# StyleGenes: Discrete and Efficient Latent Distributions for GANs

Evangelos Ntavelis [1,2] Mohamad Shahbazi[1] Iason Kastanis[2] Martin Danelljan[1]
Luc Van Gool[1,3]

[1] Computer Vision Lab, ETH Zurich, CH [2] CSEM, CH [3] KU Leuven, BE
entavelis,mshahbazi,martin.danelljan,vangool@vision.ee.ethz.ch,iason.kastanis@csem.ch

## Abstract

*We propose a discrete latent distribution for Generative Adversarial Networks (GANs). Instead of drawing latent vectors from a continuous prior, we sample from a finite set of learnable latents. However, a direct parametrization of such a distribution leads to an intractable linear increase in memory in order to ensure sufficient sample diversity. We address this key issue by taking inspiration from the encoding of information in biological organisms. Instead of learning a separate latent vector for each sample, we split the latent space into a set of* genes. *For each gene, we train a small bank of gene* variants. *Thus, by independently sampling a variant for each gene and combining them into the final latent vector, our approach can represent a vast number of unique latent samples from a compact set of learnable parameters. Interestingly, our gene-inspired latent encoding allows for new and intuitive approaches to latent-space exploration, enabling conditional sampling from our unconditionally trained model. Moreover, our approach preserves state-of-the-art photo-realism while achieving better disentanglement than the widely-used StyleMapping network.*

## 1. Introduction

Generative adversarial networks (GANs) have seen tremendous progress since the seminal work by Goodfellow et. al [8]. GANs have been successfully applied to a plethora of tasks, including conditional generation from semantic categories [2, 36, 37], images [5, 40], text [28, 31, 45], and semantic layouts [24, 26, 35, 52]. Compared to their early predecessors, recent GANs [3, 16, 25, 34] are significantly more capable of realistic and diverse generation of images, with a vast number of works aimed at designing better architectures, training objectives and training strategies [9, 14, 15, 17, 21].

The core GAN formulation, however, remained largely the same: a generator transforms a latent code *sampled*

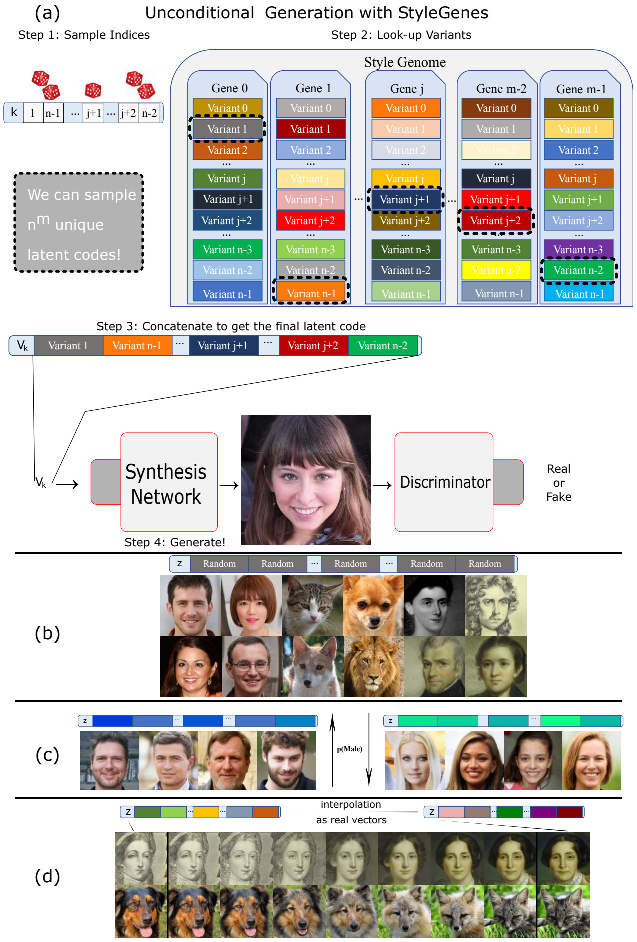


Figure 1. We propose *StyleGenes*: a biologically inspired discrete latent distribution for GANs. Our *Genome* (a) is an ordered set of smaller codebooks, we call *genes*. Each gene contains a collection of embeddings, its *variants*. For each gene, we select a variant and concatenate them to produce our latent code. We train for and perform unconditional image synthesis (b) by randomly sampling a *variant* for each *gene*. Analyzing our discrete *Genome* let us associate genes with specific attributes. We leverage this information to conditionally generate from our unconditionally trained model, without retraining the network or training any additional modules (c). Although our latent distribution is discrete, the learned style space offers emergent continuous properties, ensuring smooth interpolation between samples (d).

*from a continuous distribution* to a realistic-looking image. Initially, the latent code was sampled by a uniform distribution [8]. Quickly, however, the community converged to using a Gaussian prior [10, 41]. An important change came subsequently when Karras *et al* [17] altered the standard design of the generator network. The sampled noise is no longer given to the network as the initial input, but akin to a conditional [26] or a style transfer [12] generator, it was used to manipulate the intermediate feature maps after the convolutions. Nevertheless, the Gaussian input is mapped to an intermediate latent *style space* through a multi-layer perceptron. The motivation was that this learned space does not have to adhere to a sampling density of a fixed distribution and can be disentangled.

Interpretation and manipulation of the GANs input space, or the latent style space, has been a subject of extensive research [13, 19, 39, 42, 46]. These works usually train a separate model to make sense of the latent vector space: training a conditional normalizing flow [1] or a classifier [23] to enable conditional sampling. For generated sample manipulation, they train one vector per transformation [13], discover style channels via gradient computation [42] or apply clustering to hidden layers [6]. The use of these intricate techniques and the importance of the downstream task they are trying to tackle, raise the question on whether we can design a latent space that would permit a straightforward analysis.

In this work, we take a different approach to continuous sampling and modulate the generator with latent codes sampled from a discrete prior distribution. We set the different outcomes of this distribution to be learnable embeddings, which induces the benefit of direct optimization of the samples. A standard approach to designing such a discrete distribution of embeddings would require a memory bank of all the latent vectors. However, the advantage of an image synthesis network, is that it can generate countless novel samples. This is not feasible with such a formulation.

