Supplementary Materials
D'OH: Decoder-Only random Hypernetworks for Implicit Neural Representations

1 Initialization

In Section 3.2 we noted that we need to apply a modified initialization scheme under the random matrix hypernetwork structure we examine. Under most initiialization schemes (e.g. He [39], Xavier [33], and SIREN [77]), initialization is conducted separately for each layer - a property we want to preserve in the target network. However as we will use the *same* latent parameter vector to generate each layer we will instead need to account for this by changing the per-layer random matrices to match the desired initialization of the target network.

1.1 Derivation

Assume the entries of z are drawn independently and identically distributed from a distribution of variance $\mathrm{Var}(z)$, and that the weights of the l^{th} layer of the target network are to have variance $\mathrm{Var}(W_l)$. We seek a formula for the variance $\mathrm{Var}(B_l)$ of the distribution from which to independently and identically draw the entries of the random matrix B_l such that the entries of $B_l z$ have variance $\mathrm{Var}(W_l)$. We assume that *all* entries for both z and B_l are drawn independently of one another, and with zero mean. From Equation 2, we have:

$$\bar{W}_l = B_l z. \tag{1}$$

Recall that n denotes the dimension of z, and use superscripts to denote vector and matrix indices. Then the above equation can be written entry-wise as:

$$\mathrm{Var}(\bar{W}_l^i) = \mathrm{Var}(\sum_{j=1}^{n} B_l^{ij} z^j) \tag{2}$$

Since the entries of B_l and z are all independent, we therefore have:

$$\mathrm{Var}(\bar{W}_l^i) = \sum_{j=1}^{n} \mathrm{Var}(B_l^{ij} z^j). \tag{3}$$

Again using independence of the entries of B_l and z, we have:

$$\begin{aligned} \mathrm{Var}(\bar{W}_l^i) = &\sum_{j=1}^{n} \mathrm{Var}(B_l^{ij})\mathrm{Var}(z^j) \\ &+ \mathrm{Var}(B_l^{ij})\mathbb{E}(z^j)^2 + \mathbb{E}(B_l^{ij})^2\mathrm{Var}(z^j), \end{aligned} \tag{4}$$

which simplifies to:

$$\mathrm{Var}(\bar{W}_l^i) = \sum_{j=1}^{n} \mathrm{Var}(B_l^{ij})\mathrm{Var}(z^j) \tag{5}$$

by our zero-mean assumption on the entries of B_l and z. Invoking our identically distributed assumption finally yields:

$$\mathrm{Var}(\bar{W}_l) = n\mathrm{Var}(B_l)\mathrm{Var}(z), \tag{6}$$

so that:

$$\mathrm{Var}(B_l) = \frac{\mathrm{Var}(\bar{W}_l)}{n\mathrm{Var}(z)}. \tag{7}$$

We will use this formula to find bounds on a uniform distribution for B_l in order to achieve the variance $\mathrm{Var}(W_l)$ of the weights considered in [77]. To initialize B_l using a uniform distribution centred at 0, we must determine its bounds $\pm a$. Taking the variance of a uniform distribution, we have $\mathrm{Var}(B_l) = \frac{1}{12}(2a)^2 = \frac{a^2}{3}$. Substituting into Equation (7), we have:

$$\frac{a^2}{3} = \frac{\mathrm{Var}(\bar{W}_l)}{n\mathrm{Var}(z)}, \tag{8}$$

so that

$$a = \pm\sqrt{\frac{3\mathrm{Var}(\bar{W}_l)}{n\mathrm{Var}(z)}}. \tag{9}$$

1.2 SIREN Equivalent Initialization

We can apply Equation (9) to derive an example SIREN initialization [77].

