LVAC: Learned Volumetric Attribute Compression for Point Clouds using Coordinate Based Networks

Berivan Isik*
Stanford University
berivan.isik@stanford.edu

Philip A. Chou
Google
philchou@google.com

Sung Jin Hwang
Google
sjhwang@google.com

Nick Johnston
Google
nickj@google.com

George Toderici
Google
gtoderici@google.com

Abstract

We consider the attributes of a point cloud as samples of a vector-valued volumetric function at discrete positions. To compress the attributes given the positions, we compress the parameters of the volumetric function. We model the volumetric function by tiling space into blocks, and representing the function over each block by shifts of a coordinate-based, or implicit, neural network. Inputs to the network include both spatial coordinates and a latent vector per block. We represent the latent vectors using coefficients of the region-adaptive hierarchical transform (RAHT) used in the MPEG geometry-based point cloud codec G-PCC. The coefficients, which are highly compressible, are rate-distortion optimized by back-propagation through a rate-distortion Lagrangian loss in an auto-decoder configuration. The result outperforms RAHT by 2–4 dB. This is the first work to compress volumetric functions represented by local coordinate-based neural networks. As such, we expect it to be applicable beyond point clouds, for example to compression of high-resolution neural radiance fields.

1. Introduction

Our work addresses the problem of 3D point cloud attribute compression, using coordinate-based neural networks. Point clouds are a fundamental data type underlying 3D sampling and hence play a critical role in applications such as mapping and navigation, virtual and augmented reality, telepresence, and cultural heritage preservation, which rely on sampled 3D data [47, 55, 60, 76]. Given the volume of data in such applications, compression is important for both storage and communication. Indeed, standards for point cloud compression are underway in both MPEG and JPEG [2, 24, 36, 69].

Figure 1. Point clouds *rock*, *scooter*, *juggling*, and *basketball*.

3D point clouds, such as those shown in Fig. 1, each consist of a set of points $\{(\mathbf{x}_i, \mathbf{y}_i)\}$, where $\mathbf{x}_i$ is the 3D position of the ith point and $\mathbf{y}_i$ is a vector of attributes associated with the point. Attributes typically include color components, e.g., RGB, but may alternatively include reflectance, normals, transparency, density, spherical harmonics, and so forth. Commonly (e.g., [16, 18, 20, 21, 39, 58, 69, 82, 91]), point cloud compression is broken into two steps: compression of the point cloud positions, called the *geometry*, and compression of the point cloud *attributes*. Compression of the attributes is conditioned on the decoded geometry, as illustrated in Fig. 2. It is important to note that this conditioning is crucial in achieving good compression. This will become one of the themes of this paper.

Following successful application of neural networks in image compression [5–9, 32, 48, 53, 83, 84], neural networks have been used successfully for point cloud geometry compression, demonstrating significant gains over traditional techniques [25–27, 61, 63, 81, 88]. However, the same cannot be said for point cloud attribute compression. To our knowl-

*Work done while the first author was an intern at Google.

edge, our work is among the first to use neural networks for point cloud attribute compression. Previous attempts may have been be hindered by the inability to properly condition the attribute compression on the decoded geometry, thus leading to poor results. In our work, we show that proper conditioning improves attribute compression performance by over 30% reduction in BD-Rate. This results in a gain of 2–4 dB over region-adaptive linear transform (RAHT) coding, which is used in the "geometry-based" point cloud compression standard MPEG G-PCC.

Although learned image compression systems have been based on convolutional neural networks (CNNs), in this work we employ what have come to be called *coordinate based networks* (CBNs), also called *implicit networks*. (See [19, 79] and the references in Sec. 2.) A CBN is a network, such as a multilayer perceptron (MLP), whose inputs include the coordinates of the spatial domain of interest, e.g., $\mathbf{x} \in \mathbb{R}^3$. Thus a CBN can directly represent a nonlinear function of the spatial coordinates $\mathbf{x}$, possibly indexed with a latent or feature vector $\mathbf{z}$, as $\mathbf{y} = f_\theta(\mathbf{x})$ or $\mathbf{y} = f_\theta(\mathbf{x}; \mathbf{z})$. CBNs have recently come to the fore in accurately representing geometry and spatial phenomena such as radiance fields. However, while there has been an explosion of work using CBNs for *representing* specific objects and scenes, none of that work focuses on *compressing* those representations. (Two exceptions are [13, 33], which apply model compression to the CBNs themselves.) Good lossy compression is nontrivial, and must make the optimal trade-off between the fidelity of the reconstruction and the number of bits used in its binary representation. We show that naïve scalar quantization and entropy coding of the parameters θ and/or latent vectors $\mathbf{z}$ lead to very poor results, and that superior results can be achieved by proper normalization prior to uniform scalar quantization. This normalization amounts to using different quantization step sizes, or different numbers of bits, for different latent vectors — depending on the geometry. In addition, the entropy model and CBN must be jointly trained to minimize a loss function that penalizes not only large distortion (or error) but large bit rate as well.

Our main contributions include the following:

- We are among the first to compress point cloud *attributes* using neural networks. Our solution allows the network to interpolate the reconstructed attributes continuously across space, and offers a 2–4 dB improvement over our linear baseline, RAHT with adaptive Run-Length Golomb-Rice (RLGR) entropy coding. Note that RAHT is the transform used in the latest MPEG G-PCC standard.

- We are the first to *compress volumetric functions* modeled by *local coordinate based networks*, by training with the rate-distortion Lagrangian as the loss function (as in image compression), thereby offering scalable, high fidelity reconstructions at low bit rates. We show that naïve uniform scalar quantization and entropy coding leads to poor results, and we show formulas for normalizing the coefficients to achieve over a 30% reduction in bit rate.

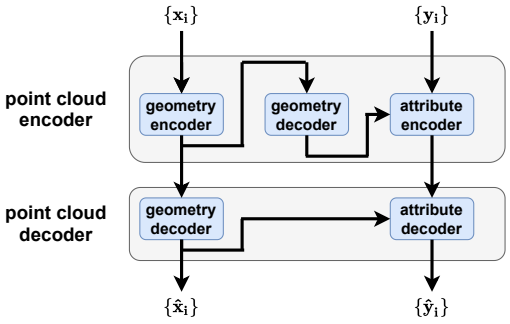

Figure 2. Point cloud codec, consisting of a geometry encoder and decoder, and an attribute encoder and decoder conditioned on the decoded geometry.

Section 2 covers related work, Sec. 3 details our Learned Volumetric Attribute Compression (LVAC) framework, Sec. 4 reports experimental results, and Sec. 5 discusses and concludes.

2. Related Work

2.1. Coordinate Based Networks

Early work that used coordinate based networks [49, 56, 71], exemplified by DeepSDF [56], focused on representing geometry *implicitly*, for example as the c-level set $\{\mathbf{x} : c = f_\theta(\mathbf{x}; \mathbf{z})\} \subset \mathbb{R}^3$ of a function $f_\theta : \mathbb{R}^3 \times \mathbb{R}^C \to \mathbb{R}$ modeled by a neural network, where $\mathbf{z} \in \mathbb{R}^C$ is a global latent vector. As a result such networks were called "implicit" networks. Much of this work focused on auto-decoder architectures, in which the latent vector $\mathbf{z}$ was determined for each instance by back propagation through the loss function. The loss function $L(\theta, \mathbf{z})$ measured a pointwise error between samples $f_\theta(\mathbf{x}_i; \mathbf{z})$ of the network and samples $f(\mathbf{x}_i)$ of a ground truth function, such as the signed distance function (SDF).

Later work that used CBNs, exemplified by NeRF [11, 52], used the networks to model not SDFs but rather other, vector-valued, volumetric functions, including color, density, normals, BRDF parameters, and specular features [31, 37, 73, 89, 94]. Since these networks were no longer used to represent solutions implicitly, their name started to shift to "coordinate-based" networks, e.g., [79]. An important innovation from this cohort was positional encoding, in which the network's positional input $\mathbf{x}$ was embedded into a higher dimensional feature space by sinusoidal maps, which greatly improved the spatial resolution of the networks [12, 45, 52, 72, 80, 95]. Another key innovation was

measuring the loss $L(\theta)$ not pointwise between samples of f_θ and some ground truth volumetric function f, but rather between volumetric renderings (to images) of f_θ and f, the latter renderings being ground truth images.

NeRF et al. focused on training the CBN $f_\theta(\mathbf{x})$ to globally represent a single scene, without benefit of a latent vector $\mathbf{z}$. However, subsequent work shifted towards using the CBN with different latent vectors for different objects [74, 90] or different regions (i.e., blocks or tiles) in the scene [15, 44, 45, 64, 78]. Partitioning the scene into blocks, and using a CBN with a different latent vector in each block, simultaneously achieves faster rendering [64, 78], higher resolution [15, 44, 45], and scalability to scenes of unbounded size [23]. However, this puts much of the burden of the representation on the local latent vectors, rather than on the parameters of the CBN. This is analogous to conventional block-based image representations, in which the same set of basis functions (e.g., 8×8 DCT) is used in each block, and activation of each basis vector is specified by a vector of basis coefficients, different for each block.

Our work borrows heavily from these works. We partition 3D space into blocks (hierarchically using trees, akin to [44, 78, 89]), and represent the color within each block volumetrically using a CBN $f_\theta(\mathbf{x}; \mathbf{z})$, allowing fast, high-resolution, and scalable reconstruction. Unlike all previous CBN works, however, we train the representation not just for fit but for efficient compression using techniques from learned image compression.

2.2. Learned Image Compression

Using neural networks for good compression is non-trivial. Simply truncating the latent vectors of an existing representation to a certain number of bits is likely to fail, if only because small quantization errors in the latents may easily map into large quantization errors in their reconstructions. Moreover, the entropy of the quantized latents is a more important determiner of the bit rate than the total number of coefficients in the latent vectors or the number of bits in their binary representation. Early work on learned image compression could barely exceed the rate-distortion performance of JPEG on low-quality 32×32 thumbnails [83]. However, over the years the rate-distortion performance has consistently improved [5–9, 32, 53, 84] to the point where the best learned image codecs outperform the latest video standard (VVC) in PSNR, albeit at much greater complexity [29], and greatly outperform conventional image codecs (by over $2\times$ reduction in bit rate) at the same perceptual distortion [48]. All current competitive learned image codecs are versions of nonlinear transform coding [5], in which the bottleneck latents in an auto-encoder are uniformly scalar quantized and entropy coded (with a hyperprior), for transmission to a decoder. The decoder uses a convolutional neural network as a synthesis transform. The codec is trained end-to-end through a differentiable proxy for the quantizer, often modeled as additive uniform noise. The loss function is a Lagragian $L(\theta) = D(\theta) + \lambda R(\theta)$, where $D(\theta)$ is an expected distortion and $R(\theta)$ is an expected bit rate (i.e., cross-entropy), and $\lambda > 0$ is a Lagrange multiplier.

Our work borrows significantly from this work, in that we use the same uniform scalar quantization and entropy model (though without a hyperprior for now) as used for the best learned image compression. Moreover, we train our representation using a similar Lagrangian loss function.

2.3. Point Cloud Compression

MPEG is standardizing two point cloud codecs: video-based (V-PCC) and geometry-based (G-PCC) [24, 36, 69]. V-PCC is based on existing video codecs, while G-PCC is based on new, but in many ways classical, geometric approaches. Like previous works [16, 18, 20, 21, 39, 58, 82, 91], both V-PCC and G-PCC compress geometry first, then compress attributes conditioned on geometry. Neural networks have been applied with some success to geometry compression [25–27, 41, 50, 51, 61, 63, 81, 88], but not to attribute compression. Exceptions may include [62], which uses learned neural 3D→2D folding but compresses with conventional image coding, and [70], which compresses attributes using a PointNet-style architecture, which is not volumetric. The attribute compression in G-PCC uses linear transforms, which adapt based on the geometry. A core transform is the region-adaptive hierarchical transform (RAHT) [20, 68], which is a linear transform that is orthonormal with respect to a discrete measure whose mass is put on the point cloud geometry [16, 67]. Thus RAHT compresses attributes conditioned on geometry. Beyond RAHT, G-PCC uses prediction (of the RAHT coefficients) and joint entropy coding to obtain superior performance [3, 40, 59].

Our work borrows heavily from RAHT, as we apply RAHT's orthonormalization formulas to our latent vectors. It turns out that this is crucial for good rate-distortion performance for point cloud attribute compression.

3. LVAC Framework

3.1. Approach to Volumetric Representation

A real-valued (or real vector-valued) function

$$f : \mathbb{R}^d \to \mathbb{R}^r \tag{1}$$

is said, informally, to be *volumetric* if $d = 3$ and *hyper-volumetric* if $d > 3$. A volumetric (or hyper-volumetric) function f may be fit by another volumetric function f_θ from a parametric family of volumetric functions $\{f_\theta : \theta \in \Theta\}$ by minimizing an error $d(f, f_\theta)$ over $\theta \in \Theta$. A simple example is linear regression. Suppose $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ is a point cloud with point positions $\mathbf{x}_i \in \mathbb{R}^3$ and point at-

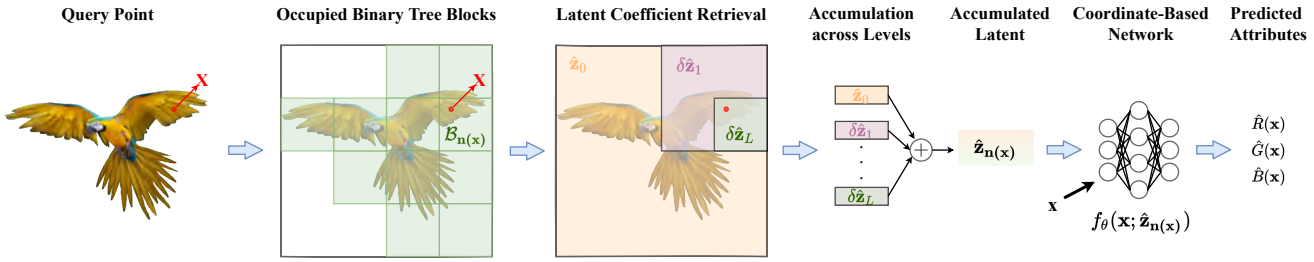

Figure 3. Querying attributes at position $\mathbf{x} \in \mathbb{R}^3$. The block $\mathcal{B}_{\mathbf{n}(\mathbf{x})}$ at target level L in which $\mathbf{x}$ is located is identified by traversing a binary space partition tree. The quantized latent $\hat{\mathbf{z}}_0$ at the root and quantized difference latents $\delta\hat{\mathbf{z}}_\ell$ at increasing levels of detail $\ell = 1, \ldots, L$ are accumulated to form a cumulative latent $\hat{\mathbf{z}}_{\mathbf{n}(\mathbf{x})}$, which is input along with $\mathbf{x}$ to the CBN at level L, to produce $\hat{\mathbf{y}} = f_\theta(\mathbf{x}; \hat{\mathbf{z}}_{\mathbf{n}(\mathbf{x})})$ (after [78]).

tributes $\mathbf{y}_i = f(\mathbf{x}_i) \in \mathbb{R}^r$. Then an affine function

$$\mathbf{y} = f_\theta(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}, \tag{2}$$

with $\theta = (\mathbf{A}, \mathbf{b})$, may be fit to the data by minimizing the squared error $d(f, f_\theta) = ||f - f_\theta||^2 = \sum_i ||f(\mathbf{x}_i) - f_\theta(\mathbf{x}_i)||^2$ over θ. Although a linear or affine volumetric function may not be able to represent adequately the complex spatial arrangement of colors of point clouds like those in Fig. 1, two strategies may be used to improve the fit. The first is to make f_θ far more complex, e.g., represented by a CBN with millions of parameters. The second is to partition the scene into blocks, and use a simpler CBN within each block. LVAC chooses the latter approach.

3.2. Latent Vectors

To be precise, in LVAC, the attributes $\mathbf{y}_i$ in a block $\mathcal{B}_\mathbf{n}$ at offset $\mathbf{n}$ are fit with a volumetric function $\mathbf{y} = f_\theta(\mathbf{x}-\mathbf{n}; \mathbf{z}_\mathbf{n})$ represented by a simple CBN, shifted to offset $\mathbf{n}$. The CBN parameters θ are learned and fixed for each point cloud. However, each block $\mathcal{B}_\mathbf{n}$ supplies its own latent vector $\mathbf{z}_\mathbf{n}$, which selects the exact volumetric function $f_\theta(\cdot; \mathbf{z})$ used in the block. The role of θ is to choose the *family* of volumetric functions best for each point cloud, or for point clouds in general. The role of $\mathbf{z}$ is to choose a member of the family best for each block. The overall volumetric function may be expressed as

$$\mathbf{y} = f_{\theta,\mathbf{Z}}(\mathbf{x}) = \sum_\mathbf{n} f_\theta(\mathbf{x} - \mathbf{n}; \mathbf{z}_\mathbf{n}) \mathbb{1}_{\mathcal{B}_\mathbf{n}}(\mathbf{x}), \tag{3}$$

where the sum is over all block offsets $\mathbf{n}$, $\mathbb{1}_{\mathcal{B}_\mathbf{n}}$ is the indicator function for block $\mathcal{B}_\mathbf{n}$ (i.e., $\mathbb{1}_{\mathcal{B}_\mathbf{n}}(\mathbf{x}) = 1$ iff the query point x is inside $\mathcal{B}_\mathbf{n}$), and $\mathbf{Z} = [\mathbf{z}_\mathbf{n}]$ is the matrix whose rows $\mathbf{z}_\mathbf{n}$ are the blocks' latent vectors.

