s_i \cdot w_{i,j,k}}{\sqrt{\sum_{i,k}(s_i \cdot w_{i,j,k})^2 + \epsilon}} \tag{5}$$

In addition, StyleGAN2 completely removes the progressive increase in resolution during training in favor of skip connections at all stages, both for the generator and the discriminator. With these changes, StyleGAN2 manages to improve the FID score on the FlickrFacesHQ (FFHQ) dataset introduced together with the StyleGAN paper from 4.40 to 2.84 at the 1024x1024 resolution.

A novel approach to face generation has been through the use of 3D-aware methods. Previously mentioned GAN methods have a limited understanding of the 3D nature of the face, having poor results under large poses. To alleviate this issue, neural radiance fields (NeRF) [37] have been incorporated into image synthesis methods. NeRFs provide a way of representing a scene with the help of fully-connected neural networks by obtaining the volume density and radiance at a certain 3D spatial location, viewed from a certain direction. Although these methods improve face generation consistency across different views, the process is computationally intensive, resulting in lower-resolution renderings. For instance, Chan et al. [7] obtain an FID score of 14.7 on the CelebA dataset at 128x128 resolution, while Sun et al. [57] reach an FID of 12.1 at the same resolution.

Another recent development in the context of face synthesis has been the introduction of Denoising Diffusion Probabilistic Models [21]. Training with this framework takes a self-supervised approach, by applying noise on top of an image and teaching the neural model to revert the process, by removing noise step by step. After the network is trained, by starting from a noisy image drawn from the Gaussian distribution, one can produce a new unconstrained image simply by removing the noise step by step. Although they achieve state-of-the-art results in terms of image quality when training on a diverse dataset [13], diffusion-based models are not optimal in terms of the computational power required both for training (where each step of denoising is taken at high resolution) and inference (because of the sequential procedure for denoising). Although some work has been directed at reducing these issues by applying the denoising steps in a lower-dimensional latent space [47], there is still a gap between these models and GAN-based approaches, that synthesize a new image in a single forward pass and yet achieve a higher FID on the face generation task.

We summarize the key aspects of the generative methods in Table 1, in order to guide the decision for our choice of generator network.

3.3.2 Obtaining training labels

Previously discussed methods focus on generating the face itself. However, an important aspect to consider when creating synthetic data for training is how to obtain the label corresponding to the synthesized image. A target label represents the information required to train

Method	Advantages	Disadvantages
3D morphable model	Full control over a series of parameters (pose, illumination, face shape)	Requires pre-existing face images or 3D assets
GAN-based models	State-of-the-art image quality and efficient inference	Harder to control explicitly
3D-aware GANs	More control in the 3D space	Lower image quality than standard GANs
Diffusion-based models	High image quality	Both training and inference are slow

Table 1: Overview of face generation methods

a network in a supervised manner. We focus on obtaining face parts semantic segmentation labels as per-pixel annotations are one of the most tedious annotations to manually perform, and automatically labeled synthetic data would potentially save costs and effort. Therefore, it is essential to obtain the segmentation map associated with each image with little manual effort.

One of the ways of automatically obtaining annotations for synthetic data, and for faces in particular, is through the use of 3D models. The FaceSynthetics dataset [61] in particular is constructed by sequencing a series of operations on top of a parametric face model, similar to the 3D morphable model mentioned earlier. These operations include randomly assigning identities, expressions, texture, hair, clothes, and backgrounds. Semantic segmentation labels, in this case, are perfect and result in high performance when training a segmentation network on synthetic data alone, with an overall F1 score of 92.0 on the Helen dataset, compared to 91.6 when training on real data alone. Although the results are impressive, this method heavily relies on having a large collection of 3D assets.

Another way of acquiring labels for generated samples is in an implicit manner, by having a conditional generative network that produces a result based on an input conditional signal. Within the context of image classification, a controllable GAN that can produce an image belonging to a certain class encoded in its input would help enhance the performance of the initial image classifier on that particular underrepresented class. Therefore, because the GAN is conditioned on an input class, training data is implicitly obtained for the opposite task of assigning a class to an image.

Besides class labels or other similar conditional signals, one can create a conditional GAN based on an input image. Through the use of this framework, Jiang et al. [22] propose SGGAN - Segmentation Guided Generative Adversarial Networks. They control the generator with the use of an input face image and a target face parts segmentation and train it to produce the same face under a novel view that matches the target segmentation. Although the framework achieves the goal of assigning semantic segmentation labels to "warped" faces, it relies on the performance of the segmentation network and it does not generate new faces.

Zhang et al. [65] provide DatasetGAN, a method for generating a large amount of training data based on the StyleGAN generator, with a small annotation effort. In their approach, a few images produced by the generator are manually annotated with the required per-pixel labels. From these images, they train a feature interpreter consisting of three fully-connected layers to map the feature maps at each depth of the generator network to a class label for each corresponding pixel. Because the feature maps have different spatial resolutions, they upscale all features to the highest resolution and concatenate them before feeding them to the style interpreter. Because this input has high dimensionality and does not fit memory constraints, they only use a random subset of feature vectors from the concatenated feature volume when training. They extensively test this framework on multiple tasks, including semantic segmentation and keypoint localization, with significant performance improvements over other methods used when only a few labeled examples are available, such as transfer learning and semi-supervised learning, obtaining a mean intersection over union (mIoU) of 53.46, compared to 45.77 and 48.17 respectively on a test set of face images annotated with 34 semantic segmentation classes.

Building on DatasetGAN, Li et al. develop BigDatasetGAN [32], which scales the framework for a much larger problem of obtaining semantic segmentation labels for a generator trained to produce ImageNet samples. Instead of StyleGAN, they employ the use of a generator network called BigGAN that was trained on ImageNet 1K [50] and label 5 samples per class in order to train a convolutional style interpreter, instead of the fully-connected one used previously. The segmentation branch added to the BigGAN network is altered. Starting from the features with the lowest spatial resolution, a feature map is passed through a convolutional layer, upsampled and concatenated to the following feature map in terms of spatial resolution, before passing again through the same operations. After each concatenation, a mix-conv layer is applied, which consists of two convolutional layers and a conditional batch normalization, where the output takes the class into account. This approach maintains the performance of the feedforward network employed in DatasetGAN, while having a much lower memory footprint, therefore not requiring the use of the pixel sampling trick.

To sum up, the most important advantages and limitations of each method are mentioned in Table 2.

Method	Advantages	Disadvantages
3D rendering	Perfect labels due to the nature of the rendering process	Requires 3D assets, including faces
Image-to-image translation	The associated segmentation label is obtained implicitly	Based on pre-existing segmentation maps
Feature interpreting network	Labels are not perfect and depend on the quality of the generated sample	Easy to integrate into any generator architecture

Table 2: Overview of methods for obtaining labeled data

3.3.3 Controllable synthesis

Conditioning a generative network on a certain variable has been a popular research topic ever since the introduction of GANs. Being able to control the generation process can be extremely useful in the context of generating synthetic training data, as it allows implicitly obtaining ground truth labels from the control signal itself. Moreover, it proves to be useful for generating samples of underrepresented classes or ones with semantic features rarely present in the original real data. Supplementing the training with synthetic data of this kind typically leads to improved performance in such edge cases.

Conditional Generative Adversarial Networks [38] are a clear extension of the original GAN architecture. The architectures of the generator and the discriminator are both modified to support another input that represents the conditional information. Thus, a generator learns to produce a sample that satisfies that particular condition, otherwise, it would be easy for the discriminator to detect it as being fake. This method relies on being able to embed the conditional signal into a vector and therefore semantic information about the generated images cannot easily be controlled. In addition, adding control requires retraining the generator and discriminator networks.

Other works rely on the disentangled representation of a GAN's latent space and use it to understand and perform the task of semantic editing.

InterFaceGAN [54] proposes a manner of enabling such editing in the latent space of GANs, leading to control over face attributes such as age, gender, or expression. They conduct the experiments on StyleGAN, showing a way of "walking" along a certain latent direction that resembles a semantic attribute. They treat every attribute as binary (with positive scores and negative scores), and train a linear SVM (support vector machine) to find the best separation hyperplane between the positive and negative samples. As a consequence of that, distancing away from the hyperplane in each direction means more and more positive (or negative) results. For instance, distancing from the age hyperplane relates to obtaining a photo of a younger or older person. Although the results of the classification accuracy with regards to the separation boundary are excellent (an accuracy of 98.7% on gender and 97.9% on age), the downside of using this framework is not having the ability to explicitly control the value of a discrete or continuous attribute (years of age, degrees of yaw as pose), instead only relying on "adding" or "subtracting" that attribute.

A method that achieves this goal is presented by Shoshan et al. [55]. They propose a way of explicitly controlling the values of semantic attributes such as identity, pose, or age, by training a StyleGAN2 generator architecture to separately encode the controllable attributes in the input style vector. The 512-dimensional vector is split into N+1 parts, where N is the number of controllable attributes and one part is left for encoding the rest of the image features. The generator is trained through contrastive learning on top of the usual adversarial training objective, by penalizing samples in a batch that should have similar attributes but result in different ones in the generated image. By doing this, the generator learns a disentangled

representation of the attributes, that allows editing a semantic feature without affecting the other ones. To gain explicit control over these attributes, a feed-forward network is trained to map the human-interpretable values to the corresponding latent code. The precision of the control obtained through the proposed method is undeniable, with the average distance between the input variable and the attribute from the resulting image being on average 2.02 years in terms of age, or 2.29° in terms of the yaw angle. However, the requirement of retraining the generator through this framework and not being able to utilize the entire input latent space to represent image features results in a lower FID score, 5.72 compared to the 3.32 obtained through the usual training of StyleGAN2 at 512x512 resolution.

Finally, the strengths and limitations of the methods presented are discussed in Table 3.

Method	Advantages	Disadvantages
Conditional GANs	High control precision	Requires retraining the generator
InterfaceGAN	Does not require retraining the generator	Is better suited for binary attributes; continuous ones are not explicitly-controlled
GAN-Control	High control precision	Requires retraining the generator

Table 3: Overview of methods for controlling the generation

3.4 Chosen solutions

With all of the information presented earlier, we further elaborate on the choices made for our system, offering a preview for the next chapter. To enable the generation of human faces alongside associated semantic segmentation labels, we provide our reasoning for the selection of each component or method involved, namely:

- the face generation network
- the automatic labeling procedure
- the method of controlling face attributes

3.4.1 Face generation network

Firstly, based on the previous analysis of the different available face generation methods listed in section 3.3.1, we choose to use StyleGAN2 as our face generator, in large part due to the quality of generated images in comparison with other methods. In addition, the choice is guided by the number of research directions that stem from it, ranging from obtaining labels to interpreting the generator's latent space.

