
CGS-GAN: 3D Consistent Gaussian Splatting GANs for High Resolution Human Head Synthesis

Florian Barthel
Fraunhofer HHI
HU Berlin

**Wieland
Morgenstern**
Fraunhofer HHI

Paul Hinzer
Fraunhofer HHI

Anna Hilsmann
Fraunhofer HHI

Peter Eisert
Fraunhofer HHI
HU Berlin



Figure 1: Example renderings at 1024^2 resolution produced by our 3D *Consistent Gaussian Splatting GAN* (CGS-GAN). Unlike prior methods that recompute the scene for each individual view to ensure high quality, our method is capable of preserving quality while synthesizing a fully 3D consistent scene that can be ported into explicit 3D settings like game engines or VR environments.

Abstract

Recently, 3D GANs based on 3D Gaussian splatting have been proposed for high quality synthesis of human heads. However, existing methods stabilize training and enhance rendering quality from steep viewpoints by conditioning the random latent vector on the current camera position. This compromises 3D consistency, as we observe significant identity changes when re-synthesizing the 3D head with each camera shift. Conversely, fixing the camera to a single viewpoint yields high-quality renderings for that perspective but results in poor performance for novel views. Removing view-conditioning typically destabilizes GAN training, often causing the training to collapse. In response to these challenges, we introduce CGS-GAN, a novel 3D Gaussian Splatting GAN framework that enables stable training and high-quality 3D-consistent synthesis of human heads without relying on view-conditioning. To ensure training stability, we introduce a multi-view regularization technique that enhances generator convergence with minimal computational overhead. Additionally, we adapt the conditional loss used in existing 3D Gaussian splatting GANs and propose a generator architecture designed to not only stabilize training but also facilitate efficient rendering and straightforward scaling, enabling output resolutions up to 2048^2 . To evaluate the capabilities of CGS-GAN, we curate a new dataset derived from FFHQ. This dataset enables very high resolutions, focuses on larger portions of the human head, reduces view-dependent artifacts for improved 3D consistency, and excludes images where subjects are obscured by hands or other objects. As a result, our approach achieves very high rendering quality, supported by competitive FID scores, while ensuring consistent 3D scene generation. Check out our project page here: <https://fraunhoferhhi.github.io/cgs-gan/>

1 Introduction

Synthesizing high-quality 3D human heads remains a major challenge in computer vision with wide applicability in film and gaming industries. However, this often requires expensive 3D capturing studios or high manual labor. For this reason, there is a high demand for automating and simplifying the synthesis process while maintaining high rendering quality. 3D GANs trained on 2D images with known camera poses have emerged as a powerful framework, enabling both generation and manipulation of facial geometry and appearance. Earlier approaches such as 2.5D GANs encode camera viewpoints directly into the generator features, allowing for multi-view synthesis but often lacking consistency across viewpoints.[32, 2, 6, 28]. 3D-aware GANs based on Neural Radiance Fields [4, 26] improve view disentanglement by rendering from an intermediate 3D volume, but suffer from slow inference and limited resolution scalability [38, 7, 8, 3, 31]. More recently, 3D Gaussian Splatting (3DGS) [21] has been integrated into generative models, providing explicit scene representations with real-time differentiable rendering capabilities. Methods like GSGAN [15] and GGHead [23] demonstrate that high-quality 3D Gaussian head synthesis is feasible within GAN frameworks. However, they rely on view-conditioning, where the camera parameters are fused with latent codes to guide synthesis. While this aids training convergence for 3DGS based GANs, it leads to strong view-dependent variations in geometry and identity.

To address this, we introduce CGS-GAN, a novel 3DGS-based generative framework that achieves stable training and 3D-consistent synthesis of human heads without the need for view-conditioning. Unlike existing methods, CGS-GAN achieves very high rendering quality from any camera pose without re-synthesizing the 3D scene for each view. This is made possible through a lightweight multi-view regularization strategy that stabilizes the training and encourages geometric consistency with minimal computational overhead. To support high-resolution synthesis and scalability, we propose an efficient generator architecture that minimizes GPU memory usage while enabling output resolutions up to 2048². Additionally, we apply random background augmentation to mitigate hole artifacts. To underline the capabilities of our new GAN framework, we curate a high-quality dataset derived from FFHQ, including entire heads by recropping from 4K sources, excluding occluded faces, and reduce view-specific biases. Our key contributions are:

- We propose a novel 3DGS-based GAN framework that synthesizes consistent 3D human heads.
- We introduce a multi-view regularization that stabilizes training and promotes 3D consistency.
- We propose a memory-efficient generator design supporting scalable, high-resolution synthesis.
- We apply a random background augmentation method that reduces common artifacts.
- We curate a high-quality FFHQ-based dataset for high-fidelity, 3D consistent head modeling.

2 Related Work

Generative adversarial networks (GANs) were first introduced by Ian Goodfellow in 2014 [13]. Ever since, GANs have received high attention for their real-time photorealistic synthesis capabilities. In its core, GANs are constructed as a non-cooperative game between a generator that synthesizes images from random latent vectors and a discriminator that differentiates between fake images and real images from a training dataset. The training pipeline is differentiable so that the generator can be optimized to produce outputs that the discriminator confuses with real images. As both networks improve during training, the generator converges towards the training data distribution. As synthesizing from random latent vectors creates only limited control, conditional GANs (cGANs) [27] have been introduced as an extension to GANs, allowing annotations to be incorporated into the GAN training framework.

Although in recent years diffusion-based approaches have surpassed the rendering quality of GANs for 2D image synthesis or video generation, GANs still stand out for their efficient real-time editing and rendering capabilities and their clear advantage not to require ground-truth data. The latter is especially important for 3D synthesis, as good 3D ground-truth data is very difficult to obtain. Consequently, current approaches that utilize diffusion based models for synthesizing 3D human heads, either take the 3D ground truth data from a pre-trained 3D GAN [25], or from 2D diffusion models [10], which are very slow to query.

2.1 3D GANs

While early 3D GANs were based on conditional 2.5D GANs [32, 2, 6, 28] or voxel grids [12, 36], there have been drastic improvements since the introduction to *Neural Radiance Field (NeRF)* [4, 26]. NeRFs provide a differentiable renderer that allows for high quality synthesis, by querying a small *Multi-layer Perceptron* MLP to decode color and opacity during ray tracing. By designing a generator that directly produces the weights of a NeRF (MLP), PI-GAN [8] and GRAF [31] were the first works to include a NeRF renderer in a 3D GAN framework. Although the rendering quality was promising, these approaches were slow to render and limited in output resolution. Addressing this, EG3D [7] proposes an efficient 3D GAN architecture that introduces a lightweight tri-plane representation, which allows using 2D features to describe a 3D volume that can be rendered using a very small NeRF MLP. This architecture considerably sped up the training and inference. Nevertheless, EG3D still relied on 2D super-resolution to render in high resolutions after the 3D rendering, which harms the 3D consistency. To achieve better 3D consistency for 3D GANs, several works [34, 9, 37] proposed extensions to EG3D that allow for removing the super-resolution part after the NeRF rendering. This was mainly achieved by moving the super-resolution in front of the NeRF rendering. While this slows down the rendering, we observe considerably better 3D consistency in the rendered 3D heads. Nevertheless, since these methods use NeRF rendering, the generator model always outputs a 2D image of the implicit 3D model, making it difficult to use the results in explicit 3D environments.