To compress the point cloud attributes $\{\mathbf{y}_i\}$ given the geometry $\{\mathbf{x}_i\}$, LVAC compresses and transmits $\mathbf{Z}$ and possibly θ as $\hat{\mathbf{Z}}$ and $\hat{\theta}$ using $R(\hat{\theta}, \hat{\mathbf{Z}})$ bits. This communicates the volumetric function $f_{\hat{\theta},\hat{\mathbf{Z}}}$ to the decoder. The decoder can then use $f_{\hat{\theta},\hat{\mathbf{Z}}}$ to reconstruct the attributes $\mathbf{y}_i$ at each point position $\mathbf{x}_i$ as $\hat{\mathbf{y}}_i = f_{\hat{\theta},\hat{\mathbf{Z}}}(\mathbf{x}_i)$, incurring distortion

$$D(\hat{\theta}, \hat{\mathbf{Z}}) = d(f, f_{\hat{\theta},\hat{\mathbf{Z}}}) = \sum_i ||\mathbf{y}_i - f_{\hat{\theta},\hat{\mathbf{Z}}}(\mathbf{x}_i)||^2. \tag{4}$$

The decoder can also use $\mathbf{y} = f_{\hat{\theta},\hat{\mathbf{Z}}}(\mathbf{x})$ to reconstruct the attributes $\mathbf{y}$ at an *arbitrary* position $\mathbf{x} \in \mathbb{R}^3$. However, LVAC minimizes the distortion $D(\hat{\theta}, \hat{\mathbf{Z}})$ subject to a constraint on the bit rate, $R(\hat{\theta}, \hat{\mathbf{Z}}) \leq R_0$. This is done by minimizing the Lagrangian $J(\hat{\theta}, \hat{\mathbf{Z}}) = D(\hat{\theta}, \hat{\mathbf{Z}}) + \lambda R(\hat{\theta}, \hat{\mathbf{Z}})$ for some Lagrange multiplier $\lambda > 0$ matched to R_0.

In the regime of interest in our work, θ has about 250-10K parameters, while $\mathbf{Z}$ has about 500K-8M parameters. Hence the focus of this paper is on compression of $\mathbf{Z}$. We assume that the simple CBN parameterized by θ can be compressed using model compression tools, e.g., [13, 33], to a few bits per parameter with little loss in performance. Alternatively, we assume that the CBN may be trained to generalize across point clouds, obviating the need to transmit θ. In Sec. 4, we explore conservative bounds on the performance of each assumption. In this section, however, we focus on compression of the latent vectors $\mathbf{Z} = [\mathbf{z}_\mathbf{n}]$.

3.2.1 Latent Vector Compression in RAHT

LVAC compresses the latent vectors $[\mathbf{z}_\mathbf{n}]$ using, essentially, RAHT. Hence, we first discuss how RAHT compresses color attributes. Though there are many ways to view RAHT, one way to view it is as compression of a piecewise constant volumetric function,

$$\mathbf{y} = f_\mathbf{Z}(\mathbf{x}) = \sum_\mathbf{n} \mathbf{z}_\mathbf{n} \mathbb{1}_{\mathcal{B}_\mathbf{n}}(\mathbf{x}). \tag{5}$$

This is the same as (3) with an extremely simple CBN: $f_\theta(\mathbf{x}; \mathbf{z}) = \mathbf{z}$. In this case, each latent $\mathbf{z}_\mathbf{n} \in \mathbb{R}^3$ directly represents a color, which is constant across block $\mathcal{B}_\mathbf{n}$. It is clear that the squared error $||f - f_\mathbf{Z}||^2$ is minimized by setting every $\mathbf{z}_\mathbf{n}$ to the average (DC) value of the colors of the points in $\mathcal{B}_\mathbf{n}$. RAHT does not quantize and entropy code the colors $\mathbf{Z} = [\mathbf{z}_\mathbf{n}]$ directly, which would be inefficient. Rather,

RAHT first transforms the $N\times C$ matrix $\mathbf{Z}$ using a geometry-dependent $N\times N$ analysis transform $\mathbf{T}_a$, to obtain the $N\times C$ matrix of transform coefficients $\mathbf{V}=\mathbf{T}_a\mathbf{Z}$, most of which may be near zero. (Here, N is the number of blocks $\mathcal{B}_\mathbf{n}$ that are *occupied*, i.e., that contain points, and $C=3$ is the number of color attributes.) Then $\mathbf{V}$ is quantized to $\hat{\mathbf{V}}$ and efficiently entropy coded. Finally $\hat{\mathbf{Z}}=\mathbf{T}_s\hat{\mathbf{V}}$ is recovered using the $N\times N$ synthesis transform $\mathbf{T}_s=\mathbf{T}_a^{-1}$.

The analysis and synthesis transforms $\mathbf{T}_a$ and $\mathbf{T}_s$ are defined in terms of a hierarchical space partition represented by a binary tree. The root of the tree (level $\ell=0$) corresponds to a large block $\mathcal{B}_{0,\mathbf{0}}$ containing the entire point cloud. The leaves of the tree (level $\ell=L$) correspond to the N blocks $\mathcal{B}_{L,\mathbf{n}}=\mathcal{B}_\mathbf{n}$ in (5), which are voxels of a voxelized point cloud. In between, for each level $\ell=0,1,\ldots,L-1$, each occupied block $\mathcal{B}_{\ell,\mathbf{n}}$ at level ℓ is split into left and right child blocks of equal size, say $\mathcal{B}_{\ell+1,\mathbf{n}_L}$ and $\mathcal{B}_{\ell+1,\mathbf{n}_R}$, at level $\ell+1$. The split is along either the x, y, or z axis depending on whether ℓ mod 3 is 0, 1, or 2. Only child blocks that are occupied are retained in the tree.

To perform the linear analysis transform $\mathbf{T}_a\mathbf{Z}$, RAHT starts at level $\ell=L-1$ and works back to level $\ell=0$, computing the average (DC) value of each block $\mathcal{B}_{\ell,\mathbf{n}}$ as

$$\mathbf{z}_{\ell,\mathbf{n}}=\frac{w_{\mathbf{n}_L}}{w_{\mathbf{n}_L}+w_{\mathbf{n}_R}}\mathbf{z}_{\ell+1,\mathbf{n}_L}+\frac{w_{\mathbf{n}_R}}{w_{\mathbf{n}_L}+w_{\mathbf{n}_R}}\mathbf{z}_{\ell+1,\mathbf{n}_R}, \quad (6)$$

where $w_{\mathbf{n}_L}=w_{\ell+1,\mathbf{n}_L}$ and $w_{\mathbf{n}_R}=w_{\ell+1,\mathbf{n}_R}$ are the *weights* of, or number of points in, the left and right child blocks of $\mathcal{B}_{\ell,\mathbf{n}}$. The global DC value of the entire point cloud is $\mathbf{z}_{0,\mathbf{0}}$. Along the way, RAHT computes the difference between the DC values of each child block and its parent as

$$\delta\mathbf{z}_{\ell+1,\mathbf{n}_L} = \mathbf{z}_{\ell+1,\mathbf{n}_L}-\mathbf{z}_{\ell,\mathbf{n}}, \quad (7)$$
$$\delta\mathbf{z}_{\ell+1,\mathbf{n}_R} = \mathbf{z}_{\ell+1,\mathbf{n}_R}-\mathbf{z}_{\ell,\mathbf{n}}. \quad (8)$$

These differences are close to zero and are efficient to entropy code. The $N\times C$ matrix of transform coefficients $\mathbf{V}=\mathbf{T}_a\mathbf{Z}$ consist of the global DC value $\mathbf{z}_{0,\mathbf{0}}$ in the first row, and the $N-1$ *right child* differences $\delta\mathbf{z}_{\ell+1,\mathbf{n}_R}$ computed in (8) in the remaining rows.

To perform the linear synthesis transform $\mathbf{T}_s\mathbf{V}$, RAHT starts at level $\ell=0$ and works up to level $L-1$, computing the *left child* differences $\delta\mathbf{z}_{\ell+1,\mathbf{n}_L}$ (7) from the *right child* differences $\delta\mathbf{z}_{\ell+1,\mathbf{n}_R}$ (8) in $\mathbf{V}$ using the constraint

$$\mathbf{0}=\frac{w_{\mathbf{n}_L}}{w_{\mathbf{n}_L}+w_{\mathbf{n}_R}}\delta\mathbf{z}_{\ell+1,\mathbf{n}_L}+\frac{w_{\mathbf{n}_R}}{w_{\mathbf{n}_L}+w_{\mathbf{n}_R}}\delta\mathbf{z}_{\ell+1,\mathbf{n}_R}, \quad (9)$$

which is obtained from (6) using (7)-(8). Then (7)-(8) are inverted to obtain $\mathbf{z}_{\ell+1,\mathbf{n}_L}$ and $\mathbf{z}_{\ell+1,\mathbf{n}_R}$ from $\mathbf{z}_{\ell,\mathbf{n}}$, ultimately computing the values $\mathbf{z}_{L,\mathbf{n}}=\mathbf{z}_\mathbf{n}$ for blocks at level L.

Expressions for the $N\times N$ matrices $\mathbf{T}_a$ and $\mathbf{T}_s$ can be worked out from the above linear operations. In particular, it can be shown that each row of $\mathbf{T}_s$ computes the color $\mathbf{z}_{L,\mathbf{n}}$ of some leaf voxel $\mathcal{B}_{L,\mathbf{n}}$ by summing the color $\mathbf{z}_0$ of the root block with the color differences $\delta\mathbf{z}_\ell$ at levels of detail $\ell=1,\ldots,L$ from the root to the leaf. Moreover, it can be shown that $\mathbf{T}_a$ and $\mathbf{T}_s$ can be orthonormalized by multiplication by a diagonal matrix $\mathbf{S}=\mathrm{diag}(s_1,\ldots,s_N)$, where

$$s_1 = (\#\text{ points in point cloud})^{-1/2}, \quad (10)$$
$$s_m = \left(\frac{w_{\ell+1,\mathbf{n}_L}(w_{\ell+1,\mathbf{n}_L}+w_{\ell+1,\mathbf{n}_R})}{w_{\ell+1,\mathbf{n}_R}}\right)^{-1/2}, \quad (11)$$

where element s_1 of $\mathbf{S}$ corresponds to row 1 of $\mathbf{V}$ (the global DC value $\mathbf{z}_{0,\mathbf{0}}$) and element s_m of $\mathbf{S}$ corresponds to row $m>1$ of $\mathbf{V}$ (a right child difference $\delta\mathbf{z}_{\ell+1,\mathbf{n}_R}$). That is, $\mathbf{S}^{-1}\mathbf{T}_a$ and $\mathbf{T}_s\mathbf{S}$ are orthonormal (and transposes of each other). This implies that the every row of the normalized coefficients $\bar{\mathbf{V}}=\mathbf{S}^{-1}\mathbf{V}$ should be quantized uniformly with the same step size Δ, or equivalently that the rows of the unnormalized coefficients $\mathbf{V}=\mathbf{T}_a\mathbf{Z}$ should be quantized with scaled step sizes $s_m\Delta$. This scaling is crucial for RAHT, as it quantizes with finer precision the coefficients that are more important. The more important coefficients are generally associated with blocks with more points. This establishes a dependency of the attribute compression on the geometry (see Fig. 2). An alternative way to understand the scaling is that it ensures that the quantization error stays the same, rather than blowing up, after the synthesis transform.

3.2.2 Latent Vector Compression in LVAC

LVAC quantizes and entropy codes the latent vectors $\mathbf{z}_\mathbf{n}\in\mathbb{R}^C$ (where now $C\gg 3$ typically) for the blocks $\mathcal{B}_\mathbf{n}$ in (3), adapting RAHT with the following *crucial differences*:

First, the blocks $\mathcal{B}_\mathbf{n}=\mathcal{B}_{L,\mathbf{n}}$ are at a *target level* of detail L, lower (i.e., coarser) than the voxel level. Thus the blocks $\mathcal{B}_{L,\mathbf{n}}$ contain say $N_x\times N_y\times N_z$ voxels, only some of which are occupied. Then the attributes (typically, colors) of the occupied voxels in $\mathcal{B}_{L,\mathbf{n}}$ are represented by the volumetric function $f_\theta(\mathbf{x}-\mathbf{n};\mathbf{z}_\mathbf{n})$ of a CBN at level L, which better models the attributes within the block at certain bit rates.

Second, since the latent vectors $\mathbf{z}_\mathbf{n}\in\mathbb{R}^C$ are not themselves the attributes of the occupied voxels, they are not a direct input to the encoder. Hence the encoder cannot apply the analysis transform $\mathbf{T}_a$ to $\mathbf{Z}=[\mathbf{z}_\mathbf{n}]$ to obtain the transform coefficients $\mathbf{V}$. Instead, LVAC learns $\mathbf{V}$ through back-propagation, without an explicit $\mathbf{T}_a$, first through the distortion measure and volumetric function (4), and then through the synthesis transform $\mathbf{T}_s$ and scaling matrix $\mathbf{S}$. The coefficients θ of the CBN may be optimized at the same time. In short, LVAC is learned while RAHT is not.

Third, learning gives LVAC the opportunity to optimize $\mathbf{V}$ not just to minimize the distortion D, but to minimize the ultimate rate-distortion objective $D+\lambda R$, which minimizes the distortion subject to a bit rate constraint.

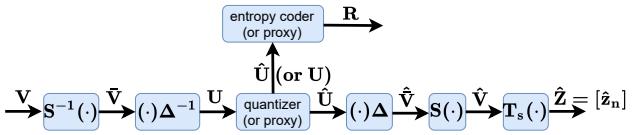

Figure 4. LVAC pipeline for compressing latents $\mathbf{Z} = [\mathbf{z_n}]$. $\mathbf{Z}$ is represented by difference latents $\mathbf{V}$, normalized by $\mathbf{S}$ across levels and blocks to obtain $\bar{\mathbf{V}}$, divided by step sizes $\mathbf{\Delta}$ across channels to obtain $\mathbf{U}$, quantized by rounding to obtain $\hat{\mathbf{U}} = \lfloor \mathbf{U} \rceil$, and reconstructed as $\hat{\mathbf{Z}} = \mathbf{T}_s \mathbf{S} \hat{\mathbf{U}} \mathbf{\Delta}$. $\mathbf{V}$ is optimized by back-propagating through $D(\theta, \mathbf{Z}) + \lambda R(\theta, \mathbf{Z})$ and the pipeline using differentiable proxies for the quantizer and entropy coder.

Figure 4 shows the compression pipeline that produces $\hat{\mathbf{Z}} = [\hat{\mathbf{z}}_\mathbf{n}]$ from $\mathbf{V}$, through which the back-propagation must be performed. The diagonal matrix $\mathbf{S}$ (defined in (10)-(11)) scales the coefficients in $\mathbf{V}$ to produce $\bar{\mathbf{V}} = \mathbf{S}^{-1}\mathbf{V}$, but is constant across channels $c = 1, \ldots, C$. The diagonal matrix $\mathbf{\Delta} = \text{diag}(\Delta_1, \ldots, \Delta_C)$ applies different step sizes Δ_c to each channel in $\bar{\mathbf{V}}$ to produce $\mathbf{U} = \bar{\mathbf{V}}\mathbf{\Delta}^{-1}$, but is constant across coefficients. The quantizer rounds the real matrix $\mathbf{U}$ elementwise to produce the integer matrix $\hat{\mathbf{U}} = \lfloor \mathbf{U} \rceil$, which is then entropy coded to produce a bit string of length R in total. The integer matrix $\hat{\mathbf{U}}$ is also transformed by $\mathbf{\Delta}$, $\mathbf{S}$, and $\mathbf{T}_s$ in sequence to produce $\hat{\mathbf{Z}} = \mathbf{T}_s \mathbf{S} \hat{\mathbf{U}} \mathbf{\Delta}$. Note that learnable parameters in Fig. 4 are $\mathbf{V}$, $\mathbf{\Delta}$, and parameters of the entropy coder.