Input Layer[1]: Assume z is initialized using $U \sim (\pm\frac{1}{n})$ and $\bar{W}_0$ by $U \sim (\pm\frac{1}{fan_{in}})$ where fan_{in} represents the input dimension of the target network:

$$\mathrm{Var}(\bar{W}_0) = \frac{1}{12}(\frac{2}{fan_{in}})^2 = \frac{1}{3fan_{in}^2} \tag{10}$$

$$\mathrm{Var}(z) = \frac{(2/n)^2}{12} = \frac{1}{3n^2} \tag{11}$$

$$\mathrm{Var}(B_0) = \frac{\mathrm{Var}(\bar{W}_i)}{n\mathrm{Var}(z)} = \frac{1/(3fan_{in}^2)}{n/(3n^2)} = \frac{n}{fan_{in}^2} \tag{12}$$

$$a_0 = \pm\sqrt{\frac{3n}{fan_{in}^2}} \tag{13}$$

[1] We follow the SIREN initialization scheme provided in the Sitzmann et al. (2020) codebase, as this has been noted by the authors to have improved performance [77]

Other Layers: $\bar{W}_i$ initialized using $U \sim (\pm\frac{1}{\omega\sqrt{h}})$, where h refers to the number of hidden units, and ω the SIREN frequency.

$$\text{Var}(\bar{W}_i) = \frac{1}{12}(\frac{2}{\omega\sqrt{h}})^2 = \frac{1}{3\omega^2 h} \tag{14}$$

$$\text{Var}(B_i) = \frac{\text{Var}(\bar{W}_i)}{n\text{Var}(z)} = \frac{1/(3\omega^2 h)}{n/(3n^2)} = \frac{n}{\omega^2 h} \tag{15}$$

$$a_i = \pm\sqrt{\frac{3n}{\omega^2 h}} \tag{16}$$

Numerical Comparison We initialize target networks with using Equations (13) and (16) for a range of input and hidden layer dimensions. The D'OH initialization correctly matches the target SIREN weight variances (Figure 1).

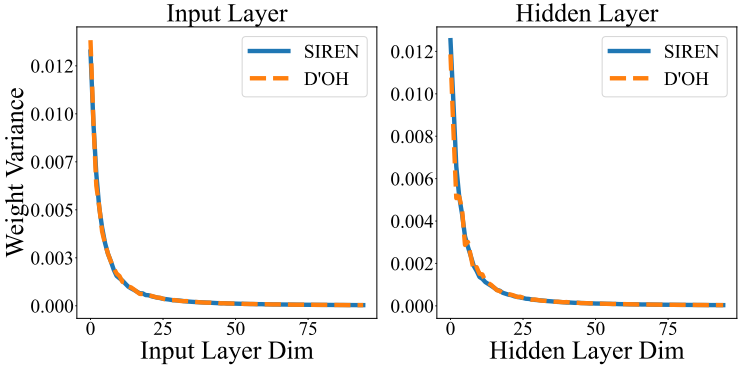


Fig. 1: Numerical comparison of layer variances between SIREN and the weights generated by D'OH (latent dim: 2000 and $\omega = 30$). Our initialization closely matches the initialization of SIREN [77].

2 Quantization, Compression, and Transmission

Quantization We outline here the design decisions for our quantization approach. We employ post-training quantization in our pipeline. While quantization-aware training (QAT) [67] has been demonstrated to reduce quantization error in the context of implicit neural representations [19, 25, 35, 78], we note this has two key disadvantages: each quantization level needs to be trained separately, while post-training quantization can evaluate multiple quantization levels at the same time; and when quantization level is considered as part of the neural architecture search (see: Figure 3) this expands the search space of satisfying

models considerably. In addition, we employ a layer-wise range-based integer quantization scheme between the min and maximum values for each weight and distribution [32]. We select an integer scheme to reduce the quantization symbol set [31,32,41]. We decided on a uniform quantization scheme rather than a non-linear quantizer such as k-means [38] due to the overhead of code-book storage, which for small networks can be substantial proportion of compressed memory [35]. In contrast, we represent each tensor with just three per-tensor components (integer tensor, minimum value, maximum value). Similar range-based integer quantization schemes are commonly described [32,41,47], and the method we use is only a subtle variation avoiding the explicit use of a zero point.

