CodecNeRF: Toward Fast Encoding and Decoding, Compact, and High-quality Novel-view Synthesis

Gyeongjin Kang[1*], **Younggeun Lee**[2*], **Seungjun Oh**[2], **Eunbyung Park**[1, 2†]

[1]Department of Electrical and Computer Engineering, Sungkyunkwan University
[2]Department of Artificial Intelligence, Sungkyunkwan University
{ggggjin99, dudrms514, asxc1324, epark}@skku.edu

Abstract

Neural Radiance Fields (NeRF) have achieved huge success in effectively capturing and representing 3D objects and scenes. However, to establish an ubiquitous presence in everyday media formats, such as images and videos, we need to fulfill three key objectives: 1. fast encoding and decoding time, 2. compact model sizes, and 3. high-quality renderings. Despite recent advancements, a comprehensive algorithm that adequately addresses all objectives has yet to be fully realized. In this work, we present CodecNeRF, a neural codec for NeRF representations, consisting of an encoder and decoder architecture that can generate a NeRF representation in a single forward pass. Furthermore, inspired by the recent parameter-efficient finetuning approaches, we propose a finetuning method to efficiently adapt the generated NeRF representations to a new test instance, leading to high-quality image renderings and compact code sizes. The proposed CodecNeRF, a newly suggested encoding-decoding-finetuning pipeline for NeRF, achieved unprecedented compression performance of more than 100× and a remarkable reduction in encoding time while maintaining (or improving) the image quality on widely used 3D object datasets.

Code — https://gynjn.github.io/CodecNeRF

1 Introduction

Neural Radiance Fields (NeRF) have been enormously successful in representing 3D scenes (Mildenhall et al. 2021). Given a handful of pictures taken from various viewpoints, it generates photorealistic images from novel viewpoints, proving beneficial for various applications, such as 3D photography and navigation (Jampani et al. 2021; Adamkiewicz et al. 2022). In addition, ongoing research endeavors have enhanced its compatibility with conventional graphics rendering engines by enabling mesh and texture extraction (Baatz et al. 2022; Tang et al. 2023), and thus it further expands its usability. Moreover, recent 3D generation and editing techniques make it more valuable as a next-generation 3D media representation, opening new possibilities and applications.

The primary reason contributing to the longstanding success of image and video is the widespread adoption of standard codec software and hardware (Wiegand et al. 2003; Sullivan et al. 2012). We simply take a picture or video with our hand-held devices, and the encoder rapidly compresses the data. The encoded data are then transmitted over network communication channels, and the receivers can consume the data with the help of fast decoding software and hardware. We envision similar usage of 3D media using NeRF: 1) senders obtain multi-view images, 2) an encoder turns those images into a NeRF representation, 3) the encoded representation is communicated through the network, 4) receivers decode the encoded data and users enjoy the contents by rendering from various viewpoints. We urge the development of an algorithmic pipeline that can achieve rapid encoding and decoding speeds, compact data sizes, and high-quality view synthesis to support this common practice.

Despite considerable technological progress, there has yet to be a fully satisfying solution to achieve all of the stated goals. Training speed (encoding time) has remarkably advanced from days to a few hours or minutes (Chen et al. 2022; Sun, Sun, and Chen 2022; Müller et al. 2022; Fridovich-Keil et al. 2022, 2023). However, due to the inherent drawback of the per-scene optimization approach, they still require powerful GPU devices and at least tens of thousands of training iterations to converge. Encoder-decoder approaches, which generate NeRF in a single network forward pass, have been proposed (Chen et al. 2021; Yu et al. 2021). However, they primarily focus on few-shot generalization and do not consider the codec aspects, and the rendering image quality is limited compared to the optimization-based approaches. On the other hand, there has been extensive investigation into compact NeRF representations to minimize encoded data sizes (Takikawa et al. 2022; Tang et al. 2022; Li et al. 2023; Rho et al. 2023; Shin and Park 2024; Li et al. 2024). While successful, the suggested methods are mostly based on the per-scene optimization approach, resulting in longer training iterations.

In this work, we introduce CodecNeRF, a neural codec for NeRF designed to accomplish the previously mentioned objectives all at once. The proposed neural codec consists of a novel encoder and decoder architectures that can produce a NeRF representation in a single forward pass. The encoder takes multi-view images and produces compact codes

*These authors contributed equally.
†Corresponding author
Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

that are transmitted to other parties through network communications. The decoder that is present on both the sender and receiver sides generates the NeRF representations given the delivered codes. This forward-pass-only approach, as demonstrated numerous times by previous neural codecs for image and video, can achieve rapid encoding/decoding times and exceptional compression performance.

The forward pass alone, however, does not guarantee the high-quality novel-view synthesis. The primary issue stems from the scarcity of instances and diversities of the existing 3D datasets, in contrast to the abundance found in the image and video domains. This shortage hampers the trained models' capability to effectively generalize to new 3D test instances. Therefore, we propose to finetune the NeRF representations on the sender side and further transmit the finetuned 'delta' information to the receiver along with the codes. Then, the receiver decodes the transmitted codes to reproduce the initial NeRF representations and apply 'delta' to obtain the final NeRF representations. Since the initial NeRF representations from the forward pass are already well formed, the subsequent finetuning requires far fewer iterations than the per-scene optimization approach, which results in significantly faster encoding time.

To reduce the overall size of the final code (codes + finetuning 'delta'), we suggest parameter-efficient finetuning (PEFT) techniques on the initial NeRF representations (Hu et al. 2021). Finetuning the entire decoder or NeRF representations substantially increases the code sizes to be transmitted, negating the advantages of employing the encoder and decoder methodology. In this work, the NeRF representation is based on K-planes method consisting of multi-resolution plane features and an MLP network. We employ widely used low-rank adaptation (LoRA) methods for the MLP and suggest a novel PEFT technique for plane features inspired by the low-rank tensor decomposition method.