There are multiple configurations of the StyleGAN generator, constructed to produce images at a specific resolution. More specifically, the three available variants of the architecture have an output resolution of 256x256, 512x512, and 1024x1024, respectively.

Before choosing the variant, we comparatively assess the quality and diversity of the generated images through the FID score and the memory requirements in inference mode for each of the three configurations. The results of our study are shown in Table 4. The FID score is computed based on 50K synthetic samples produced by each variant and the entire original FFHQ dataset, consisting of 70K images, downsampled to the respective resolution of each variant.

Variant	FID score	GPU memory used
256x256	5.74	0.95 GB
512x512	2.95	1.28 GB
1024x1024	2.75	1.86 GB

Table 4: StyleGAN2 variants comparison

With these results in mind, we choose the 512x512 generator network for developing our system. Compared to the 256x256 network, it obtains a much higher FID score of 2.95 vs. 5.74, without a large trade-off in memory used. Despite the lower resolution compared to the 1024x1024 variant, implicitly resulting in a smaller-sized network, it manages to obtain a respectable FID of 2.95 compared to 2.75, with a much lower memory footprint, which is what allows us to simultaneously dedicate a part of our training resources to the tasks of obtaining associated labels and controlling the generation. With the solution presented in the next subsection, training the annotation branch especially proves to be resource-intensive, as it requires loading multiple high-dimensional feature maps. Note that our study is not impeded by the lower resolution, but generating at a higher resolution would most likely lead to better performance on the segmentation around small-sized classes such as eyebrows, lips, or earrings. Another reason for using the 512x512 variant is so that our results are comparable to other related studies.

3.4.2 Automatic labeling procedure

Again, based on the observations in section 3.3.2, we make our choice of the method employed to acquire face semantic segmentation labels.

Even though 3D modeling methods can obtain perfect annotations, they require a large set of head scans and textures which are not publicly available in large part, making them not suitable for our work. The other mentioned method of implicitly obtaining segmentation labels by having the semantic map as one of the generator's inputs is promising and could yield high-quality annotations, yet it is not flexible, requiring retraining of the generator in case the labeling scheme is changed. Moreover, it relies on pre-existing segmentation masks, which limits the ability to synthesize completely new data points.

Therefore, in order to obtain semantic segmentation labels, we adopt the solutions described in DatasetGAN and BigDatasetGAN. They lead to acquiring high-quality annotations of entirely new faces, with only a small set of manually labeled images. This approach proves to be flexible, as it only requires the re-annotation of the small subset of sample images if the labeling requirements change. We further build our solution based on both papers, by maintaining the advantages of each one, while researching ways to improve the architecture of the additional segmentation branch used.

3.4.3 Control of face attributes

We are motivated by generating faces with explicitly-controllable attributes, which indicates the use of the method described in GAN-Control. However, we avoid the requirement of retraining the generator network from scratch with the latent space separation described in the paper. This supports our effort of creating a flexible framework that can be employed for any pre-trained StyleGAN-based generator.

Therefore, by relying on the disentangled nature of the latent space of the generator, we implement a modified version of the approach described in the paper: instead of training the generator with the custom-designed latent space vector, that has separate representations for each controllable attribute, we directly map all of the conditional variables into the latent space, and find which parts of the representation contribute the most towards the age and gender of the generated person. This way, we are able to precisely control attributes of the generated face, while still maintaining a high level of diversity. In fact, the method proposed allows for a trade-off between diversity and control precision.

The alternative provided by InterfaceGAN is promising as well, as it does not require retraining the generator. However, the framework is more aligned with the task of semantic editing of an image rather than controllable generation. The difference is that editing involves manual interaction, as attributes are not explicitly controlled, and therefore it is not a great choice for a system that generates large sets of synthetic data.

4 PROPOSED SOLUTION

In this section, we describe the main functionality of our face generation framework. After presenting an overview of the developed architecture, we separately introduce each component and mention its contribution towards our goal of producing synthetic data of faces alongside their semantic segmentation labels. Within each subsection corresponding to a module, apart from highlighting the architecture and functionality, we briefly touch on the processes involved in obtaining an effective component. In the end, we discuss the flexibility of the architecture, by mentioning how it can be adapted to produce other labels, and by indicating how the framework allows controlling only one of the two attributes, or none at all.

4.1 Architecture

The proposed framework consists of several neural-based solutions that each tackle a specific part of the generation process.

To reiterate the context, our goal is to utilize a generator network capable of producing human faces, and extend it with a way of obtaining the semantic segmentation labels associated with the generated images. At the same time, a manner of controlling image attributes should be provided.

To achieve these goals, we propose the framework illustrated in Figure 3, where all of the main components are illustrated from a high-level perspective. The generation process starts with a control attribute encoder, which maps explicit values of age and gender into a style vector w from the latent space of the generator. Thus, the requested age end gender dictate the attributes of the person obtained in the end. Style vectors w obtained this way serve as input to the face generator network by injecting them at every style block, through style modulation. The face generator network is the main component of the system, as it is capable of producing high-quality images of faces. Finally, the feature maps involved in the generation are processed by the separate segmentation network, which assigns semantic segmentation labels of human face parts to the generated image.

With the outline of our framework in mind, we detail each of the components in an order consistent with their place in the final processing pipeline.

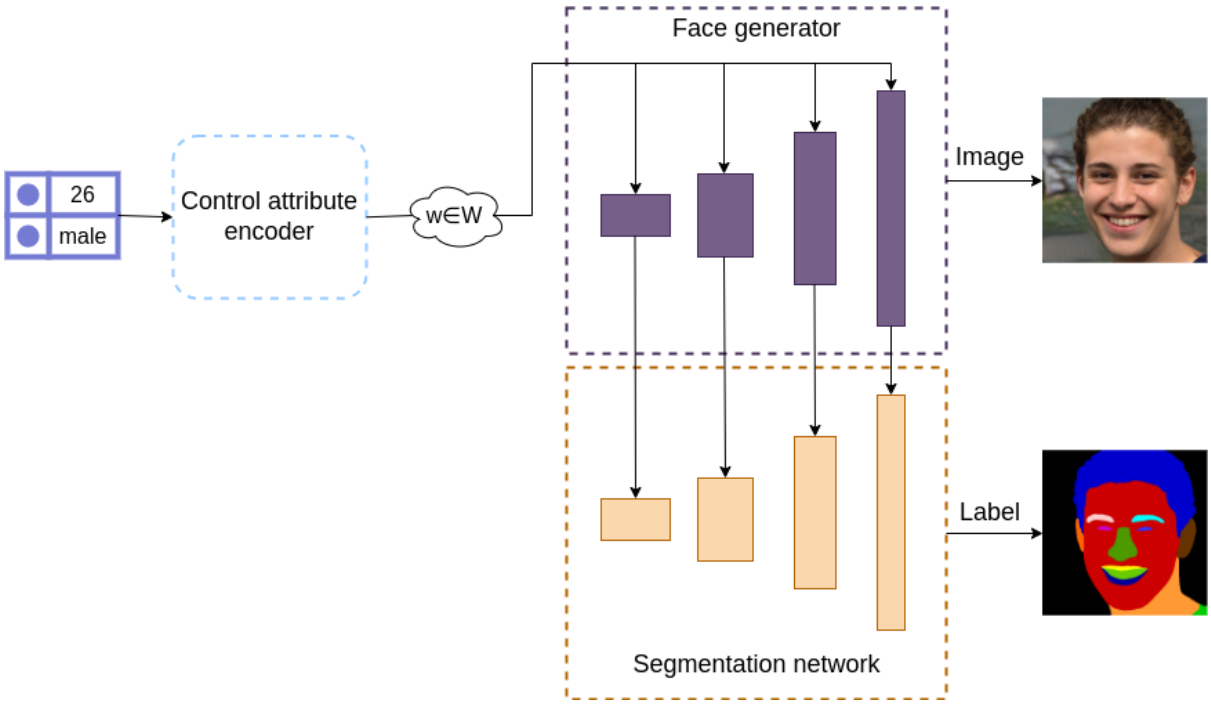


Figure 3: Overview of the proposed architecture

4.1.1 Control attribute encoder

To begin with, an attribute encoder is utilized in order to transform the explicit values of control attributes into a vector in the latent space of the generator. With proper training of this component, the predicted latent code leads to the generation of a person that satisfies the conditional constraints.

Figure 4 illustrates the detailed architecture of the control attribute encoder. The main part used for achieving control over age and gender is a feed-forward network with four layers, referred to as the attribute mapping network. Both the number of layers and the size of each layer are consistent with the mapping networks recommended by Shoshan et al. [55] in their GAN-Control framework. However, instead of training a separate attribute mapping network for each of the attributes, we train a single one that directly maps pairs of age and gender values into the style vector. The reason behind this change is that generator is not trained according to the procedure described by the authors and therefore does not have a clear separation between attributes in the 512-dimensional style vector. Because the attribute mapping network by itself produces deterministic results given a certain age and gender, we introduce randomness and variation to the generated images by combining the mapped style vector w_{attr} with another style vector randomly w_{random} generated by the original mapping network of StyleGAN2, producing style vectors $w_{combined}$. We inherit age and gender from the first one and the other image attributes from the random latent vector. We will call this procedure style mixing in the later chapters, where we detail how exactly the two vectors are combined in order to maintain both control precision and diversity.