Compression and Transmission In a typical compressed implicit neural network the entire trained and compressed network weights need to be transferred between parties. This is done by first quantizing the weights followed by a lossless entropy compressor, such as BZIP2 [75] or arithmetic coding [78]. Our method generates a target network by a low-dimensional linear code and fixed per-layer random matrices. As random matrices can be reconstructed by the transfer of an integer seed, we only quantize and compress the linear code. The recently proposed VeRA incorporates a similar integer seed transmission protocol for random matrices to improve the parameter efficiency of Low-Rank Adaptive Models [46].

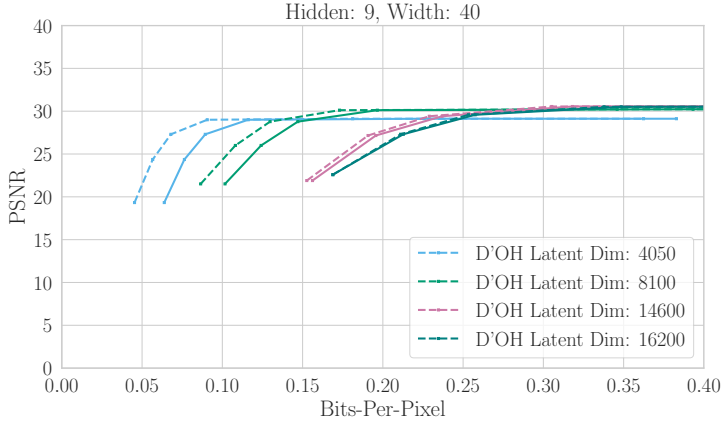
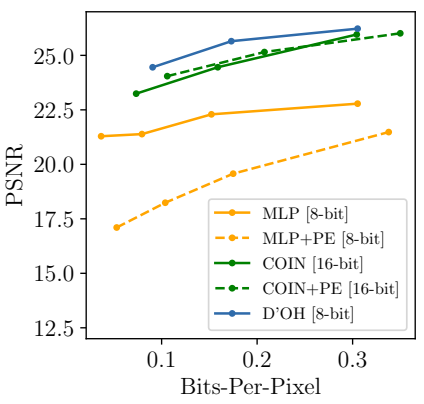
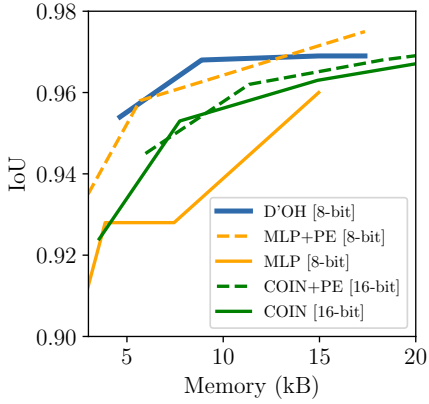


Fig. 2: Comparison of bits-per-pixel (BPP) for estimated memory footprint (parameters × bits-per-weight) [dotted] and memory after applying BZIP2 [solid] to a Python pickle of the quantized model. Rate-distortions generated by varying quantization level. The estimated is a close proxy to an actual entropy coder, but shows some discrepancy at low-rate and low-quantization levels where file overhead represent a larger proportion of code size. To account for this we report the estimated memory footprint for both D'OH and MLPs, which can be seen as a overhead-free limit for performance.

3 Positional Encoding



(a) For image experiments we find that positional encoding reduces rate-distortion performance for MLPs at 8-bit, with little change observed at 16-bits. This is likely due to the increase in parameters and interaction with quantization effects. As a result we report image benchmarks without MLP positional encoding, as the stronger benchmark. Kodak.



(b) For Binary Occupancy experiments, we find that positional encoding is necessary for MLPs to obtain good reconstruction. This is possibly due to the presence of high-frequency spatial components in the 3D shape. Thai Statue.