2.2 3DGS GANs

With the introduction of 3D Gaussian splatting (3DGS) [21], we obtained a fully explicit method which allows for fast rendering that is qualitatively similar to NeRFs. Ever since, many works have demonstrated very good rendering quality for synthesizing 3D human heads [33, 30, 11, 39]. Often, however, these methods rely on multi-view input data to overfit a single human head. This data is difficult to come by, as it either requires an expensive multi-camera setup [24] or a video of a person remaining perfectly still. As this is not required when training a 3D GAN, which learns 3D geometries from 2D data, there is a high demand for 3DGS based GANs. The first method to achieve 3DGS human head synthesis from a latent space is the GSGAN Decoder [5]. It directly converts tri-plane features from a pre-trained EG3D model into Gaussian splatting scenes, enabling all advantages of GANs (i.e. synthesizing fakes, inverting or editing) to be employed in an explicit 3DGS setting. On the downside, however, the GSGAN Decoder adds computational overhead, loses some detail and always requires a pre-trained EG3D model. The second 3DGS GAN method, GGHead [23], was proposed as the first method to train 3D human heads in a GAN training framework. Similarly to EG3D, GGHead first synthesizes 2D features using a StyleGAN2 backbone. However, instead of aligning the features in tri-plane and sampling the volume with a NeRF decoder, the output feature maps are used as UV-maps on a fixed template head mesh. Such a UV-map is constructed for each of the 3DGS attributes (color, position, scale, opacity, and rotation), where the position UV-map denotes an offset to a fixed position on the template mesh. As a result, GGHead's generator produces a 2D grid of Gaussian primitives that are used to model the surface of a head model. This method has demonstrated strong performance, producing competitive *Fréchet Inception Distance (FID)* results while rendering with explicit 3D Gaussians. The third method to achieve Gaussian splatting GAN training is GSGAN [15]. GSGAN follows a different approach to GGHead, as it does not model the surface on a head mesh, but directly predicts the Gaussian splatting scene using transformer models [35] that are specifically designed for point clouds [14, 40, 29]. Similarly to a StyleGAN architecture, GSGAN produces coarse Gaussian primitives in earlier layers, and adds fine-detail Gaussians in subsequent layers. Finally, all Gaussian primitives from all layers are concatenated into a single 3DGS scene that can be rendered and forwarded to the discriminator. With this configuration, GSGAN finds a good balance between large Gaussians that model smooth regions like skin regions, and small Gaussians that model fine details like hair. To create a relationship between the layers, GSGAN builds a hierarchical data structure, where smaller Gaussian primitives are attached to larger ones from previous layers. As a result, GSGAN produces good quality 3D heads with competitive FID results. For our method, we will use GSGAN as a basis.

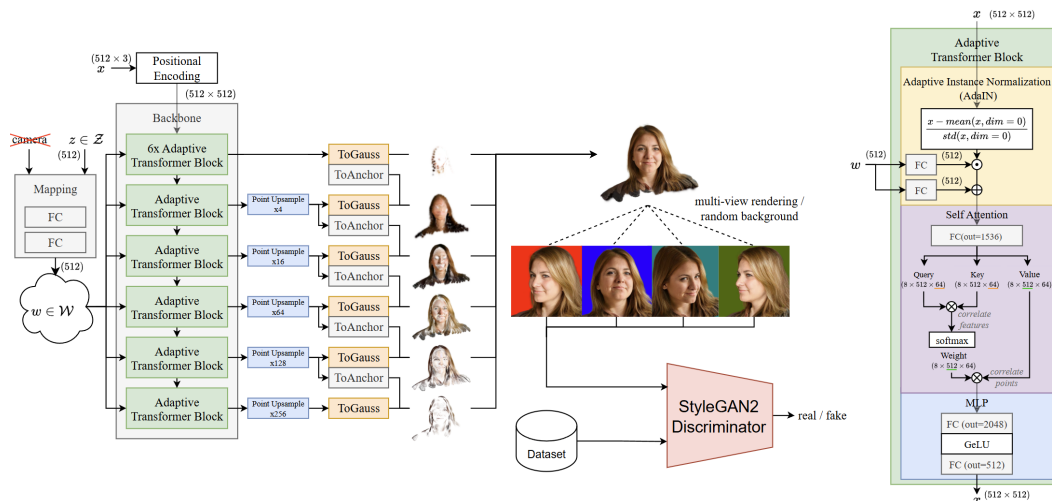


Figure 2: Overview of the proposed CGS-GAN framework. The generator is built on the generator of GSGAN [15] with key modifications for improved 3D consistency and scalability. In contrast to view-conditioned approaches, our method omits camera labels in the mapping network and instead stabilizes training through efficient multi-view rendering of the 3DGS head during each training step. Additionally, we render with random backgrounds to reduce hole artifacts.

3 Method

In this section, we present the core components of CGS-GAN, our novel framework for high-fidelity, 3D-consistent synthesis of human heads using 3D Gaussian Splatting. Our method is designed to achieve stable training without relying on view-conditioning, while enabling efficient and scalable rendering. We first introduce our tailored generator architecture, which supports efficient high quality and high resolution rendering. We then describe our multi-view regularization that enforces consistency across viewpoints during training. In addition, we propose a modified loss formulation adapted to our unconditional setting and introduce a background augmentation technique that mitigates hole artifacts and improves the geometric quality of the resulting 3D scenes.

3.1 Generator Architecture

Our generator architecture is designed to enable high-resolution, 3D-consistent head synthesis while maintaining efficiency and scalability. We use a hierarchical generator structure similar to GSGAN [15]. We first create a constant learnable feature point cloud of size (512×3) that is randomly initialized and forwarded to a positional encoding function that adds higher frequencies as in NeRF [26], resulting in size (512×512) . The encoded representation is then processed by a series of *Adaptive Transformer Blocks* from GSGAN, where each block is composed of an *Adaptive Instance Normalization Layer (AdaIN)*, a *Self Attention Layer*, and a small *Multi-Layer Perceptron (MLP)*. As shown in Figure 2 (right), the AdaIN layer normalizes the input features w.r.t. mean and standard deviation and applies a learned offset and scale, derived from a mapped latent vector $w \in \mathcal{W}$. Intuitively, this operation applies the *style* of a latent vector to the learned feature point cloud representation. This idea originates from the well established StyleGAN [19] framework, known for its state-of-the-art rendering capabilities. The subsequent Self Attention layer enables the model to establish meaningful correlation between the points. Finally, we apply a small MLP to create additional capacity [35]. From the resulting feature points, we decode a 3DGS scene by forwarding the points through fully connected layers with respective output channels for each attribute (position: 3, color: 3, scale: 3, rotation: 4 and opacity: 1) and apply tanh to clip the output to a reasonable range, while keeping differentiability. This is done for Gaussians that are rendered and Gaussians that are used as anchors or offsets for subsequent layers. To create the next layer of Gaussians, we again apply an adaptive transformer block to the feature point cloud of the previous layer and convert the result to a 3DGS scene that is offset and orientated by the anchors of the prior layer. This time, we apply point-upsampling to create four times more Gaussians than in previous layer.

Notably, unlike GSGAN, we do not apply point upsampling within the backbone network. Instead, we increase the upsampling ratio for each subsequent layer. This is a very important aspect when it comes to scaling the generator for larger resolutions. Since, with our generator the number of points in the backbone is fixed, we can flexibly add additional layers to our generator, while scaling linearly in complexity. This is not the case for GSGAN, which performs point upsampling within the backbone. By doing so, GSGAN cannot apply a transformer block to the second (or higher) layer, as the memory consumption would be too high. Instead, GSGAN applies graph convolution layers between neighboring Gaussians, which are very inefficient, as they require sorting distances between all points. As a result, our proposed generator requires substantially less GPU memory while being considerably faster. To scale our generator for higher resolutions, we add further transformer layers and double the point-upsampling factor. With this configuration, we end up with the following number of Gaussian primitives for the corresponding training resolutions: $\{256^2: 109k, 512^2: 240k, 1024^2: 502k, 2048^2: 1M\}$.

