Compressing Volumetric Radiance Fields to 1 MB

Lingzhi Li*, Zhen Shen*, Zhongshu Wang, Li Shen, Liefeng Bo
Alibaba Group, Beijing, China
{llz273714, zackary.sz, zhongshu.wzs, liefeng.bo}@alibaba-inc.com, lshen.lsh@gmail.com

Abstract

Approximating radiance fields with discretized volumetric grids is one of promising directions for improving NeRFs, represented by methods like DVGO, Plenoxels and TensoRF, which achieve super-fast training convergence and real-time rendering. However, these methods typically require a tremendous storage overhead, costing up to hundreds of megabytes of disk space and runtime memory for a single scene. We address this issue in this paper by introducing a simple yet effective framework, called vector quantized radiance fields (VQRF), for compressing these volume-grid-based radiance fields. We first present a robust and adaptive metric for estimating redundancy in grid models and performing voxel pruning by better exploring intermediate outputs of volumetric rendering. A trainable vector quantization is further proposed to improve the compactness of grid models. In combination with an efficient joint tuning strategy and post-processing, our method can achieve a compression ratio of 100× by reducing the overall model size to 1 MB with negligible loss on visual quality. Extensive experiments demonstrate that the proposed framework is capable of achieving unrivaled performance and well generalization across multiple methods with distinct volumetric structures, facilitating the wide use of volumetric radiance fields methods in real-world applications. Code is available at https://github.com/AlgoHunt/VQRF.

1. Introduction

Novel view synthesis aims to realize photo-realistic rendering for a 3D scene at unobserved viewpoints, given a set of images recorded from multiple views with known camera poses. The topic has growing importance because of its potential use in a wide range of Virtual Reality and Augmented Reality applications. Neural radiance fields (NeRF) [29] have demonstrated compelling ability on this topic by modelling and rendering 3D scenes effectively through the use of deep neural networks, which are learned to map each 3D location given a viewing direction to its corresponding view-dependent color and volume density according to volumetric rendering techniques [27]. The rendering process relies on sampling a huge number of points and feeding them through a cumbersome network, incurring considerable computational overhead during training and inference. Recent progress following radiance fields reconstruction shows that integrating voxel-based structures [23] into the learning of representations can significantly boost training and inference efficiency. These volumetric radiance fields methods typically store features on voxels and retrieve sampling points (including color features and volume densities) by performing efficient trilinear interpolation without neural network [44] or only equipped with a lightweight neural network [37] instead of cumbersome networks. However, the use of volumetric representations inevitably introduces considerable storage cost, e.g., costing over one hundred megabytes to represent a scene, which is prohibitive in real-world applications.

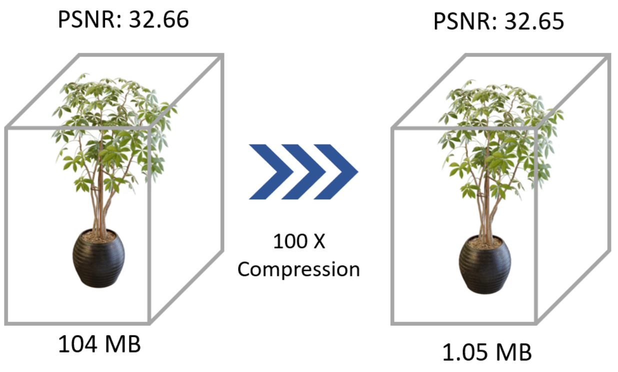
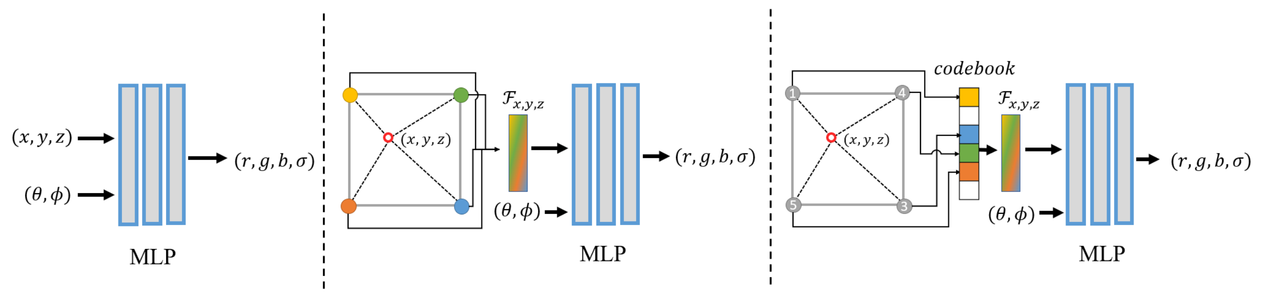


Figure 1. The pipeline can realize 100× compression rate on volumetric models while highly preserving the rendering quality.

In this paper, we aim to counteract the storage issue of representations induced by using voxel grids meanwhile retaining rendering quality. In order to better understand the characteristic of grid models, we estimated the distribution of voxel importance scores (shown in Fig. 4) and

*denote equal contribution



(a) NeRF (b) Volumetric NeRF (c) VQRF

Figure 2. (a) NeRF learns a mapping from 3D coordinate (x, y, z) and viewing direction (θ, ϕ) to color and density (r, g, b, σ). (b) Volumetric NeRF optimizes volumetric grids, estimates color features $\mathcal{F}_{x,y,z}$ and density for sampling points via tri-linear interpolation to get the final color through (or without) tiny MLPs. (c) VQRF compresses voxel features to codebook and stores per-voxel k-bits mapping index, pointing to the codebook consisting of 2^k codes.

observed that only 10% voxels contribute over 99% importance scores of a grid model, indicating that large redundancy exists in the model. Inspired by traditional techniques of deep network compression [14], we present an effective and efficient framework for compressing volumetric radiance fields, allowing about $100\times$ storage reduction over original grid models, with competitive rendering quality.

