# Benchmarking and Analyzing 3D-aware Image Synthesis with a Modularized Codebase —Supplementary Material—

**Anonymous Author(s)**
Affiliation
Address
email

## Supplementary Materials Organization:



## 1 Experimental details

### 1.1 Backbone

In this study, we adopt the state-of-the-art 3D-aware GAN, EG3D [2], as our backbone model, which we have reproduced in our modularized codebase. This codebase is utilized to perform all

Submitted to the 37th Conference on Neural Information Processing Systems (NeurIPS 2023) Track on Datasets and Benchmarks. Do not distribute.

experiments presented in this paper. Our investigation involves substituting each module in 3D GANs with alternative choices to study their individual effects. For instance, to explore the point embedder, we replace its tri-plane with feature volume or other representations. Similarly, to investigate the feature decoder, we alter the depth or activation function of the decoder. Given our modularized design, these changes can be easily implemented by adjusting the parameters in the configuration files.

In our experiments, we use only one-stage training to save computational resources. Specifically, we perform the training only at a neural rendering resolution of 64×64, without stepping the resolution up to $128 \times 128$.

### 1.2 Point embedder

In this work, we mainly explore the capacity of three point embedders: MLP, volume, and tri-plane, as well as their combinations. The MLP-based point embedder utilizes a multi-layer perceptron (MLP) to transform raw point coordinates into point features. The volume-based point embedder queries point features from the feature volume, while the tri-plane based point embedder queries point features from the tri-plane feature representation.

The experimental settings for the tri-plane based point embedder are identical to those of [2]. For the MLP-based point embedder, we adopt an MLP network to extract point features, similar to [6]. This network employs ReLU activation, 1×1 convolutions modulated by style vectors, and has a depth of 16, with a hidden layer dimension of 128 and an output layer dimension of 64. Notably, for MLP-based point features, the point embedder and the feature decoder are actually the same. As for the volume-based point embedder, we generate the feature volume using a generator that utilizes 3D convolutions, which is the same network architecture as [12]. The feature volume resolution is set as $64 \times 64$. The feature decoder for volume-based point features is an MLP network with the same architecture as the MLP-based point embedder. Other settings including geometric representation, upsampler, pose priors, *etc.*, remain consistent with our backbone model.

We also investigate the impact of composite point embedders, which entail combining two or more point embedders. Specifically, we explore combinations of MLP and volume, MLP and tri-plane, volume and tri-plane, and MLP, volume, and tri-plane. The combination of MLP and volume involves concatenating the MLP-based point features and volume-based point features along the feature channel dimension, while the other combinations follow the same principle. To conserve computational resources during training with composite point embedders, we set the MLP-based point features as the raw point coordinates, with the MLP serving as an identical mapping.

### 1.3 Feature decoder

Considering that the feature decoder typically comprises a multi-layer perceptron, it is crucial to investigate the impact of its depth and activation layer type on the performance. To this end, we conduct an empirical study on the depth and activation type of the feature decoder. Specifically, we perform experiments on three point features, including MLP, volume, and tri-plane, utilizing the experimental setup inherited from Sec. 1.2, but with varying depths of the MLP. In terms of the activation type of feature decoder, we employ both SIREN-based layers and ordinary ReLU-based layers in two settings: with and without an upsampler. And the depth of both MLPs was set to 8, with a hidden dimension of 128 and an output dimension of 64. When training the model with a SIREN-based layer, similar to [1], we set the generator learning rate to 0.00006 and the discriminator learning rate to 0.0002.

### 1.4 Geometric representation

Prior works, such as NeuS [11], have incorporated the signed distance function (SDF) into the volume rendering formula to enable the reconstruction of smooth surface models. Moreover, 3D GANs [10] have explored the use of SDF as a geometric representation for consistent generation. Hence, our

Table 1: Training time and inference speed of models with different point embedders. The tri-plane-based point embedder exhibits the highest computational efficiency.

| Point Embedder | | | FFHQ [7] 256×256 | | Cats [13] 256×256 | | Cars [3] 128×128 | |
|---|---|---|---|---|---|---|---|---|
| MLP | Volume | Tri-plane | Training Time | Inference Speed | Training Time | Inference Speed | Training Time | Inference Speed |
| ✓ | ✗ | ✗ | 5.6 Days | 29 FPS | 6.1 Days | 29 FPS | 6.9 Days | 23 FPS |
| ✗ | ✓ | ✗ | 6.8 Days | 24 FPS | 7.3 Days | 24 FPS | 8.3 Days | 20 FPS |
| ✗ | ✗ | ✓ | **2.7 Days** | **49 FPS** | **3.1 Days** | **49 FPS** | **2.7 Days** | **48 FPS** |
| ✓ | ✓ | ✗ | 6.9 Days | 24 FPS | 7.5 Days | 24 FPS | 8.4 Days | 20 FPS |
| ✓ | ✗ | ✓ | 2.8 Days | 49 FPS | 3.3 Days | 49 FPS | 2.8 Days | 48 FPS |
| ✗ | ✓ | ✓ | 3.8 Days | 38 FPS | 4.4 Days | 38 FPS | 3.9 Days | 35 FPS |
| ✓ | ✓ | ✓ | 3.9 Days | 38 FPS | 4.5 Days | 38 FPS | 4.0 Days | 35 FPS |