3.2 Multi-view Regularization

Training instability is one of the main problems when working with GANs. Usually, the generator cannot keep up with the performance of the discriminator, which causes the training to collapse. For 3D synthesis, this instability is amplified, since the generator's job of constructing a 3D head is much more complex than the discriminator's job of evaluating the realism of provided 2D images.

To encounter this training instability, we introduce a multi-view regularization that renders the generated 3D heads from multiple views per training step and averages the resulting gradients before backpropagation. This brings two advantages. Firstly, as we average the gradient on a larger sample size, the direction of the gradient is less affected by stochastic variation, improving general convergence. And secondly, as we render the same scene from multiple views per training step, we collect gradients that do not push the weights towards one specific view per latent vector, but instead evenly balance the optimization across multiple views at the same time. This is especially important for 3DGS, given that the positional gradient is very sensitive to small changes [5, 23, 15]. Usually, when using NeRF-based GANs, like EG3D [7], such an operation would be very inefficient, as the rendering with NeRF is a huge bottleneck compared to the feature synthesis part. With the efficient 3D Gaussian splatting rasterizer, we have the opposite case, where the rendering at >200 FPS is considerably faster than the backbone synthesis. As a result, our multi-view regularization only creates a neglectable overhead to the overall training time.

As the discriminator is equipped with a *Minibatch-Standard-Deviation* layer that collects statistics across the batch dimension, we have to be careful not to alter the batch statistics in any way compared to the real batches. For instance, we cannot provide the discriminator with a batch showing four times the same person from different angles, given that no person appears even twice in the real training data. Instead, if we apply the multi-view regularizations four times, we create four new batches where each person is still only shown once per batch.

3.3 Conditional Loss

In order to learn 3D geometry from 2D data, the discriminator is provided with the annotated camera position to learn which camera position belongs to which appearance. Once the discriminator has learned the 3D concepts from real data, it provides meaningful feedback for the generator how to mimic the 3D geometry. However, the way in which this feedback is created can differ quite a lot. GSGAN for instance, applies a contrastive loss function that pulls together matching camera / image pairs, while pushing non-matching pairs apart. Although this approach has proven to work, we observe that it destabilizes the training after several million training iterations. This is caused by the large training set containing quite a lot of camera / image pairs that are very similar to each other. If we now sample a batch of images and cameras that have very similar camera positions, the discriminator will produce high correlation between those non-matching pairs. Although it seems to be not bad if the discriminator confuses two very similar camera labels, it will result in very high loss terms when applying the contrastive loss. Therefore, we argue that the contrastive loss formulation is not ideal for such an optimization problem and dataset. Instead, we use the conditional training mechanism from StyleGAN [19]. Specifically, we do not directly apply a loss on the camera parameters itself but embed the camera parameters into the image features of the discriminator, which are then used to predict whether the provided image is real or fake. This conditioning, without an



Figure 3: Visualizations how GGHead (left) and GSGAN (right) rely on view-conditioning to achieve good quality renderings from any angle. As soon as the view-conditioning does not align with the camera pose, we observe worse quality. Additionally, we use a fixed latent vector for all four images, underlining that view-conditioning also changes the identity and expression.

explicit conditional loss function, is well established for 2D GANs as well as 3D GANs [7, 3]. While removing the contrastive loss, we keep both regularizations from GSGAN, which penalizes Gaussian primitives that move too far from the center and enforces the Gaussians to cluster by calculating a KNN distance between the anchors on the lowest level.

3.4 3D Consistency

While in theory, GSGAN [15] and GGHead [23] produce 3D consistent human heads, they both make use of camera conditioning, where they input the camera viewpoint to the mapping network of the generator. Specifically, during training they forward in 50% of the time the camera information to the mapping network of the generator. Although this was designed to give the generator the ability to deal with view-dependent biases in the dataset and therefore improve training convergence, we observe that the resulting generators heavily rely on this conditioning during inference as they produce poor quality renderings when conditioning the generator to a frontal view, while rendering from the side (Figure 3). Removing this view-conditioning typically destabilizes the training, even causing it to collapse in some cases. *"The necessity for such tricks is one of the main disadvantages of 3D GANs and shows that we still need more research to find better 3D generative modeling paradigms"*¹. As our training is now much more stable with our improved generator architecture, the multi-view regularization and the removed contrastive loss, we are able to fully remove the camera conditioning from the training pipeline without harming training convergence. By doing so, the generator learns to synthesize a consistent 3D head that can be rendered from any viewpoint at high quality.

3.5 Random Background

Like GGHead, we remove the background in the training dataset using the background matting network, MODNet [20]. GGHead then replaces the background with a white background. As a result, we observe that the generator sometimes exploits the white background color and uses it to produce white regions, such as reflections, by creating holes in the surface. To remove this artifact, we replace the white background with a random colored background that changes for each new image sample. This way, the generator cannot rely on any color from the background, enforcing it to produce the correct color that is required for rendering the human head itself.

4 Dataset

When training with in-the-wild images from FFHQ, we observe rendering artifacts that include large undefined floating objects in front of or next to the 3D head. Most often, these objects look like human hands, microphones, or additional human heads. They originate from the training data that includes several people with their hand covering the face, while drinking, smoking or holding a microphone. As the generator is enforced to replicate the same data distribution provided in the training data, it has to mimic the occluded training examples as well. This is a problem for several reasons: Firstly, these occluders are frequent enough in the training data to force the generator to reproduce them, however, they are not frequent enough to give the generator enough information to render them realistically. As a result, they appear in many synthetic images and in poor quality. Secondly, to avoid such artifacts during inference, often the truncation trick is applied. The truncation trick moves

¹<https://github.com/tobias-kirschstein/gghead/issues/19>

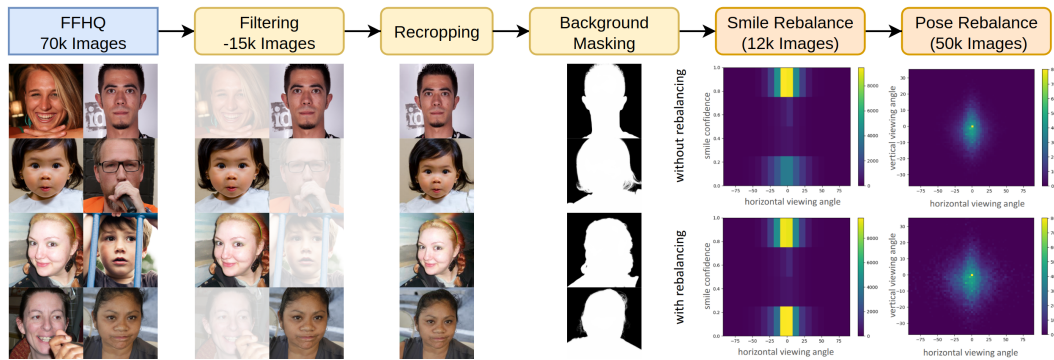


Figure 4: Our data processing pipeline: 1. remove images with occluders to enhance training, 2. recrop the entire head at 2048^2 resolution, 3. apply background masking, 4. rebalance the smiling bias, and finally 5. rebalance the camera positions.

the randomly sampled latent vector towards the mean vector, which typically synthesizes faces with higher quality and fewer occluders but trade image quality with image diversity. Thirdly, rendering the occluders in addition to the 3D head uses capacity of the generator that could be spend on better face details. This especially applies for 3DGS GANs that use a fixed number of Gaussian primitives to construct the 3D scene. For all those reasons, we argue that the overall quality drastically benefits from removing these occluders from the training data. Even though this removes some variation, we argue that effectively more variation is kept, as we no longer have to apply a strong truncation to avoid unwanted artifacts. In total we remove 15k images which is about 20% of the training data. To identify the images where occluders are present, we train a VGG-19 on a small training set curated with PicArrange [16], an application that allows querying large datasets with CLIP prompts, such as: 'a person drinking'. As a result, we propose a new subset of FFHQ, *FFHQ-Clean (FFHQC)* that only keeps training images without occluders.