Since the quantizer and entropy encoder are not differentiable (or more precisely, their derivatives with respect to $\mathbf{V}$ are trivially zero almost everywhere), they must be replaced by differentiable *proxies* during optimization. Various differentiable proxies for the quantizer are possible [4, 42], but as in [8] and others, we use the proxy $Q(\mathbf{U}) = \mathbf{U} + \mathbf{W}$, where $\mathbf{W}$ is iid unif$(-0.5, 0.5)$. Various differentiable proxies for the entropy coder are also possible. As the number of bits in the entropy code for $\mathbf{U} = [u_{m,c}]$, we use the proxy $R(\mathbf{U}) = -\sum_{m,c} \log_2 p_{\phi_{\ell,c}}(u_{m,c})$, where

$$p_{\phi_{\ell,c}}(u) = \text{CDF}_{\phi_{\ell,c}}(u + 0.5) - \text{CDF}_{\phi_{\ell,c}}(u - 0.5) \quad (12)$$

[8]. The CDF is modeled by a neural network with parameters $\phi_{\ell,c}$ that depend on the channel c and also the level ℓ (but not the offset $\mathbf{n}$) of the coefficient $u_{m,c}$. At inference time, the bit rate is $R(\lfloor \mathbf{U} \rceil)$ instead of $R(\mathbf{U})$. These functions are provided by the Continuous Batched Entropy (*cbe*) model with the Noisy Deep Factorized prior in [1].

Note that the parameters Δ_c as well as the parameters $\phi_{\ell,c}$, for all ℓ and c, must be transmitted to the decoder. However, the overhead for transmitting Δ_c is negligible, and the overhead for transmitting $\phi_{\ell,c}$ can be circumvented by using a backward-adaptive entropy code, the adaptive Run-Length Golomb-Rice (RLGR) code [43] in its place at inference time.

Figure 5. Point clouds *chair*, *basketball2*, and *jacket*.

3.3. Coordinate Based Network

Any coordinate based network can be used in the LVAC framework, but in our experiments we use a two-layer MLP,

$$\mathbf{y} = f_\theta(\mathbf{x}; \mathbf{z}) = \sigma(\mathbf{b}^3 + \mathbf{W}^{3\times H} \sigma(\mathbf{b}^H + \mathbf{W}^{H\times(3+C)}[\mathbf{x}, \mathbf{z}])) \quad (13)$$

where $\theta = (\mathbf{b}^3, \mathbf{W}^{3\times H}, \mathbf{b}^H, \mathbf{W}^{H\times(3+C)})$, H is the number of hidden units, and $\sigma(\cdot)$ is pointwise rectification (ReLU). (Here we take $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ to be column vectors instead of the row vectors we use elsewhere.) Note that there is no sinusoidal positional encoding of $\mathbf{x}$. But we also define and use a two-layer *position-attention* (PA) network,

$$\mathbf{y} = f_\theta(\mathbf{x}; \mathbf{z}) = \mathbf{b}^3 + \mathbf{z} \odot \sin(\mathbf{b}^C + \mathbf{W}^{C\times 3}\mathbf{x}), \quad (14)$$

where $\theta = (\mathbf{b}^3, \mathbf{b}^C, \mathbf{W}^{C\times 3})$ and $\odot$ is pointwise multiplication. The PA network is a simplified version of the modulated periodic activations in [45], and has many fewer parameters than the MLPs while being an efficient representation at low bit rates.

Once the latent vectors $\mathbf{Z} = [\mathbf{z_n}]$ and θ are transmitted as $\hat{\mathbf{Z}} = [\hat{\mathbf{z}}_\mathbf{n}]$ and $\hat{\theta}$, the attributes $\hat{\mathbf{y}}$ of any point $\mathbf{x} \in \mathbb{R}^3$ can be queried at the decoder, as illustrated in Fig. 3.

4. Experimental Results

4.1. Dataset and Platform

Our dataset comprises seven full human body voxelized point clouds derived from meshes created in [28, 46], shown in Figs. 1 and 5 and summarized in Tab. 1. A voxel is *occupied* if any part of the mesh intersects it, and the color $\mathbf{y}_i$ of that voxel is the average color of the mesh within the voxel. Integer voxel coordinates are used as the point positions $\mathbf{x}_i$. The voxels (and hence the point positions) have 10-bit resolution. This results in an octree of depth 10, or alternatively a binary tree of depth 30, for every point cloud. Point clouds are visualized in Meshlab [17].

We implement the LVAC framework in Python using Tensorflow. For most experiments, we train all variables

Point Cloud	# points
rock	837434
chair	791416
scooter	959388
juggling	798441
basketball	868224
basketball2	948870
jacket	805882

Table 1. Voxelized point clouds in our dataset.

(latents, step sizes, an entropy model per binary level, and a CBN at the target level) on a single point cloud, as the variables are specific to each point cloud. However, for the generalization experiments, we train only the latents, step sizes, and entropy models on the given point cloud, while using a CBN pre-trained on a different point cloud. The entire point cloud constitutes one batch. All configurations are trained in about 25K steps using the Adam optimizer and a learning rate of 0.01, with low bit rate configurations typically taking longer to converge. Each step takes 0.5-3.0 s on an NVIDIA P100 class GPU in eager mode with various debugging checks in place. The code will be made available on GitHub.

All results in the main body of this paper are reported for the *rock* point cloud, with the *basketball* point cloud used for generalization. Results for the other point clouds in the dataset are provided in the Appendix.

4.2. Baselines

Our principal baseline is RAHT, which is the core transform in the MPEG geometry-based point cloud codec (G-PCC), coupled with the adaptive Run-Length Golomb-Rice (RLGR) entropy coder [43]. Figure 6 shows the rate-distortion (RD) performance of *RAHT+RLGR* in RGB PSNR (dB) *vs* bit rate (bits per point – bpp). As PSNR is a measure of quality, higher is better. In *RAHT+RLGR*, RAHT transforms the point colors conditioned on the geometry. The resulting coefficients are uniformly scalar quantized with step sizes 2^n, for $n = 0, \ldots, 10$. The quantized coefficients are concatenated by level from the root to the leaves and entropy coded using RLGR, independently for each color component. The RD performances using RGB and YUV (BT.709) colorspaces are shown in Fig. 6 in blue with filled and unfilled markers, respectively. At low bit rates, YUV provides a significant gain in RGB PSNR, but this falls off at high bit rates.

As a secondary baseline, *level=30, model=cbe+linear (3x3)* in Fig. 6 shows the RD performance of our LVAC framework when 3-channel latents ($C = 3$) are quantized and entropy coded using the Continuous Batched Entropy (*cbe*) model with the Noisy Deep Factorized prior from Tensorflow Compression [1] followed by a simple 3×3 linear

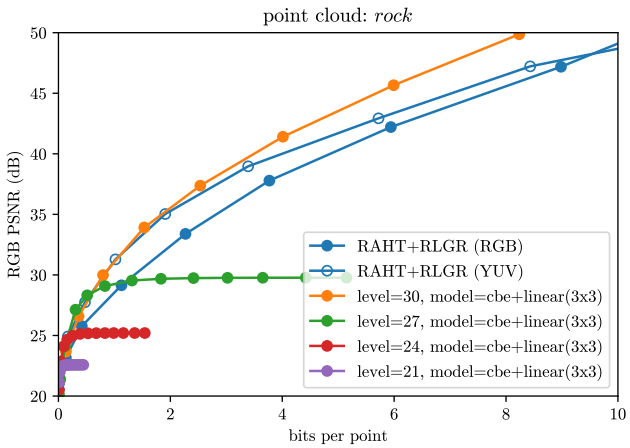

Figure 6. Baselines. *RAHT+RLGR (RGB)* and *(YUV)* are shown against 3×3 linear models at levels 30, 27, 24, and 21, which optimize the colorspace by minimizing $D + \lambda R$ using the *cbe* entropy model. Since *level=30, model=cbe+linear(3x3)* outperforms *RAHT+RLGR (YUV)* we discard the latter and use the others as baselines for more complex CBNs.

matrix as the CBN, at binary target level 30. The performance of this simple linear model agrees with that of *RAHT-RLGR (YUV)* at low rates, and outperforms it at high rates. Therefore, it is useful as a pseudo or secondary baseline and we show it in all subsequent plots along with our principal baseline *RAHT-RLGR (RGB)*.

Figure 6 also shows that at lower target levels (27, 24, 21), LVAC with the 3×3 matrix saturates at high rates, since the 3×3 matrix has no positional input, and thus represents the volumetric attribute function as a constant across each block. These constant functions serve as baselines for more complex CBNs at these levels, described next.

4.3. Coordinate Based Networks

We now compare configurations of the LVAC framework with four different CBNs: *linear(3x3)* (as a baseline), *mlp(35x256x3)*, *mlp(35x64x3)*, and *pa(3x32x3)*, at different target levels. The *mlp(35x256x3)* and *mlp(35x64x3)* CBNs are two-layer MLPs with 35 inputs (3 for position and 32 for a latent vector) and 3 outputs, having respectively 256 and 64 hidden nodes. The *pa(3x32x3)* CBN is a Position-Attention (PA) network also with 35 inputs (3 for position and 32 for a latent vector) and 3 outputs. Table 2 shows the number of parameters in these networks. All configurations use the Continuous Batched Entropy (*cbe*) model for quantization and entropy coding of the 32-channel latents.

Figure 7 (left, middle, right) shows (in green, red, purple) the RD performance of these CBNs at different target levels (27, 24, 21), along with the baselines (in blue, orange). We observe that first, at each target level $L = 27, 24, 21$, the CBNs with more parameters outperform the CBNs with fewer parameters. In particular, especially at higher bit

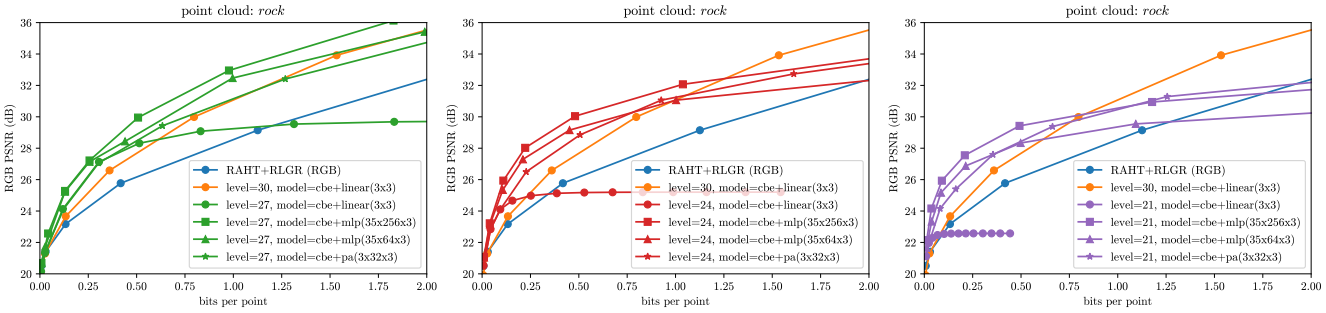

Figure 7. Coordinate Based Networks, by target level. Left, middle, right each show *mlp(35x256x3)*, *mlp(35x64x3)*, and *pa(3x32x3)* CBNs, along with baselines, at levels 27, 24, 21. More complex CBNs outperform less complex. Higher levels are better for higher bit rates.

CBN	# parameters
linear(3x3)	9
mlp(35x256x3)	9987
mlp(35x64x3)	2499
pa(3x32x3)	227

Table 2. CBNs and number of parameters.

rates, the MLP and PA networks at level L improve more than 5–10 dB over the linear network at level L, whose RD performance saturates as described earlier, for each L. Second, at each target level $L = 27, 24, 21$, there is a range of bit rates over which the MLP and PA networks improve by 2–3 dB over even the *level=30, model=cbe+linear(3x3)* baseline, which does not saturate. The range of bit rates in which this improvement is achieved is higher for level 27, and lower for level 21, reflecting that higher quality requires CBNs with smaller blocksizes.

Figure 15 in the Appendix shows these same data factored by CBN type instead of by level, to illustrate again that for each CBN type, each level is optimal for a different bit rate range.

The nature of a volumetric function $f_\theta(\mathbf{x}; \mathbf{z})$ represented by a CBN is illustrated in Fig. 8. To illustrate, we select the CBN *mlp(35x256x3)* trained on the *rock* point cloud at target level $L = 21$, and we plot cuts through the volumetric function $f_\theta(\cdot; \mathbf{z})$ represented by this CBN. Specifically, let n be a randomly selected node at the target level L, let $\hat{\mathbf{z}}_n$ be the quantized cumulative latent at that node, and let $\mathbf{x}_n = (x_n, y_n, z_n)$ be the position of a randomly selected point within the block at that node. Then we plot the first (red) component of the function $f_\theta(\mathbf{x}; \hat{\mathbf{z}}_n)$, where $\mathbf{x}$ varies from $(0, y_n, z_n)$ to (N_x, y_n, z_n), where N_x is the width of a block at level L. We do this for many randomly selected nodes n to get a sense of the distribution of volumetric functions represented at that level. (The distribution looks similar for green and blue components, and for cuts along y and z axes.) We observe that for many values of $\hat{\mathbf{z}}_n$, $f_\theta(\cdot; \hat{\mathbf{z}}_n)$ is a roughly constant function. Thus, $\hat{\mathbf{z}}_n$ must encode the colors of the palette used for these functions. However, we

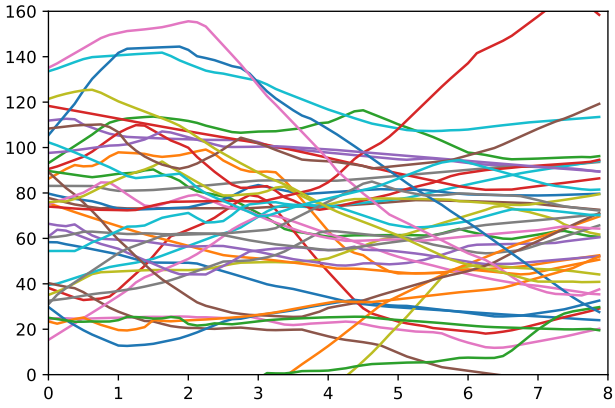

Figure 8. Cuts through the volumetric function $f_\theta(\mathbf{x}; \mathbf{z})$ represented by a CBN, for different values of $\mathbf{z}$, at target level 21.

also observe that for some values of $\hat{\mathbf{z}}_n$, $f_\theta(\cdot; \hat{\mathbf{z}}_n)$ is a ramp or some other nonlinear function across its domain. Finally, we observe almost no energy at frequencies higher than the Nyquist frequency (half the sampling rate), where the sampling occurs at units of voxels.

4.4. Generalization

In Sec. 4.3, the CBNs were optimized along with the latents, step sizes, and entropy models for a particular point cloud, *rock*. In this subsection, we take a small step towards exploring the degree to which the CBNs can be generalized across point clouds; that is, whether they can be trained to represent a universal family of volumetric functions. Towards that end, we pre-train the CBNs on the point cloud *basketball*, and fix their parameters while optimizing the other parameters (i.e., latents, step sizes, and entropy model parameters) on the point cloud *rock*. Figure 16 in the Appendix shows the results: Even with training on only a single other point cloud, the CBNs can indeed generalize across point clouds at low bit rates. At high bit rates, however, the CBNs trained on just one other point cloud do not perform well, most likely because they have been unable to

learn to represent the fine details needed for a different point cloud. Some of these results will be displayed again in the following subsection.

4.5. Side Information

When the latents, step sizes, entropy models, and CBN are all optimized for a specific point cloud, quantizing and entropy coding only the latent vectors $[\mathbf{z_n}]$ is insufficient for reconstructing the point cloud attributes. The step sizes $[\Delta_c]$, entropy model parameters $[\phi_{\ell,c}]$, and CBN parameters θ must also be quantized, entropy coded, and sent as *side information*. Sending side information incurs additional bit rate and distortion. This subsection explores the cost of this side information for both the entropy models and CBN. The side information for the step sizes is negligible, as there is only one step size for each of $C = 32$ channels.

First, we consider the side information for the entropy models. For each point cloud, there is one entropy model per binary level[1], per channel. For 26 such binary levels, 32 channels, and the Continuous Batched Entropy (*cbe*) model with Noisy Deep Factorized prior, this works out to 23296 floating point parameters. If we allocate 32 bits per floating point parameter, the bit rate would increase by 0.89 bits per point for the *rock* point cloud, which has 837434 points. Thus the RD performance would move from the solid green line to the dashed green line in Fig. 9, for *level=27, model=cbe+mlp(35x256x3)*. However, fortunately, this costly side information can be avoided, by using *cbe* during training but using RLGR during inference. Since RLGR is backward adaptive, it can adapt to Laplacian-like distributions without sending any side information. Of course its coding efficiency may suffer, but our experiments show that this degradation — to the dotted green line with open markers in Fig. 9 — is almost negligible. Henceforth we report RD performance using only RLGR during inference.

Next, we consider the side information for the CBNs. For each point cloud, there is one CBN, at the target level. For the number of parameters in our CBNs (Tab. 2), if we allocate 32 bits per floating point parameter, the side information to transmit the CBN would be 0.38, 0.095, and 0.009 bits per point for *mlp(35x256x3)*, *mlp(35x64x3)*, and *pa(3x32x3)*, respectively. Figure 10 shows the resulting RD performance for *mlp(35x256x3)*, while Fig. 17 in the Appendix shows the resulting RD performance for *mlp(35x64x3)* and *pa(3x32x3)*, at target levels 27, 24, and 21. It can be seen that 32 bits per parameter of side information to encode *mlp(35x256x3)* has a severe effect on RD performance at low bit rates, but less severe at high bit rates. For *mlp(35x64x3)*, the effect is more modest, and for *pa(3x32x3)*, it is negligible. Fortunately for the MLPs, at low bit rates, where the side information penalizes them most, and where generalization works best, they may alternatively be generalized to avoid having to transmit any side information. The RD performance of the generalized CBNs is also included in the figures.