Fig. 3: Effects of positional encoding on Image and Binary Occupancy Experiments. D'OH does not increase parameters when using positional encoding (See: 10a).

Table 1: Training configurations for Image and Occupancy Field experiments.

Dataset	**Images**	**Occupancy**
Dimensions	Kodak 768×512 DIV2K 512×512	$512 \times 512 \times 512$
Hardware	NVIDIA A100	NVIDIA A100
Optimizer	Adam $\beta = (0.99, 0.999)$	Adam $\beta = (0.99, 0.999)$
Scheduler (Exponential)	$\gamma = 0.999$	$\gamma = 0.999$
Epochs	2000	250
Batch Size	1024	20000
Loss	Mean Square Error	Mean Square Error
Perceptual Metrics	PSNR	IOU
Compression Metrics	Bits-Per-Pixel (BPP)	Memory (kB)
Target MLPs: width/hidden	20/4, 30/4, 28/9, 40/9	20/4, 30/4, 28/9, 40/9
Positional Encoding	10 frequencies	10 frequencies
Activation	Sine ($\omega = 30$)	Sine ($\omega = 30$)
Learning Rates (MLP/DOH)	$2e-4$,$1e-6$	$1e-4$, $1e-6$
Quantization levels	[4, 5, 6, 8, 16]	[4, 5, 6, 8, 16]

4 Further Benchmarks and Results

4.1 Kodak

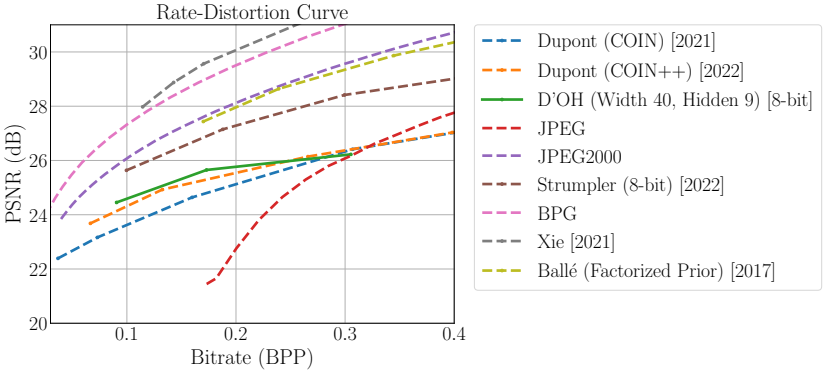


Fig. 4: Rate-Distortion on Kodak showing additional benchmarks. Our method outperforms signal agnostic codecs trained without external datasets (COIN), our method lags both advanced signal specific codecs (JPEG2000 and BPG [11]), and those that employ auto-encoding [8], invertible encoding networks [91], and meta-learned initializations [78]. We suspect that the gap with [78] is due to the use of quantization aware training (QAT). As mentioned in Section 2.2, we avoid QAT as a primary motivation for our method is to reduce the need for architecture search, including different quantization levels (the post-training quantization strategy we employ avoids this).

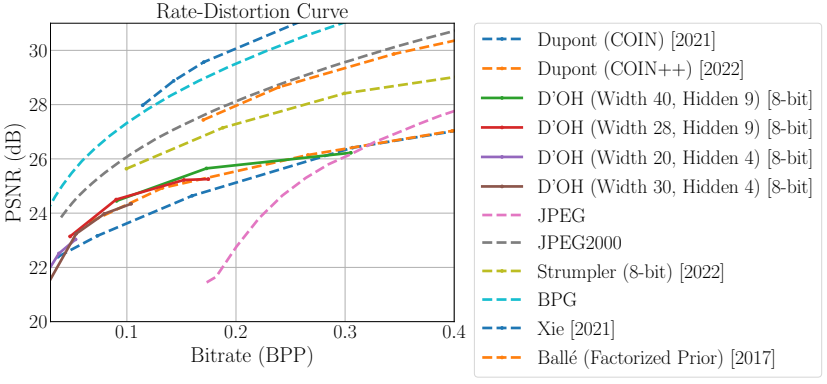


Fig. 5: Ablation running D'OH with alternative COIN target networks. We note that D'OH is able to achieve a rate-distortion improvement on each of these architectures. The resulting model overlay shows an indicative Pareto frontier of the method.

