NeRFCodec: Neural Feature Compression Meets Neural Radiance Fields for Memory-Efficient Scene Representation

Sicheng Li Hao Li Yiyi Liao* Lu Yu**

Zhejiang University

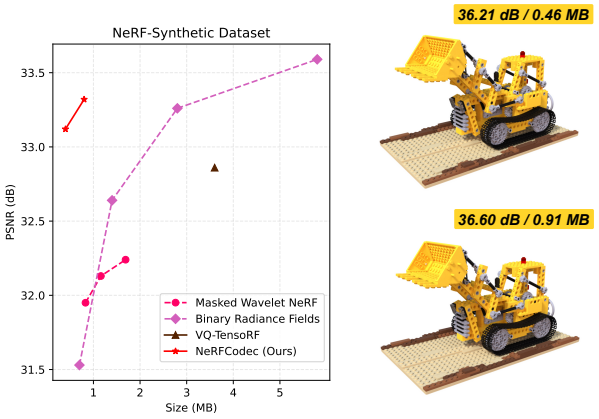
Abstract

The emergence of Neural Radiance Fields (NeRF) has greatly impacted 3D scene modeling and novel-view synthesis. As a kind of visual media for 3D scene representation, compression with high rate-distortion performance is an eternal target. Motivated by advances in neural compression and neural field representation, we propose NeRFCodec, an end-to-end NeRF compression framework that integrates non-linear transform, quantization, and entropy coding for memory-efficient scene representation. Since training a non-linear transform directly on a large scale of NeRF feature planes is impractical, we discover that pre-trained neural 2D image codec can be utilized for compressing the features when adding content-specific parameters. Specifically, we reuse neural 2D image codec but modify its encoder and decoder heads, while keeping the other parts of the pre-trained decoder frozen. This allows us to train the full pipeline via supervision of rendering loss and entropy loss, yielding the rate-distortion balance by updating the content-specific parameters. At test time, the bitstreams containing latent code, feature decoder head, and other side information are transmitted for communication. Experimental results demonstrate our method outperforms existing NeRF compression methods, enabling high-quality novel view synthesis with a memory budget of 0.5 MB.

1. Introduction

Neural Radiance Fields (NeRF) [28] have emerged as a popular scene representation for novel view synthesis. As a promising representation for immersive media, how to compress NeRF with a better storage-quality trade-off is a significant problem for efficient communication and storage.

While the deep MLPs used in NeRF [28] are parameter-efficient, the hybrid representation [36, 43], which combines feature grids and small MLPs, has become the mainstream due to its high reconstruction quality, fast training speed, and efficient rendering. However, its drawback lies in the significant storage requirements. Consequently, subsequent research efforts have emerged to reduce the storage footprint of the hybrid representation without compromising reconstruction quality.



(a) Rate-distortion performance (b) Rendered images

Figure 1. **Compression performance**.

One line of work focuses on efficient data structure design of feature grids, which involves using more parameter-efficient data structures, e.g., tensor factorization-based representations [8, 9] and multi-resolution hash grids [30], to replace dense voxel grids. These methods allow for reducing the number of parameters, e.g., from 1GB to 50MB, without sacrificing the quality. However, this size still requires further reduction.

Another line of work focuses on compressing parameters using compression techniques like quantization [24, 33, 37] and entropy coding [24]. However, most works in this field overlook another effective compression method, transform coding [18], that transforms the original data to another space based on linear [23, 40] or non-linear mapping [2]. The image compression community demonstrates that the combination of transform coding, quantization, and entropy coding leads to the most competitive compression performance [6, 34, 35]. Recently, there is a work named Masked Wavelet NeRF [32], which is the first to consider trans-

* Corresponding author. ** Co-corresponding author.

form coding together with quantization and entropy coding in NeRF compression and achieves promising performance. Despite the success, Masked Wavelet NeRF only leverages a linear transform, which has proved to be less effective than learned non-linear transform in neural image compression [10, 19, 20].

Therefore, motivated by advances in neural image compression, we introduce NeRFCodec, a NeRF compression framework that integrates non-linear transform, quantization, and entropy coding for compressing feature planes in hybrid NeRF to achieve memory-efficient scene representation. A key question is, how do we obtain the non-linear basis? Existing 2D neural image compression methods obtain such a non-linear transform by training an encoder-decoder network on millions of images, while it is infeasible to train such a neural codec on millions of 3D scenes. We find that, surprisingly, existing neural image codec trained on natural images can serve as a strong backbone for compressing feature planes of NeRF, when partially tailored and tuned to each scene. Specifically, we first pre-train a hybrid NeRF on one scene, and feed the feature planes into a well-trained 2D image neural codec by replacing its encoder and decoder heads (i.e., the first and last layers) to adjust to the target channel dimension. Next, we adapt the full encoder and the decoder head to fit this scene while keeping the remaining parts of the decoder frozen. This is supervised by rate-distortion loss, with the goal of decoding the feature planes for high image rendering quality while maintaining a low bitrate of the latent code via an entropy loss. After training, the latent code predicted by the encoder and the decoder head parameters are quantized and entropy-coded into a bitstream for transmission.

Our experimental results demonstrate that NeRFCodec pushes the frontier of the rate-distortion trade-off compared to existing NeRF compression methods [24, 32, 33]. Our method only uses 0.5 MB to represent a single scene while maintaining high visual fidelity, as shown in Fig. 1.

We summarize our contributions as follows:

- We introduce NeRFCodec, an end-to-end compression framework for plane-based hybrid NeRF. It utilizes neural feature compression, combining non-linear transformation, quantization, and entropy coding for efficient compression of plane-based NeRF representations, which advances the frontier of rate-distortion performance for compact NeRF representations.
- We propose to re-use pre-trained neural 2D image codec with slight modification and fine-tune it to each scene individually via the supervision of rate-distortion loss.
- We demonstrate that our method could achieve superior rate-distortion performance compared to existing NeRF compression methods.