To tackle this key issue, we introduce a compact representation of a discrete distribution capable of generating an exponentially large number of distinct samples. We draw inspiration from how the blueprint of a complex living organism, the DNA, can represent the great amount of diversity found in nature. Only four letters, the nucleotides, form the words, the genes, that tell the story of our biology. A virtually endless degree of variation can be obtained by combining different variants of these genes. Accordingly, we design our latent genome. We break the latent code into smaller parts, the *genes*. Each gene is sampled from a smaller set of gene *variants*. These combine into the final latent vector, analogous to the chromosome in organisms.

We introduce *StyleGenes*: an ordered collection of gene variants that are learned in conjunction with the generator, and can generate a great diversity of realistic-looking images. The nature of our latent space offers a straightforward way to interpret the associations between the discrete samples and the synthesized images. These associations can be exploited to enable downstream tasks, such as conditional generation.

Our contributions are summarized as follows.

- We introduce a compact parameterization of a discrete latent distribution for GANs, inspired by the encoding of information in biological organisms.
- Our discrete latent space formulation permits a natural and straightforward analysis of the association of genes to semantic image attributes.
- We use a pretrained classifier to integrate class-conditioning *after* training. Our analysis allows us to conditionally sample from the unconditionally trained model without the need to retrain, or train additional modules.
- The learned discrete latent space is more disentangled than the widely-used StyleGAN's W space.
- We show that despite the discrete latent distribution, the resulting style space obtains continuous properties, important for e.g. realistic interpolation and propose a method to project real images in our codebook.

We perform experiments on a variety of widely-used image generation datasets and two established GAN baselines. Our approach obtains visual results on par with the baseline continuous case, while benefiting from the intuitive gene-based approach to conditional generation manipulation offered by our StyleGenes representation. Furthermore, our approach eliminates the need of a Style Mapping network, as it can be trained using few parameters while being *faster* and yielding a more disentangled latent space.

## 2. Related Work

**Latent Code Quantization:** VQ-VAE [38] is one of the first studies to exploit discrete representations for image generation. VQ-VAE is designed to prevent the posterior collapse in VAE framework when the latent representations are paired with a powerful decoder [38]. Instead of a continuous latent space, VQ-VAE represents the latent space as a spatial grid of quantized local latent codes, which are sampled from a discrete set of learned vectors in an auto-regressive manner. VQ-VAE2 [30] is an improved version of VQ-VAE, which is capable of generating images of higher diversity and resolution by using a hierarchical multi-scale latent maps. The idea of VQ-VAE later on was extended to a GAN framework by changing the reconstruction loss and adding an adversarial one [7]. Moreover, a transformer is used to learn the auto-regressive priors for sampling the discrete local latent vector. Building on the previous approaches, RQ-VAE [20] proposes a residual feature quantization framework, which enables their model to work with smaller number of representation vectors. Feature quantization has also been used in the discriminator of GANs to increase the stability of the adversarial training [51]. This study bears similarities to the above works in formulating the latent space as a composition of discrete feature vectors. However, different to prior studies, we investigate discrete sampling of the latent code in the unsupervised GAN framework [18], without employing any encoder or self-supervised objective. These approaches deploy an auto-encoder based approach, that produce local discrete codes and need auto-regressive sampling to draw new samples. Our codebook is not trained through vector quantization, but rather through the adversarial game; it provides a global description of the image to be generated and thus does not require auto-regressive sampling.

**Latent code as a composition of smaller parts:** InfoGAN [4] aimed at bringing more interpretability and disentanglement to the latent codes of GANs by maximizing the mutual information between parts of the latent code and the corresponding generated images. Inspired by the formation of DNA from genes, DNA-GAN [43] and ELEGANT [44] also proposed dividing the latent code into smaller attribute-relevant and attribute-irrelevant parts, which are then supervised using attribute annotations to create attribute disentanglement in GANs. Similar to these studies, or method divides the style vectors into smaller codes. Additionally, the style codes in our method consist of smaller codes. However, different to InfoGAN, our method only uses a discrete set of codes to form the latent style codes. Note, we do not explicitly train our method for disentanglement and feature transfer but only for unconditional image synthesis.

**Analyzing the style space:** Steering the latent space of GANs is of high interest for many applications of image editing and conditional generation [13, 39, 46]. Previous studies' focus has primarily been on analyzing the style space, as it is more well-behaved and disentangled compared to the traditional latent space in prior GAN models. One goal of this style space analysis is to discover meaningful directions in the style space for semantic editing of images [13, 42]. Moreover, [19] uses style space to explaining and interpret the decisions made by attribute classifiers. The style space has also provided the opportunity for paired data generation using only a few annotations [50]. Recent methods utilize unconditionally pretrained models for conditional generation [1, 23]. These approaches, train a conditional normalizing flow [1] or a classifier [23] in the latent space to enable conditional sampling.In this study, we do not need to train one vector per transformation [13], compute any gradients [42] or apply clustering to hidden layers [6]. In contrast, we treat the network as a black box and, without extra training, only harness the benefits of its discrete input to enable conditional generation.

# 3. Method

In the present widely-established [14, 18] image generation paradigm, a latent vector sampled from a *continuous* multi-variate prior distribution [8] is transformed through a generator network in order to achieve the final image. In this work, we aim to offer a different approach, by starting from a *discrete* distribution. We propose to sample a set of smaller latent codes from a codebook, consisting of a collection of embeddings that are trained through the adversarial learning.

However, composing the codebook as a collection of final latent vectors leads to an intractable memory cost, as we require the generation of at least millions of unique examples. We therefore take inspiration of how biological organisms encode information as a sequence of discrete entities, called *genes*. Analogous to genes, we partition our latent vector and codebook into a sequence of *positions*. At each position, we independently sample from the set of embedding *variants* contained in the codebook, as illustrated in Figure 1-a. Even with a very compact codebook, our discrete latent sampling allows for countless combinations due to the combinatorial formulation.