[1] except binary levels in which each node has only one occupied child

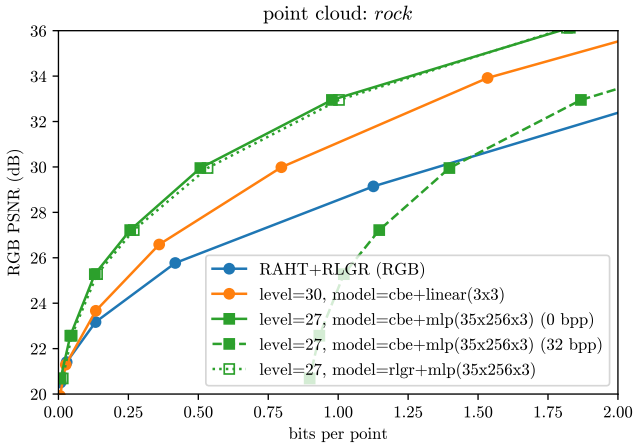

Figure 9. Side information for entropy model. Sending 32 bits per parameter for the *cbe* entropy model would reduce RD performance from solid to dashed green lines. But the backward-adaptive *RLGR* entropy coder (dotted, unfilled) obviates the need to send side information with almost no loss in performance.

Also fortunately, it is likely that 32 bits per floating point parameter is an order of magnitude more than necessary. Prior work has shown that simple model compression can be performed at 8 bits per floating point parameter [10, 77, 86] or even more aggressively at 1–4 bits per floating point parameter [30, 34, 35, 54, 75, 85, 87] with very low loss in performance, even with coordinate based networks such as NeRF [13, 33]. However, since model compression is outside the scope of our work, we simply parameterize our results by the number of bits per floating point parameter. In Sec. 4.7, we will return to the effect of side information under various values for the number of bits per parameter. First, however, we turn to a key ablation study.

4.6. Normalization

One of our main contributions is to show that naïve uniform scalar quantization and entropy coding of the latents leads to poor results, and that properly normalizing the coefficients before quantization achieves over a 30% reduction in bit rate. In this ablation study, we remove our normalization by setting the scale matrix $\mathbf{S}$ in (10)-(11) and Fig. 4 to the identity matrix, thus removing any dependency of the attribute compression on the geometry. This corresponds to a naïve approach to compression, for example by assuming a fixed number of bits per latent as in [78]. Table 3 shows that compared to this naïve approach, our normalization achieves over 30% reduction in bit rate on average over all point clouds in the dataset (computed using [14, 57]). This quantifies the reduction in bit rate due to condition-

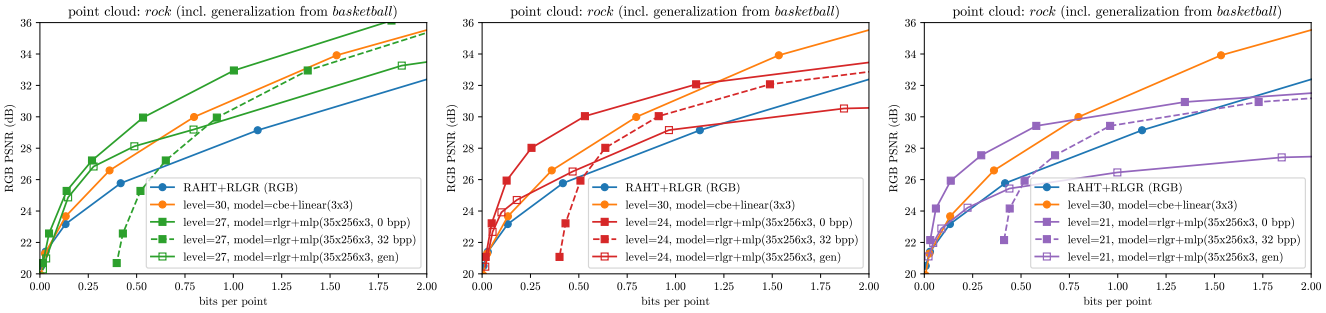

Figure 10. Effect of side information for coordinate based network *mlp(35x256x3)* at levels 27 (left), 24 (middle), and 21 (right). Sending 32 bits per parameter for the CBN would degrade RD performance from solid to dashed lines. The degradation would be inversely proportional to compression ratio if model compression is used. Alternatively, generalization (pre-training the CBN on one or more other point clouds), which works well at low bit rates, would obviate the need to transmit any side information. Generalization is indicated by "gen" in the legend.

CBN	level 30	27	24	21
linear(3x3)	-31.6%	-18.6%	-28.3%	-37.7%
mlp(35x256x3)	N/A	-29.9%	-34.3%	-27.4%
mlp(35x64x3)	N/A	-23.8%	-32.1%	-31.1%
pa(3x32x3)	N/A	-41.1%	-40.3%	-38.7%

Table 3. BD-Rate reductions due to normalization, averaged over point clouds. Normalization is crucial for good performance. Without normalization, there is no dependence on geometry.

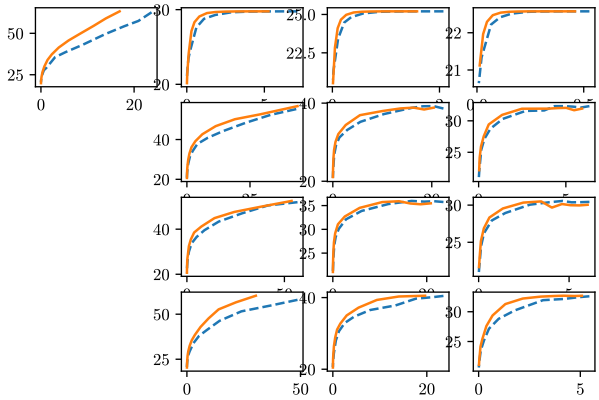

Figure 11. RD performance (RGB PSNR vs. bit rate) improvement due to normalization, corresponding to entries in Tab. 3.

ing the attribute compression on the geometry. Figure 11 shows the RD performances of the naïve (dotted blue line) and normalized (solid orange line) approaches, corresponding to the entries in Tab. 3. We observe that, except in the linear case, normalization tends to help more when the CBN is at a higher target level. This makes sense, as there are fewer latents to normalize when the CBN is at a lower level. However, we also observed in Fig. 7 that, at higher bit rates, the CBNs at lower levels are unable to outperform even the normalized linear case at level 30. Thus, while the normalized linear case is able to condition on detailed geometry all the way to individual voxels, the CBNs despite their much higher complexity perform worse than the linear models at high bit rates because they do not condition on the specific geometry within their blocks. This points to a continued need to research how to fully condition on geometry with learned attribute compression.

4.7. Convex Hull

For different bit rate ranges, and for different assumptions on the cost of side information, different configurations of the LVAC framework may be optimal. Figure 12 shows the convex hull, or Pareto fontier, of all configurations under the assumptions of 0 (left), 8 (middle), and 32 (right) bits per floating point parameter. All configurations that we have examined in this paper appear in each plot. However, only those that participate in the convex hull are listed in the legend. We observe: first, when the side information costs nothing (0 bits per parameter), the convex hull contains exclusively the largest CBN (*mlp(35x256x3)*), at higher target levels for higher bit rates. Second, as the cost of the side information increases, the smaller CBNs (*mlp(35x64x3)* and *pa(3x32x3)*), and those that are generalized from another point cloud (*mlp(35x256x3, gen)* and *mlp(35x64x3 gen)*), begin to participate in the convex hull, especially at lower bit rates. Eventually, at 32 bits per parameters, the largest CBN is excluded entirely. Third, the generalizations participate in the convex hull only at the lowest bit rates, despite not incurring any penalty due to side information. This could be because they are trained only on a single other point cloud in these experiments. Training the CBNs on more representative data would probably improve their generalization performance across a wider range of bit rates but is left for future work.

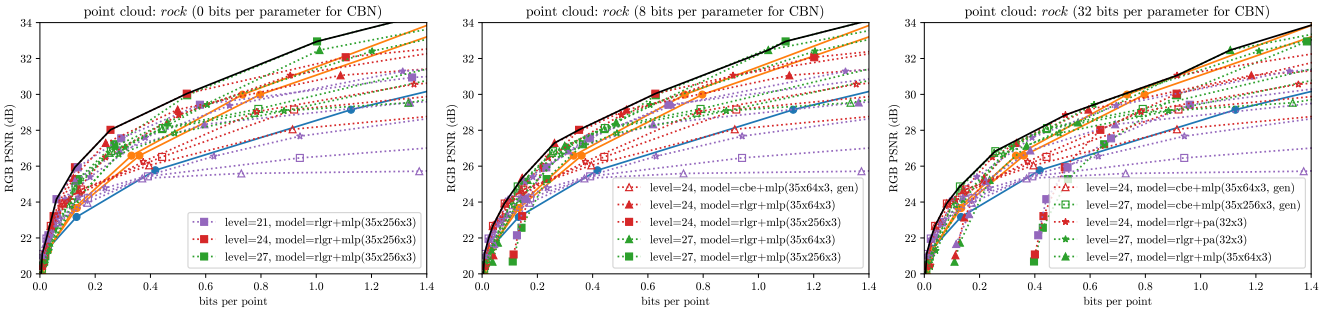

Figure 12. Convex hull (solid black line) of RD performances of all CBN configurations across all levels, including side information using 0 (left), 8 (middle), and 32 (right) bits per CBN parameter. Configurations that participate in the convex hull are listed in the legend. At 0 bits per parameter, the more complex CBNs dominate. At higher bits per parameter, the generalized and less complex CBNs begin to participate, especially at lower bit rates.

4.8. Subjective Quality

Figure 13 shows the subjective compression quality around 0.25 bpp, under the assumption of 0 bits per floating point parameter. Additional bit rates are shown in the Appendix.

4.9. Baselines, Revisited

We now return to the matter of baselines. Figure 14 shows our previous baseline, *RAHT+RLGR*, for both RGB and YUV colorspaces (blue lines). Although RAHT is the transform used in MPEG G-PCC, the reference software TMC13 v6.0 (July 2019) offers improved RD performance (green lines) compared to *RAHT+RLGR*, due principally to better entropy coding. In particular, TMC13 uses context-adaptive binary arithmetic coding with various coding modes, while *RAHT+RLGR* uses RLGR. We use *RAHT+RLGR* as our baseline because our experiments use RLGR as our entropy coder; the specific entropy coder used in TMC13 is difficult to extract from the standard. The latest version, TMC13 v14.0 (October 2021), offers even better RD performance, by introducing for example joint coding modes for color channels that are all zero (orange lines). It also introduces predictive RAHT, in which the RAHT coefficients at each level are predicted from the decoded RAHT coefficients at the previous level [3, 40, 59]. The prediction residuals, instead of the RAHT coefficients themselves, are quantized and entropy coded. Predictive RAHT alone improves RD performance by 2–3 dB (red lines). Nevertheless, LVAC with RLGR and no RAHT prediction (solid black line, from Fig. 12 (left)) is within 1 dB of TMC13 v14.0 with joint entropy coding and RAHT prediction, and it outperforms all other versions. Moreover, we believe that the RD performance of LVAC can be further improved significantly. In particular, the principal advances of TMC13 over *RAHT+RLGR* — joint entropy coding and predictive RAHT — are equally applicable to the LVAC framework. For example, joint entropy coding could be done with a hyperprior [9], and predictive RAHT could be applied to the latent vectors. These explorations are left for future work.

5. Discussion and Conclusion

This work is the first to compress volumetric functions $\mathbf{y} = f_\theta(\mathbf{x})$ modeled by local coordinate-based networks. Though we focused on RGB attributes $\mathbf{y}$, the extension to other attributes (such as signed distance, reflectance, normals, transparency, density, spherical harmonics, etc.) is straightforward. Also, though we focused on $\mathbf{x} \in \mathbb{R}^3$, extensions to hyper-volumetric functions (such as $\mathbf{y} = f_\theta(\mathbf{x}, \mathbf{d})$ where $\mathbf{d}$ is a view direction) is also straightforward. Thus LVAC should be applicable to plenoptic point clouds [38, 65, 66, 92, 93] as well as radiance fields (e.g., [44, 52, 78, 89, 94]). We believe that the main difference between plenoptic point clouds and radiance fields is the distortion measure $d(f, f_\theta)$. For point clouds, $d(f, f_\theta)$ is measured in the domain of f, such as the MSE between colors on points in 3D. For radiance fields, $d(f, f_\theta)$ is measured in the domain of *projections* or renderings of f onto 2D images, such as the MSE between colors of pixels that are renderings of f and f_θ onto 2D images. In [22], the former distortion measures are called *matching distortions*, while the latter are called *projection distortions*. The change in distortion measure from matching to projection distortions may be all that is required to apply LVAC properly to radiance field compression.

This work is also among the first to compress point cloud attributes using neural networks, outperforming RAHT, used in MPEG G-PCC, by 2–4 dB. Although MPEG G-PCC uses additional coding tools to further improve compression, such as context adaptive arithmetic coding, joint entropy coding of color, and predictive RAHT, these tools are also at our disposal, and may be the subject of further work. It should be recalled that learned image compression evolved over dozens of papers and a half dozen years, being competitive at first with only JPEG on thumbnails, and

(a) Original (b) *RAHT-RLGR (RGB)* (c) *RAHT-RLGR (YUV)* (d) *LVAC mlp(35x256x3)*

Figure 13. Subjective quality around 0.25 bpp. (a) Original. (b) 0.258 bpp, 24.6 dB. (c) 0.255 bpp, 25.9 dB. (d) 0.255 bpp, 28.0 dB.

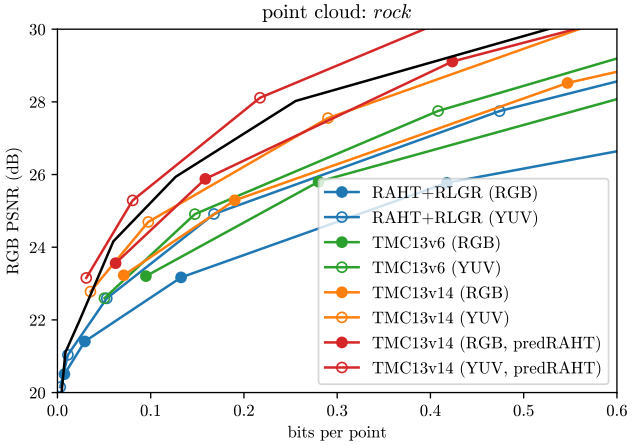

Figure 14. Baselines, revisited. In both RGB and YUV color spaces, MPEG G-PCC reference software TMC13 v6.0 improves over *RAHT+RLGR*, primarily due to context-adaptive (i.e., dependent) entropy coding. TMC13 v14.0 improves still further, primarily due to predictive RAHT. Better entropy coding (e.g., hyperprior) and predictive RAHT can also be applied to LVAC.

then successively with JPEG-2000, WebP, and BGP. Only recently has learned image compression been able to outperform VVC in PSNR [29]. Learned volumetric attribute compression (LVAC), like learned image compression, is a work in progress. Now is a particularly good time to publicise progress in this area ahead of the JPEG Pleno Call for Proposals on learned point cloud compression, with submissions scheduled in 2022 [2].

We believe that LVAC still has two big weaknesses that need to be addressed. The first is lack of conditioning on *detailed* geometry. Currently LVAC conditions on geometry by using RAHT's mechanism to normalize the latents up to the target level. Unfortunately this conditioning does not make its way to the finest levels of geometry within the CBN. We believe this is the reason that at high bit rates, CBNs at low target levels fall short of a simple linear network at the voxel level, in RD performance. In short, the CBNs are leaving big RD improvements on the table.

Another big weakness of LVAC, so far, is its auto-decoder approach, in which the encoder must perform optimization through back-propagation, in order to produce a bit stream for a point cloud. This is extremely slow relative to a feed-forward encoder. We recognize the need to develop a feed-forward encoder, and with it, the need for conditioning on the geometry to the finest levels of detail.

These provide plenty of room for future work.