4.2 Occupancy Field

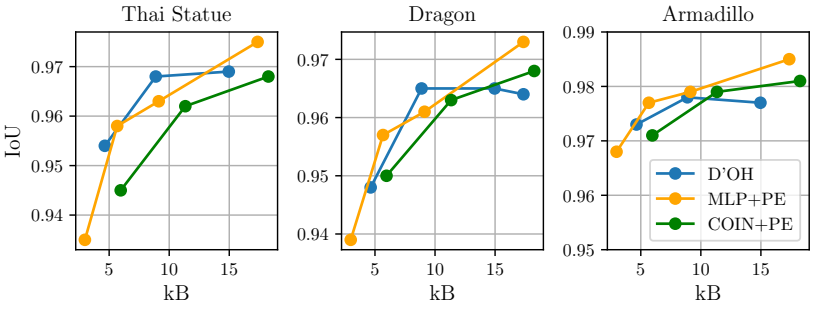


Fig. 6: Rate-distortion curves for Binary Occupancy Fields on Thai Statue, Dragon, and Armadillo. D'OH and MLP are quantized at 8-bit and COIN at 16-bit. As positional encoding is required for Binary Occupancy performance (see: Supplementary Figure 3b, we apply it as the stronger benchmark. While a large rate-distortion advantage over MLPs is observed at 6-bit quantization (see: Supplementary Table 2), when evaluated across all quantization levels and architecture D'OH shows smaller improvement or close performance to MLP models with positional encoding.

Table 2: Binary Occupancy Results for Thai Statue, Dragon, and Armadillo. D'OH performs substantially better than MLP models at low quantization levels (6-bit or lower), and MLPs without positional encoding. At higher quantization levels performance between MLPs and D'OH is comparable, with some rate distortion improvement observed for the 60% D'OH. COIN represents a MLP with 16-bit quantization [23].

		Memory	IoU↑		
Model	**Params**	**(kB)**	**Thai Statue**	**Dragon**	**Armadillo**
6-bit					
MLP (4,20)	1781	1.34	0.74	0.66	0.74
MLP (4,30)	3871	2.90	0.70	0.66	0.87
MLP (9,28)	7449	5.59	0.80	0.67	0.86
MLP (9,40)	14961	11.22	0.82	0.72	0.88
MLP+PE (4,20)	2981	2.24	0.88	0.85	0.94
MLP+PE (4,30)	5671	4.25	0.92	0.87	0.96
MLP+PE (9,28)	9129	6.85	0.93	0.87	0.96
MLP+PE (9,40)	17361	13.02	0.95	0.94	0.97
DOH (30%)	4641	3.48	0.92	0.89	0.95
DOH (60%)	8881	6.67	0.95	0.94	0.97
DOH (100%)	14961	11.22	0.95	0.95	0.97
8-bit					
MLP (4,20)	1781	1.78	0.89	0.85	0.94
MLP (4,30)	3871	3.87	0.93	0.87	0.96
MLP (9,28)	7449	7.45	0.93	0.88	0.96
MLP (9,40)	14961	14.96	0.96	0.93	0.97
MLP+PE (4,20)	2981	2.98	0.94	0.94	0.97
MLP+PE (4,30)	5671	5.67	0.96	0.96	0.98
MLP+PE (9,28)	9129	9.13	0.96	0.96	0.98
MLP+PE (9,40)	17361	17.36	0.98	0.97	0.99
DOH (30%)	4641	4.64	0.95	0.95	0.97
DOH (60%)	8881	8.88	0.97	0.97	0.98
DOH (100%)	14961	14.96	0.97	0.97	0.98
16-bit					
COIN (4,20)	1781	3.56	0.92	0.88	0.97
COIN (4,30)	3871	7.74	0.95	0.90	0.98
COIN (9,28)	7449	14.90	0.96	0.94	0.98
COIN (9,40)	14961	29.92	0.98	0.97	0.99
COIN+PE (4,20)	2981	5.96	0.95	0.95	0.97
COIN+PE (4,30)	5671	11.34	0.96	0.96	0.98
COIN+PE (9,28)	9129	18.26	0.97	0.97	0.98
COIN+PE (9,40)	17361	34.72	0.98	0.98	0.99

