STYLEGENES: DISCRETE AND EFFICIENT LATENT DISTRIBUTIONS FOR GANS

Anonymous authors
Paper under double-blind review

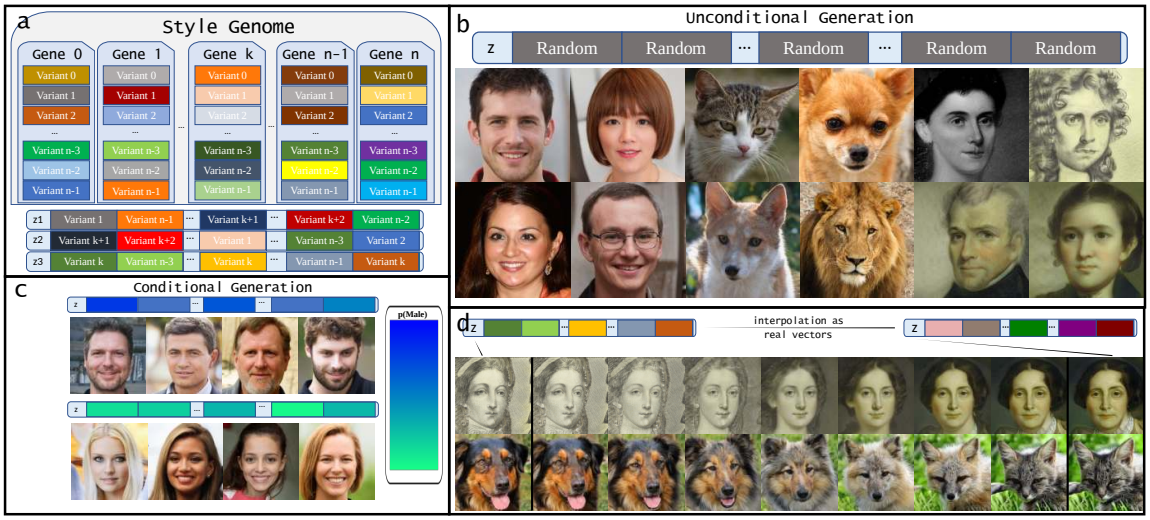

Figure 1: We propose *StyleGenes*: a biologically inspired discrete latent distribution for GANs. We train for and perform unconditional image synthesis (b) by randomly sampling a *variant* for each *gene* (a). Our approach allows for conditional generation by finding the genes that are associated with specific attributes (c). Although our latent distribution is discrete, the learned style space offers emergent continuous properties, ensuring smooth interpolation between samples (d).

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 classifier-based conditional sampling. 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 their first appearance in the seminal work by Goodfellow et. al (10). GANs have been successfully applied to a plethora of tasks, including conditional generation from semantic categories (3; 40; 39), images (7; 43), text (34; 48; 31), and semantic layouts (29; 54; 27; 38). Compared to their early predecessors, recent GANs (18; 37; 28; 4) 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 (16; 19; 17; 24; 11).

The core formulation of GANs, however, has remained largely the same: a generator transforms a latent code *sampled from a continuous distribution* to a realistic-looking image. Initially, the latent code was sampled by a uniform distribution (10). Quickly, however, the community converged to using a Gaussian prior (44; 12). An important change came subsequently when Karras *et al* (19) 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 (29) or a style transfer (14) generator, it was used to manipulate the intermediate feature maps after the convolutions. Nevertheless, the Gaussian input is mapped to an intermediate *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.

In this work, we take a different approach 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.

Our contributions are summarized as follows. **A:** We introduce a compact parameterization of a discrete latent distribution for GANs, inspired by the encoding of information in biological organisms. **B:** We exploit the discrete nature of the latent space to analyze the association of genes to semantic image attributes and develop methods for conditional generatio based on our gene analysis. **C:** We show that our learned discrete latent space is more disentangled that the widely-used Style mapping outputs. **D:** We demonstrate that despite the discrete latent distribution, the resulting style space obtains continuous properties, important for e.g. realistic interpolation. **E:** We 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 less parameters while yielding a more disentangled latent space.

2 RELATED WORK

Latent Code Quantization: VQ-VAE (41) 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 (41). 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 (33) 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 (9). 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 (23) 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 (53). 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 (20), 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 (6) 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 (46) and ELEGANT (47) 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 (15; 49; 42). 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 (45; 15). Moreover, (22) 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 (52). Recent methods utilize unconditionally pretrained models for conditional generation (2; 26). These approaches, train a conditional normalizing flow (2) or a classifier (26) in the latent space to enable conditional sampling.In this study, we do not need to train one vector per transformation (15), compute any gradients(45) or apply clustering to hidden layers (8). 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 (20; 16) image generation paradigm, a latent vector sampled from a *continuous* multi-variate prior distribution (10) 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 using as the 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,

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

where z_c is sampled from a prior distribution p_z, and θ are the generator's weights. Early works (10; 32) sample z_c from a uniform distribution. Subsequent works (44; 12) sample from a standard Gaussian distribution. Since the introduction of StyleGAN (19) 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 $\mathbb{W}$.

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

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 (19)).

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,

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

Here, v_i^j denotes the variant j of position i. The vector k of uniformly sampled indices k_i selectes the variant $v_i^{k_i}$ for each position i. The dimensionality of k is the number of positions, n_g, in our codebook. The number of variants for each positionis 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 let 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 1.34×10^{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.