goal is to examine the effects of two distinct geometric representations: density and SDF. We perform experiments on three point embedders: MLP, volume, and tri-plane. The baseline models utilizing vanilla density are identical to the model described in Sec. 1.2. For SDF-based models, the feature decoder outputs SDF values. We also incorporate sphere initialization, eikonal loss, and minimal surface loss during training, with the same loss weight as [10].

### 1.5 Upsampler

Our study aims to analyze the influence of the upsampler in 3D GANs. To ensure a fair comparison, we conduct experiments on the generation of 256×256 resolution images. When an upsampler is not utilized, the neural rendering resolution is the same as the image resolution, resulting in significantly longer training time and higher computational costs. We conduct experiments on a model without an upsampler, and disable dual discrimination since the output images are only at a fixed resolution of 256×256. Other settings remain consistent with the backbone model.

### 1.6 Pose priors

We additionally explore pose priors on the FFHQ [7] dataset. To obtain the accurate pose distribution (APD) of the dataset, we adopt the approach described in [1], where the pose distribution is modeled as a Gaussian prior. Camera poses are sampled from a normal distribution with a vertical mean of $\pi/2$ radians, standard deviation of 0.155 radians; and a horizontal mean of $\pi/2$ radians, a standard deviation of 0.3 radians. Regarding the random pose distribution (RPD), a Gaussian distribution is also assumed for the pose, with only the vertical/horizontal mean and standard deviation being randomly assigned. And we introduce the acquisition of ground-truth poses in Sec. 2.

## 2 Data details

We conduct our experiments on three datasets, including FFHQ [7], Cats [13], and ShapeNet Cars [3]. Since the original data of these datasets lacks pose labels, we perform preprocessing steps for each dataset. We follow [2] to align, crop and get pose matrix for each image in the FFHQ [7] dataset. Cats [13] contains more than $6K$ real-world cat images, and the data is preprocessed following [5]. The ShapeNet Cars [3] dataset comprises various synthetic car models. We use the dataset rendered from [2] which is composed of approximately $530K$ images. Unlike the forward-facing datasets, its camera poses encompass the full range of $360^\circ$ horizontal and $180^\circ$ vertical distributions. In our experiments, the resolution of 256 × 256 is employed for FFHQ and Cats datasets, while for the ShapeNet Cars dataset, a resolution of 128 × 128 is used.

## 3 More results

### 3.1 Efficiency comparison

We report the training time and inference speed of models utilizing various point embedders in Tab. 1. The training time is computed by training our models on 8 NVIDIA A100 GPUs for 25 million images. Inference speed is measured on a single NVIDIA A100 GPU, where we processed 1K


Figure 1: Qualitative comparison across various composite point embedders on FFHQ [7], Cats [13] and ShapeNet Cars [3], where these compound point features exhibit on-par performance in generating multi-view consistent images and high-quality geometries.

images and calculated the average FPS. As shown in Tab. 1, tri-plane-based point embedders exhibit superior computational efficiency compared to MLP-based and volume-based point embedders. MLP-based point embedders require a larger number of layers to extract point features, leading to longer processing times. Among these, volume-based point embedder is the least efficient, as it involves 3D convolutions.

### 3.2 More qualitative results

We present a qualitative comparison of various composite point embedders on FFHQ [7], Cats [13] and ShapeNet Cars [3] in Fig. 1. The qualitative result shows that combining the outputs of MLP Volume Tri-plane multiple embedders has a negligible impact on the final outcome. Additionally, we show more results of different point embedders on FFHQ [7] in Fig. 2, Fig. 3 and Fig. 4. Interestingly, we observe that models equipped with tri-plane-based point embedders generate 3D shapes with sharper noses, while those equipped with MLP-based or volume-based point embedders do not exhibit this characteristic. This phenomenon can be observed more clearly in Fig. 5. However, the underlying reason for this remains unknown.

Figure 2: Samples synthesized on FFHQ [7] with truncation 0.7 using the model with an MLP-based point embedder. For each generated identity, we show the underlying geometry under two views and appearance under three views.

Figure 3: Samples synthesized on FFHQ [7] with truncation 0.7 using the model with a volume-based point embedder. For each generated identity, we show the underlying geometry under two views and appearance under three views.

### 3.3 Code

Our code can be accessed in the supplementary file, and comprehensive training details are provided within. Upon acceptance, our code and models will be made publicly available. We ensure that all results reported in this paper can be easily reproduced.

