Deep Implicit Volume Compression (Supplementary Material)

Danhang Tang* Saurabh Singh* Philip A. Chou Christian Häne Mingsong Dou
Sean Fanello Jonathan Taylor Philip Davidson Onur G. Guleryuz Yinda Zhang
Shahram Izadi Andrea Tagliasacchi Sofien Bouaziz Cem Keskin

Google

8. Background on compression

Truncated Signed Distance Fields. A surface $\mathcal{S}$ represented in TSDF implicit form is the zero crossing of a function $\Phi(\boldsymbol{x})$:$\mathbb{R}^3 \rightarrow \mathbb{R}$ that interpolates a uniform $W \times H \times D$ 3D grid of truncated (and signed) distances from the surface. By convention, distances outside and inside the surface get positive and negative signs respectively, and magnitudes are truncated by a threshold value τ. Typically a method like marching cubes [37] is used to determine the *topology* of each voxel (*i.e*. which voxel edges intersect with the surface), as well as the offsets of the intersection points for the valid edges, which are then used to form a triangular mesh.

Lossless compression. The primary goal of general purpose lossless compression is to minimize the storage or transmission costs (typically measured in bits) of a discrete dataset $\mathcal{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$. Each data point of $\mathcal{X}$ is mapped to a variable length string of bits for storage or transmission by the sender. A receiver then inverts the mapping to recover the original data from the transmitted bits. The Shannon entropy $H = -\sum_{\boldsymbol{x}} p_{\boldsymbol{x}}(\boldsymbol{x}) \log p_{\boldsymbol{x}}(\boldsymbol{x})$ provides an achievable lower bound on the rate, *i.e*. the minimum expected number of bits required to encode an element, where $p_{\boldsymbol{x}}(\boldsymbol{x})$ is the underlying distribution of $\boldsymbol{x}$. This is achievable by encoding $\boldsymbol{x}$ to a bit string of length $-\log p_{\boldsymbol{x}}(\boldsymbol{x})$ bits. Although this length is not necessarily an integer, it can be achieved arbitrarily closely on average by an arithmetic coder [14]. With this encoding, the number of bits needed to code the entire dataset is

$$R(\mathcal{X}) = -\tfrac{1}{N} \sum_{i=1}^{N} \log p_{\boldsymbol{x}}(\boldsymbol{x}_i), \tag{4}$$

where R is referred to as the *bit rate* of the compression.

Lossy compression. In contrast, lossy compression methods can achieve significantly higher compression rates by allowing errors in the received data. These errors are typically referred to as *distortion* D. In lossy compression there is a fundamental compromise between the distortion D and the bit rate R, referred to as *rate-distortion* trade–off, where distortion can be decreased by spending more bits. Minimizing D subject to a constraint on R leads to the following unconstrained optimization problem [9, 46]

$$\arg\min_{\hat{\boldsymbol{x}}} D(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \lambda R(\hat{\boldsymbol{x}}), \tag{5}$$

where $\hat{\boldsymbol{x}}$ is a discrete lossy representation of $\boldsymbol{x}$ and λ is a trade–off parameter. Higher values of λ result in better bit rates at the expense of increased distortion.

Lossy transform coding. Often $\boldsymbol{x}$ is high dimensional, making the direct optimization of the problem above intractable. As a result, *lossy transform coding* is more commonly used instead. In lossy transform coding, a transformation is used to transform the original data $\boldsymbol{x}$ into a latent representation $\mathbf{z}=\mathcal{E}(\boldsymbol{x};\boldsymbol{\theta}_e)$ and another is used to approximately recover the original data $\hat{\boldsymbol{x}}=\mathcal{D}(\hat{\mathbf{z}};\boldsymbol{\theta}_d)$ from the lossy latent representation $\hat{\mathbf{z}}$. The transformations $\mathcal{E}$ and $\mathcal{D}$, with parameters $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_d$, respectively, are typically chosen to simplify the conversion from $\mathbf{z}$ to its lossy *discrete* version $\hat{\mathbf{z}}=Q(\mathbf{z})$ – a process called *quantization*. While $\mathcal{E}$ and $\mathcal{D}$ can be invertible transformations (*e.g*. the discrete cosine transform used for JPEG compression), in general they are not required to be. Thus, with $\boldsymbol{\theta}=\{\boldsymbol{\theta}_e, \boldsymbol{\theta}_d\}$, the original rate-distortion problem can be re-written as