We further observe that 3D consistency is slightly harmed when training our generator with the default distribution of FFHQ, since FFHQ has a strong bias for people smiling when looking into the camera. Since our model has to replicate this effect without being equipped with view-dependent components, it starts to align very thin Gaussian primitives in a way that a person smiles only when viewed from the front, harming the perceived 3D consistency. To avoid this, we rebalance the dataset by adding 12k images of persons viewed from the front who do not smile. Furthermore, we apply rebalancing for the camera view to improve the synthesis quality for side views, by duplicating 50k images from extreme poses. The full data processing pipeline is visualized in Figure 4.

5 Experiments

We compare our method with two existing Gaussian splatting based 3D GANs for human head synthesis, GGHead [23] and GSGAN [15]. For GSGAN, we remove the background synthesis. We first train all methods with the default FFHQ dataset with removed background, and then apply our new dataset, FFHQC. A detailed training configuration can be found in the appendix. As we specifically set our focus on explicit 3D human head synthesis, we do not compare our results to implicit NeRF-based 3D aware-GANs. Such comparisons can be found in [23] and [15].

To evaluate and compare our results with prior methods, we use the Fréchet Inception Distance (FID) that compares the feature statistics, created with the pre-trained InceptionV3 network, of 50K rendered images to the feature statistics of the training data. Although the FID is a well established method for calculating the image quality, it does not account for 3D consistency, as it only generates a single image per 3D head. This highly favors those methods that render in very good quality for one specific view, but render poorly from any other view. In order to properly measure the quality of the synthesized 3D model, we propose a modified FID metric, **FID_{3D}**, that measures the quality of the 3D faces without exploiting view-conditioning. Specifically, we condition prior methods on the frontal view, given that it provides the best result for most views, but render from a randomly sampled view from the dataset. For our method without conditioning FID and FID_{3D} are equal.

Table 1: FID (50kFull) results where the latent vector is conditioned on the current viewing angle (left table) and FID_{3D} results where we use a fixed frontal conditioning for all images (right table).

FID	FFHQ	FFHQ Clean			
	512	512	1024	2048	
GSGAN	5.02	5.17	/	/	
GGHead	4.34	5.37	9.91	/	
Ours	4.94	4.53	5.25	7.8	

FID _{3D}	FFHQ	FFHQ Clean			
	512	512	1024	2048	
GSGAN	10.50	7.68	/	/	
GGHead	7.90	7.78	14.27	/	
Ours	4.94	4.53	5.25	7.8	

5.1 Quantitative Results

In Table 1, we compare the FID and FID_{3D} results to prior Gaussian Splatting based 3D GAN methods. For the standard FID, that allows for view-dependent rendering, our model achieves comparable scores to GGHead and GSGAN, with GGHead achieving the best score of 4.34 for FFHQ. For our proposed dataset FFHQ, however, our model performs better than both prior methods. Here, GGHead performs worst, because of the template mesh that is optimized for frontal heads. As FFHQ includes full heads and also more views from the side, the provided template mesh might not be suitable anymore.

For the FID_{3D} results, we observe significantly worse quality for GGHead and GSGAN. This is because, the side views now have poor quality, as we restrict the models to produce consistent scenes conditioned on the frontal view. For resolutions of 1024² and 2048² our model produces low values, indicating good scaling capabilities and robust training convergence.



Figure 5: A comparison between GGHead, GSGAN and our proposed method. For prior methods we apply conditioning on the current view (first rows), resulting in good but inconsistent identities, and apply conditioning on a left side view (bottom rows), resulting in a consistent scene with poor quality for novel views. In contrast, our method creates high quality renderings for the whole rotation for FFHQ and FFHQ (last row).

5.2 Qualitative Results

Figure 5 shows example renderings of our method compared to prior methods. It illustrates how the view-conditioning helps synthesizing good renderings for steep angles however at the cost of change in person identity. For GGHead and GSGAN, we observe changes in hair color, expression

Table 2: Performance comparison between 3DGS GANs. Synthesis speed is measured on a Nvidia 4090. Training speed and GPU memory is measured using 4 Nvidia H100 with a batch size of 8.

	Synthesis Speed	Training GPU Memory	Training Speed
GSGAN@512	35.0 FPS	37 GB	43 sec / kimg
GGHead@512	167.7 FPS	15 GB	16 sec / kimg
GGHead@1024	151.7 FPS	36 GB	27 sec / kimg
Ours@512	152.5 FPS	15 GB	25 sec / kimg
Ours@1024	110.7 FPS	19 GB	64 sec / kimg
Ours@2048	61.3 FPS	38 GB	157 sec / kimg

and glasses when moving the camera without altering the latent vector. Alternatively, if we condition a fixed latent vector, in this case the left most image highlighted in red, we observe good quality for that view, but worse quality the further we rotate. Both problems are no longer present with our proposed method that removes the view-conditioning. Our method produces high quality renderings from any viewpoint, producing a single fixed 3DGS scene. Results for 1024² training are shown in Figure 1. We provide more qualitative results in the appendix.

5.3 Performance and Memory

In Table 2, we compare the synthesis speed, GPU memory usage and training speed among the 3DGS GAN methods. Instead of measuring the rendering speed of the resulting 3DGS scene, which is very fast for all methods (> 200 FPS) leveraging the efficient 3DSG rasterizer, we measure the time for synthesizing novel faces. I.e., the time it takes between forwarding a random latent vector and rendering the resulting 3D scene. With GGHead being the overall fastest with 168 FPS, our method achieves comparable results with 153 FPS. GSGAN compares as the slowest, which is due to expensive graph convolution layers. Similar results are observed in training GPU memory, where our method and GGHead require more than half of the memory of GSGAN. For higher resolutions, our model still requires comparably little GPU memory.

Although one training step of our method is slower than for GGHead, our method converges much faster. While GGHead trains for 25M iterations, taking about 4.5 days, our method already converges in less than 10M steps only taking less than 3 days.

5.4 Multi-view Regularization

Training our model without the multi-view regularization is significantly less stable, causing it to collapse after about 8M training steps. Further, as we do not generate new 3D scenes during multi-view regularization, the only computational overhead is the efficient 3DGS rendering and additional forward passes through the 2D discriminator. While generating 4x more images during training, the overall duration is only extended by 20%. We found rendering 4 views per training step a good setting, as we do not improve using 8 and perform worse using 2.

5.5 Random Background Augmentation

In Figure 6 (right), we demonstrate the effectiveness of our random background augmentation. Without random backgrounds, the model overfits to the white background causing it to create holes instead of creating white Gaussian primitives. As our method can no longer rely on a single background color, it produces every color, required for rendering the 3D head, with a Gaussian primitive.

5.6 Dataset Comparison

Rendering images with our proposed dataset, FFHQC, shows to improve the overall quality. Not only are we able to synthesize the full head (Figure 5 bottom) at high resolutions, we also observe better quality when rendering steep angles as shown in Figure 6 (top). This is caused by our camera rebalancing strategy, enforcing steeper angles to be shown more frequently. Furthermore, as we filter 15k occluded face images, we observe almost exclusively good quality 3D human heads, without rendering artifacts. Only in rare cases, we still observe some floating objects next to the face. This is

likely due to the background matting network, which sometimes confuses background and foreground. Next to removed occluders, we observe that the smiling rebalancing resolved a problem where some rendered faces were smiling when shown from a frontal view, while looking neutral from a side view as shown in Figure 6 (left).