2. Related Work

Efficient Representation of Neural Fields: The vanilla NeRF [28] proposes to represent the scene with a multilayer perception (MLP). Subsequent works demonstrate that representing the scene as voxel grids leads to significantly faster training and better reconstruction quality [21, 36, 43]. Besides, point-based methods [42, 45], like Point-NeRF [42], demonstrated its capability for high-efficiency NeRF reconstruction. However, 3D dense voxel grids and point clouds require substantial memory.

Several strategies [8, 9, 15, 17, 30, 39] have been proposed for efficient scene representation design to alleviate the stress of memory requirements. Instant-NGP [30] reduces the memory cost of high-resolution voxel grids by constructing a hash encoding and resolving hash collision implicitly by a tiny MLP decoder. Another line of work decomposes 3D feature volumes into orthogonal 2D planes or 1D vectors. EG3D [8] introduces a tri-plane representation of three perpendicular feature planes and extracts features separately from each plane as inputs for the following MLPs. TensoRF [9] is inspired by tensor decomposition to represent 3D grids with combinations of axis-aligned vectors and matrices via VM decomposition and CP decomposition. While these strategies successfully reduce the parameter count without compromising rendering quality, their raw uncompressed parameters still require at least tens of megabytes for storage.

Compression of Neural Fields Representation: Various compression techniques have been applied to the aforementioned representations of neural fields for further compression, including parameter quantization techniques, transform coding, and entropy coding.

Quantization techniques could be categorized into vector quantization and scalar quantization in terms of the parameter unit to be quantized. VQAD [37] introduces vector quantization to compress tree-based neural field representations [38]. This approach adaptively learns features within the codebook and the index assignment corresponding to various leaf nodes during training. VQRF [24] introduces a universal compression pipeline designed for pre-trained hybrid NeRF representations, including voxel pruning, vector quantization, weight quantization, and entropy coding. Masked Wavelet NeRF [32] employs quantization-aware training and 8-bit uniform scalar quantization on wavelet coefficients. BiRF [33] introduces the concept of binary neural networks [11] into the NeRF domain. It uses hash-encoded 2D planes and 3D volumes as scene representations, where each feature is quantized to either +1 or -1.

Entropy coding is a way to achieve lossless compression of a sequence of symbols, which is the foundation of advanced compression systems. cNeRF [5] individually learns a probability model for employing entropy cod-

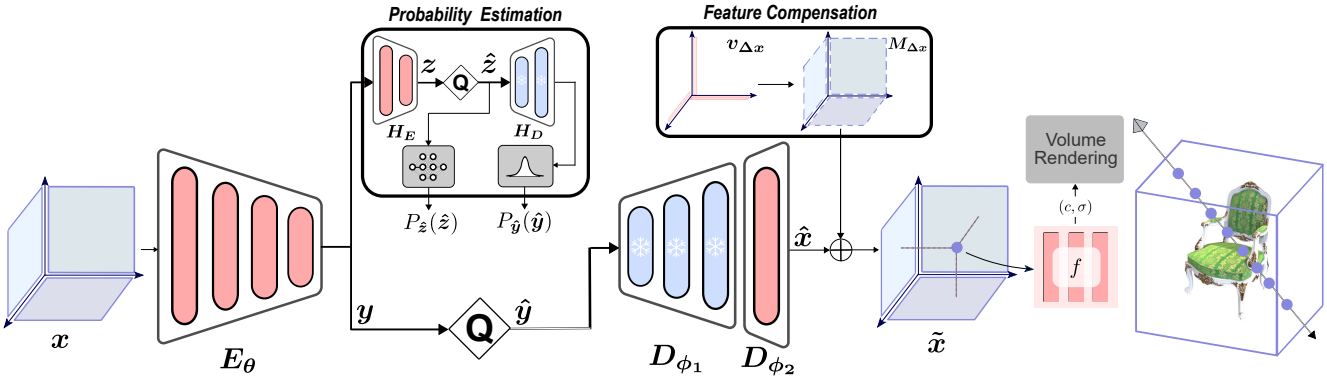


Figure 2. **NeRFCodec**. We combine the pre-trained neural 2D image codec with content-specific parameters to compress hybrid NeRF. The feature planes $\boldsymbol{x}$ are fed into feature encoder $\boldsymbol{E_\theta}$ to obtain latent code $\boldsymbol{y}$. Latent code $\boldsymbol{y}$ is quantized into $\hat{\boldsymbol{y}}$ in one branch. In another branch, latent code $\boldsymbol{y}$ is sent to the probability estimation module to get its corresponding probability $P_{\hat{\boldsymbol{y}}}$ of quantized latent code $\hat{\boldsymbol{y}}$ for entropy coding. Inside the probability estimation, it leverages a hyperprior encoder $\boldsymbol{H_E}$ to obtain hyperprior latent code $\boldsymbol{z}$ and a hyperprior decoder $\boldsymbol{H_D}$ to estimate probability distribution $P_{\hat{\boldsymbol{y}}}$. The quantized latent code $\hat{\boldsymbol{y}}$ is fed into the feature decoder $\boldsymbol{D_\phi}$ to generate reconstructed feature planes $\hat{\boldsymbol{x}}$. The feature decoder consists of feature decoder backbone $\boldsymbol{D_{\phi_1}}$ and feature decoder head $\boldsymbol{D_{\phi_2}}$. We introduce a feature compensation module to compensate for the loss of high-frequency residuals. We add feature residual matrix $\boldsymbol{M_{\Delta x}}$ represented by the outer product of feature residual vectors $\boldsymbol{v_{\Delta x}}$ to get the final feature planes $\tilde{\boldsymbol{x}}$. The final feature planes $\tilde{\boldsymbol{x}}$ cooperate with a tiny MLP f to predict the color and density of sample points for volume rendering. The red components are updated in training, while the blue components inherit parameters from pre-trained neural image compression and stay frozen. The final bitstreams include quantized latent code $\hat{\boldsymbol{y}}$, quantized hyperprior latent code $\hat{\boldsymbol{z}}$, feature decoder head $\boldsymbol{D_{\phi_2}}$, feature residual vectors $\boldsymbol{v_{\Delta x}}$, tiny MLPs f, and metadata.

ing to weights of MLP in the vanilla NeRF, which follows the strategy of prior neural network compression work [31]. VQRF utilizes entropy coding in the final step to compress simplified components of the entire model individually and then pack the bitstream together. Masked Wavelet NeRF performs run-length encoding (RLE) to masked wavelet coefficients and applies the Huffman encoding to the RLE-encoded streams to map values with a high probability to shorter bits.