We have conducted comprehensive experiments using two representative 3D datasets, Objaverse (Deitke et al. 2023) and Google Scanned Objects (Downs et al. 2022). The experimental results show that the proposed encoder-decoder finetuning method, CodecNeRF, achieved $100\times$ more compression performance and significantly reduced encoding (training) time over the per-scene optimization baseline methods while maintaining the rendered image quality. Additionally, we evaluated the CodecNeRF's performance on real scenes using the DTU dataset (Jensen et al. 2014). We perceive this outcome as unlocking new research opportunities and application avenues using NeRF. The main contributions can be summarized as follows:

- We propose CodecNeRF, an encoder-decoder-finetuning pipeline for the newly emerging NeRF representation.
- We design novel 3D-aware encoder-decoder architectures, efficiently aggregating multi-view images, generating compact codes, and making NeRF representations from the codes.
- We present the parameter-efficient finetuning approach for further finetuning the NeRF representations that consist of MLP and feature planes.
- We achieved the unprecedented compression ratio and encoding speedup of NeRF while preserving high-quality rendering.

2 Related Works

Fast training NeRF. To reduce the computational complexity, grid-based representations have been suggested as an alternative to MLP. Plenoxels (Fridovich-Keil et al. 2022) constructed a sparse voxel grid with density and color value at each voxel explicitly. DVGO (Sun, Sun, and Chen 2022) and Instant-NGP (Müller et al. 2022) utilized voxel grids that store features and densities and employed a tiny MLP to compute the final output values. TensoRF (Chen et al. 2022) decomposed 3D grids in an axis-aligned manner via VM and CP decomposition for further improving the parameter efficiency. Inspired by EG3D (Chan et al. 2022), K-Planes (Fridovich-Keil et al. 2023) employed multi-scale orthogonal 2D planes, triplanes, showing scalability to higher dimensions while maintaining the speed advantage of grid-based representations. However, those per-scene optimization methods require numerous iterations to achieve high-quality novel view synthesis. In this work, we propose an encoder-decoder architecture to encode NeRF in a single-forward pass from multi-view input images. Furthermore, we utilize a low-rank decomposition scheme for efficient finetuning, showing fast convergence with few iterations.

Compact NeRF. Follow-up studies of NeRF aim to reduce storage size while preserving the performance of the original models. TensorRF (Chen et al. 2022) and CCNeRF (Tang et al. 2022) used tensor decomposition and low-rank approximation to reduce model size. Related to quantization methods, VQRF (Li et al. 2023) introduced trainable vector quantization method and VBNF (Takikawa et al. 2022) compressed feature grid by employing vector-quantized auto-decoder. Masked wavelet representation (Rho et al. 2023) applied wavelet transform on grid-based NeRF and quantization on coefficients with trainable mask and BiRF (Shin and Park 2024) proposed binary-based radiance field that quantizes each feature with binary values. Recently, NeRF-Codec (Li et al. 2024) achieved a high compression ratio by combining pretrained neural image codec and entropy coding. However, it requires per-scene optimization process to obtain the NeRF representations, which demands significant encoding time. Compared to the aforementioned methods, our model is a forward-pass-based approach that achieves fast encoding of 3D representations.

Neural codec for images and videos. A large body of works have explored the application of learning-based methods for compressing various types of data. In the image domain, along with CNN's remarkable property as a feature extractor, encoder-decoder based methods proposed by Ballé (Ballé, Laparra, and Simoncelli 2016; Ballé et al. 2018) are established as standard approaches. These models are combined with an entropy coding, such as (Rissanen and Langdon 1979; Huffman 1952), and trained to minimize the discretized code length while weighing the trade-offs between bit-rate and representation distortion. Learning-based video compression methods, expanded from image techniques, have incorporated time axis using optical flow (Lu

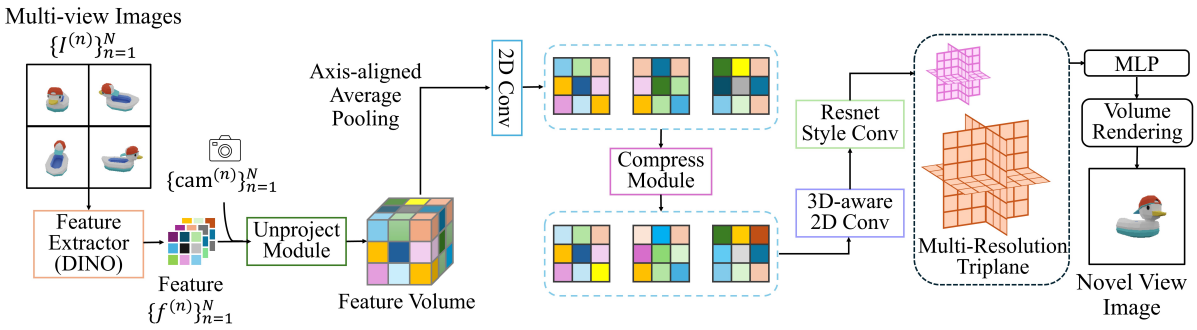

Figure 1: CodecNeRF encoder and decoder architecture.

et al. 2019), reference frames (Lin et al. 2020), and contextual learning (Li, Li, and Lu 2021). Inspired by neural compression methods in images and videos, we propose CodecNeRF, the first encoder-decoder based learned codec for NeRF, integrating neural codec with parameter-efficient fintuning in a novel way.

3 CodecNeRF

This section describes the proposed CodeNeRF pipeline with detailed architectures and finetuning methods. We explain the overall architecture (Sec. 3.1) first and present detailed methods in the following sections for each module: 3D feature construction (Sec. 3.2), 3D feature compression (Sec. 3.3), and multi-resolution triplanes (Sec. 3.4). Then, we present the training objectives used to train the proposed architecture (Sec. 3.5) and the parameter-efficient finetuning method for generating compact codes (Sec. 3.6 and 3.7).