## 3.1. Generator with continuous prior

In the classic unsupervised image synthesis literature, the generator is a function that transforms the input noise to the image domain as,

$$I = G(z_c)\theta_G\,, \quad z_c \overset{\text{iid}}{\sim} p_z\,, \quad z_c \in \mathbb{R}^d \tag{1}$$

where $z_c$ is sampled from a prior distribution $p_z$, and $\theta$ are the generator's weights. Early works [8, 29] sample $z_c$ from a uniform distribution. Subsequent works [10, 41] sample from a standard Gaussian distribution. Since the introduction of StyleGAN [17] and the models based on it, an additional model element is deployed: a Multi-Layer Perceptron. The weights of this *mapping* network, are learned in tandem with the generator's through the adversarial objective. It is used as a push-forward operator to transform the Gaussian input distribution to an intermediate latent space $\mathbb{W}$.

$$w = \text{Mapping}(z_c, \theta)\,, \quad z_c \overset{\text{iid}}{\sim} \mathbb{N}(0, I) \tag{2}$$

We propose an alternative method for learning a disentangled latent space $\mathbb{W}$, presented next.

## 3.2. A scalable codebook of learned latent codes

We aim to learn a discrete latent distribution. To this end, we first introduce a codebook of $n$ learnable embeddings. Before training, the embeddings are initialized using a standard Gaussian distribution. Through adversarial learning the embeddings are optimized, and therefore capable of representing flexible and complex style distributions. While such a formulation permits learning a set of latent codes that can generate realistic outputs, it has a fundamental flaw. The number of distinct samples we can generate scales linearly with the number of embeddings. For a latent code of length $d = 512$, we would need to learn over 35 million parameters only to be able to generate $70,000$ distinct images (the size of the FFHQ dataset [17]).

Inspired by how DNA encodes information in a discrete and compositional manner, we instead let the latent code be composed of an ordered set of positions, analogous to genes. At each position we independently and uniformly sample one of its embedding *variants* from the codebook. Then we concatenate this sequence of sampled variants into the latent code, which is used as input to the generator,

$$V_k = [v_1^{k_1}, v_2^{k_2}, ..., v_{n_g}^{k_{n_g}}] , \quad k_i \in \{1, 2, \dots, n_v\} \tag{3}$$

Here, $v_i^j$ denotes the variant $j$ of position $i$. The vector $k$ of uniformly sampled indices $k_i$ selects the variant $v_i^{k_i}$ for each position $i$. The dimensionality of $k$ is the number of positions s, $n_g$, in our codebook. The number of variants for each position is denoted $n_v$. The final image is achieved by decoding our style vector with the generator network $G$,

$$I_{z_d} = G(V_k; \theta) . \tag{4}$$

We visualize the process in Figure 1-a. Note that all embedding variants have the same length, such that $v_i^j \in \mathrm{R}^{d_g}$, where $d_g = d/n_g$ and $d$ is the total number of elements in the resulting latent code $V_k$. This formulation permits the increase of distinct samples $n_{img}$ we can generate to,

$$n_{img} = n_v^{n_g} . \tag{5}$$

For example, using a latent dimension of $d = 512$ with $n_g = 64$ genes and $n_v = 256$ variants, we can generate approximately $1.34 \times 10^{154}$ different samples; more than the estimated number of atoms in the observable universe. On the other hand, the non-compositional discrete approach using the same codebook size can only generate 256 distinct samples. In fact, by keeping $d = 512$ constant, the number of trainable parameters remains independent of the number of positions $n_g$, while allowing an exponential increase in the number of distinct samples according to Equation 5.

Note that our Genome is trained from scratch together with the synthesis network, guided only by the adversarial loss (See Figure 1).

### 3.3. Attribute-based sampling and analysis

A key feature of our discrete latent formulation, is that it provides for a simple and effective method for analysis and guided sampling. In this section, we introduce an approach to attribute-based analysis and conditional sampling, by aggregating statistics of how a set of image-specific attributes relate to individual elements in the codebook.

Let $\{a_1, \dots, a_L\}$ denote the attributes that are used to describe an image, for the specific dataset on which our generator is trained. Each attribute $a_l$ can take a finite set of values. For instance, in case of a face dataset, an attribute can describe the existence of glasses, beard, lipstick, or the hair color. In order to perform conditional image generation given a specified set of attributes, we need to estimate the conditional latent distribution $p(k|a_1, \dots, a_L)$. We assume the positions to be conditionally independent $p(k|a_1, \dots, a_L) = \prod_i p(k_i|a_1, \dots, a_L)$. We then obtain,

$$p(k_i|a_1, \dots, a_L) = \frac{p(a_1, \dots, a_L|k_i)p(k_i)}{\sum_{k_i} p(a_1, \dots, a_L|k_i)p(k_i)} = \frac{\prod_l p(a_l|k_i)}{\sum_{k_i} \prod_l p(a_l|k_i)} \tag{6}$$

The first equality is the application of Bayes' rule. In the second equality, we use that $p(k_i) = \frac{1}{n_v}$ is uniform and assume the attributes to be conditionally independent given the variant $k_i$. The latter assumption is motivated by the high degree of disentanglement that we observe across variants and positions. Further, note that this conditional independence assumptions by no means imply that the generated attributes themselves are independent. In fact, as observed in our experiments, our approach captures the strong correlations that exist between certain attributes, such as 'male' and 'beard' (see our genome analysis and Figure 2).

Equation 6 shows that the conditional distribution of the latents are fully given by the marginal attribute distribution for a given embedding variant $p(a_l|k_i)$. We aggregate statistics over the generated image samples to estimate the latter:

$$p(a_l|k_i = j) = \sum_k p(a_l|G(V_k))p(k|k_i = j) \approx \frac{\sum_{k \in S: k_i = j} p(a_l|G(V_k))}{\sum_{k \in S: k_i = j} 1} \tag{7}$$

Here, $p(a_l|G(V_k))$ is the attribute distribution of the generated image $G(V_k)$, which we estimate with a pre-trained image classifier. In the first equality, we marginalize over all possible latent vectors $k$. However, as this is intractable, we approximate the expectation value through Monte-Carlo sampling. Specifically, we pre-generate a set of images $\{G(V_k) : k \in S\}$, where the latents in $S$ are sampled from $p(k)$. We can efficiently re-use the same set of images, generated from $S$, when computing Equation 7 for all variants $k_i$ and attributes $l$.

To further increase the likelihood of sampling codebook entries with high probability of the conditioned attribute class, we scale the estimated statistics with a temperature parameter $p(a_l|k_i)^{\frac{1}{T}}$ when employed in Equation 6. This serves to increase the class consistency of the conditional sampling in our experiments.