Transform coding is an effective lossy compression technique demonstrated in traditional image and video coding. Masked Wavelet NeRF takes inspiration from JPEG 2000 [34], employing inverse wavelet transform on learnable 1D vectors and 2D planes to obtain features for subsequent queries. ACRF [16] is motivated by point cloud compression and introduces a point-based wavelet transform, region adaptive hierarchical transform (RAHT) [13], for voxel/point-based NeRF compression.

Our approach follows the compression framework that combines transform coding, quantization, and entropy coding. Compared to previous works, we introduce a non-linear transform designed for the feature planes in hybrid NeRF.

Neural Image Compression: Recent research has made rapid progress on deep learning-based neural image compression methods. The current state-of-the-art methods can approach or even surpass advanced traditional image and video codecs in terms of rate-distortion performance. Image coding standards based on the neural image compression framework, JPEG-AI [1], are also under development. The main paradigm for neural image compression is the autoencoder, which inserts a scalar quantization and an entropy coding module into the bottleneck layer. The auto-encoder is regarded as a pair of non-linear transform basis, while the latent code is regarded as non-linear transform coefficients [2]. An essential work proposed by Balle et al. [3] introduces a hyperprior network for better probability estimation of latent code in entropy coding with minor overhead of sending side information, which becomes an indispensable building block in neural image compression. Subsequently, several methods were proposed to enhance the rate-distortion performance of neural image compression, including more expressive networks in non-linear transform [10, 25], more precise probability estimation [10, 29], and generative adversarial training [27].

Despite superior natural image compression performance, directly applying these advanced methods to compressing feature planes in hybrid NeRF without modification does not achieve high rate-distortion performance, as demonstrated in our experimental section.

3. Method

In this work, we propose an end-to-end NeRF compression framework compatible with plane-based hybrid NeRF variants. Fig. 2 gives an overview of our framework, comprising neural feature compression and NeRF rendering. Neural feature compression consists of content-adaptive non-linear transform, quantization, and entropy coding. NeRF rendering follows the corresponding NeRF variants.

In the following, we first introduce preliminaries of hybrid NeRF model and neural image compression in Sec-

tion 3.1. Then, we provide a preliminary toy experiment in Section 3.2 to assess the feasibility of our methods. Next, we illustrate neural feature compression for NeRF representation compression in Section 3.3, and the training strategy in Section 3.4. Furthermore, we describe the encoding and decoding process at the test time in Section 3.5. Finally, we describe implementation details in Section 3.6.

3.1. Background

NeRF: Generally, NeRF could be regarded as a continuous scene representation function g_θ parameterized by learnable parameters θ that maps the location of a 3D point $\mathbf{x} \in \mathbb{R}^3$ and a viewing direction $\mathbf{d} \in \mathbb{S}^2$ towards the point to a volume density σ and a color value $\mathbf{c}$:

$$g_\theta : (\mathbf{x} \in \mathbb{R}^3, \mathbf{d} \in \mathbb{S}^2) \mapsto (\sigma \in \mathbb{R}^+, \mathbf{c} \in \mathbb{R}^3) \tag{1}$$

Given a target camera, the color $\mathbf{c}_r$ of a target pixel corresponding to a camera ray r is obtained via volume rendering integral approximated by the numerical quadrature:

$$\mathbf{c}_r = \sum_{i=1}^{N} T_r^i \alpha_r^i \mathbf{c}_r^i \tag{2}$$

$$\alpha_r^i = 1 - \exp(-\sigma_r^i \delta_r^i) \quad T_r^i = \prod_{j=1}^{i-1} \left(1 - \alpha_r^j\right) \tag{3}$$

where T_r^i and α_r^i denote transmittance and alpha value of a sample point $\mathbf{x}_i$ along the camera ray r.

Hybrid NeRF consists of explicit data structures and tiny MLP f. In the rendering process of hybrid NeRF, we first query features from explicit data structure based on the location of sample points, and then we use the tiny MLP f to map queried features to final density and color. The explicit data structure accounts for over 95% of the total parameters and often amounts to tens or even hundreds of megabytes, regarded as the main target for compression. In this paper, we focus on leveraging neural compression to compress plane-based hybrid NeRF, a parameter-efficient and widely used variant of hybrid NeRF.

Neural Image Compression: The basic framework of neural image compression [3, 10, 19] consists of non-linear transforms parameterized by a pair of encoder $\boldsymbol{E}$ and decoder $\boldsymbol{D}$, quantization $\boldsymbol{Q}$, and entropy coding with a learned probability estimation module.

The image $\boldsymbol{x}$ is fed into the encoder $\boldsymbol{E}$ to obtain low-dimensional latent code $\boldsymbol{y}$, which would be quantized into $\hat{\boldsymbol{y}}$. Then, the quantized latent code $\hat{\boldsymbol{y}}$ is fed into the decoder $\boldsymbol{D}$ to obtain reconstructed image $\hat{\boldsymbol{x}}$.

$$\hat{\boldsymbol{y}} = \boldsymbol{Q}(\boldsymbol{E}(\boldsymbol{x})) \quad \hat{\boldsymbol{x}} = \boldsymbol{D}(\hat{\boldsymbol{y}}) \tag{4}$$

For better modeling the probability of $\hat{\boldsymbol{y}}$, the hyperprior latent code $\boldsymbol{z}$ is introduced as side information via a hyperprior auto-encoder. $\boldsymbol{z}$ is obtained by feeding $\boldsymbol{y}$ into a hyperprior encoder $\boldsymbol{H_E}$.Then, $\boldsymbol{z}$ is quantized and fed into a hyperprior decoder $\boldsymbol{H_D}$ to recover parameters for probability distribution modeling of $P_{\hat{\boldsymbol{y}}}$. The quantized hyperprior latent code $\hat{\boldsymbol{z}}$ must also be entropy-coded and transmitted.