3.1 Overall architecture

Fig. 1 depicts the overall encoder and decoder architecture of CodecNeRF. Given N input images from different viewpoints, $\{I^{(n)}\}_{n=1}^{N}$, the goal is to produce a NeRF representation (multi-resolution triplanes). First, a 2D image feature extractor module, $\texttt{feat}_\theta$, processes all input images and generates 2D feature maps for each input image, $\{f^{(n)}\}_{n=1}^{N}$. Then, the unproject and aggregation module, $\texttt{unproj}$ and $\texttt{agg}_\phi$, lifts the 2D features to 3D features and aggregates the unprojected 3D features into a single 3D feature, $f_{3D} \in \mathbb{R}^{C\times V\times V\times V}$ (to avoid clutter notation, we assume height, width, and depth resolutions are same, V). The 3D feature, f_{3D}, is further processed by axis-aligned average pooling along each axis, resulting in three 2D features (a 2D feature for each axis), $f_{xy}, f_{yz}, f_{xz} \in \mathbb{R}^{C\times V\times V}$. These 2D features are used to generate multi-resolution triplanes by $\texttt{triplane}_\psi$, and finally, we perform the volumetric rendering to render an image using $\texttt{MLP}_\omega$. Furthermore, 2D features f_{xy}, f_{yz}, and f_{xz} are compressed by the compression module, $\texttt{comp}_\chi$, producing the minimal sizes of the codes to be transmitted. The entire pipeline is differentiable, and we train end-to-end to optimize all learnable parameters, $\{\theta, \phi, \chi, \psi, \omega\}$.

3.2 3D feature from multi-view images

In this submodule, we construct the 3D feature from multi-view input images. To extract 2D image features, we adopt a pre-trained visual transformer (ViT) (Dosovitskiy et al. 2020), specifically, DINO (Caron et al. 2021) to produce an image features $f^{(n)}$ given an input image $I^{(n)}$. We process each view image individually using the shared feature extractor, $\texttt{feat}_\theta$. Following the conventional NeRF training scheme, we assume that we can obtain camera poses for input view images beforehand. The 3D feature construction can be written as follows.

$$f_{3D} = \texttt{agg}_\phi(\{\texttt{unproj}(f^{(n)}, \text{cam}^{(n)}, \text{coord})\}_{n=1}^{N}), \quad (1)$$

where $\text{cam}^{(n)}$ denotes the camera pose for the input views. Inspired by the recent unprojection methods (Liu et al. 2024, 2023), we first construct a 3D coordinate tensor, coord $\in \mathbb{R}^{3\times V\times V\times V}$, whose resolution is V for all axis. Then, each coordinate is projected into 2D space given the camera pose, and the feature is extracted from the image feature $f^{(n)}$ using bilinear interpolation, generating the intermediate 3D feature. We use an aggregation module $\texttt{agg}_\phi$ paramaterized ϕ to combine N intermediate 3D features and produce the final 3D feature, $f_{3D} \in \mathbb{R}^{C\times V\times V\times V}$. We use a few 3D convolution layers to aggregate features and further extract useful information.

3.3 3D feature compression

The goal of 3D feature compression is to minimize the number of bits required to reconstruct the final NeRF representations, and f_{3D} from the previous stage is 3D volume, thus inefficient for storage and transmission purposes. In this work, we opt to use explicit-implicit hybrid NeRF representation, triplane (Chan et al. 2022). Triplane representation decomposes a 3D volume into three 2D planes, serving as a prevalent technique for the NeRF representations. It scales with $O(V^2)$ for the resolution V as opposed to $O(V^3)$ for a dense 3D volume.

We first transform the 3D feature into three 2D features by average pooling along each axis.

$$f_{xy}=\texttt{ap-z}(f_{3D}), f_{yz}=\texttt{ap-x}(f_{3D}), f_{xz}=\texttt{ap-y}(f_{3D}), \quad (2)$$

where $\texttt{ap-x}$ means average pooling along x axis. Then, $\texttt{comp}_\chi$ compresses the three 2D feature maps using vector quantization methods (Van Den Oord, Vinyals et al. 2017). It consists of a downsampling CNN, an upsampling CNN, and a codebook (χ includes all parameters in these three modules). First the downsampling 2D CNN module process each 2D feature map to generate low-resolution 2D feature map, $l_{xy}, l_{yz}, l_{xz} \in \mathbb{R}^{C' \times V' \times V'}$ ($C' << C$ and $V' << V$). Then, we find the closest code from the codebook to perform the vector quantization.

$$\bar{l}_{xy,i,j} = e_{\underset{k}{\text{argmin}} \|l_{xy,i,j} - e_k\|_2}, \tag{3}$$

where $e \in \mathbb{R}^{K \times C'}$ is the codebook, K is the codebook size, $e_k \in \mathbb{R}^{C'}$ denotes the k-th element of the codebook, $l_{xy,i,j} \in \mathbb{R}^{C'}$ denotes the element of l_{xy} indexed by (i, j) location, and $\bar{l}_{xy,i,j}$ is the vector quantized 2D feature map. During training, we optimize the codebook e, and the loss function for a training instance can be written as,

$$\mathcal{L}_{\text{vq}} = \left\| \text{sg}\left[l\right] - \bar{l} \right\|_2^2 + \lambda_{\text{commit}} \left\| \text{sg}[\bar{l}] - l \right\|_2^2, \tag{4}$$

where $\text{sg}\left[\cdot\right]$ is the stop-gradient operator, and λ_{commit} regulate the commitment to codebook embedding. With the slight abuse of notation, here we define $l, \bar{l} \in \mathbb{R}^{3 \times C' \times V' \times V'}$ as the concatenated tensor of three low-resolution 2D feature maps. Finally, the upsampling CNN produces three 2D feature maps with the increased resolutions, $\bar{f}_{xy}, \bar{f}_{yz}, \bar{f}_{xz} \in \mathbb{R}^{C \times V \times V}$. During training, the input to the upsampling CNN is l, but $\bar{l}$ is used during testing.

3.4 Multi-resolution triplanes

The previous works (Müller et al. 2022; Fridovich-Keil et al. 2023; Nam et al. 2024) have shown that using a multi-resolution representation efficiently encodes spatial features at different scales. It encourages spatial smoothness across different scales, superior convergence, and better accuracy. Building upon these observations, we propose a hierarchical 3D-aware convolution block, $\texttt{triplane}_\psi$, that generates a multi-resolution triplanes revised from the one introduced in (Wang et al. 2023; Wu et al. 2023). It introduces rolled-out triplanes that attend to all components from the relevant rows and columns, enabling cross-plane feature interaction.

$$(\tilde{f}_{xy}, \tilde{f}_{yz}, \tilde{f}_{xz}) = \texttt{triplane}_\psi(\bar{f}_{xy}, \bar{f}_{yz}, \bar{f}_{xz}), \tag{5}$$

where $\tilde{f}_{xy} = \{\tilde{f}^1_{xy}, \tilde{f}^2_{xy}\}$ is a set of multi-resolution triplane features for 'xy' plane, and $\tilde{f}^1_{xy} \in \mathbb{R}^{C \times V_1 \times V_1}$ and $\tilde{f}^2_{xy} \in \mathbb{R}^{C \times V_2 \times V_2}$ are different resolution features.