**A: Unsupervised Image Generation**

| | FID ↓ | | | | | Time/latent ↓ |
|---|---|---|---|---|---|---|
| | **FFHQ** | **AFHQ** | **Met/s** | **Church** | **Beds** | |
| StyleGAN2 | | | | | | |
| StyleMapping | 5.3 | **5.62** | **20.48** | 8.13 | 51.61 | 0.483 ms |
| StyleGenes | **5.11** | 5.99 | 21.00 | **6.86** | **17.84** | **0.170 ms** |
| ProjectedGANs(FastGAN) | | | | | | |
| Cont. Prior | 5.08 | 4.02 | 15.38 | **3.05** | 3.15 | **0.015 ms** |
| StyleGenes | **4.19** | **3.66** | **15.24** | 3.08 | **2.96** | 0.076 ms |

**B: Ablation Study on StyleGenome**

| **FID ↓** | **FFHQ** | | | **AFHQ** | | | **Metfaces** | | |
|---|---|---|---|---|---|---|---|---|---|
| *Genome* | # Genes | | | # Genes | | | # Genes | | |
| #Variants | 64 | 8 | 2 | 64 | 8 | 2 | 64 | 8 | 2 |
| 256 | 5.87 | 5.34 | 24.72 | 6.45 | 12.43 | 18.64 | 22.56 | 38.76 | 42.60 |
| 512 | 5.53 | 5.64 | 12.34 | **5.99** | 7.24 | 13.77 | 21.54 | 27.20 | 37.71 |
| 1024 | 5.71 | 5.20 | 6.2 | 6.11 | 10.33 | 10.47 | 21.99 | 25.05 | 32.08 |
| 2048 | 5.22 | **5.11** | 5.30 | 6.31 | 6.37 | 7.31 | 21.39 | **21.00** | 30.93 |

Table 1. **A:** Evaluation of our discrete sampling approach, *StyleGenes*, by substituting the StyleGAN2's StyleMapping network or FastGAN's Gaussian sampling for ProjectedGAN. We achieve similar or better FID to the continuous case. **B:** Ablation on different configurations of the genome and our baseline. Increasing the number of the embeddings in our codebook, the *# Variants*, increases the performance by increasing the number of parameters we are using. We can also lower the FID by breaking the latent code into more genes of smaller lengths. This increases the number of unique codes we can sample from our genome without increasing the memory/parameters.

**Predicting Attribute Presence from Latent Codes**

| Method | male | young | bald | gray-hair | h-makeup | mustache | no-beard | w-earrings | w-lipstick | mean |
|---|---|---|---|---|---|---|---|---|---|---|
| StyleMapping | 49.74% | 63.43% | 96.02% | 92.49% | 87.71% | 95.37% | 79.06% | 84.73% | 69.30% | 79.76% |
| StyleGenes | **86.71%** | **82.90%** | **97.01%** | **93.68%** | **92.09%** | **95.82%** | **91.53%** | **86.40%** | **85.89%** | **90.23%** |

Table 2. We measure disentanglement by our ability to predict an attribute's presence in a generated image from its latent code. StyleGenes' codes are much easier to associate to attributes than the StyleMapping's ones.

**Comparison with Vector Quantization approaches - FFHQ - FID ↓**

| VQ-GAN | ViT-VQGAN | Ours /w StyleGAN2 | Ours /w ProjectedGAN |
|---|---|---|---|
| 9.6 | 5.3 | 5.11 | **4.19** |

Table 3. VQ-methods are quantized and not self-learned (different losses, encoder). They learn local descriptors spatially aligned in a grid and require multiple different codes for semantically rich and diverse images. Contrary, we use a single global code, avoiding a parameter explosion with our Genome. We don't require an autoregressive sampler (e.g. transformer), and thus are faster.

## 4. Experiments

**Implementation** Our method, StyleGenes, is written in Pytorch [27]. We incorporate our sampling approach into two baseline models: (1) StyleGAN2 [18] as provided in the StyleGAN3 [16] codebase and ProjectedGANs [33] using the FastGAN [21] generator. For all datasets, we train all our models and baselines *unconditionally* using 4 GPUs following the default configuration as described in each project's code repository [16,33]. For small dataset Metfaces [15] and AFHQ [5] we use adaptive discriminator augmentation [15]. For our StyleGAN2 experiments, we train until the discriminator has seen 10 million images of resolution 256 × 256. For ProjectedGAN, we train for their reported number of iterations to reach state-of-the-art results, rounded up to the next million: 8M images for FFHQ [17] and 2M images for the other datasets.

**Datasets** We investigate the performance of our network using the *Fréchet Inception Distance (FID)* [11], on widely used datasets for unsupervised image generation:
**FFHQ** [17], is a collection of 70,000 face images scraped from flickr.com. The images were centered around the eyes and the mouth of the individual, offering strong position priors. The people depicted in the images come from a diverse background, age and poses.
**MetFaces** [15] is a dataset of image crops from art pieces of the Metropolitan Museum of Art Collection. Similarly to FFHQ the crops are centered around human faces. The dataset contain 1336 images in total. The images are under CC0 license by the Metropolitan Museum of Art. **AFHQ** [5] is a collection of 15.000 images of animal faces divided equally into three categories: cat, dog and wildlife. However, in this work we do not use the labels for conditional generation.
**LSUN Church & Bedroom** [47]. We are using two subsets of the LSUN dataset *Church* and *Bedroom*, where they contain diverse outdoor and indoor scenes respectively. We use the full LSUN Church dataset of 126,227 images and a subset of the bedroom scenes comprised of 121,000 images.

### 4.1. Unconditional Generation

In Table 1-A we see the performance of established baselines [18, 33] using StyleGenes. We compare with the continuous approach by training our baselines [15, 33] with the same hyperparameters and number of images. Our proposed discrete method produces similar results with StyleGAN's StyleMapping approach, and improves ProjectedGAN when it replaces Gaussian sampling. Note, that StyleGAN2 failed to converge in our Beds experiments.

In our ablation study (Table 1-B), we analyze the effect of the different Genome configurations to the perceptual performance of the network, while keeping the size of the resulting latent vector fixed at $d = 512$.

Increasing the number of different variants for each gene increases the number of parameters: $n_v * d$. Doing so yields better FID scores.