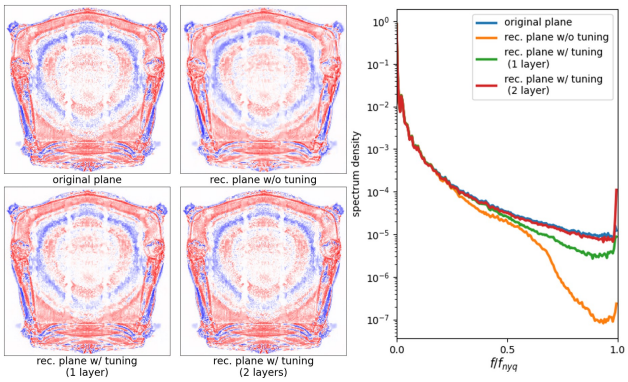


Figure 3. **Spectrum analysis of decoded feature plane**.

$$\hat{\boldsymbol{z}} = \boldsymbol{Q}(\boldsymbol{H_E}(\boldsymbol{y})) \quad P_{\hat{\boldsymbol{y}}} \leftarrow \boldsymbol{H_D}(\hat{\boldsymbol{z}}) \tag{5}$$

The probability distribution $P_{\hat{\boldsymbol{y}}}$ of quantized latent code $\hat{\boldsymbol{y}}$ is a Gaussian distribution conditioned on $\hat{\boldsymbol{z}}$. There is no prior for $P_{\hat{\boldsymbol{z}}}$, so the probability distribution $P_{\hat{\boldsymbol{z}}}$ of quantized hyperprior latent code is modeled by a factorized density model parameterized by shallow MLP.

In neural image compression, once the neural image codec is trained on millions of natural images, the architecture and parameters of the decoder are fixed. The pre-trained decoder is assumed to be capable of decoding any entropy-coded bitstream that meets standard requirements. As a result, the cost of transmitting the decoder is amortized over countless bitstreams. Therefore, when calculating the amount of data to be transmitted, only the size of the entropy-coded latent code needs to be considered.

3.2. Preliminary Analysis

Although large-scale datasets [14, 41, 44] containing millions of 3D scenes have been released, it is infeasible to train feature planes with millions of 3D data and then train a feature codec with millions of feature planes, considering time and resource consumption. Alternatively, we investigate the possibility of reusing a neural 2D image codec trained on millions of natural images and applying it to compress feature planes in hybrid NeRF with slight modification.

Thus, we design a toy experiment to assess the feasibility of our approach in a simple manner. We extract three channels of pre-trained feature planes of TensoRF [9], normalize their value range to be between 0 and 1, and feed them into a pre-trained neural 2D image codec. We visualize the reconstructed feature map and analyze its spectrum in Fig. 3. Unsurprisingly, the naïve way of resuing the 2D image codec suffers from domain gap. This leads to a significant energy loss in the high-frequency range compared

to the original. Next, we fine-tune the last layer of the decoder to reconstruct the original feature plane, while keeping the other layers frozen. We find that simply fine-tuning the last layer recovers the high-frequency energy in the reconstructed feature map to a large extent. This allows for a receiver to decode the feature plane from the latent code, when the content-adapted last layer is jointly transmitted. While fine-tuning more layers in the decoder further improves the performance, it can lead to a significant increase in the data transmission cost.

3.3. NeRFCodec

Based on previous analysis, we designed our pipeline, as shown in Fig. 2. The neural feature compression in our pipeline involves a content-adaptive feature encoder $\boldsymbol{E_\theta}$, a feature decoder $\boldsymbol{D_\phi}$ with a tuned content-adaptive decoder head $\boldsymbol{D_{\phi_2}}$, a feature compensation module and other indispensable components in neural compression including quantization and entropy coding.

Content-Adaptive Encoder: Our feature encoder $\boldsymbol{E_\theta}$ inherits the pre-trained encoder in the neural 2D image codec, except that it replaces the first layer of the pre-trained model with a new encoder head to adapt to the channel dimension of feature planes. In brief, we initially encode each feature plane $\boldsymbol{x}$ individually through content-adaptive encoder $\boldsymbol{E_\theta}$ into a latent code $\boldsymbol{y}$.

$$\boldsymbol{y} = \boldsymbol{E_\theta}(\boldsymbol{x}) \tag{6}$$

During training, all parameters $\boldsymbol{\theta}$ of our feature encoder are optimized, leading to a content-adaptive feature encoder for each scene individually. Since our encoder is customized for scene-specific optimization, the resulting latent code obtained in this way is also optimized for each scene. While it is possible to optimize the latent code directly without the need for an encoder, our ablation study indicates that the encoder-free approach is less competitive.

Content-Adaptive Decoder Head: Our feature decoder $\boldsymbol{D_\phi}$ comprises feature decoder backbone $\boldsymbol{D_{\phi_1}}$ and feature decoder head $\boldsymbol{D_{\phi_2}}$. Similar to the neural feature encoder, we also re-use the decoder in pre-trained neural image codec except for the last layer and initialize a new final layer to adjust to the number of target channel dimensions. The feature decoder takes quantized latent code $\hat{\boldsymbol{y}}$ as input and outputs reconstructed feature planes $\hat{\boldsymbol{x}}$.

$$\hat{\boldsymbol{x}} = \boldsymbol{D_\phi}(\hat{\boldsymbol{y}}) = \boldsymbol{D_{\phi_2}}(\boldsymbol{D_{\phi_1}}(\hat{\boldsymbol{y}})) \tag{7}$$

In training, the feature decoder backbone $\boldsymbol{D_{\phi_1}}$ stay fixed while the parameters in feature decoder head $\boldsymbol{D_{\phi_2}}$ are updated. Thus, the parameters in feature decoder head $\boldsymbol{D_{\phi_2}}$ need to be compressed and transmitted to the receiver side for decoding the feature planes correctly. The feature decoder head can be optimized jointly with the MLP f in hybrid NeRF, effectively predicting the attributes of each point in the scene with high quality.