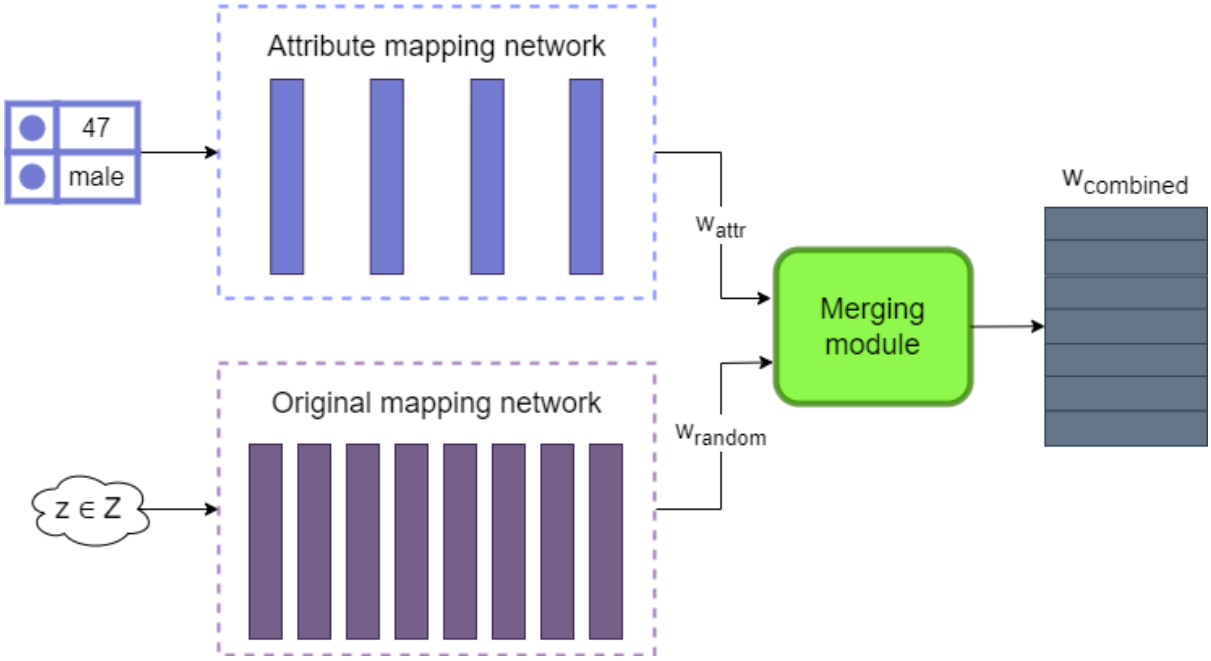


Figure 4: Detailed view of the control attribute encoder

4.1.2 Face generation network

Next, style vectors $w_{combined}$ are inserted in the StyleGAN2 generator at corresponding layers. As stated in the previous chapter, we settle on a generator that produces samples at a resolution of 512x512, with an overview of the architecture depicted in Figure 5.

Originally, a random latent variable z is mapped into a random style vector w through eight fully-connected layers. However, this component of the StyleGAN2 architecture is moved to the control attribute encoder, where it helps create the random style vector that dictates the other attributes of the image. Therefore, the image generation process starts with the style vectors $w_{combined}$ obtained from the control attribute encoder, which are injected into each style block, finally producing a 512x512 face image.

An in-depth look at the design of three consecutive style blocks is present in Figure 6, where one can notice how the style vector A is injected through the use of the modulation and demodulation of convolutional weights, a process described earlier in section 3.3.1. After the convolutional layer, a learnable bias is added along with random noise B, which helps increase the variation of the generated images.

4.1.3 Segmentation network

The final component that we describe is the segmentation network. It consists of a separate branch added as an extension to the architecture of the generator, as it relies on all of its intermediate feature maps to produce the semantic segmentation map associated with the generated image. We have implemented and tested multiple configurations, which will be

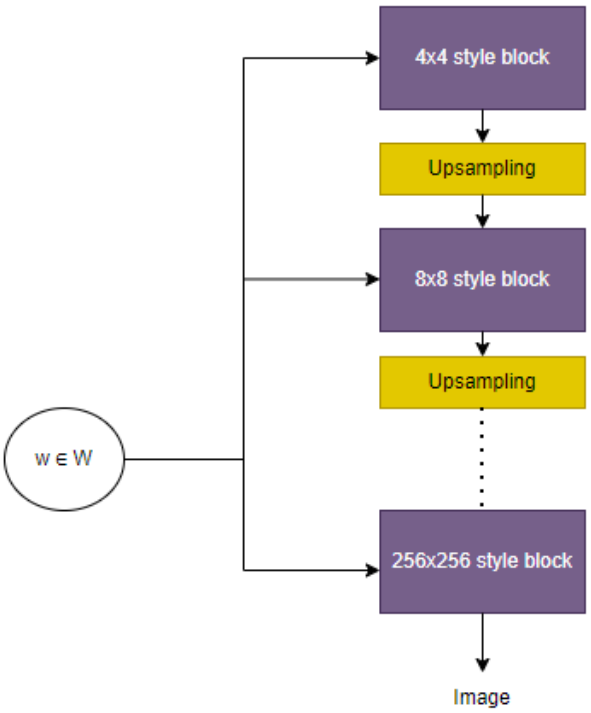


Figure 5: StyleGAN2 generator architecture

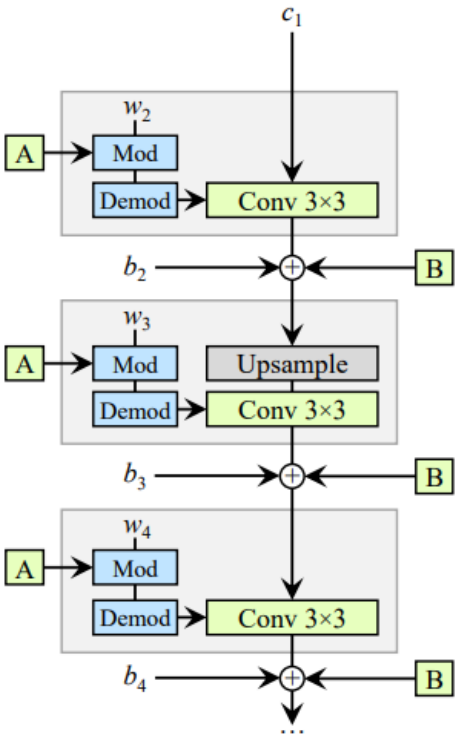


Figure 6: Style blocks [29]

further explained in the next chapter, alongside their detailed implementation and in-depth architecture.

As previously mentioned, the branch exploits the eight intermediate feature maps of the generator. Regardless of the exact implementation, in order to handle the different spatial shapes of the feature maps, the architecture has to provide a way of upsampling feature maps and a way of aggregating them. We experimented with different ways of upsampling (using bilinear upsampling and a normal convolutional layer, or a transposed convolution instead) and different ways of aggregating the feature maps (concatenation and addition). In the end, the resulting features have a spatial resolution of 512x512, the size of the output of the generator, and pass through a final few layers, namely a batch normalization layer and a final convolutional layer, obtaining the corresponding face segmentation of the generated image.

Although training is done only on a small set of previously synthesized images that are manually annotated, the pre-trained generator transfers the knowledge already present in its main generation branch to the new task of semantic segmentation. The effectiveness of this approach and the ability of the segmentation branch to generalize to other input latent vectors is assured by the continuous nature of the generator's latent space.

4.2 Framework flexibility

In this section, we discussed the architecture of our system and how it fits the problem of generating faces with semantic segmentation labels with control over the age and gender attributes. However, the framework can easily be adapted for solving other similar problems.

Firstly, we chose to integrate semantic segmentation labels into our solution as they represent an example of pixel-wise annotation and one of the most laborious annotations to obtain manually, therefore potentially saving lots of annotation effort and costs. However, the architecture of the label-assigning network can easily be modified to support other labeling tasks by simply replacing the final convolutional layer with other layers suitable for the task.

Secondly, in terms of control attributes, one can easily follow the procedure described and apply it to control other variables, such as pose, expression, or illumination.

Additionally, our framework still allows generating images unconditionally, by setting the combined latent equal to the random style vector, discarding the style vector produced by the attribute mapping network. One can also generate an image based only on a subset of the supported control attributes by assigning the other ones randomly as input to the attribute encoder.

5 IMPLEMENTATION DETAILS

The purpose of this chapter is to offer details with regard to the development of the solution. The order of the following sections relates to the steps taken during the development phase of the framework. Firstly, we mention the technologies adopted and the starting premises. Secondly, for each of the added components, namely the segmentation network and the control attribute encoder, we again stress their importance while putting emphasis on their implementation and describing the training procedure used in the case of neural modules.

5.1 Technologies and premises

The programming language of choice is Python, due to its simplicity and the availability of deep learning frameworks and libraries.

To take advantage of its preexisting implementations of neural networks and its powerful auto-differentiation engine, we develop our work using the PyTorch framework [1], as is customary for many computer vision applications. Because we experiment with different network configurations, we use Tensorboard [2] to track ongoing experiments and compare finished ones in order to guide the next developing steps.

Because our work relies on image processing and manipulation, we use the Open Computer Vision Library (OpenCV) and its implementation in Python through the cv2 package [3]. For manipulating multidimensional arrays we use Numpy [4].

We build our framework on top of the StyleGAN2 [29] generator model. More specifically, we refer to a released checkpoint [5] of the generator, previously trained on the FFHQ dataset. The working resolution of the generator is 512x512, which translates to producing images with good quality while keeping a medium resource consumption. The reasons behind this choice are presented in more detail in a previous chapter on related work. In essence, StyleGAN2 provides better image quality when compared to its predecessors, while at the same time maintaining a disentangled latent space, which justifies the performance of semantic editing tasks.

[1] `https://pytorch.org/`. Last accessed: 13 June 2023

[2] `https://www.tensorflow.org/tensorboard`. Last accessed: 13 June 2023

[3] `https://github.com/opencv/opencv-python`. Last accessed: 13 June 2023

[4] `https://numpy.org/`. Last accessed: 13 June 2023

[5] `https://catalog.ngc.nvidia.com/orgs/nvidia/teams/research/models/stylegan2/files`. Last accessed: 13 June 2023.

5.2 Face parts semantic segmentation

The current section displays the implementation details for the semantic segmentation network. As mentioned in earlier chapters, the task of face parts segmentation or face parsing refers to the precise labeling of each pixel to form a single-channel image called the segmentation map, with values corresponding to the class that particular pixel belongs to.

Generally, the solution for the face segmentation problem follows the framework of DatasetGAN. The steps include generating a small set of images, annotating them manually, and training an additional network connected to the generator to reproduce the manual annotations. In particular, however, in an attempt to decrease memory usage, which prompted the procedure described in the paper of training only on a subset of pixels, we replace the ensemble of multilayer perceptrons with a single convolutional network.

5.2.1 Labeling convention

It is essential to first define a particular convention in the form of possible classes and annotation style. Without loss of generality, we define similar labels to the ones proposed by Lee et al. [31], where they introduced the CelebAMask-HQ dataset for the task of face parsing. With this in mind, the proposed segmentation network predicts a total of 18 classes corresponding to the following components of the resulting face image: background, cloth, earring, eyeglass, hair, hat, left eye, left ear, left eyebrow, lower lip, mouth, neck, nose, right ear, right eye, right eyebrow, skin, upper lip. Note that the only category missing when comparing to the labeling of CelebAMask-HQ is "necklace", due to its rarity among the generated faces.