Note that every experiment that is in the same row in Table 1-B is using the same number of parameters. When we change the number of genes $n_g$, we also change the size of each sub-vector/variant $v_i^{k_i}$, such as $n_g * len(v_i{}^{k_i}) = d$. Thus, by increasing the number of genes we do not increase

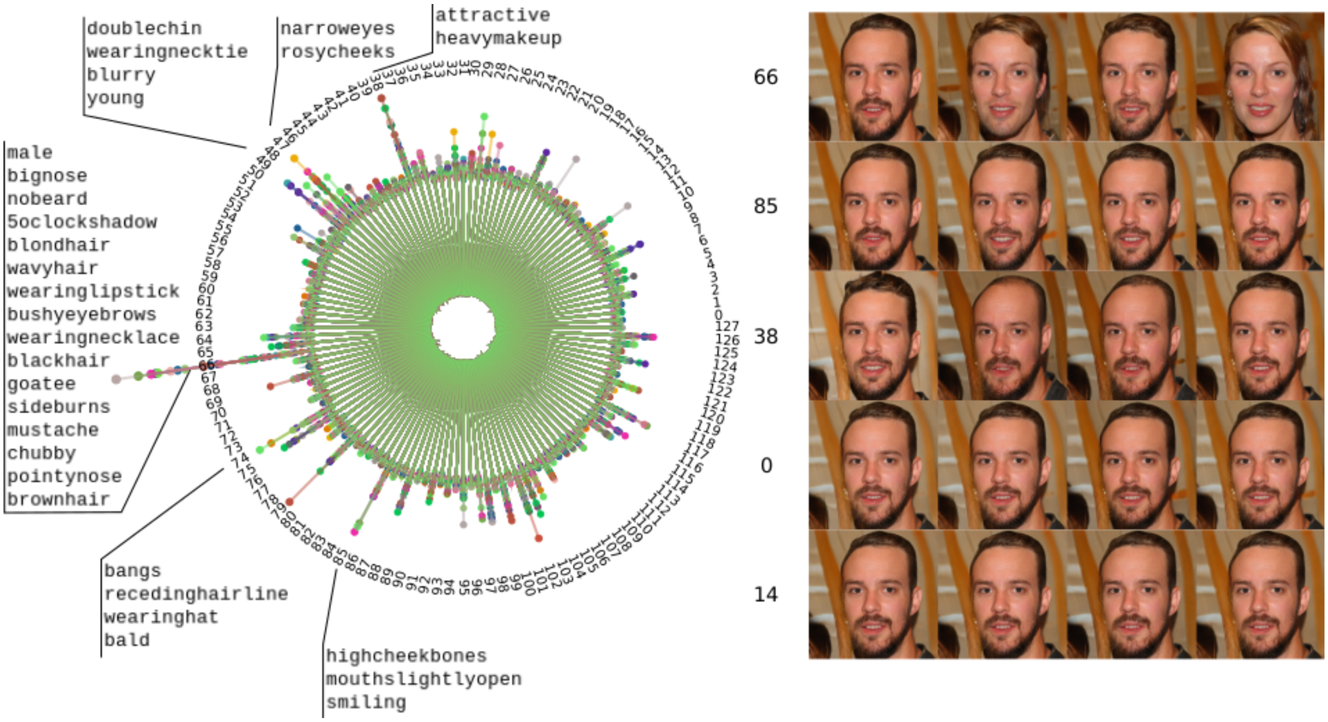


Figure 2. **Which genes affect which attribute?** The polar plot shows which genes affect which CelebA attributes the most. Each color represents a different attribute. The labels indicate the gene that exhibits the highest variability in this attribute. Most attributes are affected only by a handful of genes. We observe that specific genes affect pertinent attributes. For instance, genes 66 and 38 affect gender and hair color, while gene 85 the mouth-related features. We show how randomly changing a specific gene's variants produces different alterations to the images on the right. Changing the variants of genes 0 and 14, which do not exhibit importance for any attribute, leads to minute changes. Most genes fall into this category.

the memory footprint, but the number of different images the genome can represent is also increasing, per Equation 5. This also leads to a decrease of FID. In Table 1-right we can observe that for all three datasets, for the smallest gene length, going from variants' number of 2048 to 256 leads to a minor deterioration of performance. However, the memory footprint of the genome is decreased 8-fold.

To summarize, we observe that we can increase the performance of our network by either increase its parameters by increasing the number of variants, or by dividing it up to more genes without an increase of memory.

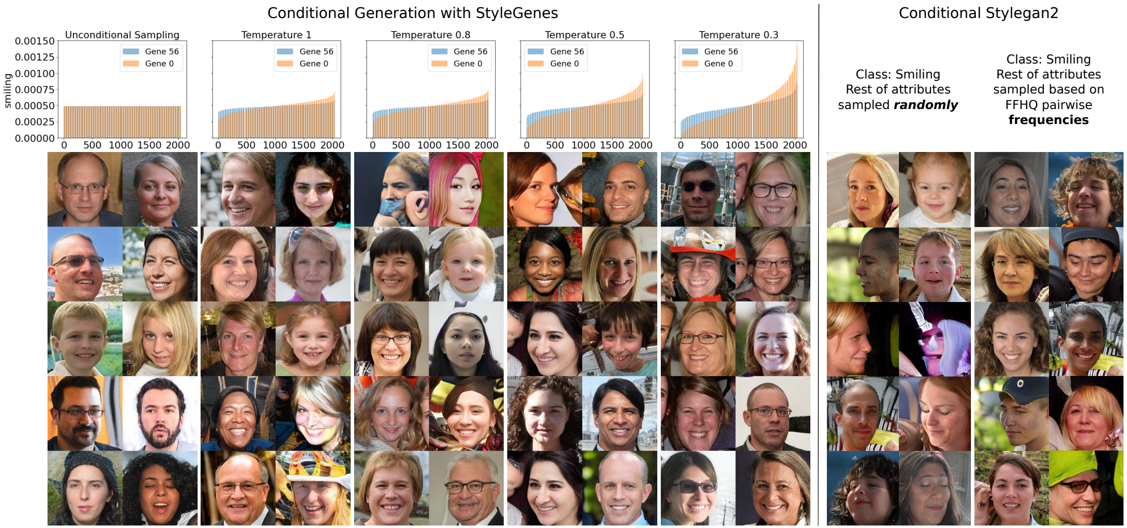
Additionally, in Table 3 we provide a comparison with methods that are using vector quantization [7, 48].