The illustration of the idea and the framework are shown in Fig. 2 and Fig. 3. The proposed framework is quite general rather than restricted to certain architecture. It is comprised of three steps, i.e., voxel pruning, vector quantization and post processing. Voxel pruning is used to omit the least important voxels which dominate model size while contributing little to the final rendering. We introduce an adaptive strategy for pruning threshold selection with the aid of a cumulative score rate metric, enabling the pruning strategy general across different scenes or base models. In order to further reduce model size, we propose to encode important voxel features into a compact codebook by developing an importance-aware vector quantization with an efficient optimization strategy, and using a joint tuning mechanism to enable the compressed models approaching to the rendering quality of original ones. We finally perform a simple post-processing step to obtain a model with quite small storage cost. For example, as shown in Fig. 1, the volumetric model with the storage cost of 104 MB and the rendering quality of PSNR 32.66 can be compressed into a tiny model costing 1.05 MB with a negligible visual quality loss (PSNR 32.65). We conduct extensive experiments and empirical studies to validate the method, showing the effectiveness and generalization of the proposed compression pipeline on a wide range of volumetric methods and varying scenarios.

2. Related Work

Neural Radiance Fields. Neural radiance fields [29] provide an effective representation to model 3D scenes, enabling high quality novel-view synthesis given some training observations of multi-viewpoints. Recent volumetric methods [2, 37, 44, 45] have shown the benefit of introducing discretized structures in training efficiency and rendering acceleration. SNeRG [15] converts NeRF to a carefully designed grid to achieve inference speedup. Plenoxels [44] directly optimizes a sparse grid with spherical harmonics to model view-dependent effects without neural networks. DVGO [37] exploits dense voxel grids to represent color features and density, [42] extends it to ultra high resolution synthesis. Some methods are proposed to integrate with a variety of data structures, e.g., octree [45], decomposed tensor and tri-planes [2] , and multi-scale hashing table [30]. These methods typically require tremendous storage overhead compared to modelling with pure MLPs.

Vector Quantization. Vector quantization (VQ) [7, 8, 11] is a classical lossy compression technique, aiming to assign a large set of vectors into a smaller set of clusters and represent each vector by one or a mixture of cluster centroids. The technique has been widely used in many real-world applications, including image compression [4, 31], video codec [21, 36] and audio codec [26, 32], as well as generative model [12,34,40] in order to improve generation quality. Some works [10] introduce vector quantization to deep neural network compression. The mostly related work in [38] achieves variable bitrate to level of detail by incorporating soft VQ with multi-resolution feature grids, which is very different from our approach to VQ. Besides using efficient hard mapping/indexing, we introduce voxel importance into VQ optimization as well as other strategies, allowing the use of VQ to be highly effective and adaptive for volumetric radiance fields compression.

Model Compression. Our method can be regarded as a sub-task of model compression, which aims to reduce overall storage size while preserving the accuracy of original models. Most model compression techniques can be cate-

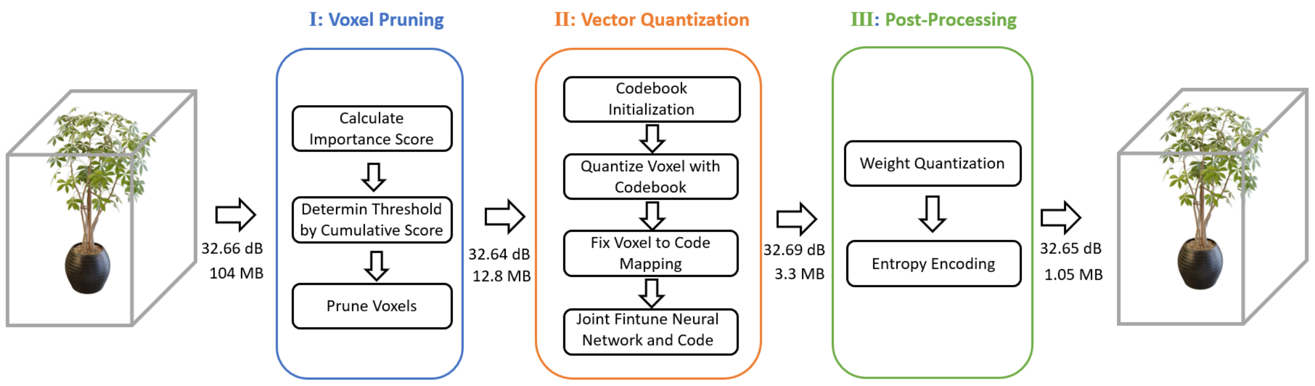


Figure 3. **Pipeline overview.** We design a three stage compression pipeline, given an DVGO model trained on *"ficus"* for example, voxel pruning reduces model size by 8×, vector quantization further improves compression ratio to around 33 ×. We can achieve a total of 100× compression after post-processing.

gorized into four groups: 1) model pruning [13, 14, 41, 43] to remove redundant connections or layers from a neural network; 2) weight quantization [14, 17, 20] to convert full-precision values to lower-bits; 3) low-rank approximation [6, 18, 35] to factorize weight matrix into lower-rank ones; 4) knowledge distillation [1, 3, 16, 33] to guide the training of compact networks via well-trained large ones. [14] introduces a pipeline, including model pruning, weight quantization and huffman coding, for general neural network compression. These techniques are employed in some volumetric radiance fields, e.g., weight quantization in [5, 45]. Empty voxel pruning in [22,45] and iterative removing (and re-including) parameters in [5] function as weight pruning. CCNeRF [39] and TensoRF [2] decompose volumetric representations via low-rank approximation.