4.3 Additional Qualitative Results - Kodak

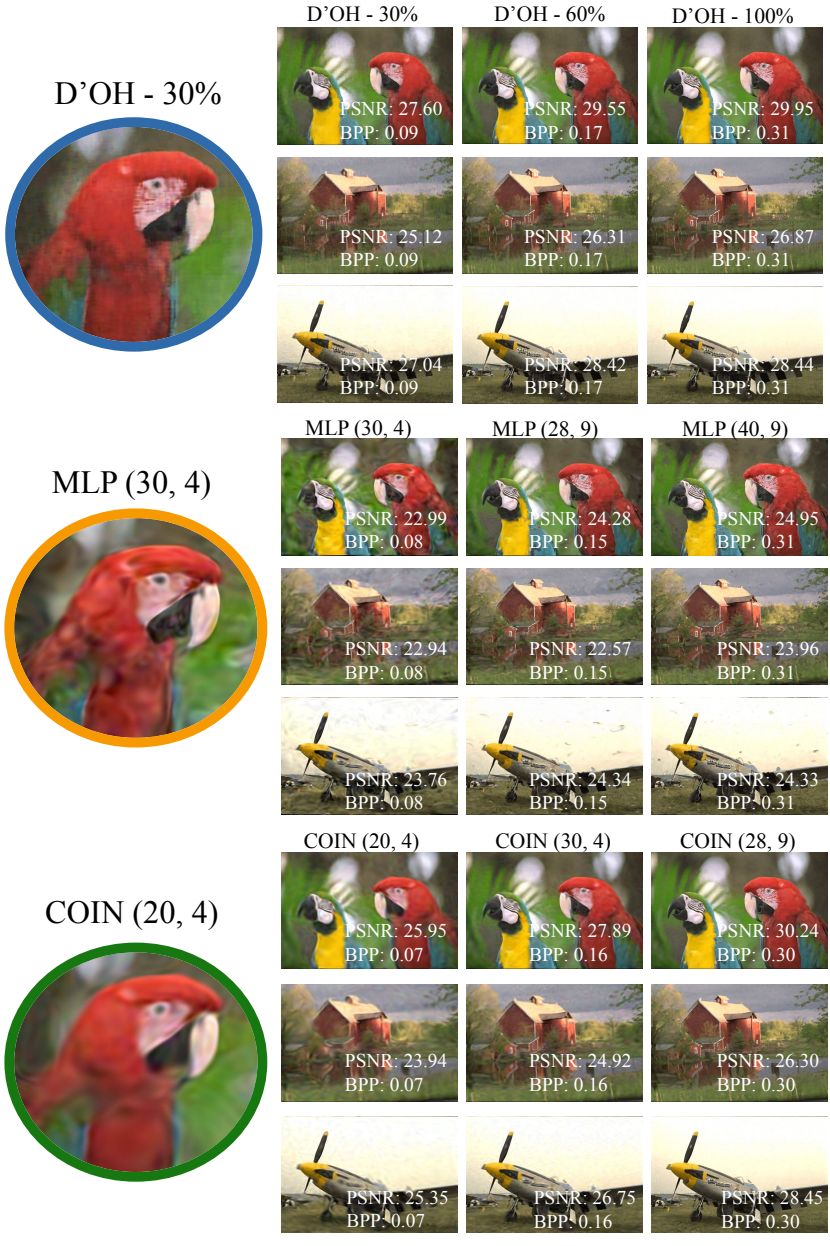


Fig. 7: Additional qualitative results on Kodak showing the comparison between 8-bit D'OH, 8-bit MLP, and COIN (a MLP quantized to 16-bits). Note that smaller COIN architectures are required to match the comparison bit-rates. D'OH is more robust to quantization than the MLP models. D'OH uses positional encoding, while the MLP models do not (see: Figure 3a - PE is detrimental to low-rate MLP performance).