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, editing, and conditional sampling, by aggregating statistics of how a set of image-specific attributes relate to individual elements in the codebook.

Let $\{a_1, \ldots, a_L\}$ denote the of 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 $p(k|a_1, \ldots, a_L)$. We assume the positions to be conditionally independent $p(k|a_1, \ldots, a_L) = \prod_i p(k_i|a_1, \ldots, a_L)$. We then obtain,

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

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).

Eq. 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 estimate the latter by aggregating statistics over generated image samples as

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

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.

4 EXPERIMENTS

Implementation Our method, StyleGenes, is written in Pytorch (30). We incorporate our sampling approach into two baseline models: (1) StyleGAN2 (20) as provided in the StyleGAN3 (18) codebase and Project-

Unsupervised Image Generation

FID ↓	FFHQ	AFHQ	Met/s	Church	Beds
StyleGAN2					
StyleMapping	5.3	**5.62**	**20.48**	8.13	51.61
StyleGenes	**5.11**	5.99	21	**6.86**	**17.84**
ProjectedGANs(FastGAN)					
Cont. Prior	5.08	4.02	15.38	**3.05**	3.15
StyleGenes	**4.19**	**3.66**	**15.24**	3.08	**2.96**

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: **Left:** Evaluation of our discrete sampling approach, *StyleGenes*, by substituting the StyleGAN2's StyleMapping network or FastGAN's continuous sampling for ProjectedGAN. Our method yields similar or better FID to the continuous case. **Right:** Ablation on different configurations of the genome and our baseline. Increasing the size of the Genome (*# Variants*) increases both its capacity and performance. 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 its memory footprint.

edGANs (36) using the FastGAN (24) 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 (18; 36). For small datases Metfaces (17) and AFHQ (7) we use adaptive discriminator augmentation (17). 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(19) and 2M images for the other datasets.

Datasets We investigate the performance of our network using the *Fréchet Inception Distance (FID)* (13), on widely used datasets for unsupervised image generation:
FFHQ (19), standing for Flickr Faces - High Quality 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 (17) 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. Both FFHQ and Metfaces are licensed under CC BY-NC 2.0 license by NVIDIA Corp.
AFHQ (7) 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. The dataset is available under CC BY-NC 4.0 license by NAVER Corp.
LSUN Church & Bedroom (50). 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-left we can see the results obtained when introduce our discrete sampling technique to established baselines (20; 36). We compare them to the results we get when we train our baselines (17; 36) with the same hyperparameters and training images, Our proposed discrete method produces similar results with StyleGAN's StyleMapping approach, and improves ProjectedGAN when it replaces gaussian sampling.

In ablation study (Table 1-right), we aim to analyze the effect of the different Genome configurations to the perceptual performance of the network. Note that by keeping the number of the variants constant, the memory footprint of our approach is also constant. Thus, every experiment that is in the same row in Table 1-right is using the same number of parameters that scale with the number of variants: $n_v * d$. Increasing this number improves the perceptual quality in terms of FID, an effect more prominent for larger genes. We call this an increase of *parameter capacity*. Alternatively, by increasing the number of genes, we also observe a decrease in FID. Note that as we want to keep the size of the resulting latent vector constant $d = 512$, we decrease each gene's length when using more. Therefore, the number of different images the genome can represent

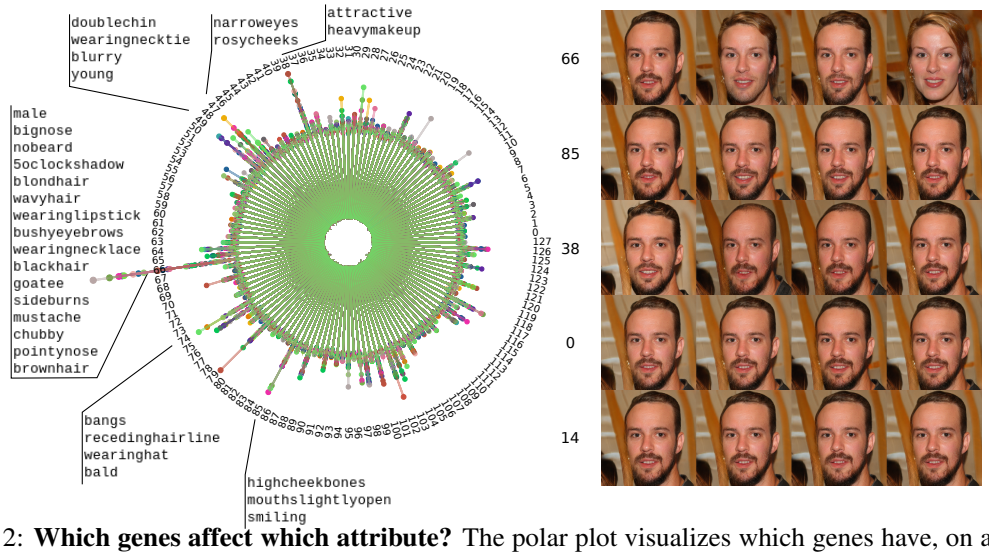

Figure 2: **Which genes affect which attribute?** The polar plot visualizes which genes have, on average, extreme (high or low) expected values towards one of the 40 CelebA attributes. Each color represents a different attribute. The labels indicate the gene that exhibits the highest variability in this 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 show extreme values for any attribute, leads to minute changes. Most genes fall into this category.