3. Problem Statement

Neural radiance fields [29] learn a continuous function that maps a 3D point $\mathbf{x} \in \mathbb{R}^3$ and viewing direction $\mathbf{d} = (\theta, \phi)$ to the view-dependent color $\mathbf{c} \in \mathbb{R}^3$ and the volume density $\sigma \in \mathbb{R}$ through the use of a multilayer perceptron (MLP) i.e. $F_\Theta : (\mathbf{x}, \mathbf{d}) \mapsto (\mathbf{c}, \sigma)$. According to the volume rendering technique [27], the pixel color $\widehat{C}(\mathbf{r})$ of a given ray $\mathbf{r} = \mathbf{o} + t\mathbf{d}$ can be estimated by accumulating the color $\mathbf{c}$ and density $\mathbf{d}$ of sampling points along the ray:

$$\widehat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i \cdot \alpha_i \cdot \mathbf{c}_i, \tag{1}$$

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i), \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{2}$$

where δ_i is the distance between adjacent points. T_i is the accumulated transmittance when reaching the point i, and α_i is ray termination probability.

Recently, volumetric radiance fields methods [37,44] introduce voxel-based structure to facilitate the learning of representations, i.e., optimizing volumetric grids and estimating the color features and density for sampling points via tri-linear interpolation. The methodology has shown significant benefit on training and inference efficiency compared to the methods relying on large neural networks. However, the use of volumetric representations inevitably introduce considerable storage cost, which might limit its usability in real world applications. To address the issue, we introduce a simple yet efficient framework to compress volumetric radiance fields with the following operations, voxel pruning, vector quantization, and post-processing with weight quantization and entropy encoding, which will be described in detail in the following sections.

4. Voxel Pruning

In order to better understand the statistics of volumetric representations, we first compute the importance scores for each voxel in the grid. Formally, according to the volume rendering technique defined in Eqn.1 and 2, we can compute the importance score of $\mathbf{x}_i$ as

$$I_i = T_i \cdot \alpha_i. \tag{3}$$

The importance score of the point assigned to its neighboring voxel $\mathbf{v}_l$ is proportional to their distance (on normalized grid interval). The importance score of the voxel $\mathbf{v}_l$ can then be obtained by accumulating the importance scores of sampling points contributing to it,

$$I_l = \sum_{\mathbf{x}_i \in \mathcal{N}_l} (1 - |\mathbf{v}_l - \mathbf{x}_i|) \cdot I_i, \tag{4}$$

where $\mathcal{N}_l$ denotes the set of the sampling points falling within the neighborhood of $\mathbf{v}_l$ and $|\mathbf{v}_l - \mathbf{x}_i| \le 1$. In practice, we shoot a batch of rays on the images of the training views and calculate the importance score of each sampling point. The importance score for each voxel can be obtained according to Eqn. 4. Then we sort the voxels with ascending importance scores and define the cumulative score rate with respect to the parameter θ as:

$$F(\theta) = \frac{\sum I_l \cdot \mathbf{1}\{I_l < \theta\}}{\sum I_l}, \tag{5}$$

where $\mathbf{1}\{\cdot\}$ denotes the binary indicator. The cumulative score rate is proportional to the expectation on the cumulative distribution of voxel importance scores. Take the *Lego* scene in the synthetic-NeRF dataset for example, we found that there exist an obviously long-tail phenomenon, i.e., **90%** of least important voxels only contribute **0.1%** of total importance score. We depict the curve of the cumulative score rate on the DVGO's model [37] in the Figure 4. As shown in the figure, most of voxels have minimal effect on rendering results, indicating that large redundancy exits in the grid model and a large amount of less important voxels can be pruned off without scarifying rendering quality.

We expect the pruning strategy to be fairly general across different scenarios or methods. In this regard, we use the quantile function to adaptively select the threshold θ_p,

$$\theta_p = F^{-1}(\beta_p), \tag{6}$$

where β_p is a hyperparameter for representing the cumulative score rate (in Eqn. 5) for pruning. For example, if setting it 0.2 for the DVGO model (in Figure 4), about 97% voxels would be pruned off.

5. Vector Quantization

We introduce vector quantization to compress the important voxels for further reducing model size. Particularly, color features typically cost much more storage compared to density modality. In this regard, we adopt the VQ strategy to only encode color features into a compact codebook, so that multiple voxel features can be replaced by a single code vector. We only need to store the codebook and the corresponding mapping index from voxels to the codebook instead of storing individual voxel features.

Let us analyze the change of storage cost after performing vector quantization. Assume there are N voxels with the feature channel dimension C, the color features of orginal model would cost $N \times C \times 16$ bits as each feature is typically saved in float16 format. We learn a codebook with the size $K \times C$ where K is supposed to be extremely smaller than N, $K \ll N$. The storage cost for saving codebook is $16KC$ bits and each voxel needs $log_2(K)$ bits to present the mapping index. In this regard, we can estimate the compression ratio r as,

$$r = \frac{16NC}{Nlog_2(K) + 16KC} \tag{7}$$

on the size of original model. For example, when K is 4096, applying the strategy would reach the maximum compression ratio of 16 for DVGO [37], 64 for TensoRF [2] and 36 for Plenoxels [44] at their default setting on the synthetic-NeRF dataset. We present the training strategy for obtaining an effective and compact codebook in the following subsections.