The proposed triplane renderer consists of two distinct MLP heads, coarse and fine, for decoding the RGBs and densities. Given a 3D coordinate $p \in \mathbb{R}^3$, the decoder collects the triplane features at three axis-aligned projected locations of $p_{xy}, p_{yz}, p_{xz} \in \mathbb{R}^2$, using bilinear interpolation. We simply concat the triplane features across the different scales and aggregate by summation.

$$f_{\text{tri}}(p) = \sum_{k \in \{xy,yz,xz\}} \texttt{concat}(\texttt{itp}(\tilde{f}^1_k, p_k), \texttt{itp}(\tilde{f}^2_k, p_k)), \tag{6}$$

$$c(p, d), \sigma(p) = \texttt{MLP}_\omega(f_{\text{tri}}(p), p, \texttt{PE}(d)), \tag{7}$$

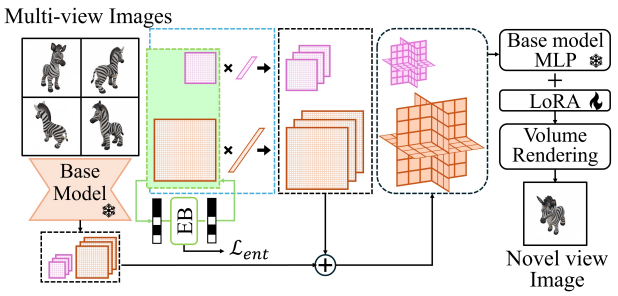

Figure 2: Parameter-efficient finetuning process.

where $\texttt{itp}(\cdot, \cdot)$ bilinearly interpolates the features given the projected 2D coordinates, $\texttt{concat}(\cdot)$ concatenate the interpolated features, $f_{\text{tri}}(p) \in \mathbb{R}^{3C}$ is the feature to be processed by an MLP network to generate $c(p)$ and $\sigma(p)$, the color and density of a point. $\texttt{PE}(d)$ is the view direction after applying the positional encoding. Finally, the volume rendering (Max 1995) is applied to render the images using the two-pass hierarchical importance sampling method proposed by NeRF (Mildenhall et al. 2021).

3.5 Training objective

Here, we train our base model in an end-to-end manner with its fully differentiable properties. We use L2 loss, denoted as $\mathcal{L}_{\text{rgb}}$ to measure the pixel-level difference and LPIPS (Zhang et al. 2018) loss, denoted as $\mathcal{L}_{\text{lpips}}$ to measure the patch-level difference between the ground truth images and rendered images.

Spatial total variation (TV) regularization encourages the sparse or smooth gradient, thereby ensuring that the feature planes do not contain erroneous high-frequency data (Chen et al. 2022; Fridovich-Keil et al. 2023). We use the standard L2 TV regularization as default to make the distribution of the triplane features smoother, as it regularizes the squared difference between the neighboring values in the feature maps.

$$\mathcal{L}_{\text{tv}} = \frac{1}{T} \sum_{k \in \{xy,yz,xz\}} \sum_{s=1}^{2} \sum_{i,j} \left(\left\| \tilde{f}^s_{k,i,j} - \tilde{f}^s_{k,i-1,j} \right\|_2^2 + \left\| \tilde{f}^s_{k,i,j} - \tilde{f}^s_{k,i,j-1} \right\|_2^2 \right) \tag{8}$$

where $T = 3C(V_1^2 + V_2^2)$ is the total number of features across all triplanes and resolutions. The final objective function for a training instance can be written as follows,

$$\mathcal{L} = \mathcal{L}_{\text{rgb}} + \mathcal{L}_{\text{vq}} + \lambda_{\text{lpips}} \mathcal{L}_{\text{lpips}} + \lambda_{\text{tv}} \mathcal{L}_{\text{tv}}. \tag{9}$$

3.6 Parameter-efficient finetuning

While the NeRF representations generated by the encoder and decoder modules are of high quality, their generalization performance on a new scene can be limited. Similar to other NeRF generalization models (Tancik et al. 2021; Bergman, Kellnhofer, and Wetzstein 2021), our approach can also leverage the finetuning of NeRF representations to

Data	Method	Iterations													
		0			500				1000				2000			
		PSNR	SSIM	MSIM	PSNR	SSIM	MSIM	SIZE	PSNR	SSIM	MSIM	SIZE	PSNR	SSIM	MSIM	SIZE
Obj	Triplanes	11.36	0.784	0.330	22.23	0.863	0.887	8.077	25.27	0.904	0.957	8.077	26.56	0.921	0.959	8.077
	Ours (PEFT)	22.45	0.871	0.901	25.95	0.902	0.945	1.024	27.61	0.921	0.961	1.024	28.57	0.932	0.968	1.024
	Ours (PEFT++)	·	·	·	25.79	0.901	0.943	0.197	27.35	0.918	0.959	0.179	28.28	0.929	0.967	0.146
GSO	Triplanes	12.55	0.836	0.364	28.32	0.940	0.964	8.077	30.13	0.955	0.970	8.077	31.54	0.964	0.975	8.077
	Ours (PEFT)	23.98	0.892	0.914	31.90	0.952	0.981	1.024	33.35	0.961	0.986	1.024	34.65	0.968	0.990	1.024
	Ours (PEFT++)	·	·	·	31.38	0.949	0.978	0.188	32.70	0.958	0.984	0.172	33.87	0.965	0.988	0.145

Table 1: Quantitative results of the proposed methods evaluated on Objaverse (denoted by 'Obj') and GSO datasets. 'PEFT++' denotes parameter efficient finetuning with entropy coding, 'MSIM' is MS-SSIM and 'SIZE' is measured in MB.

enhance visual quality for new scenes during testing time. However, effective model finetuning is severely hindered by the growing computational costs and memory storage as the model size increases. To tackle this issue, LoRA (Hu et al. 2021) is a widely used parameter-efficient finetuning (PEFT) method for adaptation of large-scale models, mainly explored in NLP and computer vision domains. We propose to adapt PEFT in our test time optimization, and to the best of our knowledge, we are the first to apply PEFT to 3D NeRF representation. We first generate multi-resolution triplanes using multi-view test images, and train only the triplanes and decoder in an efficient way.