References

[1] Tensorflow Compression. https://github.com/tensorflow/compression. 6, 7

[2] 3DG. Final call for evidence on jpeg pleno point cloud coding. Approved WG 1 document N88014, ISO/IEC MPEG JTC1/SC29/WG1, online, July 2020. 1, 12

[3] 3DG. G-PCC Codec Description v12. Approved WG 11 document N18891, ISO/IEC MPEG JTC1/SC29/WG11, Geneva, CH, October 2020. 3, 11

[4] E. Agustsson and L. Theis. Universally quantized neural compression. In *Advances in Neural Information Processing Systems 34*, 2020. 6

[5] J. Balle, P. A. Chou, D. Minnen, S. Singh, N. Johnston, E. Agustsson, S. J. Hwang, and G. Toderici. Nonlinear transform coding. *IEEE Journal of Selected Topics in Signal Processing*, pages 1–1, 2020. 1, 3

[6] J. Ballé. Efficient nonlinear transforms for lossy image compression. In *2018 Picture Coding Symp. (PCS)*, 2018. 1, 3

[7] J. Ballé, V. Laparra, and E. P. Simoncelli. End-to-end optimization of nonlinear transform codes for perceptual quality. In *2016 Picture Coding Symp. (PCS)*, 2016. 1, 3

[8] J. Ballé, V. Laparra, and E. P. Simoncelli. End-to-end optimized image compression. In *5th Int. Conf. on Learning Representations (ICLR)*, 2017. 1, 3, 6

[9] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston. Variational image compression with a scale hyperprior. In *6th Int. Conf. on Learning Representations (ICLR)*, 2018. 1, 3, 11

[10] R. Banner, I. Hubara, E. Hoffer, and D. Soudry. Scalable methods for 8-bit training of neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, page 5151–5159, 2018. 9

[11] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields, 2021. 2

[12] N. Benbarka, T. Höfer, H. ul-moqeet Riaz, and A. Zell. Seeing implicit neural representations as fourier series, 2021. 2

[13] T. Bird, J. Ballé, S. Singh, and P. A. Chou. 3d scene compression through entropy penalized neural representation functions. In *Picture Coding Symposium (PCS)*, June 2021. 2, 4, 9

[14] G. Bjøntegaard. Calculation of average psnr differences between rd-curves. Technical Report VCEG-M33, ITU-T SG16/Q6, Austin, Texas, 2001. 9

[15] Y. Chen, S. Liu, and X. Wang. Learning continuous image representation with local implicit image function, 2021. 3

[16] P. A. Chou, M. Koroteev, and M. Krivokuća. A volumetric approach to point cloud compression—part i: Attribute compression. *IEEE Transactions on Image Processing*, 29:2203–2216, 2020. 1, 3

[17] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In V. Scarano, R. De Chiara, and U. Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. 6

[18] R. A. Cohen, D. Tian, and A. Vetro. Attribute compression for sparse point clouds using graph transforms. In *IEEE Int'l Conf. Image Processing (ICIP)*, Sept 2016. 1, 3

[19] M. Czerkawski, J. Cardona, R. Atkinson, C. Michie, I. Andonovic, C. Clemente, and C. Tachtatzis. Neural knitworks: Patched neural implicit representation networks, 2021. 2

[20] R. L. de Queiroz and P. A. Chou. Compression of 3d point clouds using a region-adaptive hierarchical transform. *IEEE Transactions on Image Processing*, 25(8):3947–3956, 2016. 1, 3

[21] R. L. de Queiroz and P. A. Chou. Motion-compensated compression of dynamic voxelized point clouds. *IEEE Transactions on Image Processing*, 26(8):3886–3895, Aug. 2017. 1, 3

[22] R. L. de Queiroz and P. A. Chou. Motion-compensated compression of dynamic voxelized point clouds. *IEEE Transactions on Image Processing*, 26(8):3886–3895, 2017. 11

[23] T. DeVries, M. Angel Bautista, N. Srivastava, G. W. Taylor, and J. M. Susskind. Unconstrained scene generation with locally conditioned radiance fields. 2021. 3

[24] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai. An overview of ongoing point cloud compression standardization activities: video-based (v-pcc) and geometry-based (g-pcc). *APSIPA Transactions on Signal and Information Processing*, 9:e13, 2020. 1, 3

[25] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira. Deep learning-based point cloud coding: A behavior and performance study. In *2019 8th European Workshop on Visual Information Processing (EUVIP)*, pages 34–39, 2019. 1, 3

[26] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira. Point cloud coding: Adopting a deep learning-based approach. In *2019 Picture Coding Symposium (PCS)*, pages 1–5, 2019. 1, 3

[27] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira. Deep learning-based point cloud geometry coding: RD control through implicit and explicit quantization. In *2020 IEEE Int. Conf. on Multimedia & Expo Wksps. (ICMEW)*, 2020. 1, 3

[28] K. Guo, P. Lincoln, P. Davidson, J. Busch, X. Yu, M. Whalen, G. Harvey, S. Orts-Escolano, R. Pandey, J. Dourgarian, D. Tang, A. Tkach, A. Kowdle, E. Cooper, M. Dou, S. Fanello, G. Fyffe, C. Rhemann, J. Taylor, P. Debevec, and S. Izadi. The relightables: Volumetric performance capture of humans with realistic relighting. *ACM Trans. Graph.*, 38(6), Nov. 2019. 6

[29] Z. Guo, Z. Zhang, R. Feng, and Z. Chen. Causal contextual prediction for learned image compression. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2021. 3, 12

[30] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 9

[31] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec. Baking neural radiance fields for real-time view synthesis, 2021. 2

[32] Y. Hu, W. Yang, Z. Ma, and J. Liu. Learning end-to-end lossy image compression: A benchmark, 2021. 1, 3

[33] Berivan Isik. Neural 3d scene compression via model compression. *arXiv preprint arXiv:2105.03120*, 2021. 2, 4, 9

[34] Berivan Isik, Kristy Choi, Xin Zheng, Tsachy Weissman, Stefano Ermon, H-S Philip Wong, and Armin Alaghi. Neural network compression for noisy storage devices. *arXiv preprint arXiv:2102.07725*, 2021. 9

[35] Berivan Isik, Albert No, and Tsachy Weissman. Rate-distortion theoretic model compression: Successive refinement for pruning. *arXiv preprint arXiv:2102.08329*, 2021. 9

[36] E. S. Jang, M. Preda, K. Mammou, A. M. Tourapis, J. Kim, D. B. Graziosi, S. Rhyu, and M. Budagavi. Video-based point-cloud-compression standard in mpeg: From evidence collection to committee draft [standards in a nutshell]. *IEEE Signal Processing Magazine*, 36(3):118–123, 2019. 1, 3

[37] J. Knodt, S.-H. Baek, and F. Heide. Neural ray-tracing: Learning surfaces and reflectance for relighting and view synthesis, 2021. 2

[38] M. Krivokuća, P. A. Chou, and P. Savill. 8i voxelized surface light field (8iVSLF) dataset. input document m42914, ISO/IEC JTC1/SC29 WG11 (MPEG), Ljubljana, Slovenia, Jul. 2018. 11

[39] M. Krivokuća, P. A. Chou, and M. Koroteev. A volumetric approach to point cloud compression–part ii: Geometry compression. *IEEE Transactions on Image Processing*, 29:2217–2229, 2020. 1, 3

[40] S. Lasserre and D. Flynn. On an improvement of raht to exploit attribute correlation. input document m47378, ISO/IEC MPEG JTC1/SC29/WG11, Geneva, CH, Mar. 2019. 3, 11

[41] D. Lazzarotto, E. Alexiou, and T. Ebrahimi. On block prediction for learning-based point cloud compression. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3378–3382, 2021. 3

[42] X. Luo, H. Talebi, F. Yang, M. Elad, and P. Milanfar. The Rate-Distortion-Accuracy Tradeoff: JPEG Case Study, 2020. 6

[43] H.S. Malvar. Adaptive run-length/golomb-rice encoding of quantized generalized gaussian sources with unknown statistics. In *Data Compression Conference (DCC'06)*, pages 23–32, 2006. 6, 7

[44] J. N. P. Martel, D. B. Lindell, C. Z. Lin, E. R. Chan, M. Monteiro, and G. Wetzstein. Acorn: Adaptive coordinate networks for neural scene representation, 2021. 3, 11

[45] I. Mehta, M. Gharbi, C. Barnes, E. Shechtman, E. Ramamoorthi, and M. Chandraker. Modulated periodic activations for generalizable local functional representations, 2021. 2, 3, 6

[46] A. Meka, R. Pandey, C. Haene, S. Orts-Escolano, P. Barnum, P. Davidson, D. Erickson, Y. Zhang, J. Taylor, S. Bouaziz, C. Legendre, W.-C. Ma, R. Overbeck, T. Beeler, P. Debevec, S. Izadi, C. Theobalt, C. Rhemann, and S. Fanello. Deep relightable textures - volumetric performance capture with neural rendering. volume 39, December 2020. 6

[47] R. Mekuria, K. Blom, and P. Cesar. Design, implementation, and evaluation of a point cloud codec for tele-immersive video. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(4):828–842, 2017. 1

[48] F. Mentzer, G. D. Toderici, M. Tschannen, and E. Agustsson. High-fidelity generative image compression. *Advances in Neural Information Processing Systems*, 33, 2020. 1, 3

[49] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[50] S. Milani. A syndrome-based autoencoder for point cloud geometry compression. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 2686–2690, 2020. 3

[51] S. Milani. Adae: Adversarial distributed source autoencoder for point cloud compression. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3078–3082, 2021. 3

[52] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2, 11

[53] D. Minnen, J. Ballé, and G. Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems 31*, 2018. 1, 3

[54] D. Oktay, J. Ballé, S. Singh, and A. Shrivastava. Scalable model compression by entropy penalized reparameterization. In *International Conference on Learning Representations*, 2019. 9

[55] J. Park, P. A. Chou, and J. Hwang. Rate-utility optimized streaming of volumetric media for augmented reality. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):149–162, 2019. 1

[56] J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. pages 165–174, 06 2019. 2

[57] S. Pateux and J. Jung. An excel add-in for computing bjontegaard metric and its evolution. *ITU-T SG16 Q*, 6(7):7, 2007. 9

[58] E. Pavez, P. A. Chou, R. L. de Queiroz, and A. Ortega. Dynamic polygon clouds: representation and compression for VR/AR. *APSIPA Transactions on Signal and Information Processing*, 7:e15, 2018. 1, 3

[59] E. Pavez, A. L. Souto, R. L. De Queiroz, and A. Ortega. Multi-resolution intra-predictive coding of 3d point cloud attributes. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3393–3397, 2021. 3, 11

[60] R. Pierdicca, M. Paolanti, F. Matrone, M. Martini, C. Morbidoni, E. S. Malinverni, E. Frontoni, and A. M. Lingua. Point cloud semantic segmentation using a deep learning framework for cultural heritage. *Remote Sensing*, 12(6):1005, Mar 2020. 1

[61] M. Quach, G. Valenzise, and F. Dufaux. Learning convolutional transforms for lossy point cloud geometry compression. In *2019 IEEE Int. Conf. on Image Processing (ICIP)*, 2019. 1, 3

[62] M. Quach, G. Valenzise, and F. Dufaux. Folding-based compression of point cloud attributes. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3309–3313, 2020. 3

[63] M. Quach, G. Valenzise, and F. Dufaux. Improved deep point cloud geometry compression. *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6, 2020. 1, 3

[64] C. Reiser, S. Peng, Y. Liao, and A. Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps, 2021. 3

[65] G. Sandri, R. de Queiroz, and P. A. Chou. Compression of plenoptic point clouds using the region-adaptive hierarchical transform. In *25th IEEE Int. Conf. on Image Processing (ICIP)*, pages 1153–1157, Oct, 2018. 11

[66] G. Sandri, R. L. de Queiroz, and P. A. Chou. Compression of plenoptic point clouds. *IEEE Trans. on Image Processing*, 28(3):1419–1427, 2019. 11

[67] G. Sandri, V. F. Figueiredo, P. A. Chou, and R. de Queiroz. Point cloud compression incorporating region of interest coding. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4370–4374, 2019. 3

[68] G. P. Sandri, P. A. Chou, M. Krivokuća, and R. L. de Queiroz. Integer alternative for the region-adaptive hierarchical transform. *IEEE Signal Processing Letters*, 26(9):1369–1372, 2019. 3

[69] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. Tourapis, and V. Zakharchenko. Emerging MPEG standards for point cloud compression. *IEEE*

J. Emerging Topics in Circuits and Systems, 9(1):133–148, Mar. 2019. 1, 3

[70] X. Sheng, L. Li, D. Liu, Z. Xiong, Z. Li, and F. Wu. Deeppcac: An end-to-end deep lossy compression framework for point cloud attributes. *IEEE Transactions on Multimedia*, pages 1–1, 2021. 3

[71] V. Sitzmann, E. R. Chan, R. Tucker, N. Snavely, and G. Wetzstein. Metasdf: Meta-learning signed distance functions, 2020. 2

[72] V. Sitzmann, J N.P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020. 2

[73] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis, 2020. 2

[74] K. Stelzner, K. Kersting, and A. R. Kosiorek. Decomposing 3d scenes into objects via unsupervised volume segmentation, 2021. 3

[75] P. Stock, A. Joulin, R. Gribonval, B. Graham, and H. Jégou. And the bit goes down: Revisiting the quantization of neural networks. In *International Conference on Learning Representations*, 2019. 9

[76] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2443–2451, 2020. 1

[77] X. Sun, J. Choi, C.-Y. Chen, N. Wang, S. Venkataramani, V. Viji Srinivasan, X. Cui, W. Zhang, and K. Gopalakrishnan. Hybrid 8-bit floating point (hfp8) training and inference for deep neural networks. *Advances in Neural Information Processing Systems*, 32:4900–4909, 2019. 9

[78] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and Fidler S. Neural geometric level of detail: Real-time rendering with implicit 3d shapes, 2021. 3, 4, 9, 11

[79] M. Tancik, B. Mildenhall, T. Wang, D. Schmidt, P. Hedman, J. T. Barron, and R. Ng. Learned initializations for optimizing coordinate-based neural representations, 2021. 2

[80] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020. 2

[81] D. Tang, S. Singh, P. A. Chou, C. Häne, M. Dou, S. Fanello, J. Taylor, P. Davidson, O. G. Guleryuz, Y. Zhang, S. Izadi, A. Tagliasacchi, S. Bouaziz, and C. Keskin. Deep implicit volume compression. In *2020 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 3

[82] D. Thanou, P. A. Chou, and P. Frossard. Graph-based compression of dynamic 3d point cloud sequences. *IEEE Trans. Image Processing*, 25(4), April 2016. 1, 3

[83] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar. Variable rate image compression with recurrent neural networks. In *4th Int. Conf. on Learning Representations (ICLR)*, 2016. 1, 3

[84] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell. Full resolution image compression with recurrent neural networks. In *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 3

[85] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8612–8620, 2019. 9

[86] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7686–7695, 2018. 9

[87] Y. Xu, Y. Wang, A. Zhou, W. Lin, and H. Xiong. Deep neural network compression with single and multiple level quantization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 9

[88] W. Yan, Y. Shao, S. Liu, T. H. Li, Z. Li, and G. Li. Deep autoencoder-based lossy geometry compression for point clouds. *CoRR*, abs/1905.03691, 2019. 1, 3

[89] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa. Plenoctrees for real-time rendering of neural radiance fields, 2021. 2, 3, 11

[90] H.-X. Yu, L. J. Guibas, and J. Wu. Unsupervised discovery of object radiance fields, 2021. 3

[91] C. Zhang, D. Florêncio, and C. Loop. Point cloud attribute compression with graph transform. In *2014 IEEE Int'l Conf. Image Processing (ICIP)*, Oct 2014. 1, 3

[92] X. Zhang, P. A. Chou, M. Sun, M. Tang, S. Wang, S. Ma, and W. Gao. A framework for surface light field compression. In *IEEE Int. Conf. on Image Processing (ICIP)*, pages 2595–2599, 2018. 11

[93] X. Zhang, P. A. Chou, M. Sun, M. Tang, S. Wang, S. Ma, and W. Gao. Surface light field compression using a point cloud codec. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):163–176, Mar, 2019. 11

[94] X. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, and J. T. Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination, 2021. 2, 11

[95] J. Zheng, S. Ramasinghe, and S. Lucey. Rethinking positional encoding, 2021. 2

Appendix

A. Coordinate Based Networks

Figure 15 shows the RD performance of different networks: (left) *mlp(35x256x3)*, (middle) *mlp(35x64x3)*, and (right) *pa(3x32x3)*, along with baselines. At higher bit rates, higher target levels perform better.

B. Generalization

Figure 16 shows the RD performance of CBNs generalized from, or pre-trained on, another point cloud. Even with training on only a single other point cloud, the CBNs can indeed generalize across point clouds at low bit rates. At high bit rates, however, the CBNs trained on just one other point cloud do not perform well, most likely because they have been unable to learn to represent the fine details needed for a different point cloud.

C. Side Information

Figure 17 shows results for *mlp(35x64x3)* and *pa(3x32x3)* (top and bottom, respectively) corresponding to the results in Fig. 10 for *mlp(35x256x3)*.

D. Subjective Quality

Figure 18 shows subjective results for 0.125, 0.5, and 1.0 bpp corresponding to the subjective results in Fig. 13 for 0.25 bpp.