$$\arg\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \; D(\boldsymbol{x}, \hat{\boldsymbol{x}}; \boldsymbol{\theta}) + \lambda R(\hat{\mathbf{z}}; \boldsymbol{\phi}), \tag{6}$$

where $\hat{\boldsymbol{x}}=\mathcal{D}(\hat{\mathbf{z}};\boldsymbol{\theta}_d)$, $\hat{\mathbf{z}}=Q(\mathcal{E}(\boldsymbol{x};\boldsymbol{\theta}_e))$, and the bit rate is $R(\hat{\mathbf{z}};\boldsymbol{\phi})=-\log p_{\hat{\mathbf{z}}}(\hat{\mathbf{z}};\boldsymbol{\phi})$, with $p_{\hat{\mathbf{z}}}$ as a probability model of $\hat{\mathbf{z}}$ with parameters $\boldsymbol{\phi}$ that is learned jointly with $\boldsymbol{\theta}$. The code $\hat{\mathbf{z}}$ is converted to the corresponding variable length bit representation by entropy coding using the learned prior distribution $p_{\hat{\mathbf{z}}}$.

Quantization. Since the quantization operation is non-differentiable, training such a network in an end-to-end fash-

*indicates equal contribution.

Method	Rate Parameters (varied)	Fixed Parameters
Ours	$\lambda = \frac{1}{10^{\mu}}, \mu = i \times \frac{\log_{10} 200000}{11}$ (for $i = 0, \dots 11$)	
Tang et al. [59]	$K_{total} = 1024, 2048 \dots 5120$	numRetainedKLTBases $= 64$
Google Draco [24]	qp $= 8, \dots, 11$	qt $= 11$ skip $=$ normal
MPEG V-PCC [57]	ri configurations (for $i = 1, \dots, 5$)	geometry3dCoordinatesBitdepth $= 11$ geometryNominal2dBitdepth $= 8$ minNormSumOfInvDist4MPSelection $= 0.36$ partialAdditionalProjectionPlane $= 0.15$ minimumImageWidth $= 2560$ apply3dMotionCompensation $= 0$

Table 3: Parameters used for the experiments in Figure 7 of the main paper.

ion is challenging. Ballé et al. [2] propose simulating quantization noise during training rather than explicitly discretizing the code. Specifically, they quantize $\mathbf{z}$ by rounding to nearest integer $\hat{\mathbf{z}}{=}Q(\mathcal{E}(\boldsymbol{x};\boldsymbol{\theta}_e)){=}\lfloor\mathcal{E}(\boldsymbol{x};\boldsymbol{\theta}_e)\rceil$, which they model by adding of uniform noise during training, *i.e.* $\hat{\mathbf{z}}{=}\mathcal{E}(\boldsymbol{x};\boldsymbol{\theta}_e){+}\epsilon$, $\epsilon \sim \mathcal{U}[-0.5, 0.5]$ to simulate quantization errors; see [2] for additional details.

9. Network architecture and training

We visualize the architecture of our model in Figure 11, which is formed by a three layer encoder and decoder. While the architecture is similar to a convolutional autoencoder (implemented with convolutions in the encoder and transposed convolutions in the decoder), the main difference lies in the transformation the latent code goes through, and the additional losses that aim to minimize the bit rate as well as the reconstruction error, as visualized in Figure 12. Specifically, we add uniform noise to the code during training to simulate quantization. At test time we quantize the code and compress it with an entropy coder. Additionally, the decoder has two final convolutional heads that separate the estimation of signs and the TSDF values. The one and two layer models we experiment with are similar with fewer layers.

Figure 12 provides an overview of our training setup with the dependencies for the three terms in our training loss. Unlike a regular autoencoder which only aims to minimize the reconstruction error, we employ two additional losses $R_{\hat{\mathbf{z}}}$ and $R_{\mathbf{s}}$ to minimize the bit rates for the compressed signals for the latent code and the ground truth signs. Additionally, instead of equally weighting each element the reconstructed $\hat{\boldsymbol{x}}$, we use the ground truth signs $\mathbf{s}$ to mask the voxels that have no neighboring voxels with opposing signs and have therefore less significance.

10. Baseline parameters

The parameters used in our experiments (Figure 6) are described in Table 3, except for JP3D [55] and FVV [13] which we obtained from Tang et al. [59]. To generate a curve, we varied the corresponding rate parameter during inference, whilst keeping other parameters fixed as shown. Notations and definitions of parameters can be found in respective citations.