Parameter efficient triplane finetuning. We propose a tensor factorization scheme to efficiently finetune triplane representation. Let $\tilde{f}_k^s \in \mathbb{R}^{C\times V_s\times V_s}$, be the triplanes generated by the encoder and decoder for a scale 's' and plane 'k' ($s \in \{1, 2\}$ and $k \in \{xy, yz, xz\}$). The final triplane representations are expressed by $\tilde{f}_k^s + \Delta\tilde{f}_k^s$, and $\Delta\tilde{f}_k^s \in \mathbb{R}^{C\times V_s\times V_s}$ is constructed by a tensor product between matrices and vectors.

$$\Delta\tilde{f}_k^s = \sum_{r=1}^{R} v_r^s \circ M_{k,r}^s, \tag{10}$$

where $M_{k,r}^s \in \mathbb{R}^{V_s\times V_s}$ denotes r-th matrix for the 'k' plane and scale 's', $v_r^s \in \mathbb{R}^C$ is the r-th vector for all three planes and scale 's', and $\circ : \mathbb{R}^C \times \mathbb{R}^{V_s\times V_s} \to \mathbb{R}^{C\times V_s\times V_s}$ is a tensor product. During finetuning, we freeze $\tilde{f}_k^s$ and only updates $\Delta\tilde{f}_k^s$. We apply this scheme for every feature planes in multi-resolution triplanes and used $R = 4$. For initialization, we use a common technique, setting all matrices to random values and all vectors to zeros. It makes our delta to be zero at the start of the training.

Parameter efficient MLP finetuning. Additionally, we factorize MLP layers in decoders using the LoRA method for MLP finetuning. Using two PEFT methods, we can achieve massive reductions in trainable parameters during test time optimization (Fig. 2). The training objective is the same with the base model except for the vector quantization and LPIPS losses.

3.7 Entropy coding finetuning deltas

We leverage neural compression methods that have demonstrated efficacy in image and video domains to seek to achieve the optimal compression rate (Ballé, Laparra, and Simoncelli 2016; Ballé et al. 2018; Li, Li, and Lu 2021; Lin et al. 2020). We adopt a entropy coding model (Ballé et al. 2018) to our proposed parameter-efficient finetuning of the triplanes. We model the prior using a non-parametric density, which is convolved with a standard uniform density in a differentiable manner (please refer to (Ballé et al. 2018) for more details). Then, our training objective for the finetuning is defined as follows.

$$\mathcal{L}_{\text{ent}} = \sum_{k\in\{xy,yz,xz\}} \sum_{r=1}^{R}\sum_{s=1}^{2} -\log p(M_{k,r}^s), \tag{11}$$

$$\mathcal{L} = \mathcal{L}_{\text{rgb}} + \lambda_{\text{rate}}\mathcal{L}_{\text{ent}}, \tag{12}$$

where λ_{rate} will balance between the quantization error and the code length. We only applied the entropy model to the feature matrices for triplane features, and after finetuning, updated matrices $M_{k,r}^s$ are quantized and compressed, while other weights, $\omega_a, \omega_b, v_r^s$ are stored with 32-bit precision as a convention.

4 Experiments

4.1 Datasets

To evaluate our method, we conduct experiments on 1) Objaverse (Deitke et al. 2023) and Google Scanned Objects (GSO) (Downs et al. 2022) for object-level novel view synthesis, and 2) DTU dataset (Jensen et al. 2014) for real scenes. For Objaverse, we sourced images from One-2-3-45 (Liu et al. 2024), which consists of 46k objects, and constructed our own split of 36,796 training objects and 9,118 test objects. In GSO, we used 1,030 objects only for the evaluation. Lastly, we followed PixelNeRF (Yu et al. 2021) DTU dataset split with 88 training scenes and 15 testing scenes. Object images are rendered at a resolution of 256×256 and we cropped the DTU images to match the same resolution.

4.2 Implementation Details

To train our base model, we randomly choose 16 input images and camera poses to produce a triplane representation and predict the remaining novel views. We used two spatial resolutions $\{V_1, V_2\}$ = {64, 128} and channel size C = 32 for our multi-resolution triplanes. The MLP decoders are of 6 layers with hidden dimensions 64 for coarse and fine decoders, respectively. We set the codebook size $K = 8192$ and dimension $C' = 32$. In the finetuning stage, we first generated an initial representation using predetermined 16 view indices and finetuned the triplane features with MLP decoders in an optimization-based approach. We finetuned the model using 24 images and tested it on the remaining

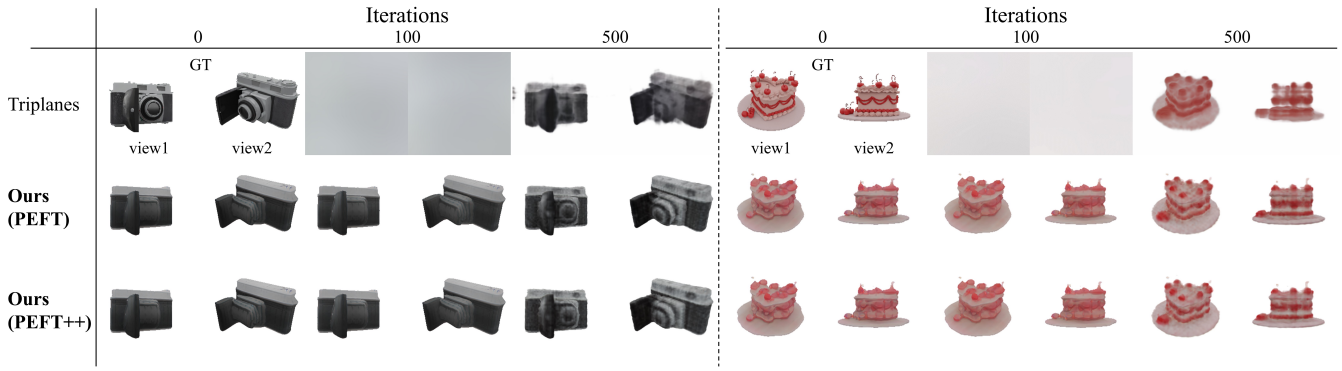

Figure 3: Novel view synthesis results on Objaverse dataset.