5.1. Codebook Initialization and Update

We use a weighted clustering strategy for initializing the codebook by the consideration that the voxels with higher importance score typically have higher impact for rendering. Formally, the voxel features $\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N\}$ are partitioned into the codebook $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_K\}$ where $N \gg K$, by minimizing the weighted within-cluster sum of squares:

$$\underset{\mathbf{B}}{\arg\min} \sum_{k=1}^{K} \sum_{\mathbf{v}_j \in \mathcal{R}(\mathbf{b}_k)} \|\mathbf{u}_j - \mathbf{b}_k\|_2^2 \cdot I_j, \tag{8}$$

where $\mathbf{u}_j$ and I_j denote the color features and the importance score of $\mathbf{v}_j$, $\mathcal{R}(\mathbf{b}_k)$ denotes the set of voxels assigned to the k-th code vector $\mathbf{b}_k$.

In practice, adopting global weighted clustering would be unsatisfactorily slow when N and K are large. We therefore apply an iterative optimization strategy to approximate the procedure. Particularly, we randomly select a min-batch of voxels from the grid at each iteration and calculate Euclidean distance between every selected voxel $\mathbf{v}_j$ and each code vector $\mathbf{b}$, and associate the voxel to the code with minimum distance. The total importance assigned to the code vector $\mathbf{b}_k$ can be estimated as $s_k = \sum I_j \mathbf{1}\{\mathbf{v}_j \in \mathcal{R}(\mathbf{b}_k)\}$ during each iteration, then $\mathbf{b}_k$ is optimized by weighted accumulating the voxel features belonging to the code according to normalized importance score,

$$\mathbf{b}_k := \lambda_d \mathbf{b}_k + \frac{(1-\lambda_d)}{s_k} \sum_{\mathbf{v}_j \in \mathcal{R}(\mathbf{b}_k)} I_j \mathbf{u}_j. \tag{9}$$

Here λ_d is the decay factor for the moving average updating of code vectors.

Code expiration. We expect all the code vectors to be effective while directly using the iterative optimization might encounter inactive code issue. Some code vectors may associate with a minimal amount (or even none) of voxels while some code vectors may be shared by a large number of informative voxels, resulting in imbalanced assignment distribution. It would degrade the representation ability of the codebook. To address the issue, we track the capacity of each code vector $\mathbf{b}_k$ according to the accumulated importance s_k assigned to it, rank all the codes in descending order and reset J codes with the lowest capacity by re-initializing via the voxel features with the top J importance in the batch.

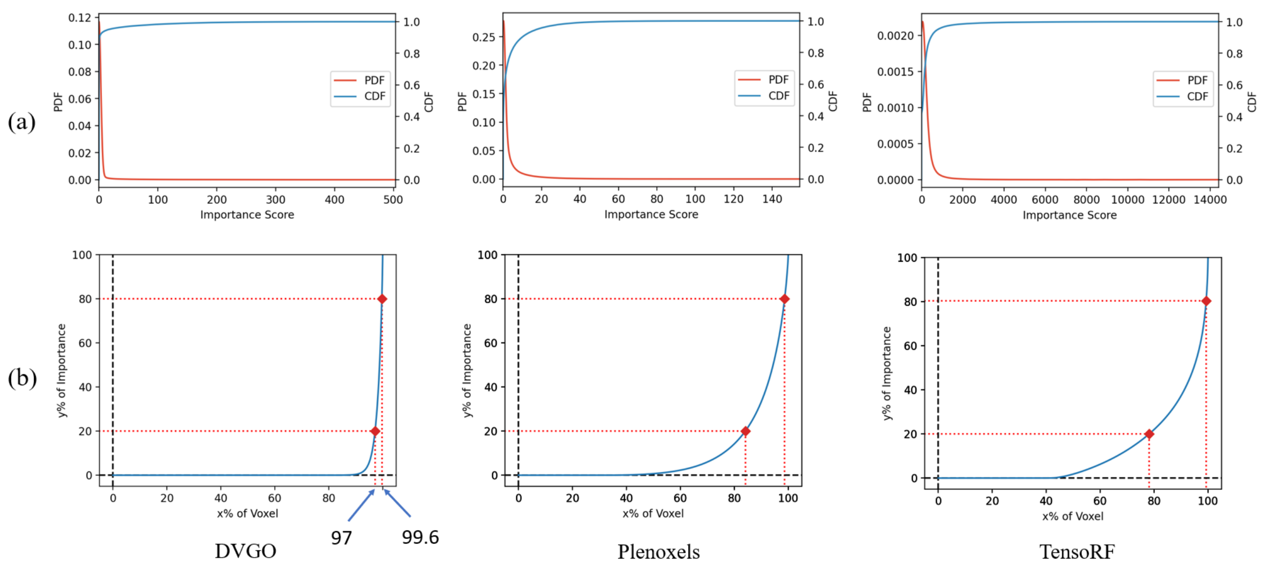


Figure 4. (a) PDF and CDF of importance score. (b) We draw the quantile-quantile curve, which means x% of least important voxels contributes to y% percent of total importance. Take DVGO (bottom left curve) for example, **97 %** of least important voxels only contribute 20% of total importance, while using the top **0.4%** $(1 - 99.6\%)$ of voxels can achieve the same amount (20%) of contribution.