Feature Compensation: Lossy compression leads to the loss of high-frequency details, as shown in Section 3.2. Thus, we introduce a high-frequency residual compensation module $\boldsymbol{C_R}$ tailored for decoded feature planes $\hat{\boldsymbol{x}}$. This method aims to increase high-frequency details on the reconstructed feature planes with a low storage cost, thereby enhancing the final rendering quality. In practice, we assign a residual matrix $\boldsymbol{M_{\Delta x}}$ to each reconstructed feature plane $\hat{\boldsymbol{x}}$ for compensating high-frequency details. To avoid the burden of directly storing the matrix, we further represent this high-frequency compensation matrix as the outer product of two orthogonal factorized vectors $\boldsymbol{v_{\Delta x}}$ with learnable parameters, following CP decomposition in TensoRF. The final feature plane $\tilde{\boldsymbol{x}}$ used for feature queries is the sum of the reconstructed feature plane $\hat{\boldsymbol{x}}$ and the residual plane $\boldsymbol{M_{\Delta x}}$ parameterized by factorized vectors $\boldsymbol{v_{\Delta x}}$.

$$\tilde{\boldsymbol{x}} = \hat{\boldsymbol{x}} + \boldsymbol{M_{\Delta x}} = \hat{\boldsymbol{x}} + \boldsymbol{v^i_{\Delta x}} \circ \boldsymbol{v^j_{\Delta x}} \tag{8}$$

At test time, these feature vectors $\boldsymbol{v_{\Delta x}}$ will be quantized and entropy-coded into the bitstreams for transmission.

Quantization: A scalar quantizer $\boldsymbol{Q}$ is introduced to quantize latent code $\boldsymbol{y}$ into $\hat{\boldsymbol{y}}$. While quantization is not differentiable, we follow the protocol strategy in neural image compression [3, 10] to add uniform noise to the latent code to simulate quantization in training.

Entropy Coding: We follow the protocols of entropy coding in neural image compression [3, 10]. An essential operation in entropy coding is symbol probability estimation. In training, the estimated probability $P_{\hat{\boldsymbol{y}}}$ and $P_{\hat{\boldsymbol{z}}}$ is used to calculate their entropy ($R_{\hat{\boldsymbol{y}}}$, $R_{\hat{\boldsymbol{z}}}$) as the lower bound of length of the bitstream when actual entropy encoding, according to Shannon rate-distortion theory [12].

$$R_{\hat{\boldsymbol{y}}} = \mathbb{E}[-\log_2 P_{\hat{\boldsymbol{y}}}(\hat{\boldsymbol{y}})] \quad R_{\hat{\boldsymbol{z}}} = \mathbb{E}[-\log_2 P_{\hat{\boldsymbol{z}}}(\hat{\boldsymbol{z}})] \tag{9}$$

Thus, their entropy is introduced as one of the minimization objectives in the optimization process. At test time, estimated probability $P_{\hat{\boldsymbol{y}}}$ and $P_{\hat{\boldsymbol{z}}}$ are fed into the engine of entropy coder to obtain the binary bitstreams.

3.4. Training

Loss Function: In training, according to the volumetric rendering process described in Section 3.1, we can render the pixel colors $\hat{C}(\mathbf{r})$ and supervise them with the ground truth RGB colors $C(\mathbf{r})$ along the sampled rays $\mathbf{r}$:

$$\mathcal{L}_{\text{recon}} = \sum_{\mathbf{r}\in\mathcal{R}} ||\hat{C}(\mathbf{r}) - C(\mathbf{r})||_2^2, \tag{10}$$

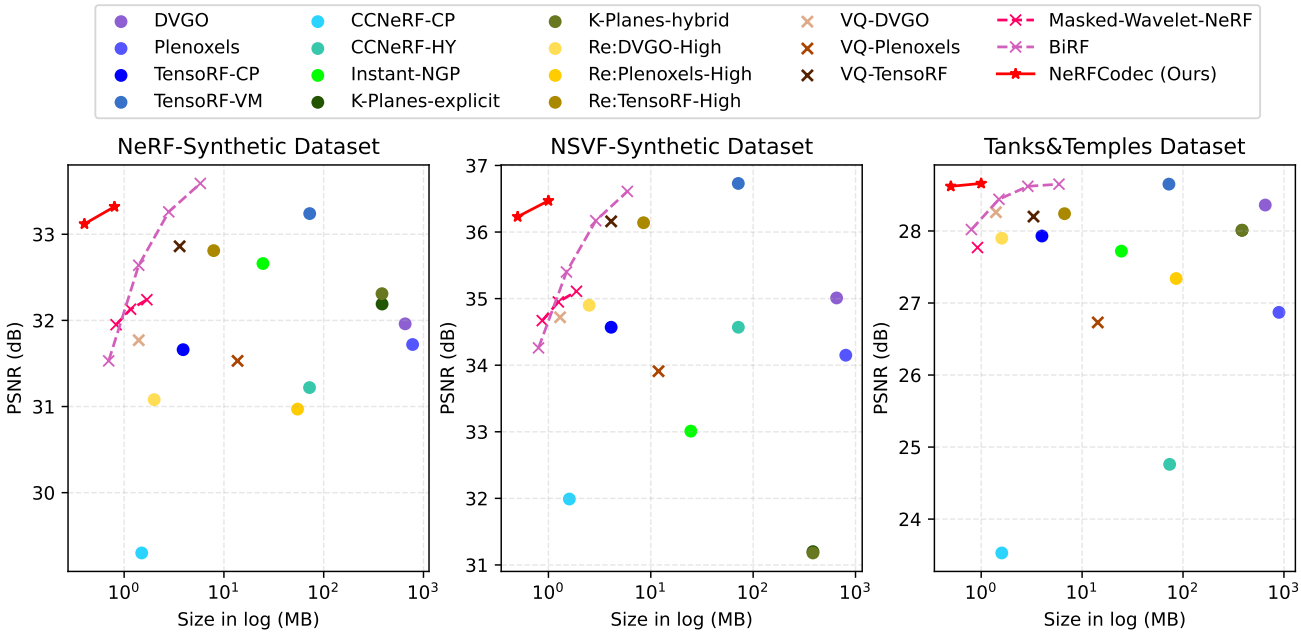


Figure 4. **Memory-quality plot**. We plot the memory footprint and visual quality score (PSNR) to compare our method with baseline methods. We use different markers for our method and each category of baseline: circles (•) for parameter-efficient data structure methods, crosses (×) for parameter compression methods, and stars (⋆) for ours.