### 4.2. Analyzing the Codebook

In this section we exploit the discrete nature of our approach to analyze the association of the latent codes and their corresponding attributes. We show how our method improves disentanglement and how to harness our analysis to use our unconditionally trained model for conditional generation. Lastly, we show how to do interpolation and projection in our setting.

**Associating variants with attributes** As described in Section 3 we run a Monte Carlo experiment to estimate the probability $p(a_l|k_i = j)$ of the variant $j$ at position $i$ resulting to the attribute $a_l$ in the output image. We randomly sample $500,000$ gene sequences from our FFHQ model and generate their corresponding images. We pass each of these images through 40 pretrained CelebA classifiers [22]. We used the weights provided by the original StyleGAN [17] repository, and the code provided by StyleSpace [42] to extract the logits for every image. For instance, let $i = 15$ and $j = 217$, and $a_l$ be a facial attribute, such as *black hair*. We estimate $p(blackhair|k_{15} = 217)$ by averaging the outputs of the *black hair* classifier for *all* style codes containing variant 217 in their $17^{th}$ position. We repeat the process for each variant and attribute to get the marginal attribute distribution for a given genome variant.

**Which genes are responsible for each attribute?** We want to test if, like its biological inspiration, our genome has specific genes that control the expression of certain attributes, such as hair color. We would like to measure the impact that changing a gene has to a certain trait of the output image. We hypothesize that if a gene controls an attribute, it will exhibit high variance in its expected values and have extreme values towards both ends. We quantify a gene's importance by calculating the *mean absolute standard score* for each gene position: the absolute distance in terms of standard



| Method | Sampling | Avg. | male |  | eye-bags |  | h-cheek/s |  | smiling |  | big-nose |  | open-mouth |  | young |  | w-lipstick |  | attractive |  | eyeglasses |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  |  | yes | no | yes | no | yes | no | yes | no | yes | no | yes | no | yes | no | yes | no | yes | no | yes | no |
|  |  | Classification accuracy (%) ↑ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| baseline | random | 74.54 | 93.32 | 54.24 | 70.62 | 87.66 | 32.78 | 96.46 | 35.20 | 97.76 | 59.02 | 85.74 | 83.24 | 93.16 | 91.44 | 51.34 | 44.50 | 88.52 | 29.46 | **98.82** | 97.92 | 99.54 |
| baseline | freq | 91.24 | 94.24 | 92.18 | 86.86 | 87.74 | 91.06 | 89.04 | 93.50 | 91.66 | 86.42 | 87.50 | 95.90 | 94.30 | **91.80** | 82.26 | 87.98 | 95.50 | 85.74 | 93.50 | 97.72 | **99.82** |
| Ours | temp-1.0 | 71.01 | 74.26 | 74.22 | 66.28 | 85.14 | 69.82 | 75.28 | 75.70 | 74.72 | 71.96 | 69.40 | 76.58 | 71.98 | 0.38 | 65.24 | 63.88 | 81.16 | 71.50 | 96.72 | 68.76 | 87.34 |
| Ours | temp-0.8 | 77.48 | 79.84 | 80.92 | 69.80 | 71.98 | 74.36 | 78.40 | 79.96 | 78.98 | 76.26 | 74.12 | 81.54 | 77.90 | 80.20 | 70.42 | 70.02 | 84.42 | 67.78 | 84.18 | 79.06 | 89.36 |
| Ours | temp-0.5 | 88.02 | 90.58 | 88.36 | 80.76 | 83.28 | 86.26 | 88.00 | 90.46 | 90.84 | 86.66 | 85.20 | 91.38 | 88.36 | 87.12 | 85.40 | 85.84 | 89.58 | 84.00 | 89.54 | 96.10 | 92.70 |
| Ours | temp-0.3 | **95.77** | **98.30** | **96.48** | **91.00** | **93.10** | **96.26** | **96.80** | **97.82** | **97.90** | **96.00** | **95.36** | **98.14** | **93.64** | 89.62 | **97.48** | **95.40** | **96.14** | **94.56** | 95.42 | **99.38** | 96.52 |
|  |  | FID ↓ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| baseline | random | 33.33 | 30.05 | 41.48 | 29.39 | 36.21 | 40.18 | 30.44 | 35.21 | 28.12 | 33.76 | 32.95 | 36.61 | 29.17 | 28.69 | 40.07 | 46.13 | 34.04 | 33.90 | 30.76 | 19.43 | 30.00 |
| baseline | freq | 10.96 | **10.11** | **10.24** | 10.48 | 11.04 | 10.72 | 11.65 | 10.91 | 11.66 | **10.08** | 10.45 | 10.95 | 11.45 | 11.26 | 10.29 | **10.70** | 10.32 | 15.40 | 10.43 | **10.00** | 11.08 |
| Ours | temp-1.0 | 11.08 | 11.11 | 11.24 | **9.98** | 11.81 | 9.63 | 9.78 | 9.96 | **9.69** | 10.88 | 9.60 | 9.94 | **9.30** | 16.86 | **9.42** | 11.99 | **9.59** | 11.20 | 17.86 | 12.16 | **9.53** |
| Ours | temp-0.8 | **10.11** | 11.04 | 10.85 | **9.88** | **10.04** | **9.60** | **9.75** | **9.67** | 9.81 | 10.72 | **9.49** | **9.89** | 9.50 | **10.17** | 9.77 | 11.43 | 9.73 | **10.39** | **9.70** | 11.15 | 9.66 |
| Ours | temp-0.5 | 12.36 | 14.41 | 13.17 | 11.86 | 11.37 | 10.24 | 10.67 | 10.46 | 11.64 | 13.59 | 13.06 | 10.28 | 10.52 | 11.83 | 14.73 | 16.31 | 11.18 | 15.91 | 10.73 | 15.34 | 9.92 |
| Ours | temp-0.3 | 28.42 | 30.92 | 28.53 | 19.43 | 17.50 | 14.97 | 14.35 | 14.24 | 79.46 | 23.46 | 28.90 | 12.99 | 14.40 | 86.22 | 32.29 | 38.63 | 18.33 | 36.92 | 15.27 | 29.86 | 11.86 |

Table 4. **Conditional Generation** We train our method unconditionally, by sampling uniformly the variants for each gene position. With our analysis we can conditionally sample the variants to generate a desired attribute, without retraining our unconditional model. We can control a FID-accuracy trade-off using the temperature. Lower values decrease variability but increase accuracy. In contrast to our method, the conditional StyleGAN baseline is limited to the attributes it was trained with. For inference they need to provide values for every attribute, and thus, to generate an image with a specific attribute, we sample the rest *randomly* or use the real dataset's conditional *frequencies*.

deviations that the gene variants have on average with the codebook's mean expected value for the particular attribute: $s_l^i = \Sigma_j \frac{|p(a_l|k_i=j) - \mu_{p(a_l|k_i)}|}{\sigma_{p(a_l|k_i)}}$.