Which voxel needs quantization? In order to achieve a good balance between rendering quality loss and compression rate, we save a fraction of mostly important voxels without passing them through vector quantization, where the fraction rate is determined by the quantile of the cumulative score in Eqn. 5,

$$\theta_k = F^{-1}(\beta_k), \qquad (10)$$

where β_k denotes the hyperparameter and θ_k represents the keeping threshold. The voxels with the important score larger than θ_k are directly stored, named as non-vector-quantized voxels (*non-VQ voxels*). As shown in Fig. 4, the statistics reveal that the top 1% of voxels can contribute over 60% of the importance, saving a fraction of voxels facilitate rendering quality preservation, only with minimal increase on storage. Using such a strategy can achieve a better trade-off compared to compressing all through VQ.

5.2. Joint Finetune

Performing voxel pruning and vector quantization can generally compress a grid model to 5% of the original size, while we found it bring in unacceptable performance loss (31.88 dB drops to 31.32 db in Table 5). In order to further improve the representation ability of the compressed models, we propose to fine-tune the remaining voxel features (as well as MLPs if the original method used) jointly with the VQ optimization. The insight is similar to tuning models after weight pruning and quantization in deep network compression [14], as we expect the compressed grid model to approach the rendering quality of the original model.

The joint fine-tuning phase is highly efficient. Take the compression on an DVGO model for example. We **fix** the voxel-to-codebook mapping during fine-tuning as we apply a hard mapping index, and update four parts including 1) code vectors in the codebook, 2) volume density, 3) non-VQ voxels and 4) small network originally used in the DVGO. As the size of voxel gradients are extremely large but sparse, we update each code vector by synchronizing weights across the voxels assigned to it for every i iterations, which can boost training efficiency.

6. Post-Processing

We can further reduce model size via the post-processing step, comprised of weight quantization [9] and entropy encoding [25]. We use a simple uniform weight quantization on volume density and the color features of non-VQ voxels without operating on codebook as it is fairly compact. An 8-bit weight quantization casts full-precision floating number to unsigned integers.

We store two boolean masks to identify which voxel have been pruned, vector quantized or directly saved without VQ. The storage cost for a compressed DVGO model comes from the saving of following six components, 1) the 2-bits mask, 2) code vectors (i.e., codebook), 3) mapping indexes between voxels and codebook, 4) 8-bit quantized density, 5) 8-bit quantized non-VQ voxels and 6) small MLPs the method originally used. We encode them with entropy encoding (LZ77 [24, 46]) and pack them together to get the final storage cost.

Methods	Synthetic-NeRF PSNR (dB)↑	Synthetic-NeRF SSIM ↑	Synthetic-NeRF SIZE (MB)↓	Synthetic-NSVF PSNR (dB)↑	Synthetic-NSVF SSIM ↑	Synthetic-NSVF SIZE (MB)↓	LLFF PSNR (dB)↑	LLFF SSIM ↑	LLFF SIZE (MB)↓	Tanks&Temples PSNR (dB)↑	Tanks&Temples SSIM ↑	Tanks&Temples SIZE (MB)↓
NeRF [29]	31.01	0.947	2.5	-	-	-	26.50	0.811	5.0	25.78	0.864	5.0
mip-NeRF [29]	33.06	0.961	2.5	-	-	-	26.66	0.819	2.5	-	-	-
CCNeRF-CP [39]	30.55	0.935	4.4	-	-	-	-	-	-	27.01	0.878	4.4
TensoRF-CP [2]	31.56	0.949	3.9	34.48	0.971	3.9	-	-	-	27.59	0.897	3.9
DVGO [37]	31.90	0.956	105.9	34.90	0.975	119.8	-	-	-	28.29	0.910	113.4
VQ-DVGO	31.77	0.954	1.4	34.72	0.974	1.3	-	-	-	28.26	0.909	1.4
Plenoxels [44]	31.71	0.958	259.8	34.12	0.977	283.3	26.43	0.842	2006.2	26.84	0.911	367.7
VQ-Plenoxels	31.53	0.956	13.7	33.91	0.976	11.9	26.28	0.839	40.0	26.73	0.908	14.3
TensoRF [2]	33.09	0.963	67.6	36.72	0.982	71.6	26.70	0.836	179.8	28.54	0.921	72.6
VQ-TensoRF	32.86	0.960	3.6	36.16	0.980	4.1	26.46	0.824	8.8	28.20	0.913	3.3

Table 1. **Quantitative comparison.** We compare VQRF with origin NeRF, uncompressed volumetric radiance fields and other methods focusing on model size. Compared to all the baselines, our method achieve the best trade-off between rendering quality and model size.

7. Experiments

7.1. Datasets

Synthetic-NeRF is first introduced by [29] and has been widely adopt by following works. It contains 8 scenes rendered at 800×800 resolution. Each scene contains 100 rendered views for training and 200 views for testing.

Synthetic-NSVF [23] includes additional 8 objects with more complex geometry and lighting condition compared to Synthetic-NeRF. Images are rendered with 800×800 pixels.

LLFF [28, 29] includes 8 forward-facing scenes captured by mobile phone cameras in the real world. We follow the setting in [29] by using the images of 1008×756 pixels.