In addition, the face segmentation results displayed in the next sections follow a color scheme suitable for distinguishing between all classes.

5.2.2 Manual annotation

Having defined the required classes, a set of 50 images is obtained with the help of the generator. In addition, keeping track of the input latent vectors that produced each image is key in this approach, as they are further employed during training. To make sure the resulting images are reproducible, the random noise component of each style block is set to zero (noise is not injected) in the generator forward pass. Therefore, we save the input latents as binary .npy files alongside the image they produced.

The next important step is the manual annotation of the set of 50 generated images. For completing this task, we employ the use of a platform called Dataloop [6] which is specifically designed for large annotation tasks. It supports popular label types such as image classes,

[6] `https://console.dataloop.ai/welcome`. Last accessed: 13 June 2023

bounding boxes, segmentation, or keypoint marking. As an alternative, one can use COCO Annotator [7]. However, we found that using the annotation style based on connected points, as is the case for the COCO Annotator, results in a coarse and imprecise segmentation. Therefore, we choose the fine-grained style obtained with the help of the Dataloop annotating platform.

Multiple examples of the manual annotation for synthetic images are displayed in Figure 7.

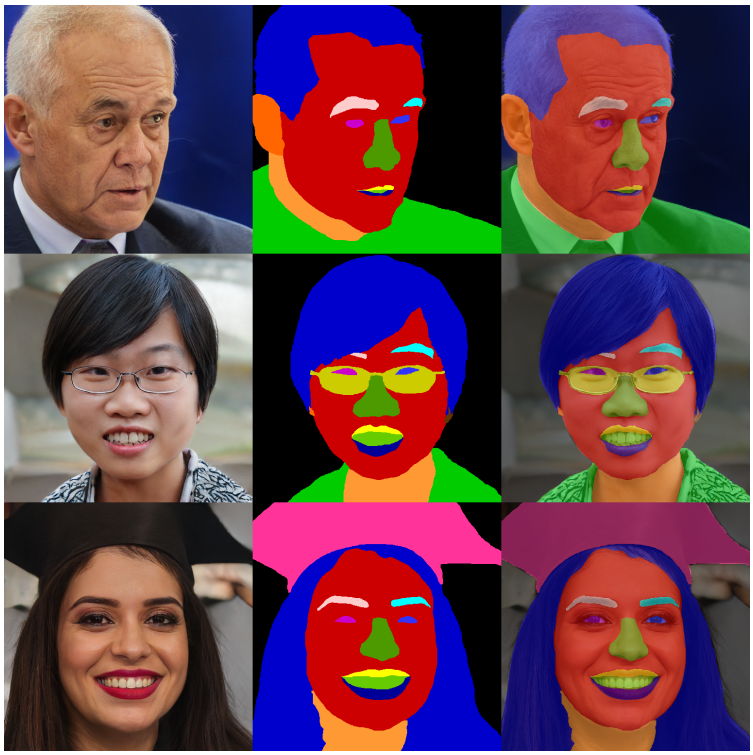


Figure 7: Generated images and their manual annotation. The left column shows the image originally generated by StyleGAN2. The middle column displays the annotated segmentation mask for the corresponding image. The last column is the result of blending the original synthetic image and its segmentation, for visualization purposes.

5.2.3 Segmentation network architecture

We experiment with multiple configurations of the segmentation network and in the end choose the approach that maximizes the mean F1 score on the validation set. Details and results of the testing procedure are shown in the following chapter, which is dedicated to evaluating the solution. For future reference, the list of architecture configurations employed are:

1. **config-1** - an architecture similar to the generator, sequentially upscaling feature maps from the lowest spatial resolution (4x4) to the highest (512x512)

[7] `https://github.com/jsbroks/coco-annotator`. Last accessed: 13 June 2023

2. **config-2** - an architecture where feature maps are processed independently, merged, and finally interpreted together
   (a) **config-2a** - feature maps are processed independently by one convolutional layer
   (b) **config-2b** - feature maps are processed independently by two convolutional layer
   (c) **config-2c** - feature maps are processed independently by three convolutional layer
   (d) **config-2b-balanced** - same as config-2b, except a class-balancing loss is employed

All of the architectures are based on the general architecture illustrated in the previous chapter, Figure 3, with the main purpose of the network being to transform the feature maps from all layers of the generator into a final segmentation map. Instead of the resource-intensive solution of upsampling all feature maps to the final 512x512 resolution, concatenating them, and using an ensemble of multilayer perceptrons (MLP) to predict per-pixel classification, as stipulated in the original DatasetGAN approach, we rely on the insights provided by the following work, BigDatasetGAN, where the feature interpreter takes the form of a CNN. Therefore, our segmentation network is also a convolutional neural network, with a different architecture.

In between convolutional layers, use the ReLU activation function for nonlinearity. Moreover, we found that using a batch normalization layer right before the final convolutional layer marginally improved the results, but more importantly had an effect on reducing the training time taken until convergence.

We will further detail the two main configurations (**config-1** and **config-2**) of the segmentation branch and show the iterative process through which the best results were achieved.

The first approach (**config-1**) displayed in Figure 8 is based on replicating the architecture of the generator. With this in mind, the segmentation branch is created to match the shapes of the generator feature maps at each respective layer. To leverage the knowledge of the generator network, the feature maps are introduced at each layer through addition. The alternative of using concatenation requires a further reduction in the channel dimension in order to fit the memory constraints, which negatively affects the performance. For upscaling purposes, to further match the architecture of the generator, we use transposed convolutions.

The second configuration (**config-2**) is presented in Figure 9 and consists of passing each of the eight feature maps of the generator through a set of convolutional layers. The resulting feature maps are then upsampled to the output resolution of 512x512 using bilinear interpolation and concatenated along the channel dimension. The resulting tensor is processed by a batch normalization layer and a final convolutional layer that maps the channels into the number of classes, 18. Aside from being helpful in solving the task of semantic segmentation, the convolutional layers also reduce the number of channels before the upsampling operation. The channel dimensionality reduction is required before the upsampling operation for memory efficiency. The performance is not severely affected in this case by the channel reduction when compared to the first configuration, as a feature map with low spatial dimensions still provides useful information for the final computation. In the previous configuration, a feature

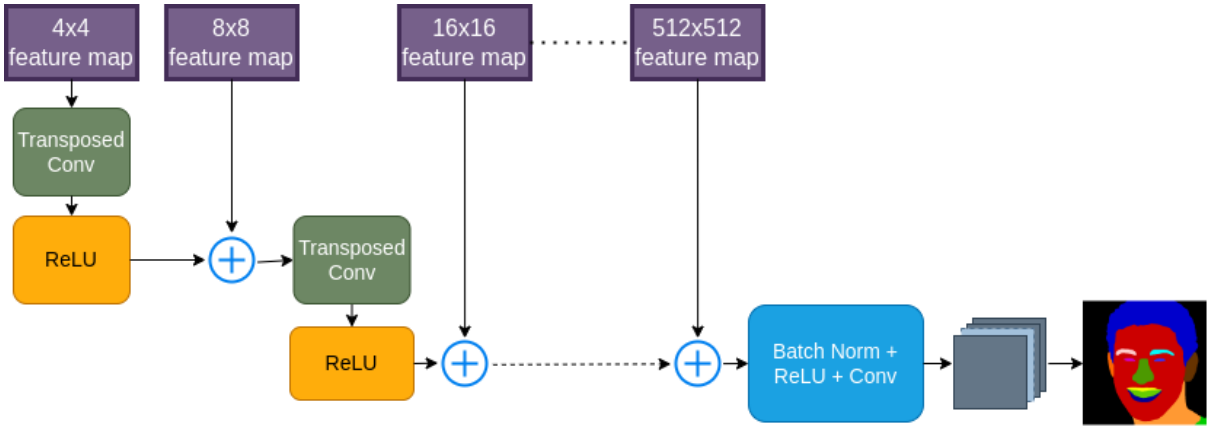


Figure 8: Segmentation network configuration 1

map with low spatial size passes through more convolutional layers than one with high spatial dimensions, and therefore its contribution to the result diminishes if each of the convolutions involves a channel reduction.

The variations of config-2 are named based on the number of convolutional layers that process each of the feature maps: **config-2a** with one layer per feature map, **config-2b** with two, and **config-2c** with three. **Config-2b-balanced** further builds on config-2b not by modifying the architecture, but by employing a class-balanced loss in training.

5.2.4 Training

For training and evaluating purposes, the dataset of 50 data points is divided as follows: 30 for training, 10 images for validation, and 10 images for testing the performance of the segmentation. We will continue by detailing the training process, saving the discussion for the performance on the testing set for the next chapter.

Despite the small size of the training dataset, the segmentation branch is still capable of reaching generalization in the task of semantic segmentation, as the training procedure detailed further is an extension of a learning technique called transfer learning. Transfer learning generally represents a way of exploiting a model previously trained to solve a particular task, in order to adapt and reuse its knowledge to another similar problem, which may consist of training on a smaller dataset. While this typically means reusing the same network and possibly changing the last few layers, in our case, this concept is applied when training an entirely new network or branch. We leverage the concepts embedded in the feature maps of the generator to train the segmentation branch with very few training samples. This approach proves to be both compelling in terms of performance and training time.