In Figure 2 we see the score for a gene in a specific position. The genes are placed circularly around the plot. Each color represents one of the 40 attributes. Most genes do not significantly affect any of the attributes, instead controlling local image details. For each attribute, only a handful of genes have high standard scores. On the right side of Figure 2, we see how changing the variants of specific genes alters the output image. We sample a gene sequence and start substituting the variant of one gene at random. Manipulating the genes that exhibit high scores in the polar plot, such as genes 66, 85, and 38, leads to visible changes in the image. However, most genes do not exhibit large scores for any of the attributes. Changing these genes' variants results in barely noticeable changes.

**Conditional Generation** In the previous steps we acquired the marginal attribute distribution for a given variant. We use this information to conditionally generate an image with a desired attribute $a_l$. In Table 4 we can see samples produced by our method. To generate unconditionally we sample the variant for each gene position uniformly. However, as described in Section 3 we can now infer the conditional latent distribution $p(k|a_l)$ and use it to sample the variants instead. In Figure 4 we can see the results of our conditional sampling. Decreasing the temperature $t$ increases the likelihood of the presence of the desired attribute, however, can also limit the variability of the conditional outputs, as it is outlined in increased FID scores in Table 4.

To gauge the ability of our method to generate conditionally, we train a StyleGAN2 model with pseudo-labels from the CelebA classifiers. Training conditionally, limits the model's generation to a fixed set of attributes. Additionally, every attribute needs to have a value, 0 or 1, in order to generate a sample. For our experiment, we sample all other attributes than the one we aim to generate either *randomly* or by using their co-occurrence *frequency* in FFHQ.

In Table 4, we find we compare similarly to our baseline. However, we are not limited by a predefined number of attributes and can be extended to more without training. Moreover, we can use the temperature value to control the

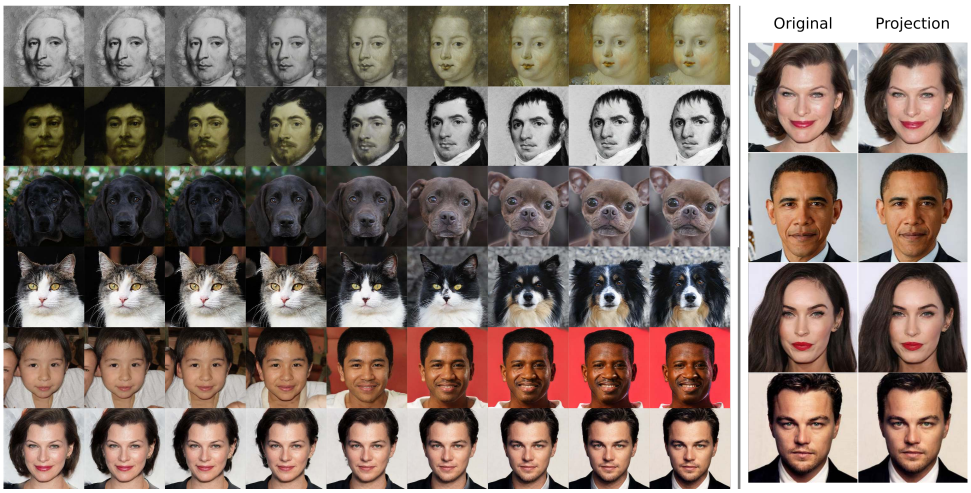


Figure 3. **Interpolation and Inversion.** Our latents are trained in a discrete fashion, but have real values. Thus, it is possible to interpolate between them. Our network can generate realistic results from codes outside of the genome's values. Inspired by this feature, we extend PTI [32] to add real images in our genome.

trade-off between variability and accuracy. Lastly, training conditionally with a small dataset can lead to poor performance and mode collapse [36]. With our method we are conditionally sampling gene variants from our unconditionally trained model and, thus, we do not face the same problem.

**StyleGenome and Disentanglement** The StyleGAN's [17] motivation to design the StyleMapping Network to make the sampling density determined by the mapping and not to be limited to any fixed distribution; they aimed for the resulting space W to be more disentangled. We explore how disentangled our *StyleGenes* are compared to the output space of StyleMapping. We train a Multi-Layer Perceptron to predict the presence of an attributed in an image from its latent code. We randomly sample 50.000 codes from each of the two representations. Then we extract the fake images' attributes using the pretrained CelebA classifiers, and appropriately prepared the train/val/test subsets. We find that StyleGenes outperforms the StyleMapping's accuracy on every attribute we tested, with a 10% average increase, as shown in Table 2. In Figure 2, we see that certain genes affect a group of pertinent attributes. During training, each variant is sampled independently from the rest. We hypothesize that this pushes the variants to be semantically self-contained compared to the StyleMapping approach, which maps an undivided vector to the W space.

**Interpolation**. During training we sample the latent codes from our discrete codebook, but their values lie on $\mathbb{R}^d$. We want to gauge whether the learned genome comprises samples that lie on a smooth surface. We sample two codes and interpolate between them. In Figure 3 we can see interpolation results for all three datasets. The transition is smooth and the subsequent samples are semantically coherent and realistic. By optimizing on discrete samples we are able to learn a continuous distribution.

**Adding real images in the codebook**. We extend the Pivotal Tuning Inversion [32] to project real images into our codebook in Figure 3. We start by concurrently optimizing a set of vectors in the underlying continuous space to produce the images we aim to invert. Then, we find the indices of the nearest-neighbor variant for each gene in the codebook. We train both the generator and the codebook to recreate the images, based only on these indices. We substitute PTI's locality regularization with our codebook-perseverance regularization: we push randomly sampled codes to keep their syntheses unchanged via an LPIPS [49] loss. We find this step important to retain the perceptual quality of the genome.