Tanks & Temples [19] is reconstructed from video dataset captured in the real world. All the images are captured at 1920 × 1080 resolution. Here we use five scenes *(Barn, Caterpillar, Family, Ignatius, Truck)* without background.

7.2. Implementation Details

We apply the pipeline on three representative volumetric methods, i.e., DVGO [37], Plenoxels [44] and TensoRF [2]. When adapting VQRF to certain method, we first obtain a non-compressed grid model following the default training configuration of each method. The pruning quantile β_p is set to 0.001 for all the three methods. The keeping quantile β_k is set to 0.6 for DVGO and Plenoxels, and 0.7 for TensoRF. We use 4096 as the default codebook size for all the experiments. Codebook initialization takes 1000 iterations with a batchsize of 10000 voxel points per iteration. Moving average factor λ_d is 0.8, and code expiration number J is set to 10. We use VQ-DVGO as the default method for all the experiments and ablation studies in Section 7. Please refer to the supplemental material for more details.

7.3. Results

Quantitative results. We compare our work with original NeRF and other uncompressed volumetric radiance fields in Table 1. Here the 'VQ-' prefix the corresponding volumetric radiance fields compressed by using our pipeline. All the reported model sizes of original DVGO, Plenoxels, and TensoRF are calculated after using a standard zip compression for a fair comparison.

As shown in Table 1, our method realizes significant benefit on model size with competitive rendering quality compared to all the baseline methods across various datasets. Specifically, VQ-DVGO achieves the highest compression ratio, with 75× on the synthetic-NeRF dataset and an average of 83× on the three datasets, while effectively preserving rendering quality with negligible drop (less than 0.2 dB on PSNR). The performance advantage is consistent with the phenomenon illustrated in Fig. 4, i.e., compared to the models of Plenoxels and TensoRF there exists a greater redundancy in DVGO models due to the use of dense grids. On the other hand, our pipeline can also achieve obvious storage cost reduction on Plenoxels and TensoRF. Plenoxels has already employed sparse voxel grids by pruning off empty voxels to reduce model size, and using decomposed tensor structures in TensoRF naturally introduces compactness on model size. Our method can still realize over 20× compression ratios on both methods with comparable rendering quality. We can also achieve better performance on both rendering quality and storage cost over the compressed setting "TensoRF-CP" proposed in TensoRF, showing the generalizability and effectiveness of our pipeline for pursuing high-performing compact models.

Visual results. We compare the rendering results of the compressed models and corresponding original ones on the

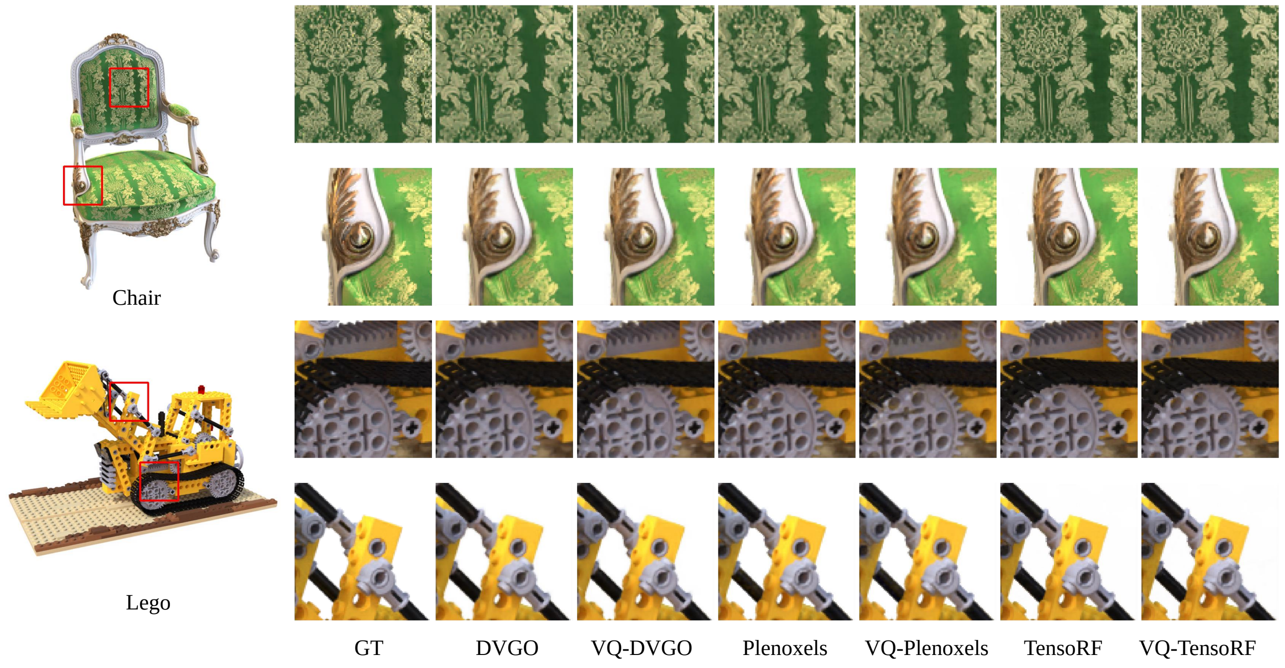


Figure 5. **Qualitative Comparison on Sythetic-NeRF.** We can hardly observe visual artifacts on the rendering results of the compressed models compared to the original models, even in zoom-in images.