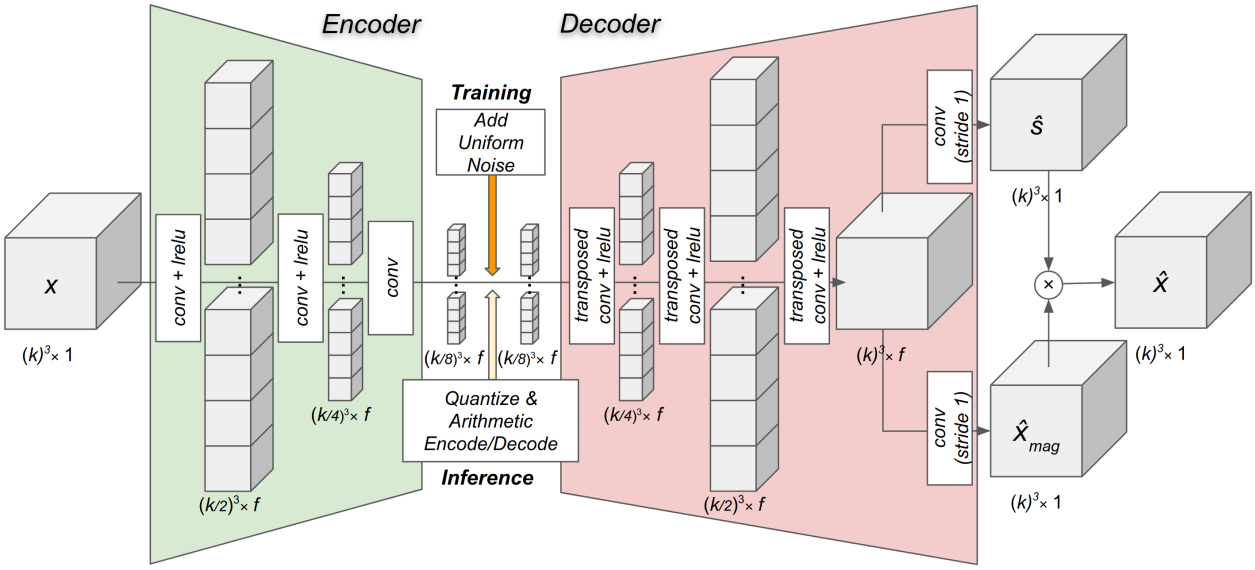

Figure 11: **Network architecture.** The encoder $\mathcal{E}$ consists of three convolutional layers and the decoder $\mathcal{D}$ has three transposed convolution layers, each with a stride of two. $\mathcal{D}$ has two convolutional heads with a stride of one, which separates sign prediction from TSDF estimation. Refer to Section 4 in main paper for details.

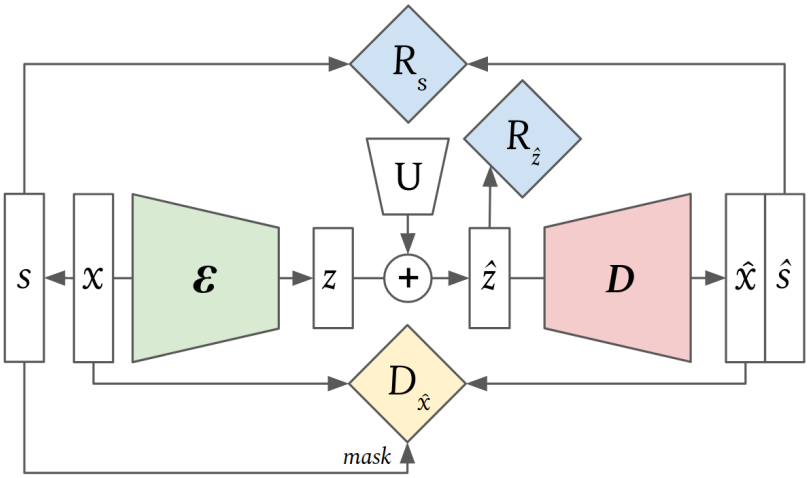

Figure 12: **Training losses.** During training, we employ three different losses as explained in Section 4. Here, the distortion loss $D_{\hat{\mathbf{x}}}$ makes use of the ground truth signs $\mathbf{s}$ to mask the voxels that have no neighboring voxels with opposing signs and have therefore less significance. $R_{\mathbf{s}}$ is the cross entropy between the predicted and actual signs, which is used to minimize the bit rate for compressed ground truth signals. $R_{\hat{\mathbf{z}}}$ is an estimate of the differential entropy of the noisy latent code, also used to minimize the bit rate for the compressed latent code $\hat{\mathbf{z}}$.