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 the presence of an attribute in a generated image from its latent code. We find it is much easier to associate our StyleGenes' codes to attributes, than with the StyleGAN's style codes.

is also increasing, per Eq. equation 5. We call this an increase of *combinatorial capacity*. Increasing the combinatorial capacity does not increase the parameters of the genome. 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.

4.2 Analyzing the Codebook

In this section we aim to analyze the properties that arise due to the discrete nature of our approach. Moreover, we show how to use them for conditional generation in FFHQ.

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 (25). Their weights are included in the original StyleGAN (19) code repository, and we used the code provided by StyleSpace (45) to extract the logits for every image.

Conditional Generation In the previous step we acquired the marginal attribute distribution for a given genome variant. We use this information to conditionally generate an image with a desired attribute a_l. To

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 3: **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. We can control a FID-accuracy trade-off using the temperature. Lower values decrease variability but increase accuracy. For our baseline, a conditional StyleGAN, we need to provide values for every attribute. We either sample them *randomly* or use the real dataset's conditional *frequencies*.

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 3 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. This effect is outline in increased FID scores in Table 3

To gauge the ability of our method to generate conditionally, we train a conditional StyleGAN2 model with pseudo-labels from the CelebA classifiers. In Table 3, we find we compare similarly to our baseline. However, we do not require a predefined number of classes and our method can be extended to more classes without training. Moreover, we can use the temperature value to control the trade-off between variability and accuracy. Lastly, training conditionally with a small dataset can lead to poor performance and mode collapse (39).

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 quantify this by calculating the *mean absolute standard score* for each gene position: the absolute distance in terms of standard 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

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 (35) to add real images in our genome.

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.

StyleGenome and Disentanglement The StyleGAN's (19) 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 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-bottom.

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 (35) approach 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 want to invert. Then, we find the indices of the nearest-neighbor variant for each gene position 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 regularization: we push randomly sample gene-variants to keep their syntheses unchanged via an l2 and LPIPS (51) loss. We find this step important to retain the perceptual quality of the codebook.

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 and conditional sampling. Lastly, we show that we can generate samples between the genome's discrete elements, indicating that the samples are on a smooth style surface.

6 ETHICAL DISCUSSION

Our methodology permits exploiting the discrete latent code of an image generator trained in an unsupervised manner, to generate conditionally and manipulate the output samples. However, it comes with some limitations. First, while GANs aim to mimic the distribution of real images there is a gap between the real and fake distribution (21). We apply a classifier trained on real images to infer the labels of fake images. The classifiers are limited by the entanglements of attributes in the training dataset. This issue is also discussed in StyleSpace (45), where they note that the classifier may fail to predict a lipstick on a male face. We hypothesize that this effect is further intensified because, while FFHQ offers significant diversity in "age, ethnicity and accessories", CelebA contains celebrity faces and thus is more limited in those factors. Lastly, the biases of the labelers of the images can be propagated to the conditional image generation and manipulation results (See *attractive* in Figure 3. Therefore, one should be wary to apply this approach to a real life application.

Please not that in this work we train our model using portraits of real people, using the Flickr-Faces-HQ dataset (19). As described in *https://github.com/NVlabs/ffhq-dataset*, the images were collected to adhere to privacy rules and were filtered to only include samples intented for redistribution. Moreover, if an individual identifies themselves in the dataset, they can request the removal of their face from the collection.

REFERENCES

[1] R. Abdal, Y. Qin, and P. Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4431–4440, 2019.

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

[3] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[4] E. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *arXiv*, 2020.

[5] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. D. Mello, O. Gallo, L. Guibas, J. Tremblay, S. Khamis, T. Karras, and G. Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *CVPR*, 2022.

[6] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. 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.

[7] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[8] E. Collins, R. Bala, B. Price, and S. 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.

[9] P. Esser, R. Rombach, and B. Ommer. Taming transformers for high-resolution image synthesis. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12868–12878, 2021.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. 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.

[11] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In I. Guyon, U. V. 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.

[12] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In I. Guyon, U. V. 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.

[13] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In 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.

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

[15] A. Jahanian, L. Chai, and P. Isola. On the "steerability" of generative adversarial networks, 2020.

[16] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[17] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila. Training generative adversarial networks with limited data. In *Proc. NeurIPS*, 2020.

[18] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila. Alias-free generative adversarial networks. In *Proc. NeurIPS*, 2021.

[19] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[20] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020.

[21] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. Improved precision and recall metric for assessing generative models. *CoRR*, abs/1904.06991, 2019.

[22] O. Lang, Y. Gandelsman, M. Yarom, Y. Wald, G. Elidan, A. Hassidim, W. T. Freeman, P. Isola, A. Globerson, M. Irani, and I. Mosseri. Explaining in style: Training a gan to explain a classifier in stylespace. *arXiv preprint arXiv:2104.13369*, 2021.

[23] D. Lee, C. Kim, S. Kim, M. Cho, and W.-S. Han. Autoregressive image generation using residual quantization. *arXiv preprint arXiv:2203.01941*, 2022.

[24] B. Liu, Y. Zhu, K. Song, and A. Elgammal. Towards faster and stabilized {gan} training for high-fidelity few-shot image synthesis. In *International Conference on Learning Representations*, 2021.