The probability of the latent code is used to calculate its theoretical average minimum code length, which is the entropy of the latent code. The entropy of the latent code, serving as a proxy for the actual average encoding length, is incorporated into the final optimization objective to minimize the storage of latent code.

$$\mathcal{L}_{\text{entropy}} = R_{\hat{\mathbf{y}}} + R_{\hat{\mathbf{z}}} \tag{11}$$

The neural feature codec and small MLP for attribute regression will be jointly optimized by combining the rendering loss and the entropy loss. The overall training loss for our system is defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{recon}} + \lambda_{\text{entropy}} \mathcal{L}_{\text{entropy}} \tag{12}$$

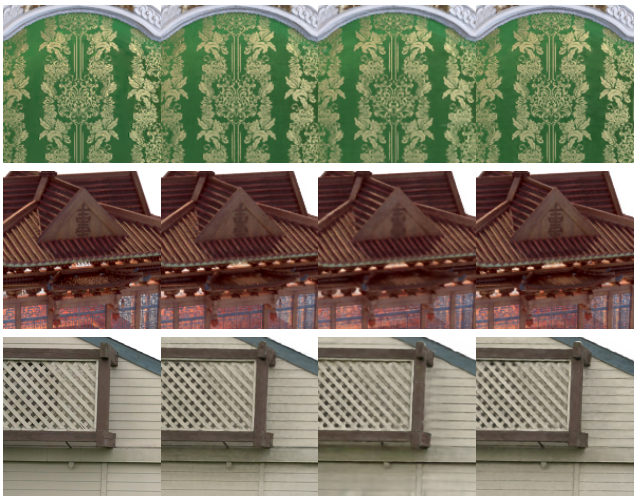
Multi-Stage Training: First, we pre-train the TensoRF with 30,000 iterations to obtain feature planes $\boldsymbol{x}$ and tiny MLP f for each scene. Then, we perform a warm-up to the neural feature codec only with the feature plane reconstruction loss, leading to a better starting point for formal training. Next, we proceed with the joint training of the neural feature codec and the neural radiance field for 100,000 iterations with $\mathcal{L}_{\text{total}}$ as the loss function. Finally, we perform quantization-aware training on the network parameters to be transmitted,e.g. those from "decoder head" and "MLP in renderer", for 10,000 iterations.

3.5. Test-Time Encoding and Decoding Process

At test time, the neural feature codec performs actual entropy coding and outputs bitstreams of latent code for transmission. In practice, the orthogonal feature planes are fed into the codec one by one to obtain their respective bitstreams of latent code. Apart from latent code, the content-specific network parameters in the neural feature decoder would be compressed and transmitted for the successful decoding of feature planes. Besides, other minority learnable parameters, such as MLPs, will be quantized and entropy-encoded into bitstreams. Thus, at the sending end, the bitstreams of different components, including latent code, content-specific parameters, small MLPs etc., would be multiplexed into a total bitstream. Then, at the receiving end, the total bitstream is first demultiplexed into multiple substreams representing different components. Each substream is independently entropy-decoded and reassembled to its corresponding location for reconstructing feature planes and volume rendering. It is worth noting that we only need to reconstruct feature planes once, and subsequent querying operations for each sampling point are executed on the reconstructed planes. Thus, the decoding operation will only incur a short decoding waiting time and will not affect the rendering efficiency.

3.6. Implementation Details

Our method is compatible with different plane-based NeRF approaches. In practice, we choose two variants of plane-based NeRF, TensoRF-VM [9] and HexPlane [7], as the representative implementations of neural fields to validate compatibility. The implementation of neural feature codec is based on the CompressAI library [4]. Specifically, we choose the neural image codec proposed by Cheng et al. [10] as the backbone of the neural feature codec for the



GT Ours MWNeRF [32] TensoRF [9]

Figure 5. **Qualitative comparison on NeRF-Synthetic.**

following experiment session. In fact, other end-to-end image codecs [3, 19] can also be substituted into our NeRFCodec framework.

4. Experiments

In this section, we evaluate our method's performance through quantitative comparisons with baselines, memory breakdown analysis, and thorough ablation studies to validate our design decisions. We mainly present the experimental results using TensoRF-VM as the neural field in this section and demonstrate the results using HexPlane as the neural field in the supplementary materials.

Datasets: We evaluate our performance using both synthetic and real-world datasets. We use two synthetic datasets: the NeRF-Synthetic dataset [28] and the NSVF-Synthetic dataset [26]. We also choose the real-world dataset Tanks and Temples [22] and follow the NSVF evaluation protocol in scene selection and background masking.

Baselines: We categorize baseline methods into two types: a) methods focus on parameter-efficient data structure and b) methods focus on parameter compression. Representatives of methods focusing on parameter-efficient data structure include DVGO [36], Plenoxels [43], TensoRF [9], CCNeRF [39], Re:NeRF [15], Instant-NGP [30], and K-Planes [17]. Representatives of methods focusing on parameter compression include VQRF [24], Masked Wavelet NeRF [32], and BiRF [33].