We pre-compute the feature maps for each image in the dataset by running the forward pass of the generator with the previously saved latent vectors. Saving the features and loading them at training time implies that we no longer need to load the generator model when training,

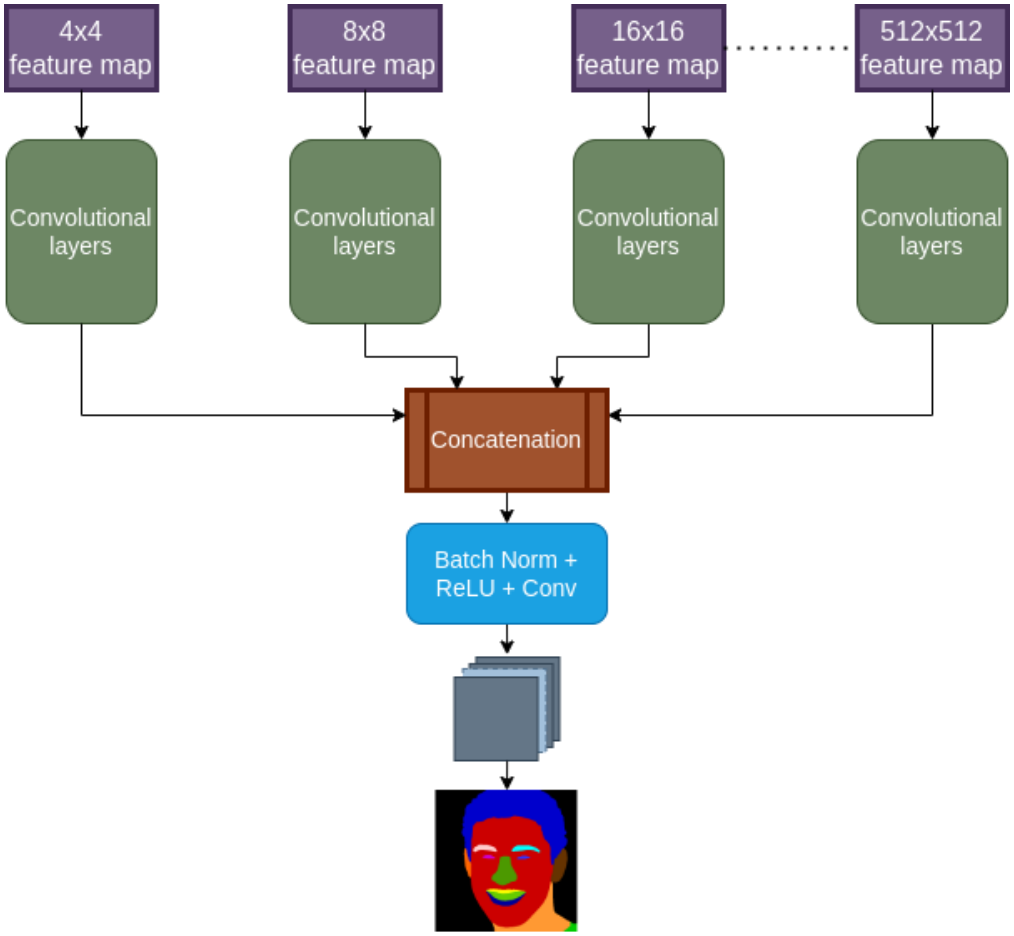


Figure 9: Segmentation network configuration 2

thereby saving both computational effort and GPU memory. The alternative represents loading only the latent vectors at training time and extracting the feature maps on-the-fly, before using them as input to the segmentation branch. Note that in this second approach, for a successful training, the weights of the generator have to be frozen.

An example of the configuration file used for training the segmentation branch is displayed in Listing 1, Appendix A. We used a batch size of 2 and trained for 100 epochs using the Adam optimizer with a learning rate of 0.001 and a weight decay of 0.0001.

The training minimizes the cross-entropy loss between the logits of the segmentation branch output of shape 18x512x512 and the ground truth segmentation map of shape 1x512x512. In the output of the network, each pixel has a probability of belonging to each of the 18 classes, hence the 18 channels. The ground truth map has a single value per pixel - the class it actually belongs to.

The PyTorch implementation of cross-entropy loss first applies a softmax operation over the output logits and then computes the negative log-likelihood, both operations being summarized in the following expression:

$$L_{ce} = -\frac{1}{B} * \sum_{(y,t)} log(\frac{e^{y_t}}{\sum_i e^{y_i}}), \tag{6}$$

with B being the batch size, y being the network output for a certain pixel, and t being that pixel's class.

At inference time, for validation and testing purposes, obtaining the predicted segmentation map $pred$ reduces to finding the class with the highest score (corresponding to the highest probability), for each pixel in the network output $logits$:

$$pred_{i,j} = \arg\max_k (logits_{k,i,j}). \tag{7}$$

For **config-2b-balanced**, we also try a technique described by Cui et al. [10] for handling class-imbalanced datasets. The cross-entropy loss for each class is weighted differently, based on the frequency of a certain class among the training set and a hyperparameter β that controls the degree to which underrepresented classes are weighted more than common ones. The final loss, therefore, has the equation:

$$L = \sum_{c \in C} \frac{1-\beta}{1-\beta^{n_c}} * L_c, \tag{8}$$

where C is the set of classes, n_c is the number of pixels that correspond to class c in the training set, and L_c is the cross-entropy loss computed only on pixels of class c in the ground-truth segmentation map. We use a value of 0.9999 for β.

5.3 Control attribute encoder

In this section, we detail our approach to gaining control over the generation process.

Inspired by the GAN-Control framework, we train a small feed-forward network in order to encode attributes into the latent space of the generator. The mapped style vector projects into a face image with the features required. In our case, the control variables employed are age and gender.

As opposed to GAN-Control, we do not require a specially-designed generator, one that has its latent space split into segments that independently control each attribute. Instead, we start from the original StyleGAN2 generator and show that its latent space is sufficiently disentangled for the purposes of gaining control over a few attributes. This decision implies directly encoding all of the controlled attributes into a style vector with a single feed-forward network, instead of the approach of using one per attribute.

5.3.1 Dataset used

To obtain training data for the mapping network, we employ the use of the same age predictor as the one used in GAN-Control, namely DEX [48], a VGG-16 network [56]. For the gender attribute, since it was not one of the attributes controlled in the GAN-Control framework, we use another VGG-16 network trained for this task, which was released together with the age predictor in the same paper. The architecture of the VGG-16 network is detailed in Table 11 in the Appendix, where one can see a sequencing of 13 convolutional layers followed by a final three fully-connected layers. The final output has a size equal to the number of classes being predicted. In the case of the age predictor, that number is 101, and in the case of the gender predictor, it is 2.

With this in mind, we run the generator in inference mode, constructing a dataset of 100.000 samples with pairs of style vectors w and their respective generated images. We run the VGG-16 networks on every image, obtaining a label for the age and gender of the synthetic face. Naturally, we do not need to save the images thereby reducing the time taken to construct the dataset. Given the final output of the VGG-16 networks, gender directly corresponds to the highest among the two output values. Age is obtained through what is called the softmax expected value, which reframes the usual problem of regressing the age into a classification problem, yielding better results. Given the 101 output values of the VGG-16 age predictor network, the equation of the softmax expected value is:

$$E = \sum_{i=0}^{100} k * \frac{e^{y_k}}{\sum_i e^{y_i}}, \tag{9}$$

where y is the 101-dimensional output of the network which is first normalized through a softmax operation before computing the expected age value.

This dataset is constructed because the problem of controlling the attributes of the generated face reduces to reversing the previous process: we train a simple fully-connected network to map the age and gender back into the latent space of the generator, based on the saved associated style vector.

5.3.2 Mapping network architecture

We use a multilayer perceptron consisting of four layers in order to obtain a mapping from the given attributes to a style vector that generates an appropriate face. The architecture for this encoder is consistent with the one suggested in GAN-Control in terms of the hidden layers. What differs is that the input layer consists of two units, as we encode both the age and gender attributes at the same time. An overview of the layers that constitute the attributes mapper is listed below in Table 5. In addition, note the output size of 512 which matches the size of a style vector.

Layer type	Input shape	Output shape
Linear	2	512
Linear	512	512
Linear	512	512
Linear	512	512

Table 5: Attribute mapping network architecture

As suggested in the paper, we use the Leaky ReLU activation function with a negative slope of 0.2 as the choice of nonlinearity between each linear layer. The Leaky ReLU activation function has the following formula:

$$LeakyReLU(x) = max(0, x) + a \cdot min(0, x), \tag{10}$$

where a is the negative slope.

5.3.3 Training procedure

Given the dataset of 100.000 pairs of style vectors with the associated age and gender, we train the mapping network to solve the inverse problem: given the age and gender, find an encoding into a suitable style vector.

Firstly, the dataset is split into a training and a validation part. The validation subset has a total of 1000 examples, enough to measure how well the model is performing at a certain point in training.

We use the configuration file found in Listing 2, Appendix A. As the training process is slow and an epoch is quite large, we train based on iterations instead of epochs, using the same optimizer hyperparameters as the ones employed for training the segmentation network. We evaluate the network on the validation set every 2000 iterations and monitor the performance, stopping the training when convergence has been reached.

Initially, we followed the approach described in GAN-Control, where each attribute encoder is trained using a combination of a regression loss L_{reg} and an attribute loss L_{attr}. The regression loss is either an L1 or an L2 distance between the target style vector and the predicted one. We choose the mean squared error (MSE) loss for our work. As for the attribute loss, the actual function used depends on the attribute at hand. The loss is based on the notion of reconstruction, as it is computed between the input attributes and the projected ones. The projected attributes are obtained by first mapping the input attributes to a predicted style vector (with the help of the mapping network), then an image is produced by the trained generator, and finally, the attribute predictors are used on the image.

To this end, the loss that we minimize is composed of three terms and is described in the

following equation:

$$L = 10 * L_{reg} + 0.01 * L_{age} + L_{gender}, \tag{11}$$

where L_{reg} is the MSE between the target style vector and the one predicted by the network, L_{age} is the MSE loss between the input age and the reconstructed age, and L_{gender} is the cross-entropy loss employed on the input gender and the reconstructed gender. The weighting factors are manually chosen so that the loss values lie in a similar range at the start of the training.

Note that this training requires loading the mapping network, the generator network, and the two VGG-16 networks into memory at the same time. Because of that, we simulate a larger batch size of 64 using gradient accumulation, despite the real batch size being only 2. Gradient accumulation refers to the idea of computing gradients based on small batches and aggregating them for multiple steps, before finally updating the weights. This technique is done to increase training stability.

In a further effort to simplify the training process and understand the effect and significance of the loss components, we conduct a study by training using only the regression loss. We show that this method approaches a similar control precision and leads to exponentially faster training, as the loss only requires the output of the mapping network. In addition, by not steering the optimization in the direction of the predicted age and gender on the projected image, we obtain a mapping network that is less biased to the attribute predictors.

5.3.4 Style mixing

By only using the attribute mapping network and passing the predicted values to the generator, we obtain high-quality images, depending on the requested age and gender, but at the cost of diversity. This unwanted consequence is bypassed by the GAN-Control framework through the fact that in training their specialized version of the generator, a part of the latent code is left aside for encoding all other image attributes besides the controlled ones.

While this approach proves to be useful in their case, we want to have a method of turning a pre-trained generator model into a powerful synthetic data generator.