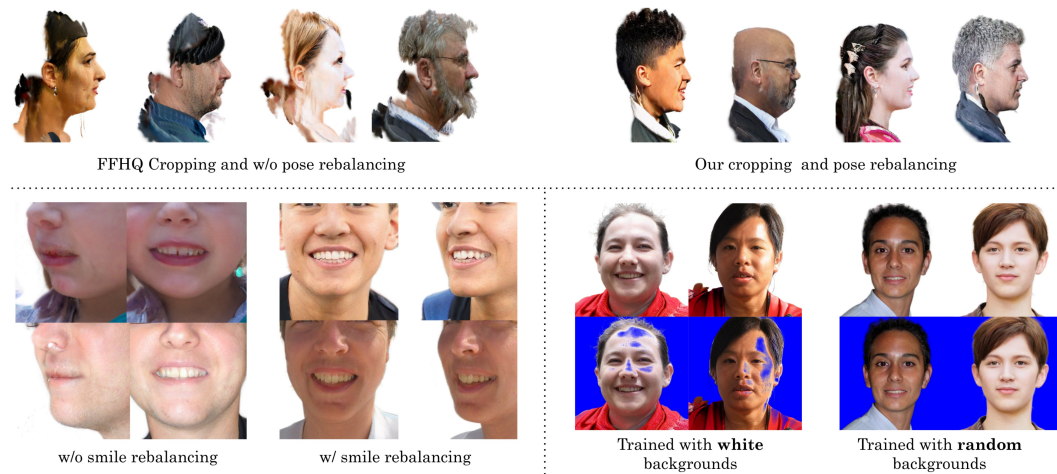


Figure 6: Top: Improved side views by training with entire heads and rebalancing the camera pose. Bottom left: Renderings w/ and w/o smiling bias. Bottom right: Training w/ and w/o random backgrounds.

6 Conclusion & Outlook

We propose a novel Gaussian splatting GAN framework, **CGS-GAN**, that is capable of efficiently synthesizing 3D consistent human heads at high-quality and high resolutions of up to 2048^2 . While prior work suffers from either rendering inconsistent persons or producing 3D heads that show poor quality for some viewpoints, our method achieves very good quality from any viewing angle, while producing a fixed 3DGS scene that can be ported into explicit 3D environments. This allows for applications such as inverting a 3D head from a 2D photograph and directly importing it into a game engine. This potentially saves time and resources, as no high-end 3D capturing studio is required. A straight forward extension to this work could be the inclusion of further training samples showing the back of the head, like in [3], allowing for full 360° head synthesis. Furthermore, even though our method already trains very fast, requiring less iterations while possessing an efficient forward pass, we could apply optimized attention implementations to reduce the training time even further, making development of novel 3D GAN methods easier and more accessible. Finally, combining our method with 3D morphable models could turn our static human heads into animatable talking 3D head avatars.

Acknowledgments and Disclosure of Funding

This work has partly been funded by the German Research Foundation (project 3DIL, grant no. 502864329), the European Commission (Horizon Europe project Luminous, grant no. 101135724) as well as the Fraunhofer Society in the Max Planck-Fraunhofer collaboration project NeuroHum.

References

- [1] R. Abdal, W. Yifan, Z. Shi, Y. Xu, R. Po, Z. Kuang, Q. Chen, D.-Y. Yeung, and G. Wetzstein. Gaussian shell maps for efficient 3d human generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9441–9451, 2024.
- [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] S. An, H. Xu, Y. Shi, G. Song, U. Y. Ogras, and L. Luo. Panohead: Geometry-aware 3d full-head synthesis in 360deg. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20950–20959, 2023.
- [4] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *ICCV*, 2021.
- [5] F. Barthel, A. Beckmann, W. Morgenstern, A. Hilsmann, and P. Eisert. Gaussian splatting decoder for 3d-aware generative adversarial networks. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 7963–7972, 2024.
- [6] S. Brehm, F. Barthel, and R. Lienhart. Controlling 3D objects in 2D image synthesis. *SN Comput. Sci.*, 4(1), Nov. 2022.
- [7] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. De Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16123–16133, 2022.
- [8] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5799–5809, 2021.
- [9] X. Chen, Y. Deng, and B. Wang. Mimic3d: Thriving 3d-aware gans via 3d-to-2d imitation. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2338–2348. IEEE Computer Society, 2023.
- [10] X. Chen, M. Mihajlovic, S. Wang, S. Prokudin, and S. Tang. Morphable diffusion: 3d-consistent diffusion for single-image avatar creation. 2024.
- [11] X. Chu and T. Harada. Generalizable and animatable gaussian head avatar. *Advances in Neural Information Processing Systems*, 37:57642–57670, 2024.
- [12] M. Gadelha, S. Maji, and R. Wang. 3d shape induction from 2d views of multiple objects. In *2017 International Conference on 3D Vision (3DV)*, pages 402–411, 2017.
- [13] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.
- [14] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu. Pct: Point cloud transformer. *Computational visual media*, 7:187–199, 2021.
- [15] S. Hyun and J.-P. Heo. GSGAN: Adversarial learning for hierarchical generation of 3d gaussian splats. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [16] K. Jung, K. U. Barthel, N. Hezel, and K. Schall. Picarrange-visually sort, search, and explore private images on a mac computer. In *International Conference on Multimedia Modeling*, pages 452–457. Springer, 2022.
- [17] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.
- [18] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.
- [19] 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.
- [20] Z. Ke, J. Sun, K. Li, Q. Yan, and R. W. Lau. Modnet: Real-time trimap-free portrait matting via objective decomposition. In *AAAI*, 2022.
- [21] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.
- [22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
- [23] T. Kirschstein, S. Giebenhain, J. Tang, M. Georgopoulos, and M. Nießner. GGHead: Fast and Generalizable 3D Gaussian Heads. In *SIGGRAPH Asia 2024 Conference Papers*, SA '24, New York, NY, USA, 2024. Association for Computing Machinery.

- [24] T. Kirschstein, S. Qian, S. Giebenhain, T. Walter, and M. Nießner. Nersemble: Multi-view radiance field reconstruction of human heads. *ACM Trans. Graph.*, 42(4), jul 2023.
- [25] Y. Lan, F. Tan, D. Qiu, Q. Xu, K. Genova, Z. Huang, S. Fanello, R. Pandey, T. Funkhouser, C. C. Loy, and Y. Zhang. Gaussian3diff: 3d gaussian diffusion for 3d full head synthesis and editing. In *ECCV*, 2024.
- [26] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.
- [27] M. Mirza and S. Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.
- [28] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt, and Y.-L. Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7588–7597, 2019.
- [29] A. Nichol, H. Jun, P. Dhariwal, P. Mishkin, and M. Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022.
- [30] S. Qian, T. Kirschstein, L. Schoneveld, D. Davoli, S. Giebenhain, and M. Nießner. Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20299–20309, 2024.
- [31] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20154–20166. Curran Associates, Inc., 2020.
- [32] A. Shoshan, N. Bhonker, I. Kviatkovsky, and G. Medioni. Gan-control: Explicitly controllable gans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14083–14093, October 2021.
- [33] J. Tang, D. Davoli, T. Kirschstein, L. Schoneveld, and M. Niessner. Gaf: Gaussian avatar reconstruction from monocular videos via multi-view diffusion. *arXiv preprint arXiv:2412.10209*, 2024.
- [34] A. Trevithick, M. Chan, T. Takikawa, U. Iqbal, S. De Mello, M. Chandraker, R. Ramamoorthi, and K. Nagano. What you see is what you gan: Rendering every pixel for high-fidelity geometry in 3d gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22765–22775, 2024.
- [35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [36] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- [37] J. Xiang, J. Yang, Y. Deng, and X. Tong. Gram-hd: 3d-consistent image generation at high resolution with generative radiance manifolds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2195–2205, 2023.
- [38] Y. Xue, Y. Li, K. K. Singh, and Y. J. Lee. Giraffe hd: A high-resolution 3d-aware generative model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18440–18449, 2022.
- [39] J. Zhang, Z. Wu, Z. Liang, Y. Gong, D. Hu, Y. Yao, X. Cao, and H. Zhu. Fate: Full-head gaussian avatar with textural editing from monocular video. *arXiv preprint arXiv:2411.15604*, 2024.
- [40] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021.