E. Additional Point Clouds

Figure 19, Fig. 20, Fig. 21, Fig. 22 & Fig. 23, Fig. 24, Fig. 25, Fig. 26 & Fig. 27, Tab. 4, Fig. 28, and Fig. 29 show results for point clouds *chair*, *scooter*, *juggling*, *basketball*, *basketball2*, and *jacket* corresponding respectively to Fig. 6, Fig. 7, Fig. 15, Fig. 16, Fig. 9, Fig. 10, Fig. 17, Tab. 3, Fig. 11, and Fig. 12 for *rock*.

rock	level			
CBN	30	27	24	21
linear(3x3)	-36.5%	-26.1%	-29.6%	-35.2%
mlp(35x256x3)	N/A	-35.8%	-39.4%	-28.7%
mlp(35x64x3)	N/A	-32.7%	-34.4%	-27.8%
pa(3x32x3)	N/A	-41.6%	-39.8%	-37.5%
chair	level			
CBN	30	27	24	21
linear(3x3)	-36.8%	-26.5%	-34.1%	-47.4%
mlp(35x256x3)	N/A	-34.8%	-39.5%	-28.0%
mlp(35x64x3)	N/A	-34.3%	-47.5%	-38.4%
pa(3x32x3)	N/A	-45.2%	-44.7%	-45.6%
scooter	level			
CBN	30	27	24	21
linear(3x3)	-32.6%	-19.5%	-34.6%	-41.8%
mlp(35x256x3)	N/A	-32.1%	-38.1%	-20.6%
mlp(35x64x3)	N/A	-27.1%	-30.2%	-39.1%
pa(3x32x3)	N/A	-41.9%	-42.2%	-40.6%
juggling	level			
CBN	30	27	24	21
linear(3x3)	-26.9%	-8.5%	-23.4%	-32.4%
mlp(35x256x3)	N/A	-22.7%	-35.0%	-27.7%
mlp(35x64x3)	N/A	-16.3%	-35.2%	-30.5%
pa(3x32x3)	N/A	-47.7%	-47.6%	-39.1%
basketball	level			
CBN	30	27	24	21
linear(3x3)	-30.0%	-19.7%	-27.7%	-28.9%
mlp(35x256x3)	N/A	-20.9%	-29.2%	-42.3%
mlp(35x64x3)	N/A	-5.8%	-28.4%	-32.8%
pa(3x32x3)	N/A	-31.1%	-27.7%	-24.5%
basketball2	level			
CBN	30	27	24	21
linear(3x3)	-29.2%	-14.6%	-20.8%	-36.5%
mlp(35x256x3)	N/A	-34.5%	-24.0%	-15.9%
mlp(35x64x3)	N/A	-28.7%	-27.8%	-24.1%
pa(3x32x3)	N/A	-41.2%	-42.5%	-41.7%
jacket	level			
CBN	30	27	24	21
linear(3x3)	-29.3%	-15.2%	-27.7%	-41.9%
mlp(35x256x3)	N/A	-28.8%	-35.1%	-28.4%
mlp(35x64x3)	N/A	-22.0%	-21.3%	-25.1%
pa(3x32x3)	N/A	-39.1%	-38.0%	-41.7%

Table 4. BD-Rate reductions due to normalization, for each point cloud. See Tab. 3 for aggregate results.

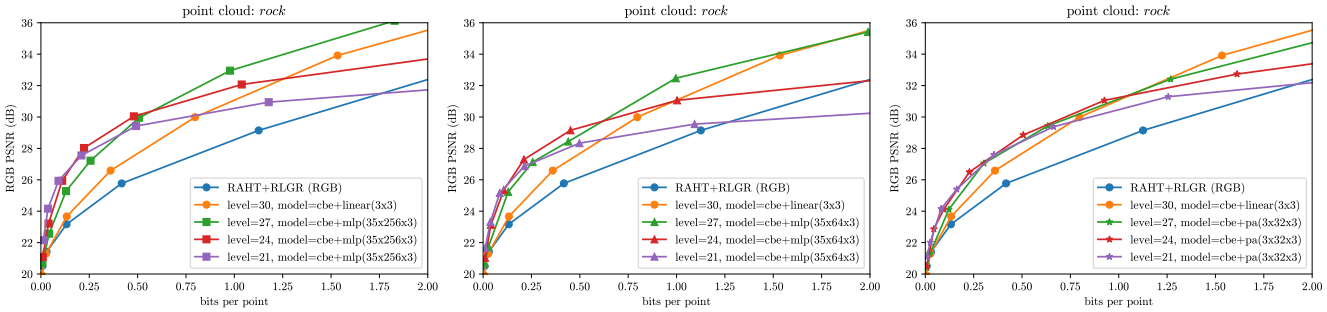

Figure 15. Coordinate Based Networks, by network.

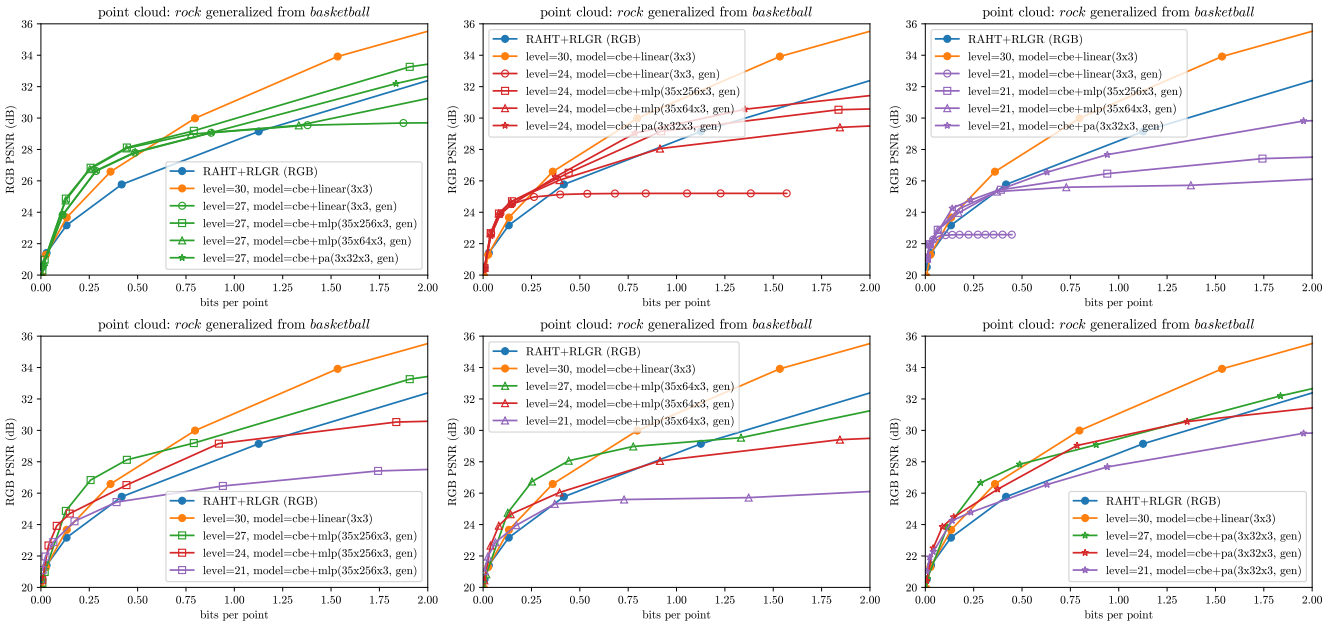

Figure 16. Coordinate Based Networks with generalization, by level (top row) and by network (bottom row). CBNs that are generalized (i.e., pre-trained on another point cloud) are able to outperform the baselines at low bit rates.

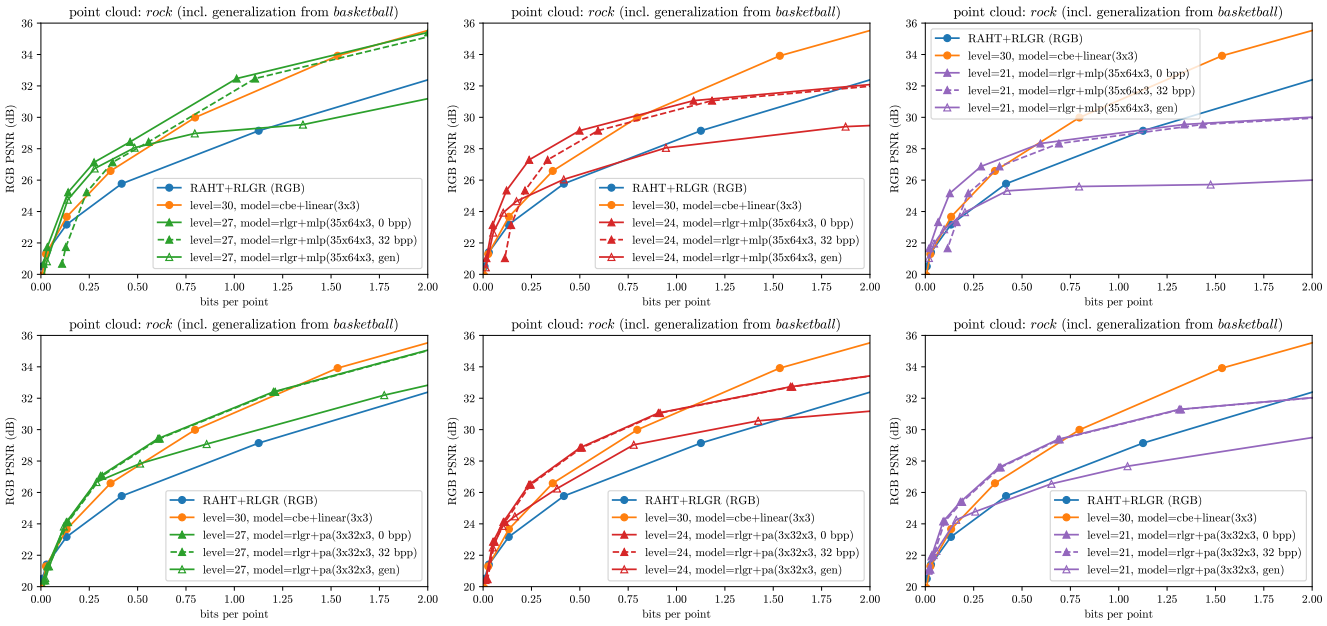

Figure 17. Effect of side information for coordinate based networks *mlp(35x64x3)* (top) and *pa(3x32x3)* (bottom) at levels 27 (left), 24 (middle), and 21 (right). See Fig. 10 for *mlp(32x256x3)*.

0.129 bpp, 23.2 dB | 0.127 bpp, 24.3 dB | 0.127 bpp, 25.9 dB

0.540 bpp, 26.5 dB | 0.536 bpp, 28.2 dB | 0.531 bpp, 30.0 dB

1.005 bpp, 28.7 dB
RAHT-RLGR (RGB)

1.019 bpp, 31.3 dB
RAHT-RLGR (YUV)

1.003 bpp, 33.0 dB
LVAC mlp(35x256x3)

Figure 18. Subjective quality of point cloud *rock*. Each row is a different bit rate. Original is shown in Fig. 13. Zoom in to see differences. See Fig. 13 for 0.25 bpp.

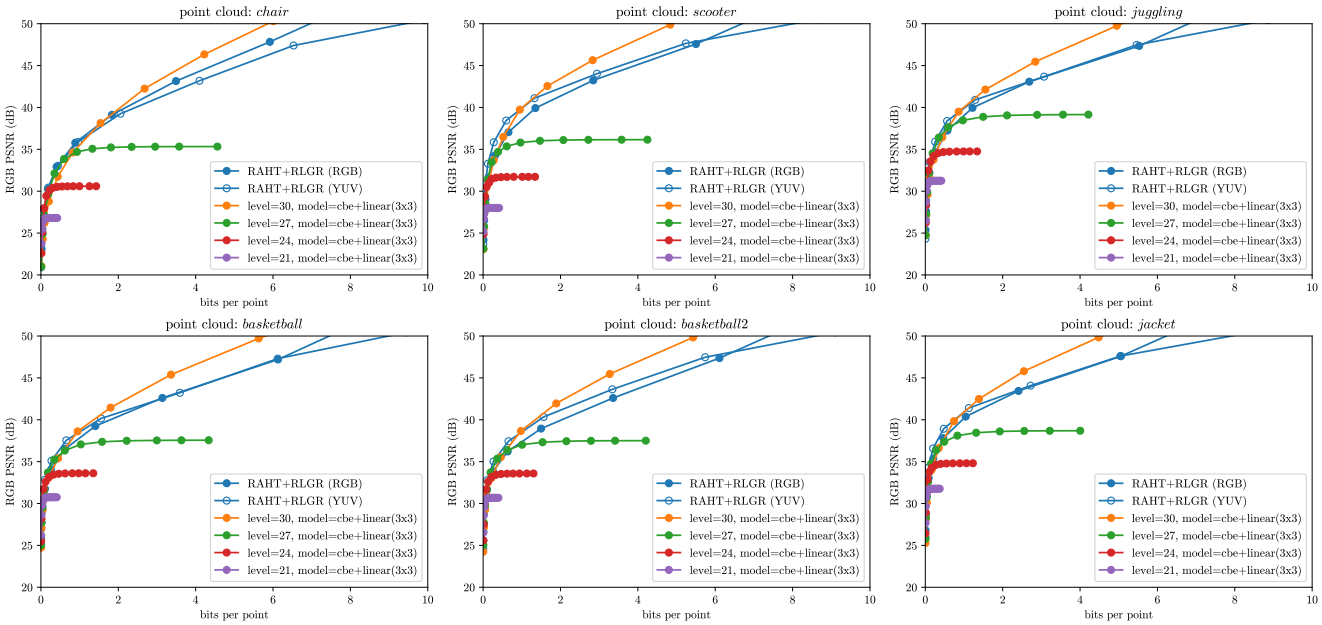

Figure 19. Baselines for six point clouds. *RAHT+RLGR (RGB)* and *(YUV)* are shown against 3×3 linear models at levels 30, 27, 24, and 21, which optimize the colorspace by minimizing $D + \lambda R$ using the *cbe* entropy model. See Fig. 6 for point cloud *rock*.

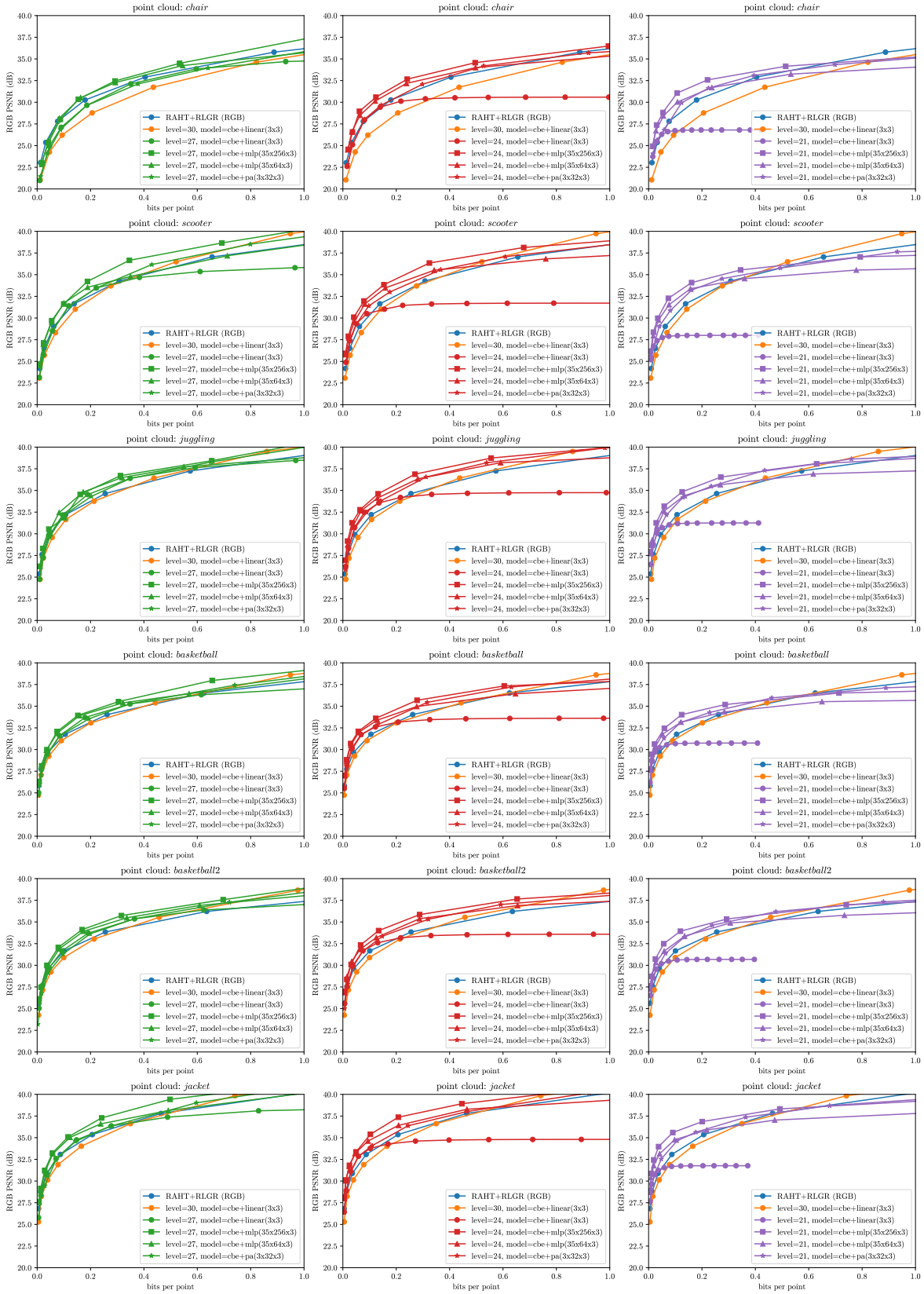

Figure 20. Coordinate Based Networks, by target level. Each row is a different point cloud. Left, middle, right columns each show *mlp(35x256x3)*, *mlp(35x64x3)*, and *pa(3x32x3)* CBNs, along with baselines, at levels 27, 24, 21. See Fig. 7 for point cloud *rock*.