How we achieve this is by leveraging the high disentanglement of StyleGAN's latent space. We combine two style vectors: one predicted by the attribute mapper and one randomly generated by the original StyleGAN2 latent mapper. The resulting style vector is then injected at each layer of the generator. This method relies on a result published by the authors of StyleGAN, whereby they show how injecting a style vector at a certain layer controls a specific semantic part of the generated image.

Therefore, we find the layers that have the biggest influence on the age and gender of the generated person. To accomplish this, we first establish a baseline for the mean age error

and gender accuracy of the model when using random style vectors as input to the generator. Naturally, as this does not include any sort of control component, it results in a 15.77 mean age error and 49.5% gender accuracy, as listed in Table 6. After the baseline has been established, we analyze how much the control precision increases when injecting the attribute-controlled style vector at each possible position, while preserving the random style vector as input to every other layer. The results of this analysis are presented in Table 6. There are eight total layers, of which layers 2, 3 and 4 have the highest contribution toward the goal of controlling age and gender, as they improve the precision of the control the most.

Layer	Age error (years)	Gender accuracy (%)
baseline	15.66	51.20
1	16.12	52.40
2	13.80	55.30
3	13.32	65.80
4	10.80	73.50
5	14.59	51.30
6	15.50	51.10
7	15.63	51.40
8	15.64	51.40

Table 6: Effect of injecting the random style vector at each layer

Given these results, we conclude that in order to obtain as much diversity as possible, under the constraint of maintaining a high control precision, we have to inject the predicted style latent at layers 2, 3, and 4, as the other layers have little to no effect on the age and gender of the generated image. The random style vector is therefore used as input to the other layers.

5.4 System overview

In this chapter, we detailed the implementation of each component. After establishing how the modules are constructed alongside their capabilities, we briefly present how the overall system was assembled.

To begin with, the proposed framework is intended as a solution for synthesizing large datasets with reliable annotations. The final goal is to have synthetic training data as a supplement to real data, by improving the distribution of the real data in underrepresented cases, through control of the generation, or by completely replacing real data in cases where it is hard to obtain or annotate.

Therefore, we design our system in order to have flexibility. One can synthesize data based on all of the control attributes, as mentioned previously. In addition, there exists the possibility of generating based on a subset of control attributes, by randomly assigning the others at inference time. Lastly, the control component can be fully removed by only using the random style vector as input to the generator.

6 EVALUATION AND TESTING

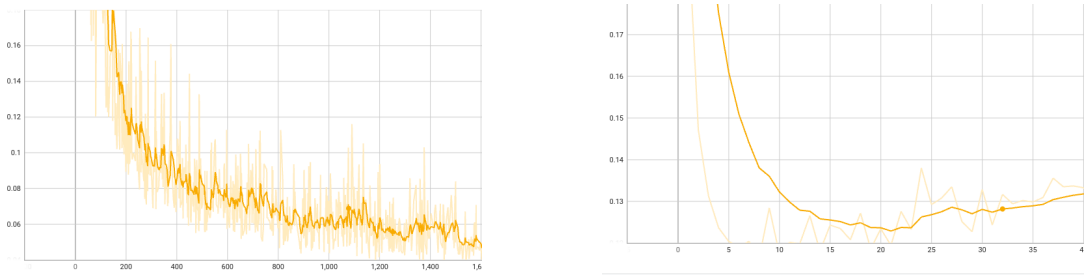
The aim of this chapter is to present the results obtained through the previously detailed methods. We evaluate our framework both quantitatively, through metrics, and qualitatively, by visually inspecting the images and semantic segmentation maps produced by the system.

6.1 Semantic segmentation evaluation

Firstly, to evaluate the training process, we analyze the loss curves during the training of the label-assigning network (segmentation network). The training itself converges fast, with an example of the evolution of the training loss and validation loss being shown in Figure 10. The training slightly suffers from overfitting, as seen from the increase of the validation curve in later epochs, suggesting that a larger number of training samples would help. However, for the purposes of this paper, we find that the number of annotated examples is sufficient for drawing conclusions and for serving as a baseline for the employed method. In practice, if integrating such a system for improving the performance of an existing network through synthetic data, one would need to think about balancing the effort in the number of annotated images against the performance required.

In order to choose the best configuration of the segmentation network, we compare their results on the set of 10 validation samples. We use two metrics that are relevant to the task of semantic segmentation: F1 score and IoU (intersection over union) [68, 69].

The F1 score is obtained by combining precision and recall into a single metric. Specifically, it is computed as the harmonic mean of the two. In this case, the F1 score is computed on the entire image, by comparing the predicted segmentation map with the ground-truth annotated one. Given the notations of TP, FP, FN, and TN for the number of true positive,



(a) Training loss

(b) Validation loss

Figure 10: Typical segmentation network loss curves with respect to the number of training steps

false positive, false negative, and true negative pixels, precision, recall, and F1 are computed using the equations:

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{14}$$

IoU is defined similarly, using the following equation:

$$IoU = \frac{TP}{TP + FP + FN} \tag{15}$$

With this in mind, we present the results of the previously listed configurations on the validation set in Table 7. Note that we exclude the background class from the computations, as is customary for this task [68, 69]. All configurations score similarly, suggesting that the method of transferring the generator's knowledge to the problem of predicting segmentation labels is well formulated. The worst-performing configuration is the first one, indicating that an architecture similar to the generator does not yield the best results. This is because the problem does not require learning to "generate" the label; instead, a better solution is to directly adapt the problem of face generation to face segmentation.

Configuration	Mean F1 score (%)	Mean IoU (%)
config-1	90.59	78.79
config-2a	91.04	79.42
config-2b	91.51	80.13
config-2c	91.09	79.63
config-2b-balanced	91.96	80.73

Table 7: Validation results for the segmentation network

After choosing the best performing model (config-b2-balanced), we compute the F1 metric per class and the mean F1 on the test set, in order to gain an insight into what classes have the best and the worst results. These insights are summarized in Table 8. As expected, classes that generally relate to a lot of image pixels (background, skin) have a better F1 than classes with few image pixels (earring, eyebrow, lip). However, the performance on the test set is in line with the one in the validation set.

We also judge the performance of the predicted semantic segmentation maps in comparison to other face parts segmentation networks, previously trained on CelebAMask-HQ, and therefore producing a set of labels similar to ours. We conduct the tests on our test set and also list

the results in Table 8. Note that the performance of our segmentation is similar or better in most cases, suggesting that training data produced by our framework may indeed have a positive effect when training for the downstream task of face parts semantic segmentation.

	background **eyeglass** **cloth**	**mouth** **lower lip** **upper lip**	**l_ear** **r_ear** **earring**	**hat** **hair** **nose**	**l_eyebrow** **r_eyebrow** **skin**	**l_eye** **r_eye** **neck**	**mean**
	98.41	93.48	64.99	95.84	54.37	42.20	
BiSegNet [64]	93.22	91.38	66.83	95.88	48.69	40.54	73.84
	89.77	88.90	25.20	92.33	97.01	94.78	
	97.96	94.45	91.77	95.98	84.50	90.38	
DML-CSR [68]	94.39	91.06	94.86	94.14	85.06	89.58	91.10
	89.38	88.72	79.14	93.45	96.92	95.06	
	97.73	94.73	93.08	90.70	86.13	92.85	
Proposed system	97.02	90.26	95.33	94.91	86.30	91.52	92.01
	80.22	91.26	78.93	95.23	96.61	94.80	

Table 8: Comparison of mean and per-class F1 score (%) on the test set

Note that there may exist annotation differences between our manual annotation and the annotation of CelebAMask-HQ (for instance, hair segmentation granularity, or boundaries of the nose), therefore rendering the comparison not exact, but still representative. To comment more on the results, we observe that BiSegNet produces a high-quality segmentation, but its failure cases include mismatching the left and right facial parts, which considerably reduces the F1 score on those classes. DML-CSR produces a result very similar to ours, with slightly better results on the hat and cloth classes, suggesting that we may benefit from having more training data with those particular classes.

We display the predicted segmentation maps and visually compare them to the corresponding manual annotation. A subset of the results can be seen in Figure 11. The model generally performs well, with the most typical errors consisting of a more coarse segmentation of the hair or less precision around smaller parts of the face (eyebrows, eyes).

Results of generating completely new images alongside the semantic segmentation predicted by the trained segmentation network are displayed in Figure 12. For more samples, refer to Figure 17 in Appendix C.

6.2 Evaluation of attribute-guided generation

Figure 13 shows the evolution of the validation loss and metrics when considering all three losses. As one can observe, the gender accuracy and age mean error improve over time and reach almost perfect values. However, the reconstruction loss improves until a certain point, where it remains for the rest of the training. This can be explained as follows: as we are using an encoder, the natural consequence is that for a certain age and gender, in order to minimize the reconstruction error, a mean latent is obtained.

Figure 11: Segmentation test results visualized. For each set, the left column represents the originally generated image. In the middle column, the predicted semantic segmentation can be observed and compared to the last column, which contains the ground truth manual annotations.

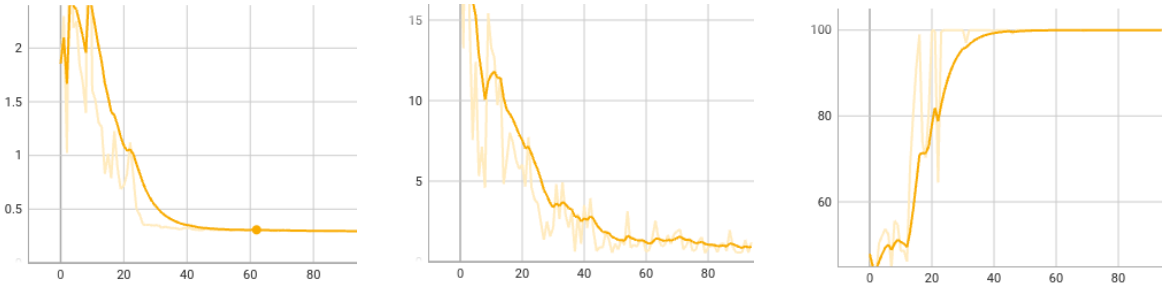
Figure 12: New generated faces with the associated semantic segmentation annotation after training

The exact values for the age error and gender accuracy on the validation set are listed in Table 9 for the model that uses all three losses. Considering the very high accuracy obtained with this method (0.85 mean age error in years, and 100% gender accuracy), it is safe to say that the model learns a mapping biased to the attribute-predicting models. That is, given a certain user input of age and gender, the mapping network produces a style vector whose associated image is perfectly aligned with the VGG-16 networks used.