A Technical Appendices and Supplementary Material

In the following sections, we give a more training details of our GAN framework A.1, provide further information about the components of the generator A.2, compare our generator architecture to GSGAN A.3, discuss the limitations and broader impacts of this work A.4, make further comparisons to prior NeRF-based approaches A.5, provide further ablations for the multi-view regularization A.6, render further results A.11, and show some failure cases with possible solutions for future work A.7. Additionally, we give more details about the contrastive loss function used in GSGAN and show how it destabilizes the training A.8, give more details about the dataset filtering process A.9, and, finally, show that the resulting scenes can be used in explicit environments such as Unity A.10. More interactive results can be viewed on the project page that we include in the supplementary materials.

A.1 Training Details

Given that GAN training can be very difficult to set up, a lot of works, including this one, base their implementation on an already well functioning GAN frameworks. While there are often significant changes for the generator architecture, we observe very similar hyperparameter settings, adversarial loss functions and regularization. Chronologically, there has been the *Progressive Growing of GANs* [17], then *StyleGAN* [18] and *StyleGAN2* [19], and specifically for 3D rendering *EG3D* [7]. Those works all build upon the prior one, thus inheriting most training setting. The same goes for GSGAN [15], which we use as a starting point for our work, that applies most settings from EG3D.

As a result, our training settings are closely orientated to all prior methods. Specifically, this includes an Adam optimizer [22] with a learning rate of 0.0025 for the generator and 0.002 for the discriminator and a R1 regularization of 1.0, which penalizes the discriminator if it creates too large gradients for real images. At the beginning of the training, we blur the real and fake images for a period of 200k training iterations, to enable smooth training convergence. We set the batch size to 32, which is collated of 4 batches of size 8 for each of the GPUs. During training, we track the FID with 20k fake images to identify the best checkpoint. Further, we average the generator weights in an *Exponential Moving Average* EMA generator, which produces better results than the training generator. For our 2D discriminator, we use the StyleGAN2 discriminator from [19].

Different to GSGAN, which represents the 3DGS scene in a 3D cube with side length of 2 (-1 to 1), we go back to the EG3D representation, using a cube with side length of 1 (-0.5 to 0.5). GSGAN, also modifies the field of view of the rendering camera from 12° to 22° , given that it uses a much larger scene. We, however, also revert this setting to the original 12° used in EG3D.

A.2 GAN Architecture

In Figure 7, we provide an overview of the point upsampling layer and the toGauss / toAnchor layer that we use in our generator architecture. The point upsampling is constructed of a repeat branch that simply duplicates the points four times and a learnable branch that produces four times more features and then reshapes the features into the point dimension. Intuitively, each point is the offspring to four

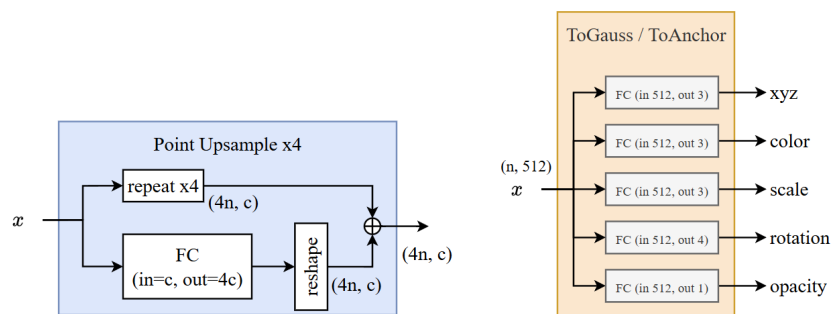


Figure 7: Overview of the point upsampling layer (left) and the toGauss / toAnchor layer (right) used in our generator.

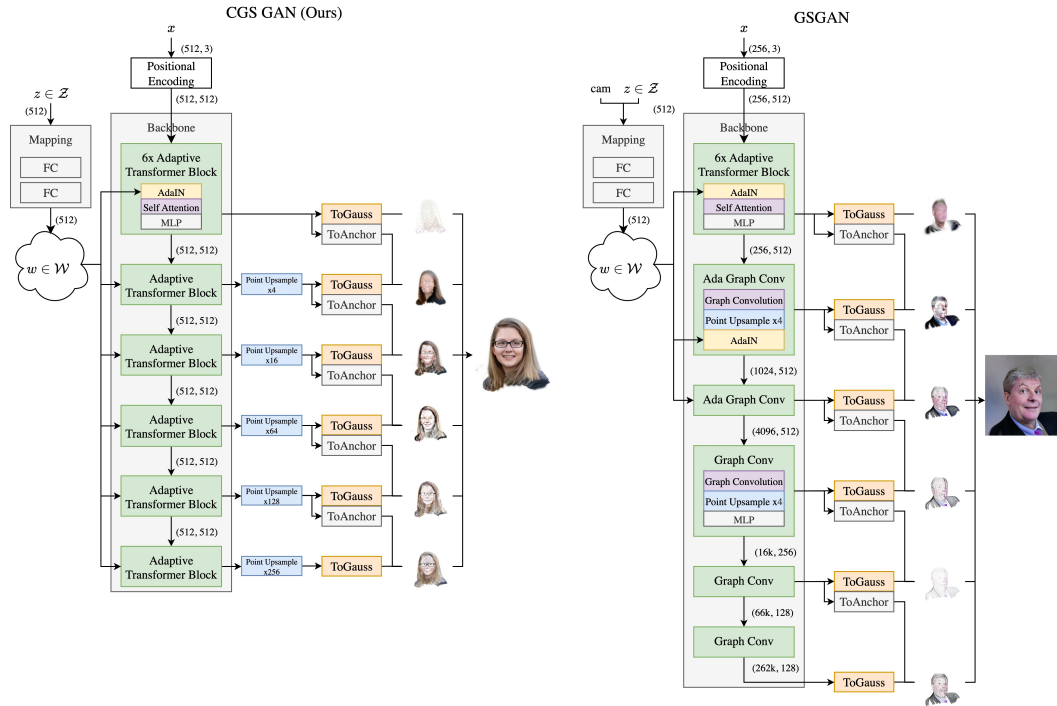


Figure 8: Comparison between our generator (left) and the generator of GSGAN (right).

new points with slightly changed features each. The toGauss / toAnchor is simply constructed of fully connected layers that map each high dimensional 512D point into a 3DGS feature. Additionally we also apply a weak tanh activation between -20 and 20. Otherwise, features such as rotations infinitely rise to values of 1000. For the rendered scene, this makes no difference since the magnitude of the quaternion is neglected, however, for achieving stable gradients it can make a big difference.

A.3 Comparison to GSGAN

In Figure 7, we compare our generator architecture (left) to the architecture of GSGAN (right). The core differences are the removed upsampling layers in the backbone network, the removed graph convolution components, and the removed background generator.

Upsampling: Instead of upsampling the feature points in the backbone network, which causes quadratic memory consumption for each subsequent layer, our backbone maintains a fixed number of feature points and performs the upsampling for each output layer individually. This way, we can scale our network for a very large number of Gaussians, without requiring a lot of GPU memory.

Graph Convolution: Since the memory consumption is too high using attention layers with large inputs, GSGAN applies graph convolution layers after the first block in the backbone. This however, slows down the forward pass, as graph convolution layers require expensive neighborhood search algorithms like KNN. As we do not upsample the points in the backbone, we can instead use efficient attention layers for all following layers.