## 4 Limitations and future work

**Efficiency.** Training our 3D GAN models is a time-consuming process, especially when utilizing MLP-based and volume-based point embedders. To draw meaningful conclusions, we must conduct a plethora of experiments. Currently we are uncertain about how to improve the efficiency of our models' training. Each experiment requires careful hyperparameter tuning, which is a challenging task. However, due to computational limitations, multiple runs of our experiments are infeasible. Consequently, we identify the "optimal" settings using a limited number of attempts.

**Training stability.** We have observed that training 3D GANs can be rather unstable. Some experiments are highly sensitive to hyperparameters such as the $\gamma$ value of R1 regularization [9], learning rate, *etc*. However, our study does not investigate this aspect in depth. Future research addressing hyperparameter sensitivity and training stability may lead to significant reductions in training costs and more compelling results.

**Universality.** We trained all our 3D-aware image synthesis models on simple, object-level datasets, such as FFHQ [7] for face generation, and our conclusions are based on these datasets. However, our paper does not explore the extension of 3D GANs to a higher degree of universality, which represents

Figure 4: Samples synthesized on FFHQ [7] with truncation 0.7 using the model with a tri-plane-based point embedder. For each generated identity, we show the underlying geometry under two views and appearance under three views.

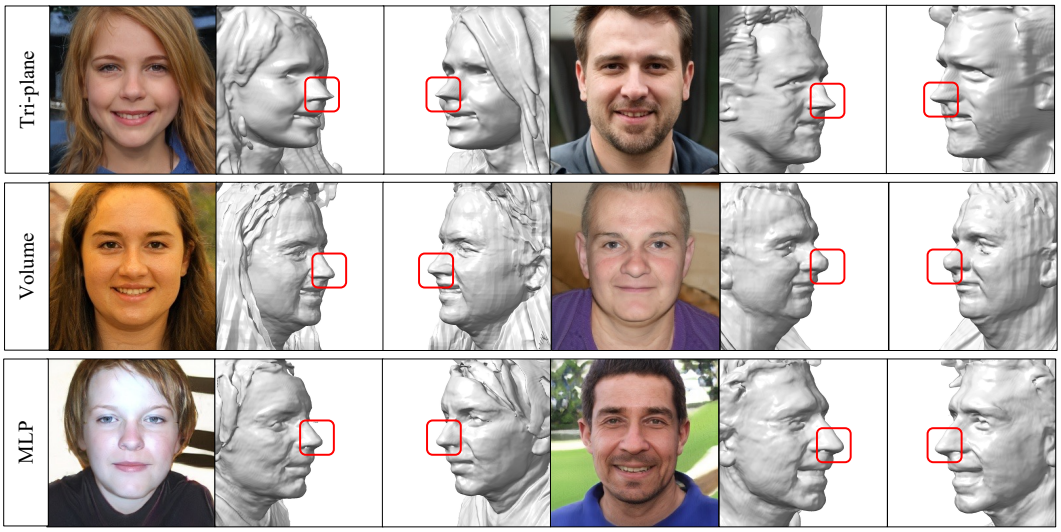


Figure 5: Qualitative comparison across different single point embedders on FFHQ [7], zoom in for better viewing. Models equipped with tri-plane-based point embedders generate 3D shapes with sharper noses, which can appear unnatural.

a promising research direction for the future. This universality pertains to generating diverse objects (e.g., ImageNet [4] or Microsoft CoCo [8]), dynamic objects or scenes, and large-scale scenes.

# 5 Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Please refer to the supplementary files to access our code.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In addition to the descriptions provided in our paper and *Supplementary Material*, comprehensive training details can be found in the supplementary files of our code.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Our models cannot handle multiple runs due to computational infeasibility.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

3. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

# References


[1] 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[2] 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[3] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2009.

[5] Y. Deng, J. Yang, J. Xiang, and X. Tong. GRAM: Generative radiance manifolds for 3D-aware image generation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[6] J. Gu, L. Liu, P. Wang, and C. Theobalt. StyleNeRF: A style-based 3D-aware generator for high-resolution image synthesis. In *Int. Conf. Learn. Represent.*, 2021.

[7] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Eur. Conf. Comput. Vis.*, 2014.

[9] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for GANs do actually converge? In *Int. Conf. Mach. Learn.*, 2018.

[10] R. Or-El, X. Luo, M. Shan, E. Shechtman, J. J. Park, and I. Kemelmacher-Shlizerman. StyleSDF: High-resolution 3D-consistent image and geometry generation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[11] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Adv. Neural Inform. Process. Syst.*, 2021.

[12] Y. Xu, S. Peng, C. Yang, Y. Shen, and B. Zhou. 3D-aware image synthesis via learning structural and textural representations. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[13] W. Zhang, J. Sun, and X. Tang. Cat head detection-how to effectively exploit shape and texture features. In *Eur. Conf. Comput. Vis.*, 2008.