Total loss	**Age L1 error**	**Gender accuracy (%)**
$L_{reg} + L_{age} + L_{gender}$	0.85	100%
L_{rec}	1.78	100%

Table 9: Comparison of validation results based on loss employed

In addition, we train the network using only the MSE loss on latent vectors, (L_{reg}). With the results of 1.78 mean age error and 100% gender accuracy, this training appears to perform slightly worse than the method that uses all losses. However, the result is strictly based



(a) Latent MSE (b) Age mean error (c) Gender accuracy

Figure 13: Attribute mapping network validation results

on learning to map a requested age and gender into a mean style vector for the requested attributes.

The control precision is as impressive as mentioned previously only because the attribute mapping network converts the queried attributes into a generic face with those attributes, as seen from Figure 14, one can observe samples produced by the generator network given a random age and gender. The collapse in variety is further attested to by the results illustrated in Figure 15, where one can observe that fixing the gender attribute and modifying the age attribute results in an interpolation across the "age dimension", where other image attributes barely change.

Figure 14: Faces synthesized based on attribute mapping network, without style mixing

Figure 15: Results of fixing the gender attribute and increasing the age attribute

With the added diversity from the style mixing operation previously discussed in section 5.3.4, Table 10 summarizes the final accuracy of the control, while at the same time comparing the solution to the other ones listed in the previous chapter on related work. We estimate the mean age error in years and the gender accuracy based on a set of control attributes extracted from 10000 images of FFHQ, as indicated in GAN-Control.

When compared to the InterfaceGAN framework, we score a lower gender accuracy of 95.53% vs. 98.7%. However, the results are not directly comparable, as they compute the metric using a set of images with high confidence in the predicted gender attribute. In contrast, our result is bounded by the performance of the gender-predicting model. In comparison with GAN-Control, we do not reach the same level of control in age, with a mean error of 4.47 years compared to 2.02, but we maintain the high-quality results of the original StyleGAN2 generator, as we do not retrain it specifically for enabling high-precision control.

Method	Age L1 error	Gender accuracy (%)
InterfaceGAN [54]	N/A	98.7
GAN-Control [55]	2.02	N/A
Proposed system	4.47	95.53

Table 10: Comparison of control accuracy and FID

Visual results that intuitively explain the functionality of style mixing are displayed in Figure 16. The final image is a composition of the two others, by extracting the age and gender from the first one and all of the other image attributes from the other.

Figure 16: Faces generated with style mixing. Left column shows the image generated with the style vector obtained from the attribute mapping network. The middle column represents a randomly generated face. Last column shows the result of style mixing, where age and gender of the left person is preserved, and other attributes come from the middle person.

7 CONCLUSIONS AND FUTURE WORK

Overall, this paper presents an end-to-end framework for generating synthetic data with faces, that can be modified to suit the purposes of other research activities or certain commercial uses, thus validating the relevancy of our work in a larger context.

To begin with, we established in the introduction chapters how using synthetic data for training neural networks has gained momentum over the past few years, due to its many advantages. To support this notion, we motivate decisions made when designing our framework based on the advantages that synthetic data offers.

Firstly, to leverage the privacy-preserving nature of synthetic data and the high level of quality and diversity of recent GAN architectures, we built our solution on top of a StyleGAN2 generator network trained to generate human faces.

Secondly, to support the ease of obtaining training labels for synthetic data, we conduct experiments based on the approach suggested by Zhang et al. [65]. By adapting the feature interpreter network to a convolutional architecture, instead of the resource-intensive ensemble of fully-connected networks, we achieve great results in the problem of face semantic segmentation, with a mean F1 of 92.01%.

Thirdly, to reinforce the idea of using synthetic data to complement real data in underrepresented cases, we train and use an attribute mapping network, which may later be used to generate a large amount of synthetic data with attributes rarely present in the real data. We reach a high control precision, with a 4.47 mean age error and a 95.53% gender accuracy.

Future work

Our work can be extended in a couple of directions.

Firstly, in order to improve the quality of the obtained training labels, one can study the effects of adding more training samples with manually annotated labels. Even though using the current set of 30 training samples results in overfitting after a number of epochs, we hypothesize that the performance saturates with a relatively small number of examples.

Secondly, a semantic segmentation network should be trained on the generated data, in order to effectively judge the exact impact that synthetic data has. Training on mixed real and synthetic data would show if generated data actually improves results over using real data exclusively.

BIBLIOGRAPHY

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] John M Abowd and Lars Vilhuber. How protective are synthetic data? In *Privacy in Statistical Databases: UNESCO Chair in Data Privacy International Conference, PSD 2008, Istanbul, Turkey, September 24-26, 2008. Proceedings*, pages 239–246. Springer, 2008.

[3] Alankrita Aggarwal, Mamta Mittal, and Gopi Battineni. Generative adversarial network: An overview of theory and applications. *International Journal of Information Management Data Insights*, 1(1):100004, 2021.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[5] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, page 187–194, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[6] Fadi Boutros, Marco Huber, Patrick Siebke, Tim Rieber, and Naser Damer. Sface: Privacy-friendly and accurate face recognition using synthetic data, 2022.

[7] Eric R. Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis, 2021.

[8] Aayush K Chaudhary, Rakshit Kothari, Manoj Acharya, Shusil Dangi, Nitinraj Nair, Reynold Bailey, Christopher Kanan, Gabriel Diaz, and Jeff B Pelz. Ritnet: Real-time semantic segmentation of the eye for gaze tracking. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3698–3702. IEEE, 2019.

[9] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[10] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples, 2019.

[11] Jessamyn Dahmen and Diane Cook. Synsys: A synthetic data generation system for healthcare applications. *Sensors*, 19(5):1181, 2019.

[12] Saloni Dash, Andrew Yale, Isabelle Guyon, and Kristin P Bennett. Medical time-series data generation using generative adversarial networks. In *Artificial Intelligence in Medicine: 18th International Conference on Artificial Intelligence in Medicine, AIME 2020, Minneapolis, MN, USA, August 25–28, 2020, Proceedings 18*, pages 382–391. Springer, 2020.

[13] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

[14] D.C Dowson and B.V Landau. The fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450–455, 1982.

[15] European Commission. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance), 2016.

[16] Tiziano Fagni, Fabrizio Falchi, Margherita Gambini, Antonio Martella, and Maurizio Tesconi. Tweepfake: About detecting deepfake tweets. *Plos one*, 16(5):e0251415, 2021.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[18] David Güera and Edward J Delp. Deepfake video detection using recurrent neural networks. In *2018 15th IEEE international conference on advanced video and signal based surveillance (AVSS)*, pages 1–6. IEEE, 2018.

[19] Tal Hassner, Shai Harel, Eran Paz, and Roee Enbar. Effective face frontalization in unconstrained images, 2014.

[20] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[22] Songyao Jiang, Zhiqiang Tao, and Yun Fu. Segmentation guided image-to-image translation with adversarial networks, 2019.

[23] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[24] James Jordon, Lukasz Szpruch, Florimond Houssiau, Mirko Bottarelli, Giovanni Cherubin, Carsten Maple, Samuel N. Cohen, and Adrian Weller. Synthetic data – what, why and how?, 2022.

[25] Indu Joshi, Marcel Grimmer, Christian Rathgeb, Christoph Busch, Francois Bremond, and Antitza Dantcheva. Synthetic data in human analysis: A survey, 2022.

[26] Alexander Kapitanov, Karina Kvanchiani, and Sofia Kirillova. Easyportrait – face parsing and portrait segmentation dataset, 2023.

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

[28] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

[29] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan, 2020.

[30] Minchul Kim, Feng Liu, Anil Jain, and Xiaoming Liu. Dcface: Synthetic face generation with dual condition diffusion model, 2023.

[31] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation, 2020.

[32] Daiqing Li, Huan Ling, Seung Wook Kim, Karsten Kreis, Adela Barriuso, Sanja Fidler, and Antonio Torralba. Bigdatasetgan: Synthesizing imagenet with pixel-wise annotations, 2022.

[33] Tianye Li, Timo Bolkart, Michael J Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4d scans. *ACM Trans. Graph.*, 36(6):194–1, 2017.

[34] Yingzhou Lu, Huazheng Wang, and Wenqi Wei. Machine learning for synthetic data generation: a review. *arXiv preprint arXiv:2302.04062*, 2023.

[35] Richard T Marriott, Sami Romdhani, and Liming Chen. A 3d gan for improved large-pose facial recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13445–13455, 2021.

[36] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning, 2022.

[37] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[38] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[39] Sophie J. Nightingale and Hany Farid. Ai-synthesized faces are indistinguishable from real faces and more trustworthy. *Proceedings of the National Academy of Sciences*, 119(8):e2120481119, 2022.

[40] Sergey I. Nikolenko. Synthetic data for deep learning, 2019.

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[42] Krut Patel. Mnist handwritten digits classification using a convolutional neural network (cnn), Sep 2019.

[43] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3d face model for pose and illumination invariant face recognition. In *2009 sixth IEEE international conference on advanced video and signal based surveillance*, pages 296–301. Ieee, 2009.

[44] Xi Peng, Xiang Yu, Kihyuk Sohn, Dimitris Metaxas, and Manmohan Chandraker. Reconstruction-based disentanglement for pose-invariant face recognition, 2017.

[45] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[46] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[47] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[48] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Dex: Deep expectation of apparent age from a single image. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 252–257, 2015.

[49] Donald B Rubin. Statistical disclosure limitation. *Journal of official Statistics*, 9(2):461–468, 1993.

[50] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

[51] Fatemeh Sadat Saleh, Mohammad Sadegh Aliakbarian, Mathieu Salzmann, Lars Petersson, and Jose M. Alvarez. Effective use of synthetic data for urban scene semantic segmentation, 2018.

[52] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016.

[53] Viktor Seib, Benjamin Lange, and Stefan Wirtz. Mixing real and synthetic data to enhance neural network training – a review of current approaches, 2020.

[54] Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. Interfacegan: Interpreting the disentangled face representation learned by gans, 2020.

[55] Alon Shoshan, Nadav Bhonker, Igor Kviatkovsky, and Gerard Medioni. Gan-control: Explicitly controllable gans, 2021.

[56] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[57] Jingxiang Sun, Xuan Wang, Yong Zhang, Xiaoyu Li, Qi Zhang, Yebin Liu, and Jue Wang. Fenerf: Face editing in neural radiance fields, 2022.

[58] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[59] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards end-to-end speech synthesis, 2017.

[60] Mika Westerlund. The emergence of deepfake technology: A review. *Technology innovation management review*, 9(11), 2019.