views. Note that the 16 images used to generate the initial triplane representations are a subset of the 24 training images, and the same images are all used to train baselines for a fair comparison. For the DTU dataset, we choose 8 input images for base model training and 16 images for finetuning. We set the LoRA's rank to 4 in every layer of the decoder.

4.3 Results

Object-level Benchmarks. To assess the efficacy of our method in test time optimization, we employed K-planes (Fridovich-Keil et al. 2023) as our baseline model, which has shown fast training and compact representation. We revised the architecture based on our method for a fair comparison, and this model will be referred to as 'Triplanes'. We evaluated the performance under two different scenarios starting from our generated triplane initializations: 1) parameter-efficient finetuning (PEFT) and 2) parameter-efficient finetuning with the proposed entropy coding method (PEFT++).

For the quantitative metrics, we report the standard image quality measurements, PSNR, SSIM (Wang et al. 2004), and MS-SSIM (Wang, Simoncelli, and Bovik 2003). We also measure the storage requirements of the representations to show the compression performance of our method. As shown in Tab. 1, our method shows fast encoding progress from its initialized representation and improvement on all metrics. Thanks to the pretrained generalization capability of our method, our model outperforms the per-scene optimization baseline, Triplanes, in novel view synthesis. The qualitative results in Fig. 3 present the novel view synthesis results across the finetuning iteration, illustrating the generalization ability and fast encoding speed of our methods. In Tab. 2, we report the component-level storage comparison (on the Objaverse dataset) after 10K iterations. Our method achieved 100× compactness with better quality compared to the baseline model.

We also compared our model with a large reconstruction model, GeoLRM (Zhang et al. 2024), a Gaussian-based reconstruction model. Since GeoLRM has the property that can scale up to dense views, we choose it as the baseline for a fair comparison with our model. We utilized the GSO dataset, and the same 16 views were used as input, and the remaining views were used to evaluate the performance. Tab. 3 shows that our model outperforms the baseline on PSNR and SSIM while often missing the detailed visual quality as depicted in Fig. 4. We suspect that the small code size in the bottleneck layer incurs the difficulty in providing enough details. Nevertheless, after finetuning with our proposed method, we can attain high-quality 3D representations quickly with a very compact size, as shown in Tab. 1 and Fig. 4. We evaluated all objects in Tab. 3, and 100 objects in Tab. 1

Component	Total size in MB (codes + finetuning deltas)			
	Triplanes	PEFT	PEFT++	W/O FT
Codes	.	0.013	0.013	0.013
Feature	7.864	0.984	0.031	.
MLP	0.213	0.027	0.027	.
Total	8.077	1.024	0.071	0.013

Table 2: Memory footprint in component level.

	PSNR	SSIM	MS-SSIM	Codes (MB)
GeoLRM	23.73	0.890	0.922	.
Ours	24.49	0.897	0.918	0.015

Table 3: Initialization performance on GSO dataset.

Scene-level Benchmarks. We further demonstrate the applicability of our method on real scenes using the DTU dataset. Two representative optimization-based methods, TensoRF (Chen et al. 2022) and 3D-GS (Kerbl et al. 2023), are used as baselines. As shown in Tab. 4, our method was finetuned for less than a minute to evaluate its fast encoding ability. Fig. 5 presents the novel view synthesis results on the DTU dataset. Both quantitative and qualitative results indicate that our method surpasses TensoRF and 3D-GS in terms of quality, even with a compact representation size.

In-depth Evaluations. We conducted a detailed comparison of our method with both fast and compact specialized models on Objaverse dataset. For fast training NeRF, we adopted TensoRF (Chen et al. 2022), DVGO (Sun, Sun, and Chen 2022), and Plenoxels (Fridovich-Keil et al. 2022) and for compact NeRF, we employed CCNeRF (Tang et al. 2022), MDWT (Rho et al. 2023), VQRF (Li et al. 2023),

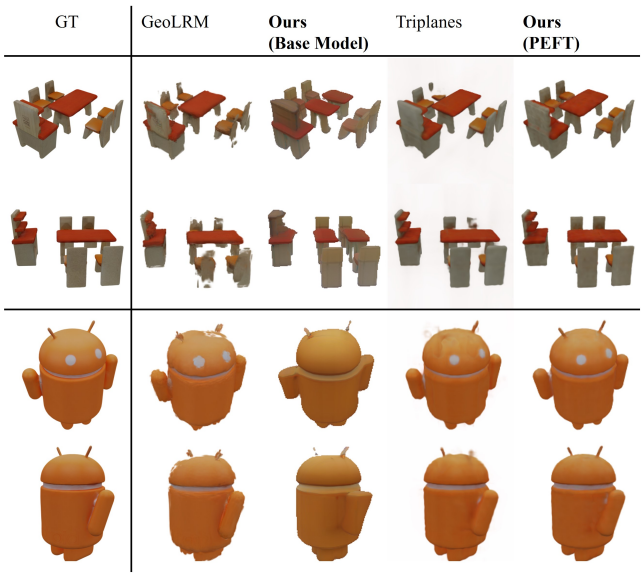

Figure 4: Novel view synthesis results on GSO dataset. 'Triplanes' and 'Ours (PEFT)' is finetuned with 1K iterations.

Method	PSNR	SSIM	Train (s)	Size (MB)
TensoRF	14.58	0.517	57.3	5.430
3D-GS	15.77	0.581	64.0	156.9
Ours (PEFT)	20.05	0.650	41.3	1.023
Ours (PEFT++)	20.07	0.646	58.8	0.160

Table 4: Perfromance comparison on DTU dataset.