## 5. Conclusion

In this work we introduce *StyleGenes*. Inspired by how information is encoded in the DNA by only four basic building blocks, we design a discrete sampling approach for GANs. We define our StyleGenome, an ordered collection of gene variants. We uniformly sample a variant for each gene to form a sequence. Its concatenation is the style code used by the generator to synthesize an image. Our discrete sampling technique achieves an FID score on par with its continuous counterpart, while enabling an intuitive way to analyze the latent code. We use pretrained classifiers to aggregate attribute statistics, enabling attribute-based analysis. Our analysis enables conditional sampling out of our unconditionally trained model. Lastly, we show that we can generate samples between the genome's discrete elements, indicating that the samples are on a smooth style surface, and devise an approach to incorporate real images in our genome.

**Acknowledgements** This work was partly supported by CSEM and the ETH Future Computing Laboratory (EFCL), financed by a gift from Huawei Technologies.

## References

[1] Rameen Abdal, Peihao Zhu, Niloy J. Mitra, and Peter Wonka. Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows. *ACM Trans. Graph.*, 40(3), May 2021. 2, 3

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 1

[3] Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *arXiv*, 2020. 1

[4] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 2180–2188, Red Hook, NY, USA, 2016. Curran Associates Inc. 3

[5] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1, 5

[6] Edo Collins, Raja Bala, Bob Price, and Sabine Süsstrunk. Editing in style: Uncovering the local semantics of gans. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5770–5779, 2020. 2, 3

[7] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12868–12878, 2021. 2, 6

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014. 1, 2, 3

[9] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 1

[10] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 2, 3

[11] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017. 5

[12] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017. 2

[13] Ali Jahanian, Lucy Chai, and Phillip Isola. On the "steerability" of generative adversarial networks, 2020. 2, 3

[14] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. 1, 3

[15] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Proc. NeurIPS*, 2020. 1, 5

[16] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Proc. NeurIPS*, 2021. 1, 5

[17] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3, 4, 5, 6, 8

[18] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020. 3, 5

[19] Oran Lang, Yossi Gandelsman, Michal Yarom, Yoav Wald, Gal Elidan, Avinatan Hassidim, William T. Freeman, Phillip Isola, Amir Globerson, Michal Irani, and Inbar Mosseri. Explaining in style: Training a gan to explain a classifier in stylespace. *arXiv preprint arXiv:2104.13369*, 2021. 2, 3

[20] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. *arXiv preprint arXiv:2203.01941*, 2022. 2

[21] Bingchen Liu, Yizhe Zhu, Kunpeng Song, and Ahmed Elgammal. Towards faster and stabilized {gan} training for high-fidelity few-shot image synthesis. In *International Conference on Learning Representations*, 2021. 1, 5

[22] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015. 6

[23] Weili Nie, Arash Vahdat, and Anima Anandkumar. Controllable and compositional generation with latent-space energy-based models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 13497–13510. Curran Associates, Inc., 2021. 2, 3

[24] Evangelos Ntavelis, Andrés Romero, Iason Kastanis, Luc Van Gool, and Radu Timofte. SESAME: Semantic Editing of Scenes by Adding, Manipulating or Erasing Objects. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 394–411, Cham, 2020. Springer International Publishing. 1

[25] Evangelos Ntavelis, Mohamad Shahbazi, Iason Kastanis, Radu Timofte, Martin Danelljan, and Luc Van Gool. Arbitrary-scale image synthesis. In *2022 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2022*, 2022. 1

[26] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2

[27] 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 H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 5

[28] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery, 2021. 1

[29] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. 3

[30] Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *ArXiv*, abs/1906.00446, 2019. 2

[31] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1060–1069, New York, New York, USA, 20–22 Jun 2016. PMLR. 1

[32] Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *ACM Trans. Graph.*, 2021. 8

[33] Axel Sauer, Kashyap Chitta, Jens Muller, and Andreas Geiger. Projected gans converge faster. In *NeurIPS*, 2021. 5

[34] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. volume abs/2201.00273, 2022. 1

[35] Edgar Schönfeld, Vadim Sushko, Dan Zhang, Juergen Gall, Bernt Schiele, and Anna Khoreva. You only need adversarial supervision for semantic image synthesis. In *International Conference on Learning Representations*, 2021. 1

[36] Mohamad Shahbazi, Martin Danelljan, Danda Pani Paudel, and Luc Van Gool. Collapse by conditioning: Training class-conditional GANs with limited data. In *International Conference on Learning Representations*, 2022. 1, 8

[37] Mohamad Shahbazi, Zhiwu Huang, Danda Pani Paudel, Ajad Chhatkuli, and Luc Van Gool. Efficient conditional gan transfer with knowledge propagation across classes. In *2021 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021*, 2021. 1

[38] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NIPS*, 2017. 2

[39] Andrey Voynov and Artem Babenko. Unsupervised discovery of interpretable directions in the gan latent space. In *International Conference on Machine Learning*, pages 9786–9796. PMLR, 2020. 2, 3

[40] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1

[41] Tom White. Sampling generative networks, 2016. 2, 3

[42] Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12863–12872, June 2021. 2, 3, 6

[43] Taihong Xiao, Jiapeng Hong, and Jinwen Ma. Dna-gan: Learning disentangled representations from multi-attribute images. *International Conference on Learning Representations, Workshop*, 2018. 3

[44] Taihong Xiao, Jiapeng Hong, and Jinwen Ma. Elegant: Exchanging latent encodings with gan for transferring multiple face attributes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–187, September 2018. 3

[45] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. *arXiv preprint arXiv:1711.10485*, 2017. 1

[46] Ceyuan Yang, Yujun Shen, and Bolei Zhou. Semantic hierarchy emerges in deep generative representations for scene synthesis. *International Journal of Computer Vision*, 2020. 2, 3

[47] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 5

[48] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved vqgan, 2021. 6

[49] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 8

[50] 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. In *CVPR*, 2021. 3

[51] Yang Zhao, Chunyuan Li, Ping Yu, Jianfeng Gao, and Changyou Chen. Feature quantization improves GAN training. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11376–11386. PMLR, 13–18 Jul 2020. 3

[52] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. Sean: Image synthesis with semantic region-adaptive normalization. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5103–5112, 2020. 1