Background Generator: To render the appearance of the background, GSGAN uses a scaled down version of the main generator, to produce points on a sphere, which are added in the background. As we remove the background in our training data, we also remove the background generator for our architecture.

A.4 Limitations & Broader Impacts

Although our proposed model is capable of synthesizing high-fidelity renderings showing the full head, we still cannot render the back of the head, as such images are not provided in FFHQ. Including

Table 3: Further comparisons to EG3D, GSM and Mimic3D (FID@512).

Method	FID
EG3D [7]	3.28
Mimic3D [9]	4.27
GSM [1]	28.19
Ours	4.94

Table 4: Comparison between different number of views during multi-view regularization.

Num Views	FID
w/o Multi-view	6,44
Multi-view 2	6,13
Multi-view 4	4,90
Multi-view 8	4,96

additional images showing the back of the head like in [3], would likely achieve even better results. Further, even after removing 15k images showing occluders, we still observe some artifacts being present in some images. This needs further attention, either by filtering more images or improving the background matting network that sometimes fails to differentiate hats from backgrounds or masks multiple persons in an image. Additionally, our high resolution methods are still slow to compute, making further development in this field not accessible for many smaller research groups with fewer computational capacity. Therefore, we are aiming to apply more efficient CUDA implementations to reduce the training time even further. Further, we provide our training checkpoints for all resolutions, together with the optimizer momentum values, allowing to finetune our models without the necessity of re-training the full pipeline from scratch.

Building automatic tools for synthesizing realistic human heads is both a scientific advancement with many applications in the industry while also a dangerous tool that can be misused for producing fraud information. This is an ongoing and complex problem that needs more attention. Furthermore, FFHQ has a strong ethnic bias, which is very problematic when applying such methods in production. Finally, using such generative models to produce art, can be problematic as the training data might be used against the consent of the respective artist.

A.5 Further Comparisons

In Table 3 we provide further comparisons to NeRF-based approaches, such as EG3D and Mimic3D, while also another Gaussian splatting based method, *Gaussian Shell Maps* (GSM), which was originally designed for full body 3D human avatars. We observe that EG3D produces the overall best results. Nevertheless, all except our method takes advantage of the view conditioning method. Further, EG3D also applies 2D super-resolution layers after NeRF rendering, making it even less 3D consistent.

A.6 Multi-view Regularization Ablation

In Table 4 we provide an ablation for different number of views shown during the multi-view regularization. We observe better quality with increasing number of views. After four views, however, the performance does not improve further. Instead we observe slightly worse FID results. Therefore, we have selected four views as our default setting in all experiments. Notably, the training does not fully collapse, when removing the multi-view regularization from our model. We, however, observe significantly worse results and worse training convergence, as shown in Figure 9. Here, the output of the discriminator is considerably lower, when removing the multi-view regularization. This means that the discriminator easily detects the fakes and produces very high gradients that harm the training.

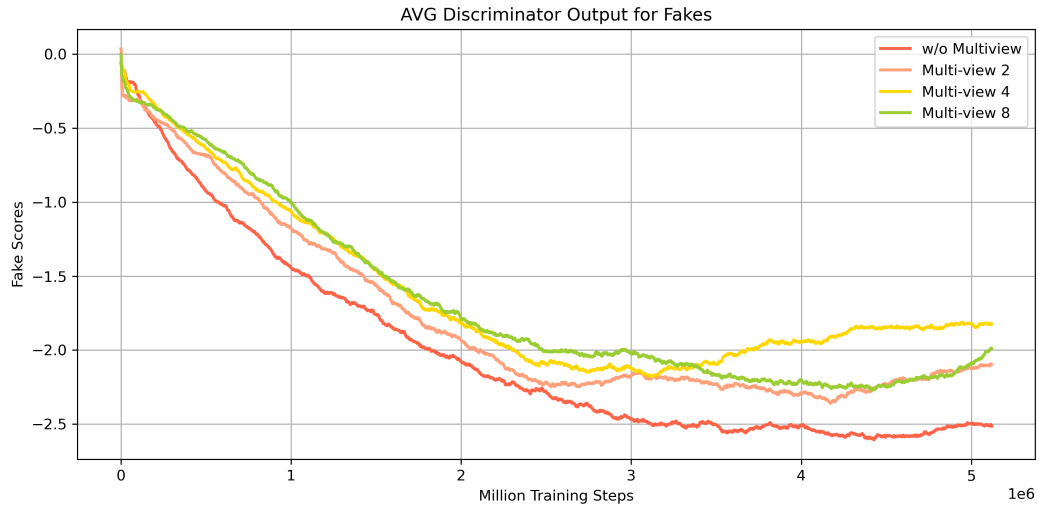


Figure 9: Comparison of the discriminators output signal for fake images when using different settings for the multi-view regularization. Higher scores denote better training convergence.

A.7 Failure Cases

While the vast majority of rendered images look very good, in some cases we still observe occluders to be present in the rendered output images. This is shown in the top row of Figure 10. We believe that this is due to the masking network that sometimes masks multiple persons or confuses the background with a hat as shown in Figure 11. Additionally, we notice that some heads have a hole in the hair region when rotating them very far to the side. Such effect can likely be avoided by using more training samples with steeper viewing angles.



Figure 10: Top row shows some rare cases where the face is still obscured by floating objects. This is likely fixed by using a better masking network. And the bottom row shows holes that become visible when turning $>90^\circ$. Such viewing angles are never shown during training.



Figure 11: Real training examples with respective masks, where the masking network masks multiple persons, or where the background is confused with a hat.

A.8 Contrastive Loss

In the method section (Section 3) of our main paper, we argue that the contrastive loss function of GSGAN [15] is not ideal for 3D GAN training with FFHQ. This is because a lot of camera view are very similar in the dataset, and the network is highly penalized if it confuses two almost identical camera views. Specifically, the contrastive loss is formalized as [15]:

$$\mathcal{L}_{\text{pose}} = -\log \left(\frac{\exp(\text{sim}(p_I, p_{\theta}^+)/\tau)}{\sum_{b=1}^B \exp(\text{sim}(p_I, p_{\theta}^b)/\tau)} \right). \quad (1)$$

Here, sim denotes the cosine similarity, p_I an embedding of the image, p_{θ}^+ the camera embedding of the matching camera, p_{θ}^b the camera embedding of all non matching cameras in batch B , and τ a temperature scaling parameter. If we now have a p_{θ}^+ that is very similar to another camera embedding p_{θ}^b within the training batch, we receive a very high loss that harm the training convergence. This can be seen in Figure 12, where the contrastive loss decreases for the first few million training steps and then suddenly increases again after about 3.8 million steps. At this point the training completely collapses and does not recover again.



Figure 12: Contrastive loss tracked over the course of the training. After 3.8 million steps, the loss increases again, causing the training to collapse.

A.9 Dataset Filtering

In Figure 13, we give an overview of the categories that we applied to identify bad training images. While mainly focusing on images where the face is obscured by an object, we also found images that generally harm the training convergence. Examples for such images are shown in the 'depiction' class and the 'blurry' class in Figure 13. Consequently, we removed those images along with the other images, with occluders.



Figure 13: An overview of the main categories for images that we identify as bad training data. Categories like 'Hand', '>1 Face' and 'Microphone' were the most frequent ones.

To identify, which images should be removed we use PicArrange [16], which is an application that allows for querying large datasets with text prompts. A screenshot of the application is shown in Figure 14, where we query the dataset with the prompt 'food' and receive all matching images.