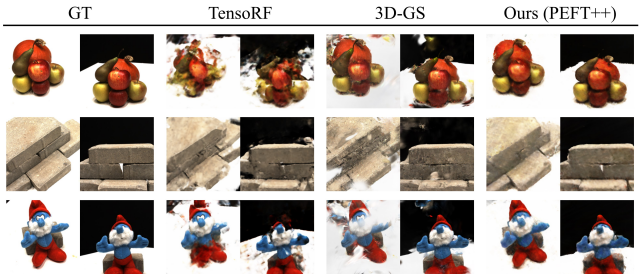

Figure 5: Novel view synthesis results on DTU dataset.

and BiRF (Shin and Park 2024). We showed different compression rates by varing the rank size in decoder's LoRA (1, 4, and 8). As illustrated in Fig. 6, our method demonstrates the ability to achieve fast encoding and a high compression ratio, outperforming representative models in both aspects.

Feature Visualization We also visualized the delta feature maps across finetuning iterations based on our entropy coding method. As shown in Fig. 7, the feature maps following the entropy coding, eliminate unnecessary components across the different resolutions, and get a high compression ratio resulting from quantization. This observation aligns with our intuition that employing multiple spatial resolutions would help reduce the quantity of information at each level, thus making the use of entropy models an ideal strategy.

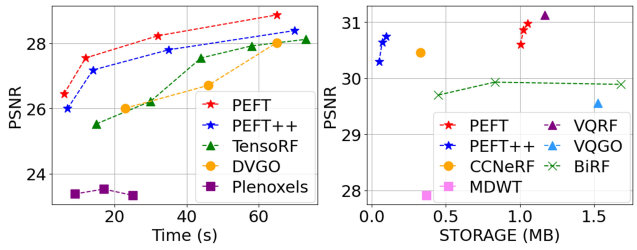

Figure 6: In-depth evaluations. Fast training NeRF (right graph) and Compact NeRF (left graph).

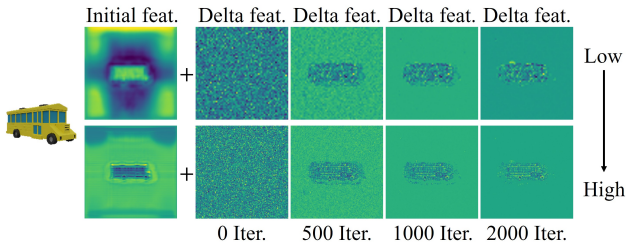

Figure 7: Visualization of delta feature maps finetuned with entropy coding. The averaged YZ planes across the channel dimensions are shown in the different resolutions.

5 Limitations and Future Works

While CodecNeRF demonstrates promising performance in encoding speed and compression ratio, it still has limitations. First, technical enhancements are needed to handle complex scenes, such as large environments (e.g., Mip-NeRF 360 datasets (Barron et al. 2022)). Exploring block-wise or hierarchical coding and training on large 3D datasets could improve adaptability. Second, supporting other representations, like i-NGP or 3D-GS, requires modifications, such as point-based neural encoders and decoders. Enhancing rendering quality and encoding speed may involve larger architectures and incorporating learned 2D priors (Radford et al. 2021; Rombach et al. 2022) as supervision. Finally, advanced neural codecs and weight-pruning techniques could further optimize compression performance.

6 Conclusion

In this work, we introduced CodecNeRF, a novel encoding-decoding-finetuning pipeline designed for fast encoding and decoding, compact codes, and high-quality renderings. To our knowledge, this is the first attempt to propose a neural learned codec for emerging 3D representations, such as NeRF. Our experimental results demonstrated a significant performance improvement over strong NeRF baseline models across commonly used 3D object datasets, including Objaverse and Google Scanned Objects, as well as the scene-level DTU dataset. We believe that our framework has the potential to uncover new research avenues and expand the applications of NeRF, demonstrating compelling evidence that 3D representation can be encoded with comparable efficiency to traditional image and video data.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (RS-2019-II190421, Artificial Intelligence Graduate School Program(Sungkyunkwan University)) and the National Research Foundation (NRF) grant (RS-2024-00337548). This work was also supported by the Culture, Sports, and Tourism R&D Program through the Korea Creative Content Agency grant funded by the Ministry of Culture, Sports and Tourism in 2024 (Project Name: Research on neural watermark technology for copyright protection of generative AI 3D content, RS-2024-00348469), and Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-IT2401-01.

References

Adamkiewicz, M.; Chen, T.; Caccavale, A.; Gardner, R.; Culbertson, P.; Bohg, J.; and Schwager, M. 2022. Vision-only robot navigation in a neural radiance world. *IEEE Robotics and Automation Letters*, 7(2): 4606–4613.

Baatz, H.; Granskog, J.; Papas, M.; Rousselle, F.; and Novák, J. 2022. NeRF-Tex: Neural Reflectance Field Textures. In *Computer Graphics Forum*, volume 41, 287–301. Wiley Online Library.

Ballé, J.; Laparra, V.; and Simoncelli, E. P. 2016. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*.

Ballé, J.; Minnen, D.; Singh, S.; Hwang, S. J.; and Johnston, N. 2018. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*.

Barron, J. T.; Mildenhall, B.; Verbin, D.; Srinivasan, P. P.; and Hedman, P. 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5470–5479.

Bergman, A.; Kellnhofer, P.; and Wetzstein, G. 2021. Fast training of neural lumigraph representations using meta learning. *Advances in Neural Information Processing Systems*, 34: 172–186.

Caron, M.; Touvron, H.; Misra, I.; Jégou, H.; Mairal, J.; Bojanowski, P.; and Joulin, A. 2021. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, 9650–9660.

Chan, E. R.; Lin, C. Z.; Chan, M. A.; Nagano, K.; Pan, B.; Mello, S. D.; Gallo, O.; Guibas, L.; Tremblay, J.; Khamis, S.; Karras, T.; and Wetzstein, G. 2022. Efficient Geometry-aware 3D Generative Adversarial Networks. In *CVPR*.