[25] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

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

[27] E. Ntavelis, A. Romero, I. Kastanis, L. Van Gool, and R. Timofte. SESAME: Semantic Editing of Scenes by Adding, Manipulating or Erasing Objects. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 394–411, Cham, 2020. Springer International Publishing.

[28] E. Ntavelis, M. Shahbazi, I. Kastanis, R. Timofte, M. Danelljan, and L. Van Gool. Arbitrary-scale image synthesis. In *2022 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2022*, 2022.

[29] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[31] O. Patashnik, Z. Wu, E. Shechtman, D. Cohen-Or, and D. Lischinski. Styleclip: Text-driven manipulation of stylegan imagery, 2021.

[32] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[33] A. Razavi, A. van den Oord, and O. Vinyals. Generating diverse high-fidelity images with vq-vae-2. *ArXiv*, abs/1906.00446, 2019.

[34] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In M. F. Balcan and K. 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.

[35] D. Roich, R. Mokady, A. H. Bermano, and D. Cohen-Or. Pivotal tuning for latent-based editing of real images. *ACM Trans. Graph.*, 2021.

[36] A. Sauer, K. Chitta, J. Muller, and A. Geiger. Projected gans converge faster. In *NeurIPS*, 2021.

[37] A. Sauer, K. Schwarz, and A. Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. volume abs/2201.00273, 2022.

[38] E. Schönfeld, V. Sushko, D. Zhang, J. Gall, B. Schiele, and A. Khoreva. You only need adversarial supervision for semantic image synthesis. In *International Conference on Learning Representations*, 2021.

[39] M. Shahbazi, M. Danelljan, D. P. Paudel, and L. V. Gool. Collapse by conditioning: Training class-conditional GANs with limited data. In *International Conference on Learning Representations*, 2022.

[40] M. Shahbazi, Z. Huang, D. P. Paudel, A. Chhatkuli, and L. Van Gool. Efficient conditional gan transfer with knowledge propagation across classes. In *2021 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021*, 2021.

[41] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. In *NIPS*, 2017.
[42] A. Voynov and A. Babenko. Unsupervised discovery of interpretable directions in the gan latent space. In *International Conference on Machine Learning*, pages 9786–9796. PMLR, 2020.
[43] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. 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.
[44] T. White. Sampling generative networks, 2016.
[45] Z. Wu, D. Lischinski, and E. 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.
[46] T. Xiao, J. Hong, and J. Ma. Dna-gan: Learning disentangled representations from multi-attribute images. *International Conference on Learning Representations, Workshop*, 2018.
[47] T. Xiao, J. Hong, and J. 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.
[48] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. *arXiv preprint arXiv:1711.10485*, 2017.
[49] C. Yang, Y. Shen, and B. Zhou. Semantic hierarchy emerges in deep generative representations for scene synthesis. *International Journal of Computer Vision*, 2020.
[50] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.
[51] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.
[52] Y. Zhang, H. Ling, J. Gao, K. Yin, J.-F. Lafleche, A. Barriuso, A. Torralba, and S. Fidler. Datasetgan: Efficient labeled data factory with minimal human effort. In *CVPR*, 2021.
[53] Y. Zhao, C. Li, P. Yu, J. Gao, and C. Chen. Feature quantization improves GAN training. In H. D. III and A. 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.
[54] P. Zhu, R. Abdal, Y. Qin, and P. Wonka. Sean: Image synthesis with semantic region-adaptive normalization. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5103–5112, 2020.

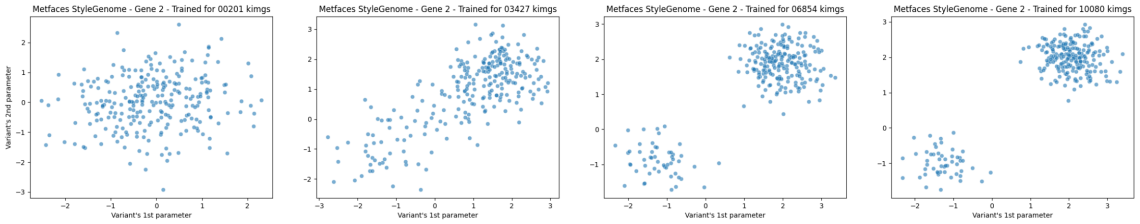

Figure 4: We can observe how the density of the discrete latent distribution is changing through training. The variants' values are learned through the adversarial game to produce images, along with the synthesis network. that match the real distribution.

A SUPPLEMENTARY MATERIAL

A.1 EVOLUTION OF STYLEGENOME DURING TRAINING

As we discussed in Section 4., the StyleGAN's (19) 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. In Table 2-top we can see how the adversarial game is altering the density of our discrete distribution throughout the training. Our learnable embeddings are aligning, together with the Synthesis network, to better match the real images distribution.

A.2 PROJECTING A REAL IMAGE VIA LATENT OPTIMIZATION

In the main paper we discussed how the discrete nature of our proposed *StyleGenome* enables us to conditionally generate and manipulate images. Even if the genome has a finite number of images it can generate, this number is large enough to produce countless different samples. However, it is interesting to approach the inverse problem. We have a specific image, how can we *project* it to the latent space?

In the main paper we have shown that we can generate realistic samples using real vectors in between two of our gene sequences. Similarly, we tackle the task of projection to the latent space as a continuous optimization problem. In contrast to StyleGAN's(19) projection (1), where a set of samples from a gaussian distribution is passed through the mapping network and then averaged, we average the real values of randomly selected gene sequences. In the first two columns of Figure 5, we can see the real images and our method's projections. We optimize the style vectors modulating each layer of the synthesis network (W^+space) (1). The vector is optimized in order to minimize mean squared error and the perceptual distance (51) between the synthesized image and the original image we want to project.