4.1. Comparison with Baselines

We report the average visual scores (PSNR, SSIM) of test views and measure the average memory footprint of all scenes in each dataset. We plot the memory requirement and visual quality metrics of both our approach and the baseline method in Fig. 4. We leave the quantitative comparison of SSIM to the supplementary materials. We achieve multi-bitrate points by using different numbers of channels in the feature planes. In Fig. 4, we observe that our method achieves superior rate-distortion performance compared to the baseline methods across three different datasets. We observe an improvement compared to Masked Wavelet NeRF, which we attribute to the use of a non-linear transform instead of a wavelet transform. We also show the qualitative comparison in Fig. 5.

4.2. Memory Breakdown

In Table 1, we report the memory footprint of each component in the final bitstream, including the memory of compressed feature planes, compressed decoder adaptor, compressed MLP for attribute regression, and other side information. We also report the original sizes of the uncompressed components and the corresponding compression ratios. It's worth noting that the non-linear transform allows for a compression ratio over 1000 times.

	Compressed (MB)	Original (MB)	Ratio
Feature planes	0.052	69.953	×1345.3
Decoder head	0.269	1.761	×6.5
MLP in renderer	0.042	0.160	×3.8
Feature vectors	0.049	0.234	×4.8
Residual vectors	0.041	0.234	×5.7
Total	0.453	72.342	×159.7

Table 1. **Memory breakdown.**

4.3. Ablation Study

We conduct ablation studies on NeRFCodec at a low rate point using *Chair* scene from NeRF-Synthetic dataset. We leave more qualitative results to supplementary materials.

Is content-adaptive encoder needed for compression? Our method follows an auto-encoder architecture, fine-tuning the feature encoder for each scene to obtain the latent code. Alternatively, we explore a potential approach: bypassing the feature encoder and directly learning the latent code using the feature decoder, forming an auto-decoder architecture. To compare these approaches, we implement the auto-decoder scheme by removing the feature encoder and setting the latent code as learnable parameters. The remaining experimental settings, including quantization, entropy coding on the latent code, and decoder tuning strategy, remain consistent with the auto-encoder scheme. For a fair comparison, we conduct a thorough hyperparameter search to identify the optimal learning rate for optimizing the latent code in the auto-decoder scheme. In Table 2, we show the rate-distortion performance comparison between auto-encoder and auto-decoder schemes. The experimental re-

sults suggest that the auto-encoder scheme exhibits certain advantages in rate-distortion performance, possibly due to the feature encoder providing a more optimal initial value and optimization dynamics for the latent code.

	Size (MB)	PSNR (dB)	SSIM
auto-decoder	0.461	34.41	0.973
auto-encoder	0.453	35.08	0.981

Table 2. **Ablation on compression scheme.**

Can we re-use pre-trained 2D neural image codec without architecture modification? An alternative to leverage off-the-shelf neural image codec to compress plane-based hybrid NeRF is to split the feature planes into three channels, normalize, and feed it into the feature codec for compression. In this paradigm, the neural image codec could also be jointly fine-tuned with hybrid NeRF via rate-distortion loss. We report the experimental results in Table 3. We observed that the bitrate consumption of this alternative is significantly higher than our approach. This could be attributed to the fact that they do not fully exploit inter-channel information during compression.

	Size (MB)	PSNR (dB)	SSIM
img. codec	1.786	29.96	0.923
tuned img. codec	1.150	33.25	0.961
Ours	0.453	35.08	0.981

Table 3. **Ablation on re-using neural image codec.**

Does the entropy loss contribute to storage saving? During training, we introduce entropy loss to constrain the predicted probability distribution by the probability model to be as close as possible to the actually unknown marginal distribution of the latent code. When this entropy loss is minimized as much as possible during the optimization process, the actual code length during practical encoding will be close to the theoretically shortest code length. We report the impact of entropy loss on the final storage savings of latent code in Table 4. Compared to the scheme without entropy loss, we observed that the scheme with entropy loss significantly reduces the final code length. This highlights the importance of introducing entropy loss during training.

	Size (MB)	PSNR (dB)	SSIM
w/o entropy loss	0.701	34.98	0.980
w/ entropy loss	0.453	35.08	0.981

Table 4. **Ablation on entropy loss.**

Do we need a reconstruction loss for the feature plane? Intuitively, one might think that the reconstruction loss between the reconstructed feature plane and the pre-trained feature plane could benefit higher-quality NeRF reconstructions. However, in practice, we find that adding a reconstruction loss to the reconstructed feature planes leads to degradation in rendering quality, reported in Table 5. We hypothesize that the neural feature decoder can learn to synthesize features more suited for the following attribute regression without direct feature reconstruction loss.

	Size (MB)	PSNR (dB)	SSIM
w/ feature rec. loss	0.479	32.81	0.965
w/o feature rec. loss	0.453	35.08	0.981

Table 5. **Ablation on feature reconstruction loss.**

Can a neural feature codec trained on a small-scale feature dataset generalize to new scenes? In cases where training the neural feature codec on a massive amount of feature planes is infeasible, we try to train a neural feature codec with a few scenes and test its generalization on unseen objects. Specifically, we collect feature planes trained on eight objects from the NeRF-Synthetic dataset. Next, we use these planes to train a neural feature codec on these scenes via rendering loss and entropy loss. When evaluating on the training scenes, we find it could render reasonable images but with a slight degradation in visual scores. However, when the codec trained on NeRF-Synthetic datasets is applied to the feature planes pre-trained from NSVF-Synthetic *without any test-time optimization*, it fails to synthesize reasonable rendering results. The neural feature codec struggles to generalize with limited training data. Hence, our strategy of adapting well-trained 2D image codecs for NeRF compression holds value, considering the difficulty in obtaining large-scale plane feature datasets.

5. Conclusion

In this paper we propose NeRFCodec, an end-to-end compression framework for plane-based hybrid NeRF. The main idea is to leverage non-linear transform, quantization, and entropy coding for compressing feature planes in hybrid NeRF to achieve memory-efficient scene representation. The experiments show that our method only uses a memory budget of 0.5 MB to represent a single scene while achieving high-quality novel view synthesis. As a limitation, training the non-linear transform is time-consuming. Moreover, we need to train a specialized neural feature codec for each scene individually. In the future, we plan to scale up the data collections of feature planes and train a generalized neural feature codec that generalizes well on unseen objects via training on large-scale datasets.