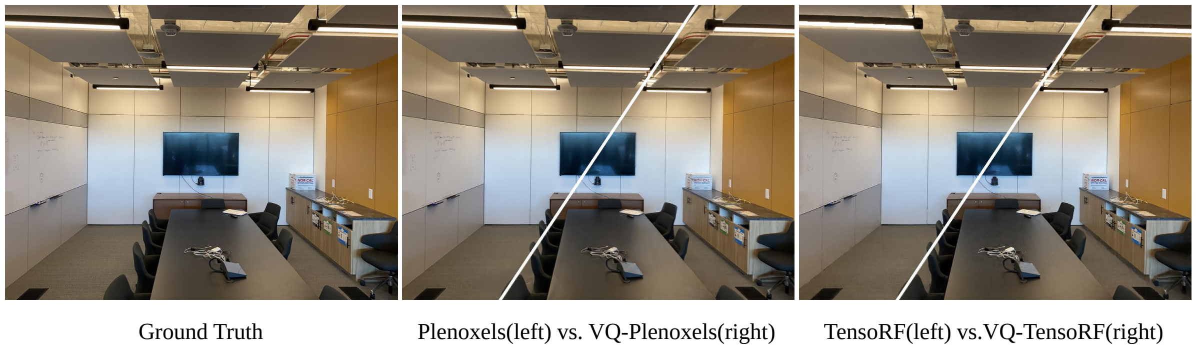

Ground Truth
Plenoxels(left) vs. VQ-Plenoxels(right)
TensoRF(left) vs.VQ-TensoRF(right)

Figure 6. Visual quality comparison of origin model versus our compressed model on real forward-facing dataset.

four datasets in Fig. 5, Fig. 6 and Fig. 7. We provide more visual comparison in the supplemental material. Visual differences are hard to be observed in a variety of 3D scenes, including synthetic bounded, forward-facing and real-world bounded scenes across different volumetric baselines.

7.4. Ablation Study

Pruning and keeping percentage. We estimate the effect of using different β_p and β_k in Table 2. Using a moderate pruning parameter β_p can benefit rendering quality and model size both. A very compact model can be obtained with a relatively large β_p, while some informative voxels may be eliminated as well which would hurt rendering quality. Using a larger keeping parameter β_k could benefit visual performance yet reduce compression ratio. We choose a moderate setting (0.001 and 0.6) as default, and a trade-off between rendering quality and storage cost can be achieved flexibly by adjusting the two hyper-parameters.

Training and testing overhead. We provide the training and inference time in Table 3. Our compression pipeline is very efficient and introduces marginal overhead on the original training pipeline. VQRF generally requires three to four minutes to compress one model. The inference speed of these compressed models are comparable (or even higher) to the original ones.

Codebook size. Codebook capacity is another important factor to influence compression performance. We conduct the experiments by using from extremely small to large sizes, and estimate the effect on rendering quality and storage cost in Table 4. The benefit of using a extremely small

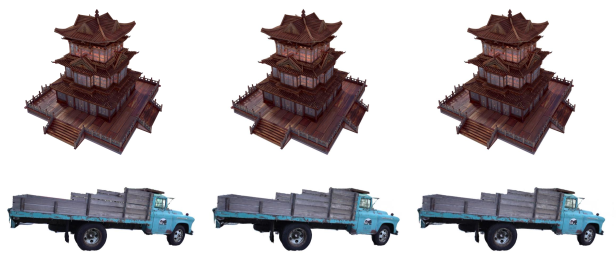


Figure 7. Rendering results on synthetic-NSVF dataset and Tanks&Temples dataset show that our method can well generalize to the datasets with different distribution.

β_p	β_k	PSNR↑	SSIM↑	LPIPS ↓	SIZE↓
0	0	30.38	0.941	0.077	2.54
	0.3	31.13	0.948	0.067	2.73
	0.6	31.60	0.953	0.060	3.06
	0.9	31.94	0.956	0.054	3.86
0.001†	0	31.04	0.947	0.068	0.93
	0.3	31.49	0.951	0.062	1.11
	0.6†	31.77	0.954	0.057	1.43
	0.9	31.96	0.956	0.054	2.21
0.01	0	30.90	0.946	0.070	0.74
	0.3	31.31	0.950	0.040	0.92
	0.6	31.59	0.953	0.059	1.24
	0.9	31.80	0.956	0.055	2.02
0.1	0	27.02	0.920	0.084	0.49
	0.3	27.35	0.935	0.077	0.67
	0.6	27.64	0.939	0.054	0.99
	0.9	28.27	0.945	0.065	1.67

† denote our default choice of β_p and β_k

Table 2. Evaluating pruning quantile β_p and keeping quantile β_k.

size is marginal on the total storage cost while the drop on PSNR is obvious. Using a larger size can usually improve the ability of preserving rendering quality, which tends to reach saturation when using a value larger than 4096. In other word, using a extremely large codebook brings minimal improvement on rendering quality while increasing storage cost obviously.

Step-by-step analysis. We conduct a step-by-step experiment to validate each module in the proposed framework. The results are presented in Table 5. We calculate the size after a zip compression for fair comparison. Compared to the uncompressed baseline, using voxel pruning achieves 5× compression with a negligible PSNR drop, and directly using vector quantization brings additional 4× reduction while leading to PSNR decrease of 0.5dB. By virtue of joint finetuning, rendering quality can be effectively recovered without affecting model size. Our model size can finally reach the level of 1 MB after applying weight quantization.