Training with our discrete set of genes we are able to learn a dense continuous space that enables projection of images to the level shown in Figure 5. As we can see in the rest of the columns of the figure, however, the closest variants to the real sub-vectors producing the projected image, can not recreate the original well. In Figure 5, we show three different approaches for computing the distance of the projected latent code's sub-vectors to the genome's variants: Manhattan distance, euclidean, and cosine similarity.

In practise we use this approach to initiate our Codebook inversion, as described in Section 4

A.3 IMAGE MANIPULATION

Using the computed expected values, we can create an ordering of variants for each attribute. We aim to manipulate generated samples towards an attribute. For example, we would like to change the color of a generated person's hair. We start from a sampled gene sequence and substitute the variants of the most critical

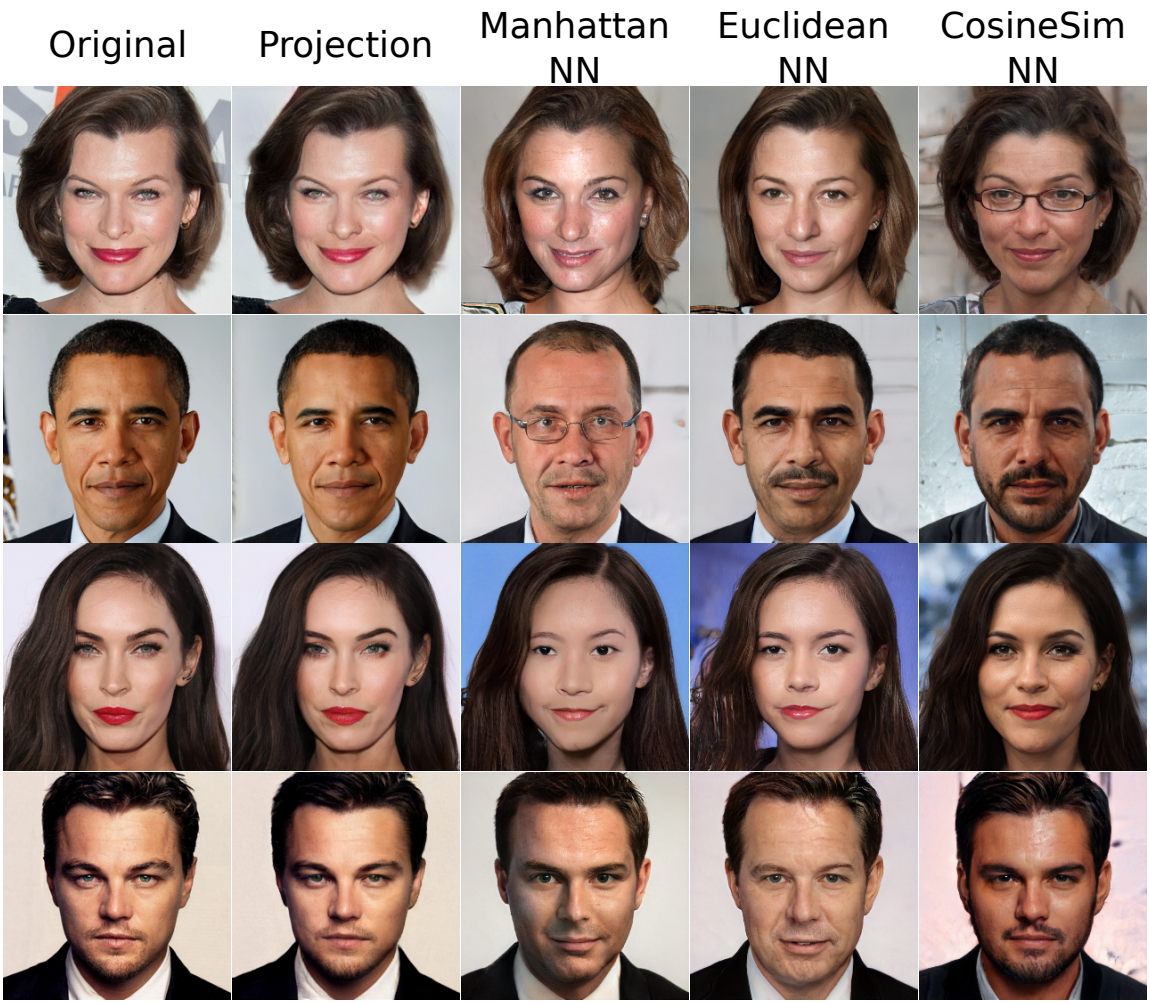

Figure 5: **Image Inversion with latent code optimization.** While projecting specific faces to our discrete latent space cannot guarantee good results. Optimizing in our underlying continuous space is able to produce projection samples of high quality and fidelity. In the three last rows we find the closest variants of each gene to the real projected vector, using different distance functions. We use this us an intermediate step for our codebook inversion, explained in Section 4.

genes with ones that exhibit higher/lower attribute expectations. In Figure 6 we showcase such an application. Moreover, while our discrete representation is by design unordered (shuffling the variants will not affect our method), we can create an ordering based on an attribute's expected value. After locating the position of the original variant, we can control the degree to which the attribute will be expressed by traversing the ordered collection.