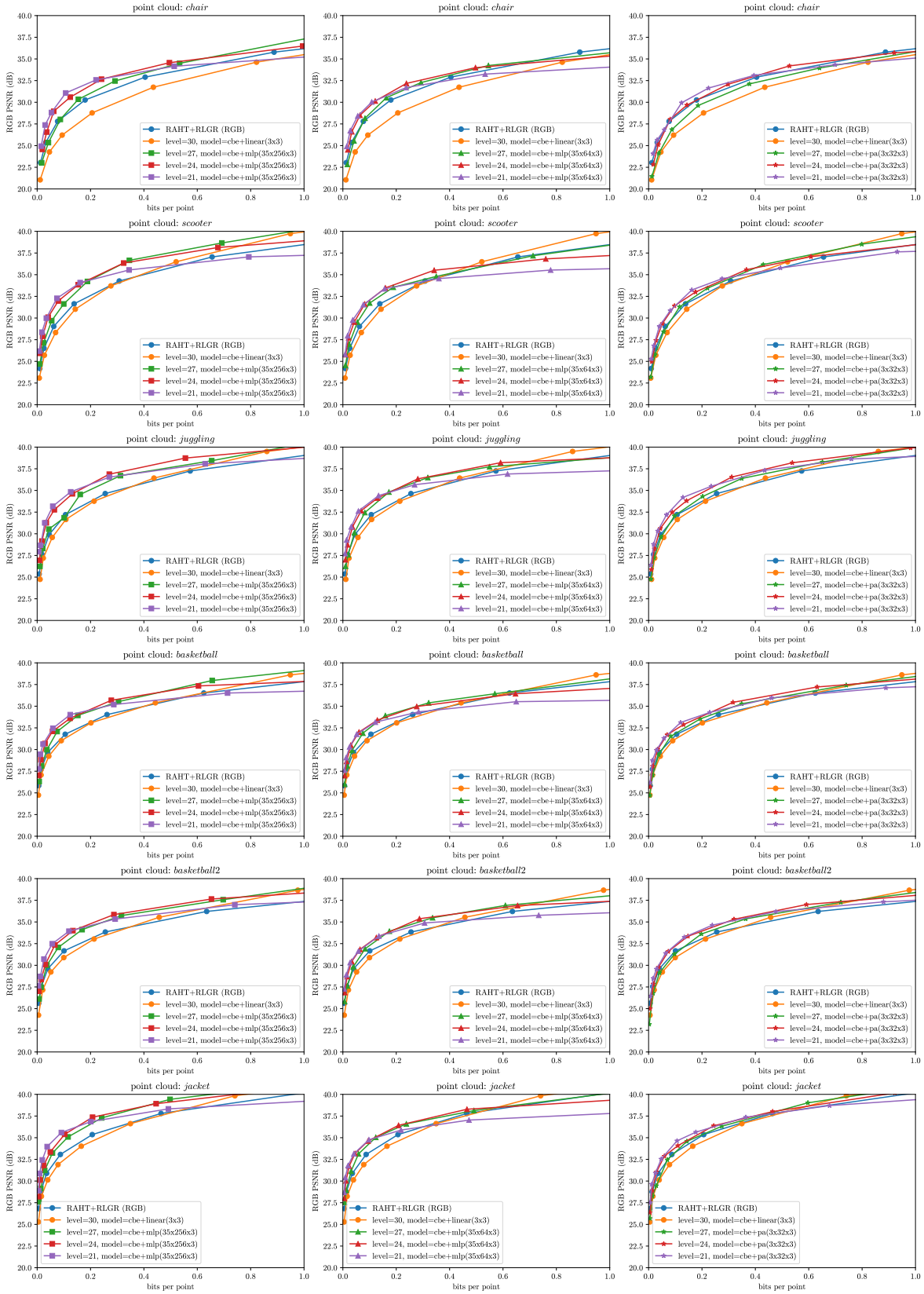

Figure 21. Coordinate Based Networks, by network. Each row is a different point cloud. Left, middle, right columns each show levels 27, 24, and 21, along with baselines, for CBNs *mlp(35x256x3)*, *mlp(35x64x3)*, and *pa(3x32x3)*. See Fig. 15 for point cloud *rock*.

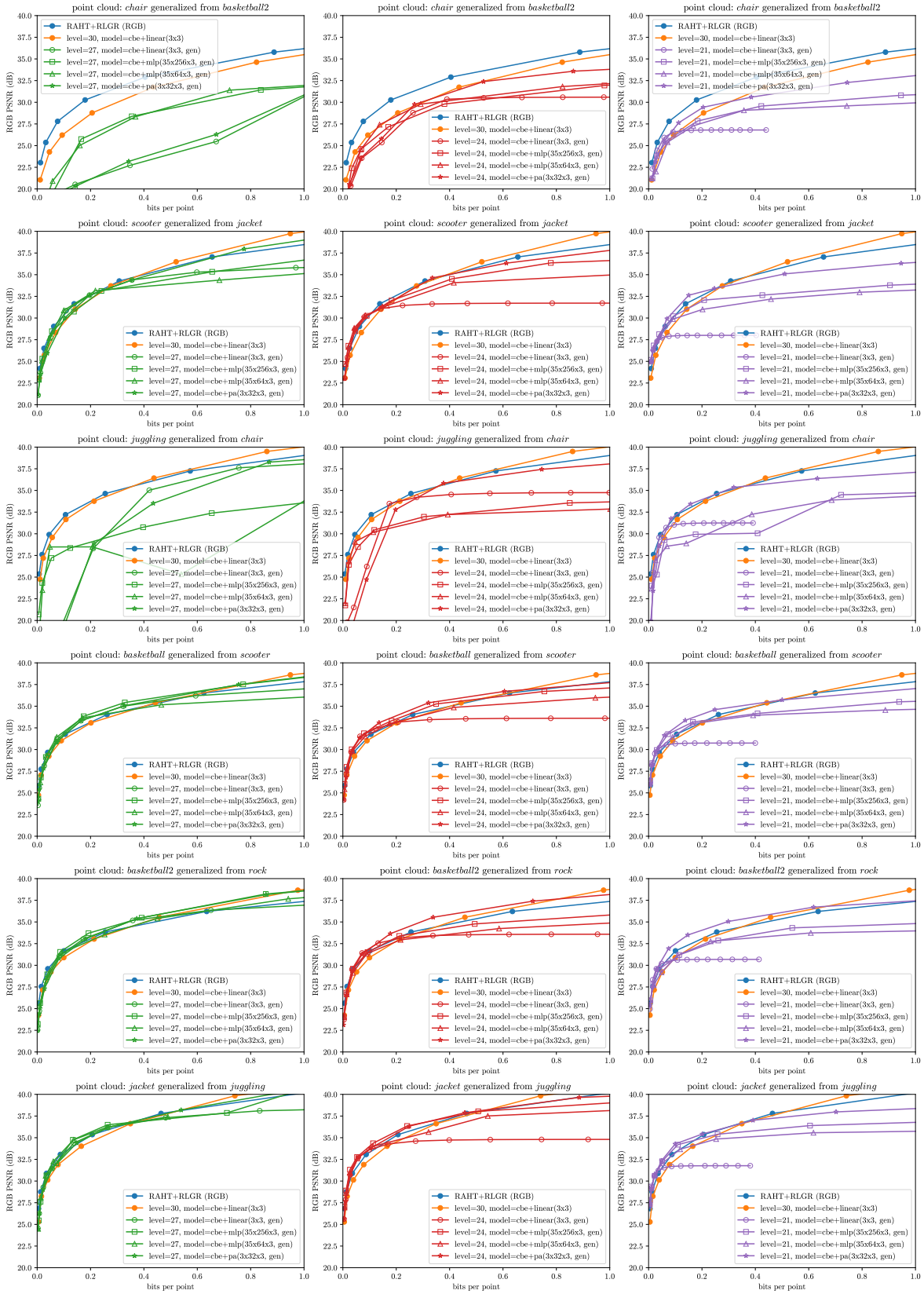

Figure 22. Coordinate Based Networks with generalization, by target level. Each row is a different point cloud. Left, middle, right columns each show *mlp(35x256x3)*, *mlp(35x64x3)*, and *pa(3x32x3)* CBNs, along with baselines, at levels 27, 24, 21. See Fig. 16 (top) for point cloud *rock*.

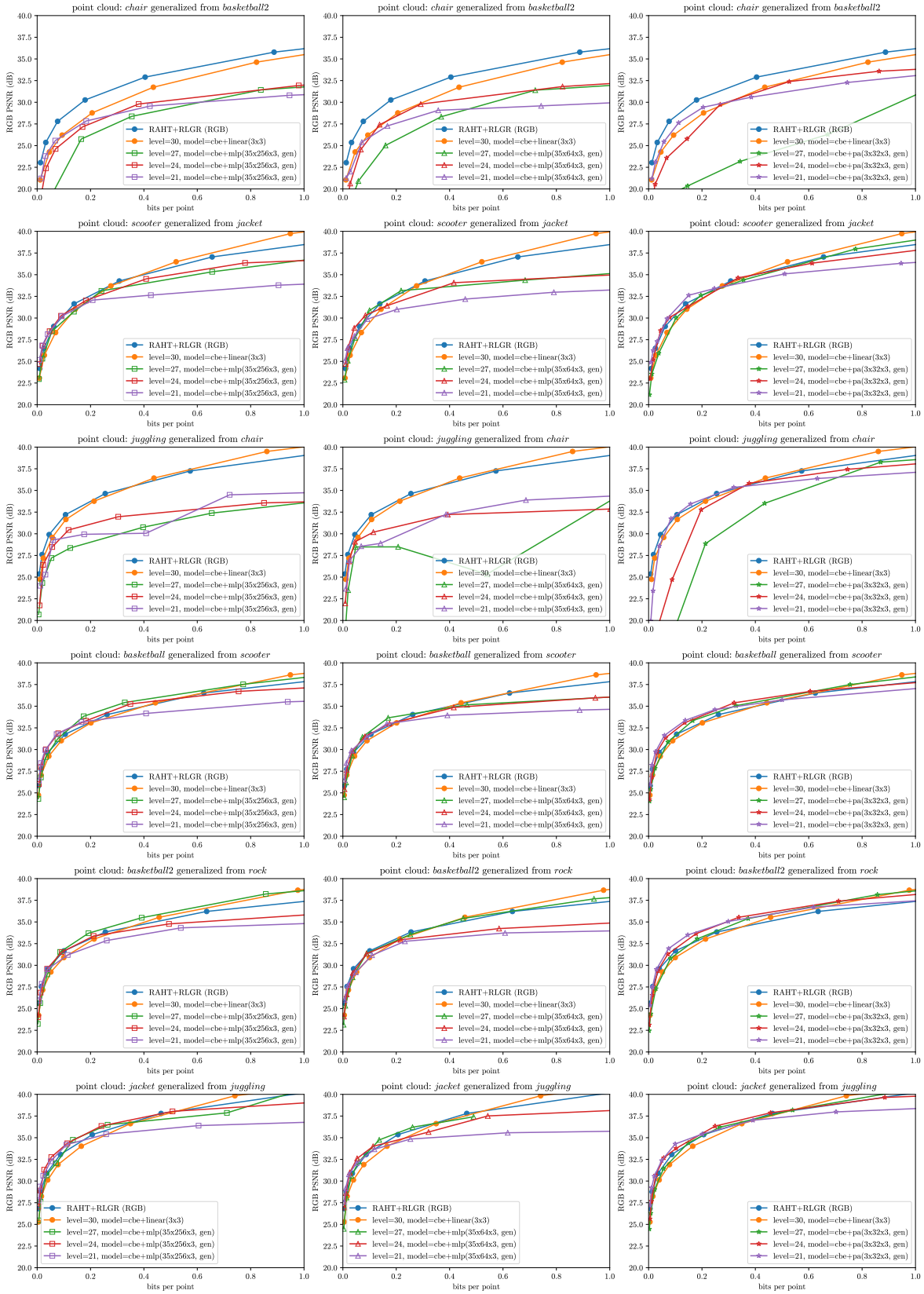

Figure 23. Coordinate Based Networks with generalization, by network. Each row is a different point cloud. Left, middle, right columns each show levels 27, 24, and 21, along with baselines, for CBNs *mlp(35x256x3)*, *mlp(35x64x3)*, and *pa(3x32x3)*. See Fig. 16 (bottom) for point cloud *rock*.

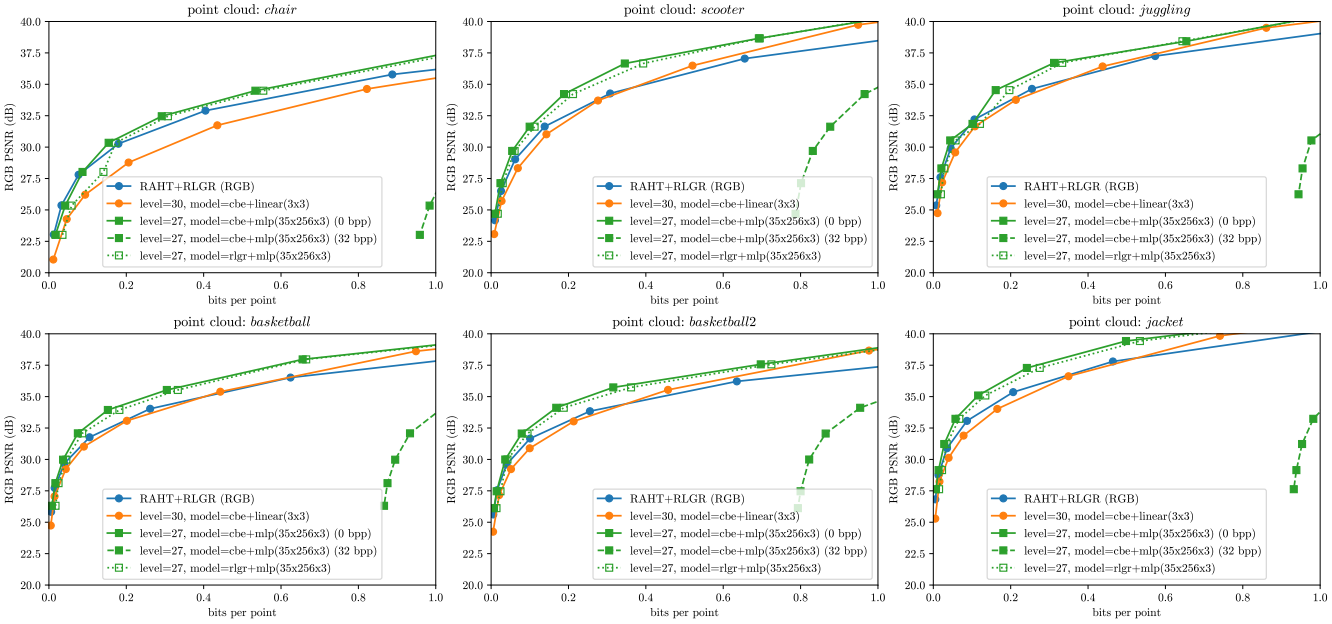

Figure 24. Side information for entropy model. Sending 32 bits per parameter for the *cbe* entropy model would reduce RD performance from solid to dashed green lines. But the backward-adaptive *RLGR* entropy coder (dotted, unfilled) obviates the need to send side information with almost no loss in performance. See Fig. 9 for point cloud *rock*.