Chen, A.; Xu, Z.; Geiger, A.; Yu, J.; and Su, H. 2022. Tensorf: Tensorial radiance fields. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, 333–350. Springer.

Chen, A.; Xu, Z.; Zhao, F.; Zhang, X.; Xiang, F.; Yu, J.; and Su, H. 2021. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 14124–14133.

Deitke, M.; Schwenk, D.; Salvador, J.; Weihs, L.; Michel, O.; VanderBilt, E.; Schmidt, L.; Ehsani, K.; Kembhavi, A.; and Farhadi, A. 2023. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13142–13153.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Downs, L.; Francis, A.; Koenig, N.; Kinman, B.; Hickman, R.; Reymann, K.; McHugh, T. B.; and Vanhoucke, V. 2022. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, 2553–2560. IEEE.

Fridovich-Keil, S.; Meanti, G.; Warburg, F. R.; Recht, B.; and Kanazawa, A. 2023. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12479–12488.

Fridovich-Keil, S.; Yu, A.; Tancik, M.; Chen, Q.; Recht, B.; and Kanazawa, A. 2022. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5501–5510.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Huffman, D. A. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9): 1098–1101.

Jampani, V.; Chang, H.; Sargent, K.; Kar, A.; Tucker, R.; Krainin, M.; Kaeser, D.; Freeman, W. T.; Salesin, D.; Curless, B.; et al. 2021. Slide: Single image 3d photography with soft layering and depth-aware inpainting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 12518–12527.

Jensen, R.; Dahl, A.; Vogiatzis, G.; Tola, E.; and Aanæs, H. 2014. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 406–413.

Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4).

Li, J.; Li, B.; and Lu, Y. 2021. Deep contextual video compression. *Advances in Neural Information Processing Systems*, 34: 18114–18125.

Li, L.; Shen, Z.; Wang, Z.; Shen, L.; and Bo, L. 2023. Compressing volumetric radiance fields to 1 mb. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4222–4231.

Li, S.; Li, H.; Liao, Y.; and Yu, L. 2024. NeRFCodec: Neural Feature Compression Meets Neural Radiance Fields for Memory-Efficient Scene Representation. In *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 21274–21283.

Lin, J.; Liu, D.; Li, H.; and Wu, F. 2020. M-LVC: Multiple frames prediction for learned video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3546–3554.

Liu, M.; Xu, C.; Jin, H.; Chen, L.; Varma T, M.; Xu, Z.; and Su, H. 2024. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *Advances in Neural Information Processing Systems*, 36.

Liu, Y.; Lin, C.; Zeng, Z.; Long, X.; Liu, L.; Komura, T.; and Wang, W. 2023. Syncdreamer: Generating multiview-consistent images from a single-view image. *arXiv preprint arXiv:2309.03453*.

Lu, G.; Ouyang, W.; Xu, D.; Zhang, X.; Cai, C.; and Gao, Z. 2019. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11006–11015.

Max, N. 1995. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2): 99–108.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1): 99–106.

Müller, T.; Evans, A.; Schied, C.; and Keller, A. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4): 1–15.

Nam, S.; Rho, D.; Ko, J. H.; and Park, E. 2024. Mip-grid: Anti-aliased grid representations for neural radiance fields. *Advances in Neural Information Processing Systems*, 36.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PMLR.

Rho, D.; Lee, B.; Nam, S.; Lee, J. C.; Ko, J. H.; and Park, E. 2023. Masked Wavelet Representation for Compact Neural Radiance Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20680–20690.

Rissanen, J.; and Langdon, G. G. 1979. Arithmetic coding. *IBM Journal of research and development*, 23(2): 149–162.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10684–10695.

Shin, S.; and Park, J. 2024. Binary radiance fields. *Advances in Neural Information Processing Systems*, 36.

Sullivan, G. J.; Ohm, J.-R.; Han, W.-J.; and Wiegand, T. 2012. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12): 1649–1668.

Sun, C.; Sun, M.; and Chen, H.-T. 2022. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5459–5469.

Takikawa, T.; Evans, A.; Tremblay, J.; Müller, T.; McGuire, M.; Jacobson, A.; and Fidler, S. 2022. Variable bitrate neural fields. In *ACM SIGGRAPH 2022 Conference Proceedings*, 1–9.

Tancik, M.; Mildenhall, B.; Wang, T.; Schmidt, D.; Srinivasan, P. P.; Barron, J. T.; and Ng, R. 2021. Learned initializations for optimizing coordinate-based neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2846–2855.

Tang, J.; Chen, X.; Wang, J.; and Zeng, G. 2022. Compressible-composable nerf via rank-residual decomposition. *Advances in Neural Information Processing Systems*, 35: 14798–14809.

Tang, J.; Zhou, H.; Chen, X.; Hu, T.; Ding, E.; Wang, J.; and Zeng, G. 2023. Delicate textured mesh recovery from nerf via adaptive surface refinement. *arXiv preprint arXiv:2303.02091*.

Van Den Oord, A.; Vinyals, O.; et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Wang, T.; Zhang, B.; Zhang, T.; Gu, S.; Bao, J.; Baltrusaitis, T.; Shen, J.; Chen, D.; Wen, F.; Chen, Q.; et al. 2023. Rodin: A generative model for sculpting 3d digital avatars using diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4563–4573.

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. P. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4): 600–612.

Wang, Z.; Simoncelli, E. P.; and Bovik, A. C. 2003. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, 1398–1402. Ieee.

Wiegand, T.; Sullivan, G. J.; Bjontegaard, G.; and Luthra, A. 2003. Overview of the H. 264/AVC video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7): 560–576.

Wu, R.; Liu, R.; Vondrick, C.; and Zheng, C. 2023. Sin3dm: Learning a diffusion model from a single 3d textured shape. *arXiv preprint arXiv:2305.15399*.

Yu, A.; Ye, V.; Tancik, M.; and Kanazawa, A. 2021. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4578–4587.

Zhang, C.; Song, H.; Wei, Y.; Chen, Y.; Lu, J.; and Tang, Y. 2024. Geolrm: Geometry-aware large reconstruction model for high-quality 3d gaussian generation. *arXiv preprint arXiv:2406.15333*.

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 586–595.