Figure 6: **Image Manipulation** First we acquire the most significant genes for each CelebA attribute. We can edit an images by substituting the variants of the significant genes with ones having a low (left side) or high (right side) expectation of a certain attribute.

Variant Pruning - FID↓							
0	8	16	32	64	128	256	512
5.873 ± 0.060	5.814 ± 0.061	5.787 ± 0.024	5.784 ± 0.037	**5.760** ± 0.047	5.909 ± 0.052	6.114 ± 0.044	6.270 ± 0.045

Table 4: Using our discriminator as a heuristic, we compute the expected value of *realness* for each variant. We use these values to substitute the most fake variants with duplicates of the most real ones. The number on the top of each columns shows how many variants are pruned. Our pruning approach can decrease the FID. Pruning too much, however, decreases the size of the genome and can increase the score. We report the FID computed with 50k images, averaged over five runs.

A.4 Pruning the genome

There is another common technique applied to continuous latent spaces of GANs that we have not addressed: the truncation trick(3). As we have an unordered set of variants, applying the trick is not straight-forward. However we develop a technique to limit the erroneous samples that our model can synthesize. Using the discriminator as a heuristic, we replicate the process we follow in the main paper to produce the expected value of an attribute. Similarly, we derive the expected value of *realness* of a particular variant as the average discriminator output over a set of samples that contain this variant.

As with the truncation trick, we limit the number of samples the network can generate by removing certain variants off the *gene pool*. In Table 4 we compute the FID score for different genome scenarios. We remove x number of variants that exhibit the lowest *realness* in expected values and substituted them with the ones that exhibit the highest. The number on the top of each column denotes this number for each scenario. The configuration we used for this experiment has 1024 variants per gene, making the scenario of the rightmost column have half the size of the genome of the leftmost one.

We find out that our pruning approach can increase the perceptual quality of the method in terms of FID. However, we hypothesize that limiting the number of variants to a larger degree can decrease the variability of the generated samples and hurt the score. Note, that we use the discriminator's score as an easy heuristic. However, samples produced using variants with extreme fake expected values are not necessarily erroneous, nor the ones with the realest values are guaranteed to be perceptually good. However, on average they produce a better FID score as shows in Table 4.

Figure 7: **Image Generation on FFHQ** 1024×1024 **(19)** using *StyleGenes* along with a StyleGan2 synthesis network. All images were produced by gene sequences selected at random and without cherry-picking.

A.5 Different configurations

In the main paper we show results for three different datasets for images generated at 256×256 resolution using the StyleGAN2(20; 17) synthesis network.

We also train our *StyleGenes* approach for FFHQ images of resolution 1024×1024 for 2 million images seen by the discriminator. Similarly with our main paper experiments, our results, with an FID score of 7.60 are on par with the mapping network's FID of 7.61. In Figure 7 we can observe results generated by our approach.

For a second experiment, we substitute the StyleGAN2 backbone with StyleGAN3-T (18) and train for the Metfaces dataset. Again, we observe similar FID scores for our method (27.17) and the standard approach (27.44)). We trained both networks for 6 million images per discriminator. We can see the samples synthesized by StyleGenes in Figure 8.

Lastly, we also train for 3d-aware image synthesis using EG3D (5) for the cats sub-dataset of AFHQ. We train the method we both its original StyleMapping approach and our StyleGenes, using the same setting. We get an fid score of 4.17 for the baseline and 4.15 for our approach.

A.6 The benefits of the mapping network

In the nominal work of the first StyleGAN (19), the authors motivate the design of the mapping network as a mean to unwarp the gaussian prior in a way that permits only sampling from valid combinations of attributes. Moreover, they argue that a benefit of the newly acquired latent space is that it does not follow a predetermined distribution and it can learn its own sampling density.

Our approach also shares these benefits. Having a set of learnable discrete samples means that these *move* during training, and can alter their density. While we uniformly sample the variant of each gene, training pushes these variants to be closer or further apart. Moreover, using pretrained CelebA(25) classifiers we can quantify the correlation of certain attributes. In Figure 10 we see the Pearson correlation computed over the