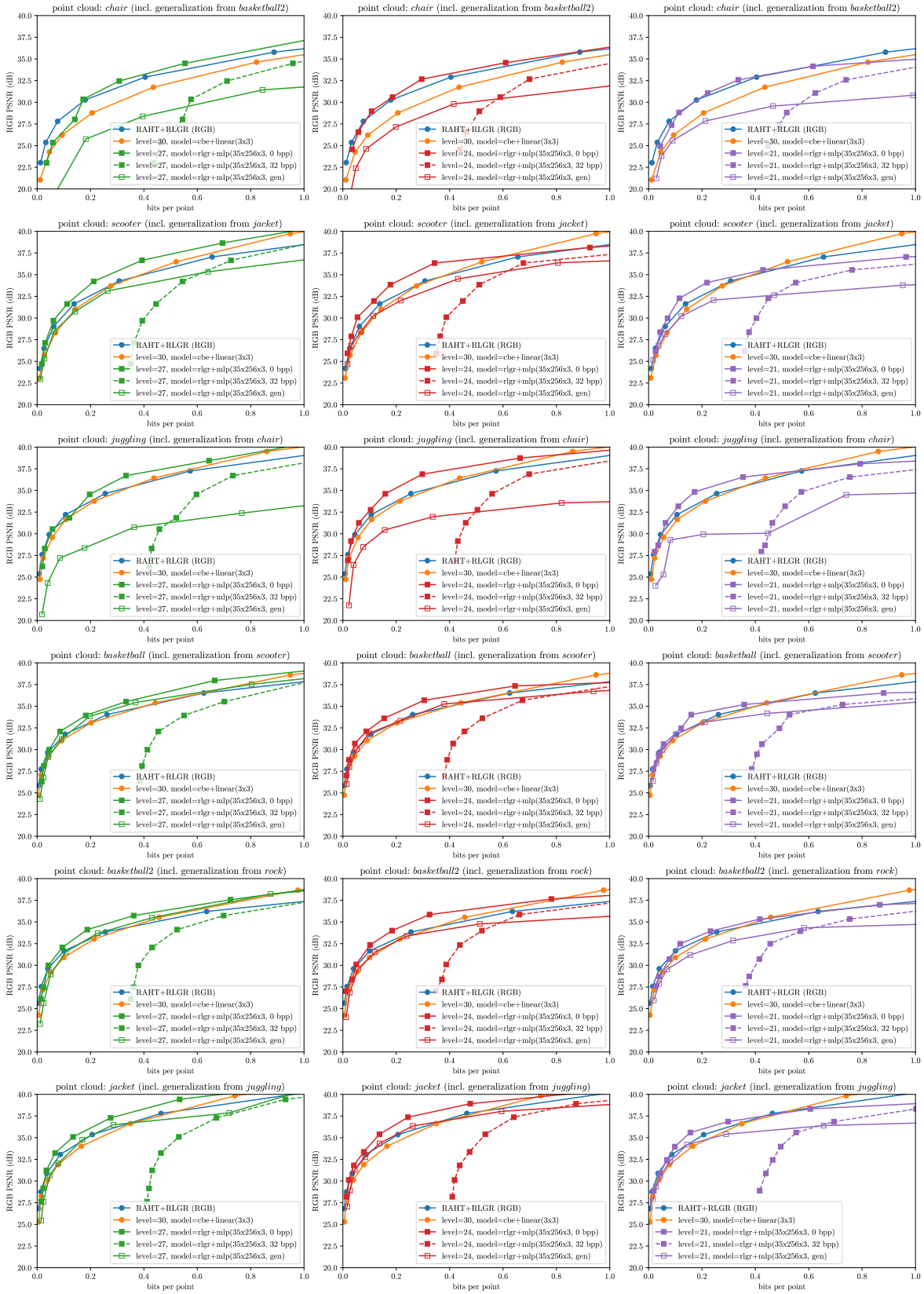

Figure 25. Effect of side information for coordinate based network *mlp(35x256x3)* at levels 27 (left), 24 (middle), and 21 (right). Each row is a different point cloud. See Fig. 10 point cloud *rock*.

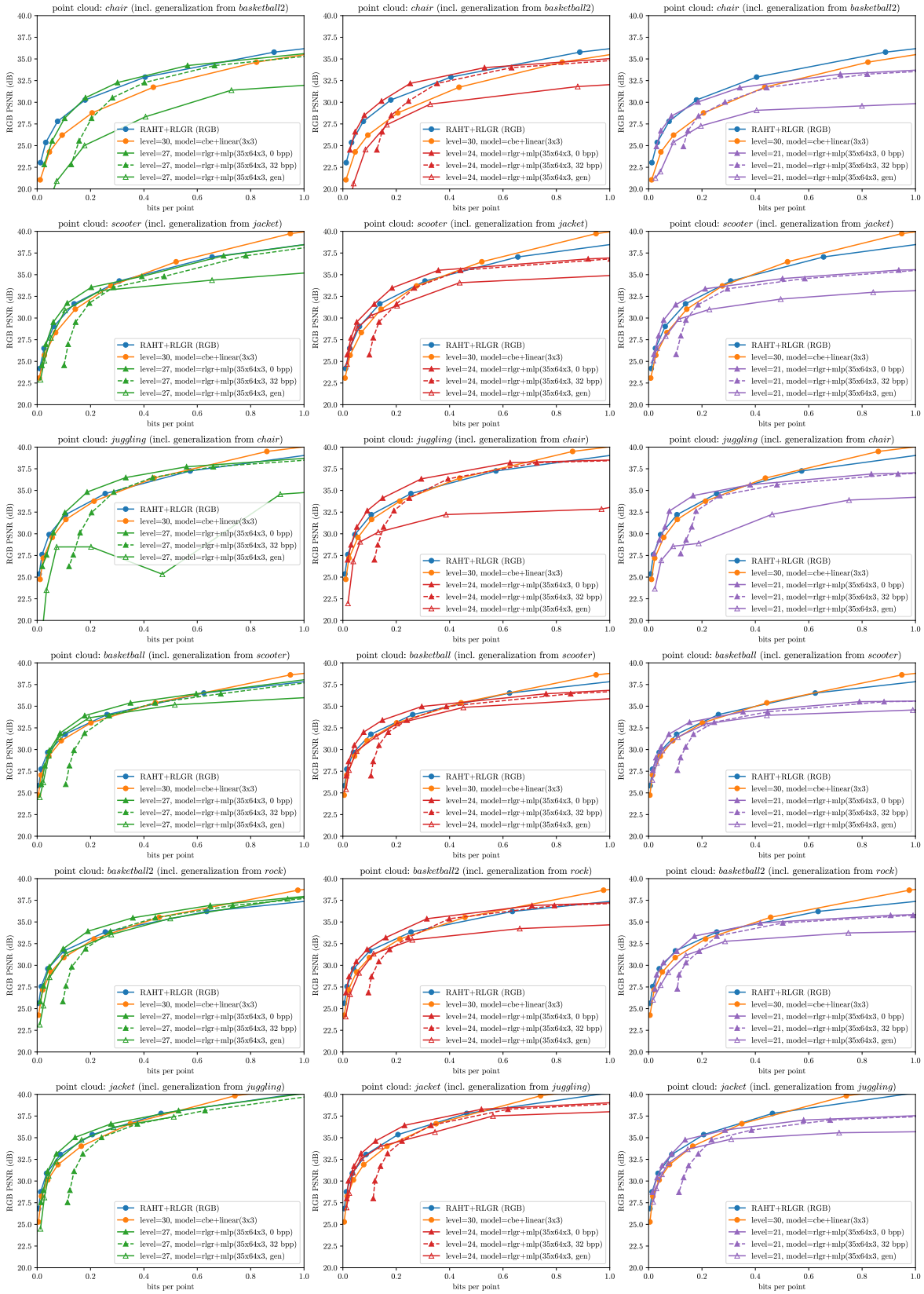

Figure 26. Effect of side information for coordinate based network *mlp(35x64x3)* at levels 27 (left), 24 (middle), and 21 (right). Each row is a different point cloud. See Fig. 17 (top) for point cloud *rock*.

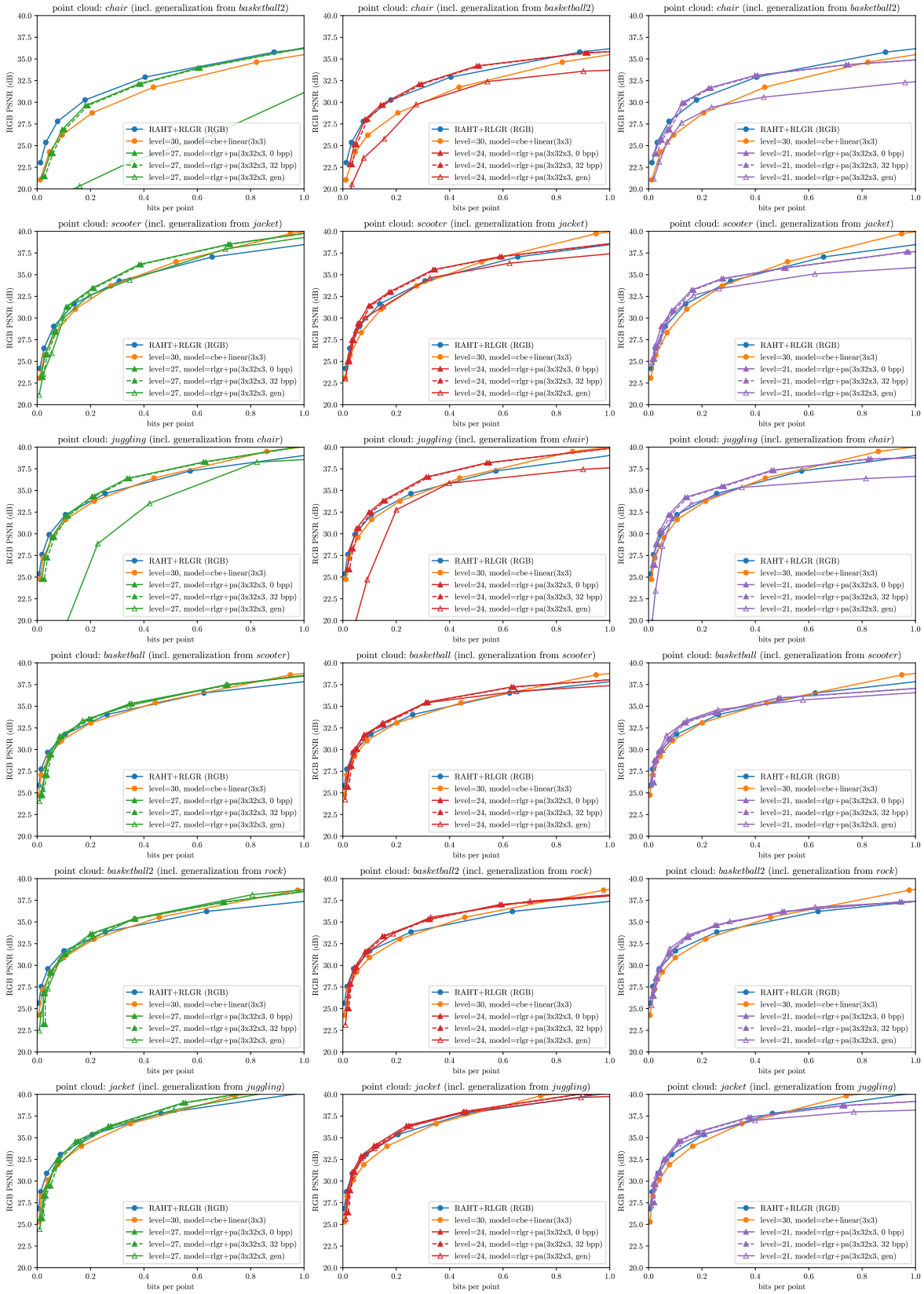

Figure 27. Effect of side information for coordinate based network *pa(3x32x3)* at levels 27 (left), 24 (middle), and 21 (right). Each row is a different point cloud. See Fig. 17 (bottom) for point cloud *rock*.

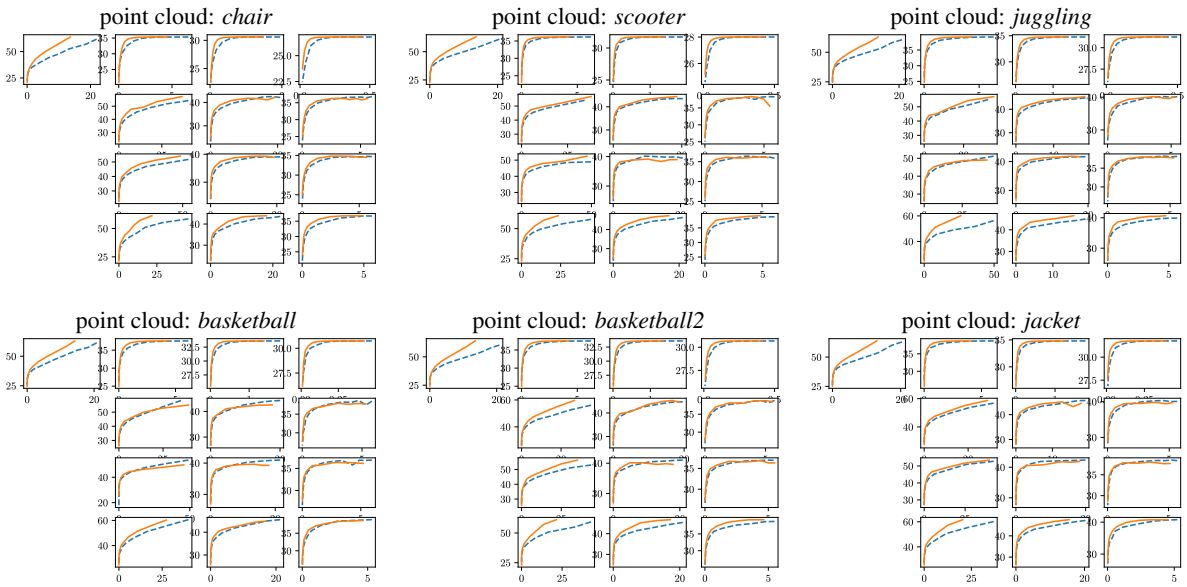

Figure 28. RD performance (RGB PSNR vs. bit rate) improvement due to normalization, corresponding to entries in Tab. 4. See Fig. 11 for point cloud *rock*.

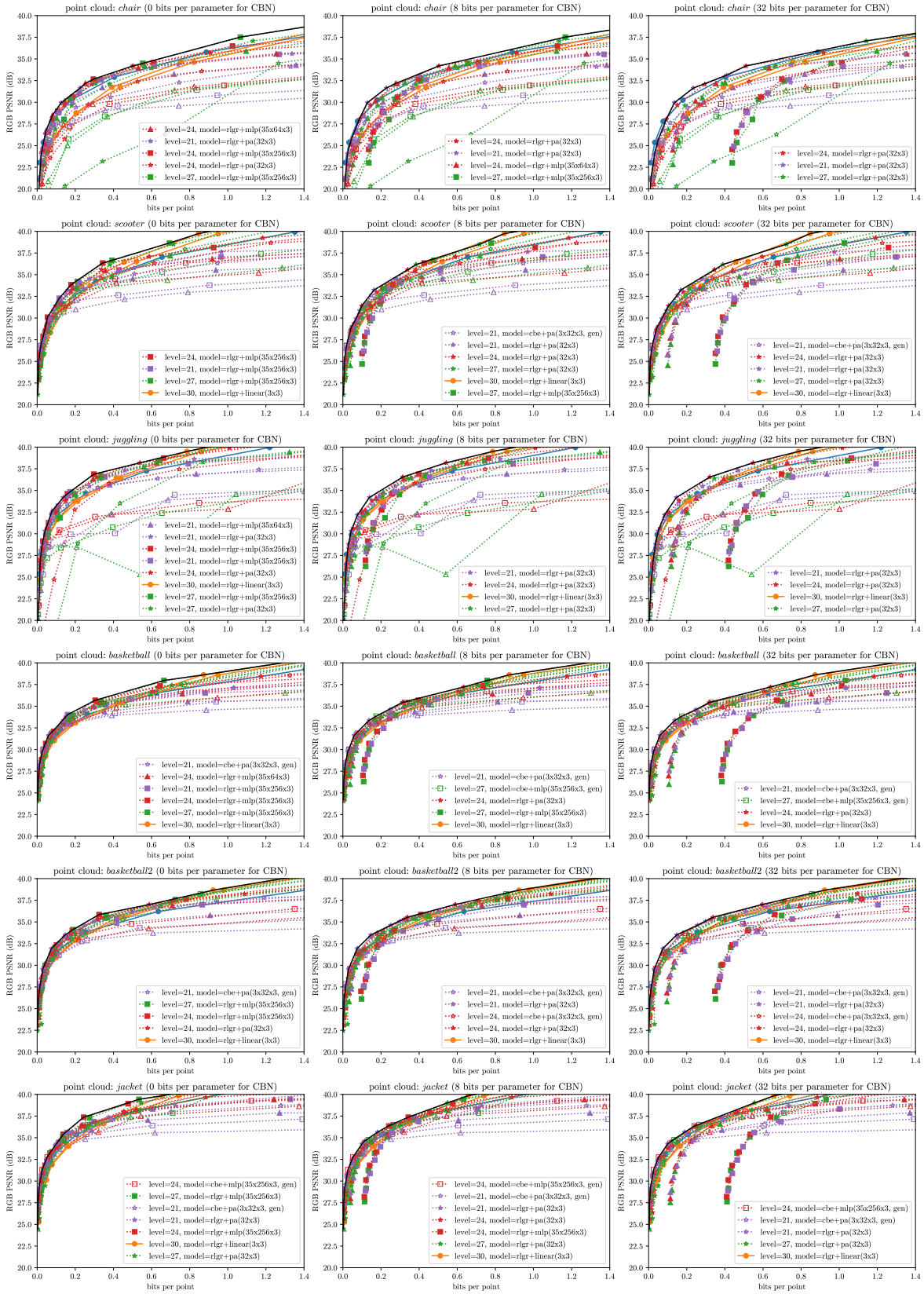

Figure 29. Convex hull (solid black line) of RD performances of all CBN configurations across all levels, including side information using 0 (left), 8 (middle), and 32 (right) bits per CBN parameter. Each row is a different point cloud. See Fig. 12 for point cloud *rock*.