Method	PSNR↑ (dB)	Size↓ (MB)	Train↓ (min)	Test↓ (ms)
DVGO	31.90	105.9	5	160
VQ-DVGO	31.77	1.4	5+3	180
Plenoxels	31.71	259.8	10	41
VQ-Plenoxels	31.53	13.7	10+3	33
TensoRF	33.09	67.6	12	750
VQ-TensoRF	32.86	3.6	12+4	725

Table 3. Comparison of training and testing overhead.

	PSNR↑	SSIM↑	LPIPS↓	SIZE↓
16	30.63	0.944	0.071	1.040
64	31.19	0.949	0.065	1.134
256	31.44	0.951	0.062	1.164
1024	31.62	0.953	0.059	1.308
4096	31.77	0.954	0.057	1.431
16384	31.81	0.955	0.056	1.630

Table 4. Ablation study of codebook size.

	PSNR↑	SSIM↑	LPIPS↓	SIZE↓
baseline	31.90	0.956	0.054	105.9
+voxel pruning	31.88	0.956	0.054	19.0
+vector quantization	31.32	0.952	0.061	4.8
+joint finetune	31.79	0.954	0.036	4.8
+weight quantization	31.77	0.954	0.057	1.4

Table 5. Step-by-step analysis on performance gain.

Limitation and Discussion. The proposed compression is not lossless. It can achieve significant compression on modern voxel grid-based methods with minimal quality loss (less than 0.2dB) while the method may reach bottleneck on model size, i.e., the quality may drop obviously (in Table 2) if pursuing a further compression to reach a KB-level (e.g., aggressive voxel pruning, more compact quantization).

8. Conclusion

In this paper we proposed VQRF, a novel and versatile compression framework designed for volumetric radiance fields. Our framework employs an adaptive voxel pruning mechanism, a learnable vector quantization, and post-processing techniques to considerably reduce storage cost, even down to 1 MB, without degrading rendering quality. We deploy the framework on several modern volumetric methods, such as DVGO, Plenoxels and TensoRF. Extensive experiments confirm the effectiveness and generalization of VQRF, demonstrating its impressive compression performance on multiple methods across different datasets.

References

[1] Lei Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? *arXiv preprint arXiv:1312.6184*, 2013.

[2] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Proceedings of the European Conference on Computer Vision*, 2022.

[3] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 742–751, 2017.

[4] Pamela C Cosman, Karen L Oehler, Eve A Riskin, and Robert M Gray. Using vector quantization for image processing. *Proceedings of the IEEE*, 81(9):1326–1341, 1993.

[5] Chenxi Lola Deng and Enzo Tartaglione. Compressing explicit voxel grid representations: fast nerfs become also small. *arXiv preprint arXiv:2210.12782*, 2022.

[6] Emily Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *arXiv preprint arXiv:1404.0736*, 2014.

[7] William H Equitz. A new vector quantization clustering algorithm. *IEEE transactions on acoustics, speech, and signal processing*, 37(10):1568–1575, 1989.

[8] Allen Gersho and Robert M Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012.

[9] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

[10] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

[11] Robert Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984.

[12] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10696–10706, 2022.

[13] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pages 1737–1746. PMLR, 2015.

[14] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[15] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul E. Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[17] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[18] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

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

[20] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[21] Yoon Yung Lee and John W Woods. Motion vector quantization for video coding. *IEEE Transactions on Image Processing*, 4(3):378–382, 1995.

[22] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Ping Tan. Streaming radiance fields for 3d video synthesis. In *Advances in Neural Information Processing Systems*, volume 35, pages 13485–13498, 2022.

[23] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *Advances in Neural Information Processing Systems*, 2020.

[24] Jean loup Gailly and Mark Adler. zlib. `https://zlib.net/`.

[25] David JC MacKay, David JC Mac Kay, et al. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[26] John Makhoul, Salim Roucos, and Herbert Gish. Vector quantization in speech coding. *Proceedings of the IEEE*, 73(11):1551–1588, 1985.

[27] Nelson L. Max. Optical models for direct volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 1995.

[28] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics*, 2019.

[29] 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 *Proceedings of the European Conference on Computer Vision*, 2020.

[30] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022.

[31] Nasser M Nasrabadi and Robert A King. Image coding using vector quantization: A review. *IEEE Transactions on communications*, 36(8):957–971, 1988.

[32] Kuldip K Paliwal and Bishnu S Atal. Efficient vector quantization of lpc parameters at 24 bits/frame. *IEEE transactions on speech and audio processing*, 1(1):3–14, 1993.

[33] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

[34] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

[35] Roberto Rigamonti, Amos Sironi, Vincent Lepetit, and Pascal Fua. Learning separable filters. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2754–2761, 2013.

[36] Thomas Sikora. The mpeg-4 video standard verification model. *IEEE Transactions on circuits and systems for video technology*, 7(1):19–31, 1997.

[37] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[38] Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. Variable bitrate neural fields. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022.

[39] Jiaxiang Tang, Xiaokang Chen, Jingbo Wang, and Gang Zeng. Compressible-composable nerf via rank-residual decomposition. In *Advances in Neural Information Processing Systems*, 2022.

[40] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[41] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao. Improving the speed of neural networks on cpus. 2011.

[42] Zhongshu Wang, Lingzhi Li, Zhen Shen, Li Shen, and Liefeng Bo. 4k-nerf: High fidelity neural radiance fields at ultra high resolutions. *arXiv preprint arXiv:2212.04701*, 2022.

[43] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *arXiv preprint arXiv:1608.03665*, 2016.

[44] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[45] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[46] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.