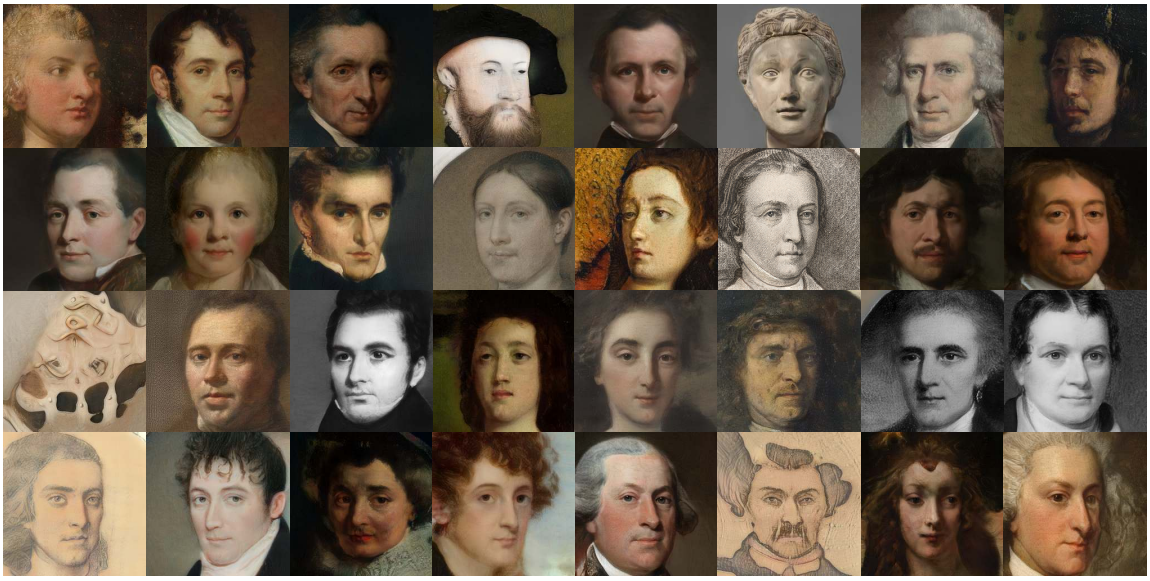

Figure 8: **Generation on Metfaces (17)** using *StyleGenes* along with a ***StyleGan3*** synthesis network. All images were produced by gene sequences selected at random and without cherry-picking.

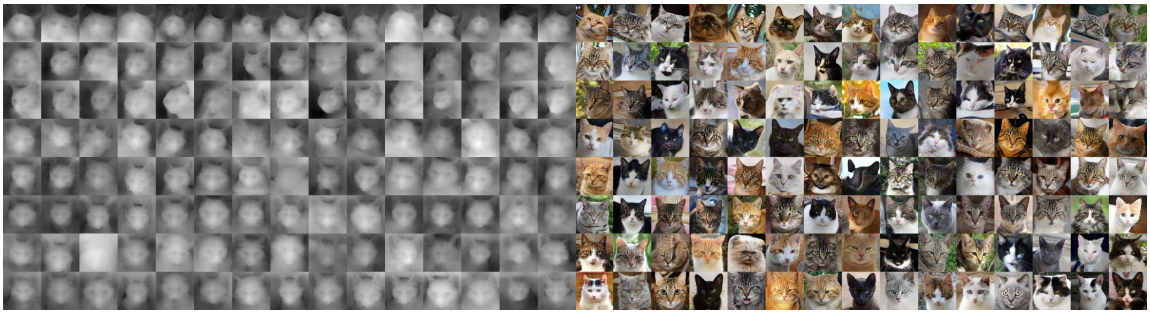

Figure 9: **Generation on Cats (17)** using *StyleGenes* along with a ***EG3D*** (5) synthesis network. All images were produced by gene sequences selected at random and without cherry-picking.

vector of classifiers' outputs for 50 thousand images. We show results for real images of the FFHQ dataset, as well as generated images from a StylegGAN using the mapping network and our approach. We can observe that the correlation values are similar between our proposed method and the mapping network.

A.7 More visual results.

In the following figures we provide additional results:

- In Figure 12 we present more unconditional samples for FFHQ (19).
- In Figure 13 we present more unconditional samples for AFHQv2 (7).
- In Figure 14 we present more unconditional samples for Metfaces (17).
- In Figure 15 we present more conditionally generated samples for FFHQ (19).

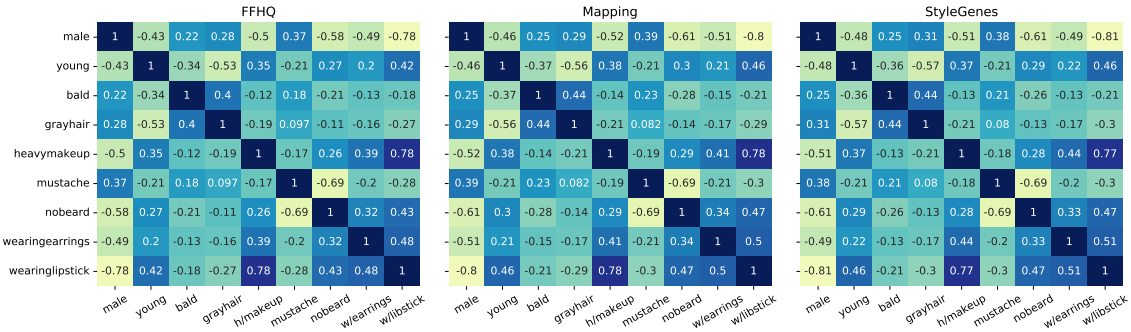

Figure 10: **Attribute Correlation.** One motivation behind the design of the mapping network is to forbid the sampling of invalid combinations, that are not present in the original dataset. By using pretrained classifiers, we show that the correlation between the certain attributes is similar between real FFHQ images and those produced by both the mapping network and our discrete sampling method.