Acknowledgements: This work is supported by the National Natural Science Foundation of China under Grant No. U21B2004, No. 62071427, No. 62202418, Zhejiang University Education Foundation Qizhen Scholar Foundation, and the Fundamental Research Funds for the Central Universities under Grant No. 226-2022-00145. Yiyi Liao and Lu Yu are with Zhejiang Provincial Key Laboratory of Information Processing, Communication and Networking (IPCAN), Hangzhou 310007, China.

References

[1] João Ascenso, Elena Alshina, and Touradj Ebrahimi. The jpeg ai standard: Providing efficient human and machine visual data consumption. *IEEE Multimedia*, 30(1):100–111, 2023. 3

[2] Johannes Ballé, Philip A Chou, David Minnen, Saurabh Singh, Nick Johnston, Eirikur Agustsson, Sung Jin Hwang, and George Toderici. Nonlinear transform coding. *IEEE Journal of Selected Topics in Signal Processing*, 15(2):339–353, 2020. 1, 3

[3] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2018. 3, 4, 5, 7

[4] Jean Bégaint, Fabien Racapé, Simon Feltman, and Akshay Pushparaja. Compressai: a pytorch library and evaluation platform for end-to-end compression research. *arXiv preprint arXiv:2011.03029*, 2020. 6

[5] Thomas Bird, Johannes Ballé, Saurabh Singh, and Philip A Chou. 3d scene compression through entropy penalized neural representation functions. In *2021 Picture Coding Symposium (PCS)*, pages 1–5. IEEE, 2021. 2

[6] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J Sullivan, and Jens-Rainer Ohm. Overview of the versatile video coding (vvc) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3736–3764, 2021. 1

[7] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 130–141, 2023. 6

[8] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J. Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3d generative adversarial networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2

[9] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022. 1, 2, 4, 6, 7

[10] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 7939–7948, 2020. 2, 3, 4, 5, 6

[11] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in Neural Information Processing Systems (NeurIPS)*, 28, 2015. 2

[12] Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012. 5

[13] Ricardo L De Queiroz and Philip A Chou. Compression of 3d point clouds using a region-adaptive hierarchical transform. *IEEE Transactions on Image Processing*, 25(8):3947–3956, 2016. 3

[14] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 13142–13153, 2023. 4

[15] Chenxi Lola Deng and Enzo Tartaglione. Compressing explicit voxel grid representations: fast nerfs become also small. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1236–1245, 2023. 2, 7

[16] Guangchi Fang, Qingyong Hu, Longguang Wang, and Yulan Guo. Acrf: Compressing explicit neural radiance fields via attribute compression. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2024. 3

[17] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 12479–12488, 2023. 2, 7

[18] Vivek K Goyal. Theoretical foundations of transform coding. *IEEE Signal Processing Magazine*, 18(5):9–21, 2001. 1

[19] Dailan He, Ziming Yang, Weikun Peng, Rui Ma, Hongwei Qin, and Yan Wang. Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5718–5727, 2022. 2, 4, 7

[20] Yueyu Hu, Wenhan Yang, Zhan Ma, and Jiaying Liu. Learning end-to-end lossy image compression: A benchmark. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 44(8):4194–4211, 2021. 2

[21] Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. Relu fields: The little non-linearity that could. In *ACM Trans. on Graphics*, pages 1–9, 2022. 2

[22] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017. 7

[23] Adrian S Lewis and G Knowles. Image compression using the 2-d wavelet transform. *IEEE Transactions on image Processing*, 1(2):244–250, 1992. 1

[24] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Liefeng Bo. Compressing volumetric radiance fields to 1 mb. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4222–4231, 2023. 1, 2, 7

[25] Jinming Liu, Heming Sun, and Jiro Katto. Learned image compression with mixed transformer-cnn architectures. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 14388–14397, 2023. 3

[26] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 7

[27] Fabian Mentzer, George D Toderici, Michael Tschannen, and Eirikur Agustsson. High-fidelity generative image compression. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:11913–11924, 2020. 3

[28] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik,

Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 1, 2, 7

[29] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018. 3

[30] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. on Graphics*, 2022. 1, 2, 7

[31] Deniz Oktay, Johannes Ballé, Saurabh Singh, and Abhinav Shrivastava. Scalable model compression by entropy penalized reparameterization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019. 3

[32] Daniel Rho, Byeonghyeon Lee, Seungtae Nam, Joo Chan Lee, Jong Hwan Ko, and Eunbyung Park. Masked wavelet representation for compact neural radiance fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 20680–20690, 2023. 1, 2, 7

[33] Seungjoo Shin and Jaesik Park. Binary radiance fields. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 1, 2, 7

[34] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36–58, 2001. 1, 3

[35] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012. 1

[36] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 7

[37] Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. Variable bitrate neural fields. In *ACM Trans. on Graphics*, pages 1–9, 2022. 1, 2

[38] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 11358–11367, 2021. 2

[39] Jiaxiang Tang, Xiaokang Chen, Jingbo Wang, and Gang Zeng. Compressible-composable nerf via rank-residual decomposition. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:14798–14809, 2022. 2, 7

[40] Andrew B Watson et al. Image compression using the discrete cosine transform. *Mathematica journal*, 4(1):81, 1994. 1

[41] Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Jiawei Ren, Liang Pan, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, et al. Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 803–814, 2023. 4

[42] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5438–5448, 2022. 2

[43] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 7

[44] Xianggang Yu, Mutian Xu, Yidan Zhang, Haolin Liu, Chongjie Ye, Yushuang Wu, Zizheng Yan, Chenming Zhu, Zhangyang Xiong, Tianyou Liang, et al. Mvimgnet: A large-scale dataset of multi-view images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 9150–9161, 2023. 4

[45] Qiang Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. Differentiable point-based radiance fields for efficient view synthesis. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–12, 2022. 2