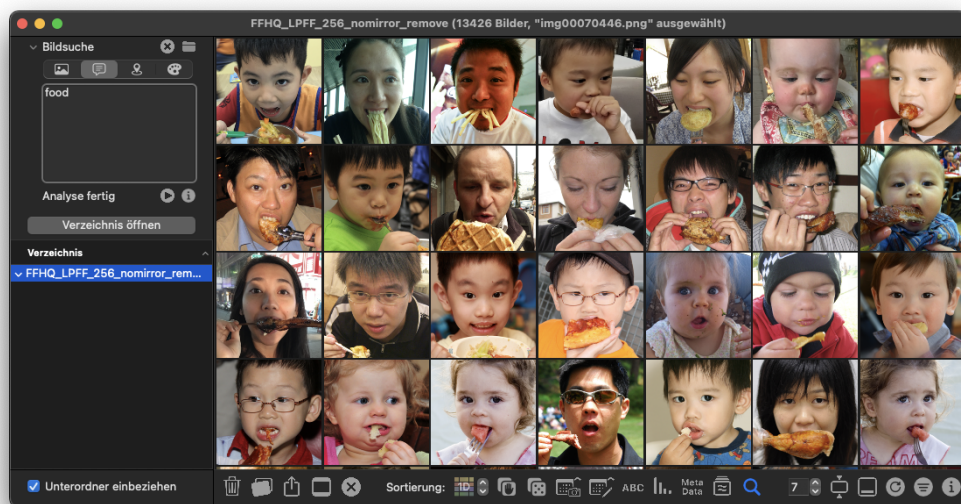


Figure 14: Screenshot of PicArrange [16], which was used to identify bad bad training images, by writing text prompts like "food" in the upper left corner. PicArrange then returns the images with the highest CLIP correlation to the respective prompt.

A.10 Explicit 3D Environments

In Figure 15, we demonstrate that our 3D heads can be imported into explicit 3D environments, such as Unity. Using the 3DGS plugin by Aras Pranckevičius (<https://github.com/aras-p/UnityGaussianSplatting>), the 3D faces can simply be placed into 3D scenes and interact with other 3D objects. Combining this with the inversion capabilities of our models, we can now create realistic 3D assets just from a single 2D image.



Figure 15: Our faces imported into unity using the Unity Gaussian splatting plugin by Aras Pranckevičius: <https://github.com/aras-p/UnityGaussianSplatting>

A.11 Rendering Results

In Figure 16 and Figure 17, we demonstrate uncurated renderings of our proposed method, trained on FFHQ at 512 resolution. Apart from some outliers, we observe very high quality with few occluders being present. Further, in Figure 18 and Figure 19 we demonstrate the difference between the original FFHQ dataset and our adapted FFHQ dataset. We observe fewer occluders and render the full head instead of clipping just above the hair line.

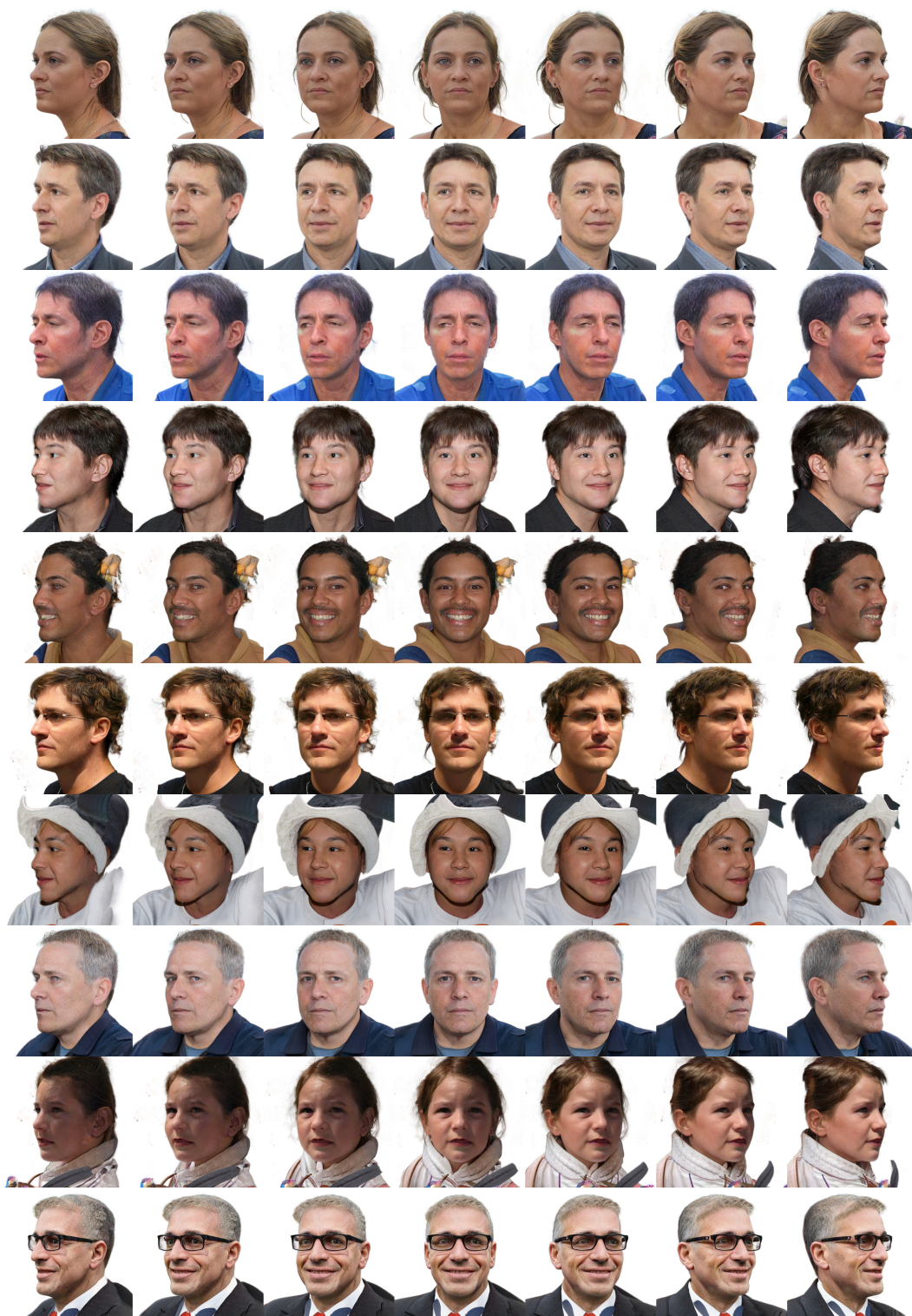




Figure 17: Example renderings of our method trained on FFHQ at 512 resolution ($\psi = 0.8$).



Figure 18: Rendering examples of our model with **FFHQ** ($\psi = 1.0$).



Figure 19: Rendering examples of our model with **FFHQ** showing the full head with fewer occluders ($\psi = 1.0$).

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: We give a brief overview of our work in the abstract and list our main contributions again in the introduction.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We discuss the limitations for both our method and our curated dataset in more detail in the appendix, additionally showing failure cases.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: No theoretical results or proofs are made in our paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Next to the details in our main paper, we provide more detailed information about specific training settings in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The training data that we use is publicly available and the code will be published along with detailed instructions how to reproduce our results. We also provide the code to the reviewers.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We report our training details in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Training our models and the models of related work takes up a lot of time and resources, not allowing us to run them multiple times.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We report memory consumption and training time in 5.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We review the NeurIPS code of Ethics and ensure that there is no conflict with our work.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We provide a short section about the broader impact of our work in the appendix.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our paper does not pose such risks as we do not use any additional data other than publicly available image datasets.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We only use FFHQ which was published along with the StyleGAN paper which is cited. FFHQ has a CC BY-NC-SA 4.0 license.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA] .

Justification: We do not release new assets. We only change the pre-processing of existing public datasets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [No]

Justification: We exclusively use LLM for writing, editing, or formatting.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.