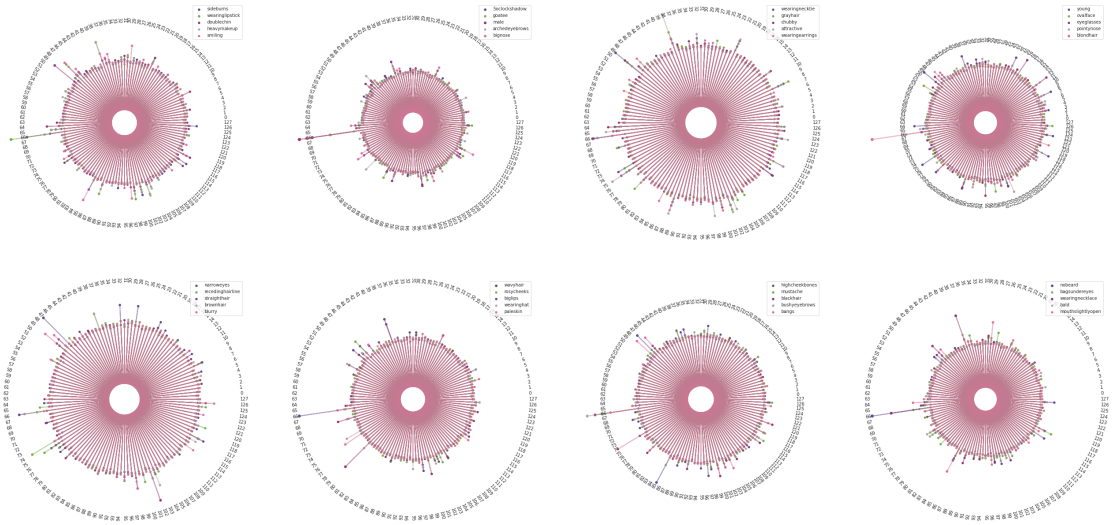

Figure 11: **Attributes' absolute standard scores.** We are presenting a clearer version of Figure 2 of the main paper. One can easily see that for most attributes only specific genes produce high variability. These are the genes we use to conditionally generate and manipulate samples. Most genes are only utilized for small changes in the images on their own, but combining them can create diverse outputs. Please zoom for a more detailed view.

Figure 12: **Unconditional generation on FFHQ.** All images were produced by gene sequences selected at random and without cherry-picking.

Figure 13: **Unconditional generation on AFHQv2.** All images were produced by gene sequences selected at random and without cherry-picking.

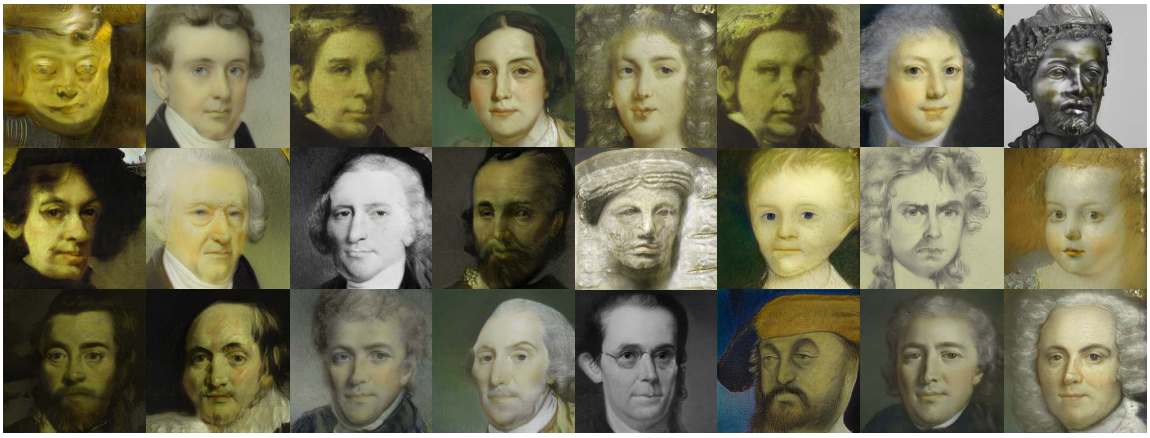

Figure 14: **Unconditional generation on Metfaces.** All images were produced by gene sequences selected at random and without cherry-picking.

Figure 15: **Conditional generation on FFHQ.** Results for more attributes on conditional generation using our analysis with pretrained CelebA classifiers

B GENOME CODEBASE

In this chapter we share the sampling code of *StyleGenes*. Using the StyleGAN3 (18) codebase: `https://github.com/NVlabs/stylegan3`, adding this function will enable training with our approach.

```
# Modified from:
# github.com/NVlabs/stylegan3/
class Genome(torch.nn.Module):
    def __init__(self,
        dim,  # style latent dimensionality.
        num_ws, # number of W copies used to modulate the synthesis network
        num_variants = 2048,
        gene_length = 8,
    ):
        super().__init__()
        self.dim = dim
        self.num_ws = num_ws

        #Discrete genome
        self.gene_length = gene_length
        assert (self.dim % self.gene_length) == 0
        self.num_genes = self.dim // self.gene_length
        self.num_variants = num_variants
        genes = torch.randn((self.num_genes, \
                             self.num_variants, \
                             self.gene_length), \
                             requires_grad=True)
        self.genes = torch.nn.Parameter(genes)

    # We use the same arguments as the Mapping network
    # in order to seamlessly substitute the function call
    def forward(self, z, \
                      c, \
                      truncation_psi=1, \
                      truncation_cutoff=None, \
                      update_emas=False):

        # We randomly select one variant per gene
        # to create the gene sequence
        ws = torch.stack([ \
                self.genes[
                    range(self.num_genes), \
                    list( \
                        torch.randint( \
                            size=(self.num_genes,), \
                            high=self.num_variants) \
                    ) \
                ].view(-1) for _ in range(z.shape[0]) \
            ], dim=0) # z.shape[0] = batch size

        ws = ws.unsqueeze(1).repeat([1, self.num_ws, 1])
```

```
    return ws
```