[61] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Matthew Johnson, Virginia Estellers, Thomas J. Cashman, and Jamie Shotton. Fake it till you make it: Face analysis in the wild using synthetic data alone, 2021.

[62] Andrew Yale, Saloni Dash, Ritik Dutta, Isabelle Guyon, Adrien Pavao, and Kristin P Bennett. Privacy preserving synthetic health data. In *ESANN 2019-European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2019.

[63] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications, 2023.

[64] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation, 2020.

[65] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. Datasetgan: Efficient labeled data factory with minimal human effort, 2021.

[66] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network, 2017.

[67] Yuyang Zhao, Na Zhao, and Gim Hee Lee. Synthetic-to-real domain generalized semantic segmentation for 3d indoor point clouds, 2022.

[68] Qi Zheng, Jiankang Deng, Zheng Zhu, Ying Li, and Stefanos Zafeiriou. Decoupled multi-task learning with cyclical self-regulation for face parsing. In *Computer Vision and Pattern Recognition*, 2022.

[69] Yinglin Zheng, Hao Yang, Ting Zhang, Jianmin Bao, Dongdong Chen, Yangyu Huang, Lu Yuan, Dong Chen, Ming Zeng, and Fang Wen. General facial representation learning in a visual-linguistic manner, 2022.

[70] Kuan Zhu, Haiyun Guo, Zhiwei Liu, Ming Tang, and Jinqiao Wang. Identity-guided human semantic parsing for person re-identification, 2020.

A CONFIGURATION FILES

```yaml
DATASET_TRAIN:
  PATH: "face_seg_dataset/train"

DATASET_VALID:
  PATH: "face_seg_dataset/val"

TRAIN_SETUP:
  DEVICES: [0]
  WORKERS: 2
  BATCH_SIZE: 2
  EPOCHS: 100
  OPTIMIZER: "adam"
  LEARNING_RATE: 0.001
  WEIGHT_DECAY: 0.0001

MODEL:
  PRETRAINED: "weights/pretrained/ffhq-512.pt"
  SIZE: 512
  TRUNCATION: 1.0
  TRUNCATION_MEAN: 4096
  CHANNEL_MULTIPLIER: 2
  LATENT: 512
  N_MLP: 8
  N_CLASSES: 18
```

Listing 1: Segmentation network training configuration file

```yaml
DATASET_TRAIN:
  PATH: "attributes_train_dataset/train_df.pkl"

DATASET_VALID:
  PATH: "attributes_train_dataset/val_df.pkl"

TRAIN_SETUP:
  DEVICES: [0]
  WORKERS: 2
  BATCH_SIZE: 64
  EVAL_EVERY: 2000
  OPTIMIZER: "adam"
  LEARNING_RATE: 0.001
  WEIGHT_DECAY: 0.0001

MODEL:
  PRETRAINED: "weights/pretrained/ffhq-512.pt"
  SIZE: 512
  TRUNCATION: 1.0
  TRUNCATION_MEAN: 4096
  CHANNEL_MULTIPLIER: 2
  LATENT: 512
  N_MLP: 8
  N_CLASSES: 18
```

Listing 2: Attribute mapping network training configuration file

B DETAILED ARCHITECTURES

Layer type	Input shape	Output shape
convolutional	3x224x224	64x224x224
convolutional	64x224x224	64x224x224
convolutional	64x224x224	64x224x224
max pool	64x224x224	64x112x112
convolutional	64x112x112	128x112x112
convolutional	128x112x112	128x112x112
max pool	128x112x112	128x56x56
convolutional	128x56x56	256x56x56
convolutional	256x56x56	256x56x56
convolutional	256x56x56	256x56x56
max pool	256x56x56	256x28x28
convolutional	256x28x28	512x28x28
convolutional	512x28x28	512x28x28
convolutional	512x28x28	512x28x28
maxpool	512x28x28	512x14x14
convolutional	512x14x14	512x14x14
convolutional	512x14x14	512x14x14
convolutional	512x14x14	512x14x14
maxpool	512x14x14	512x7x7
linear	25088	4096
linear	4096	4096
linear	4096	$n\_classes$

Table 11: VGG-16 architecture [56]. $n\_classes$ is the number of output classes the network was trained to classify. In the case of the network trained to predict ages, that number is 101 and in the case of the gender predictor, the number of classes is 2.

C MORE DATA GENERATION RESULTS

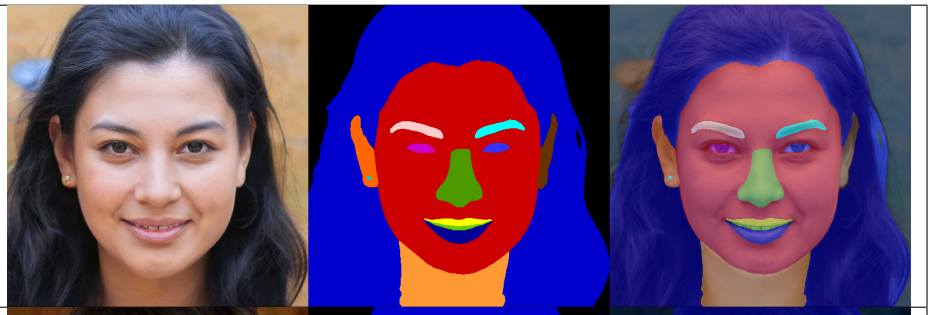
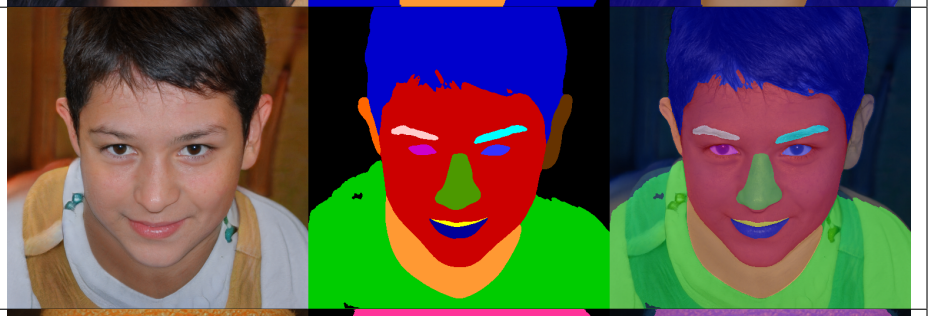
Age 30, woman	
Age 15, man	
Age 65, woman	
Age 50, man	

Table 12: Samples produced with all components enabled

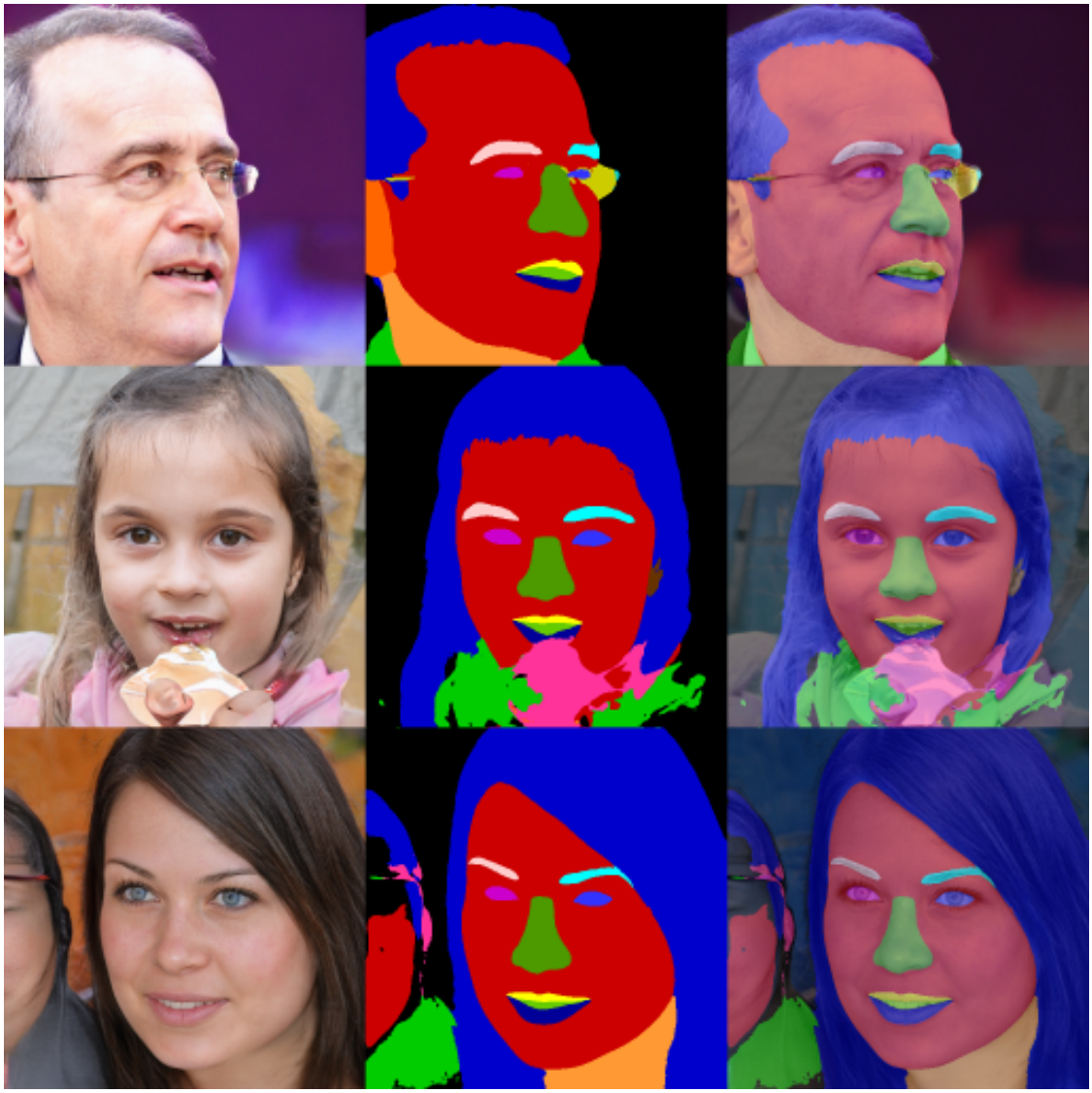


Figure 17: Failure cases of semantic segmentation. The main issues are related to the face generation itself: uncertain eyeglasses, generation artifacts, or part of a second person present in the image.