4.4 Additional Qualitative Results - Occupancy Field

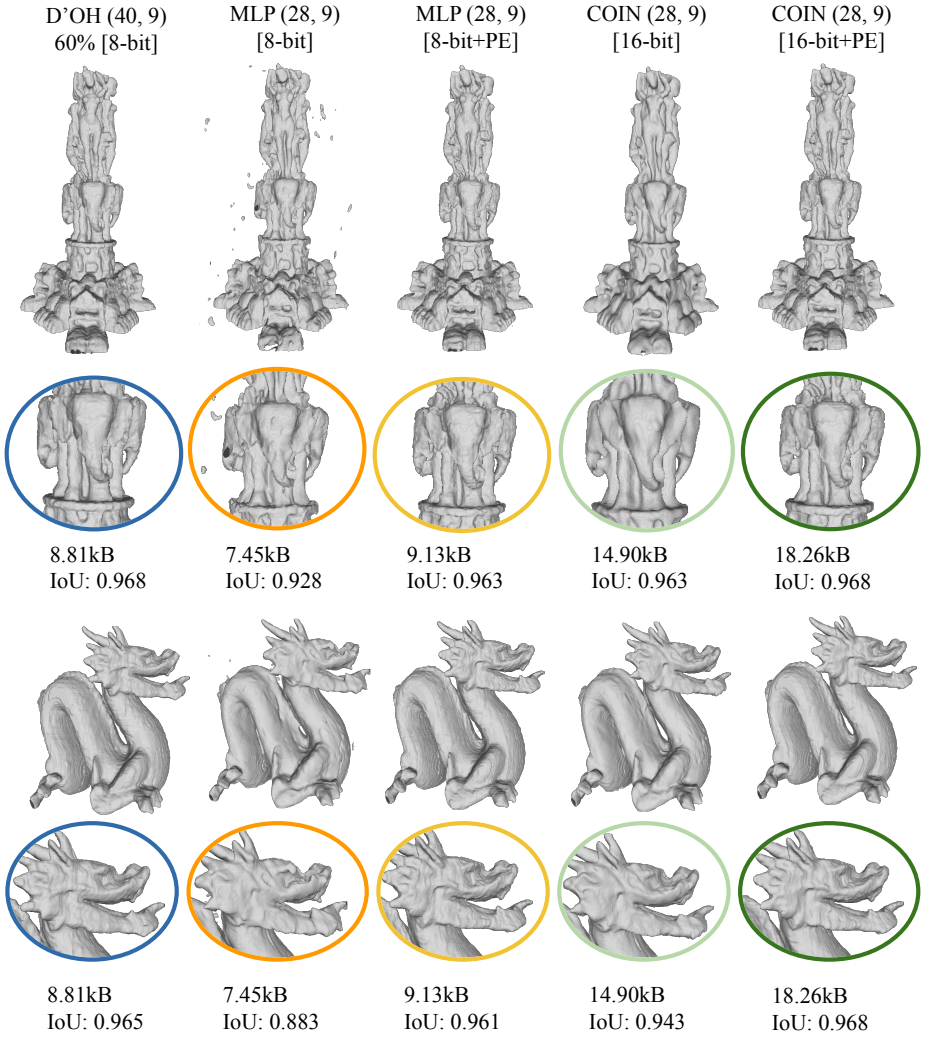


Fig. 8: Binary Occupancy qualitative results on Thai Statue and Dragon. D'OH shows a large performance improvement over MLP models without positional encoding (which lose high frequency information), and shows a small rate-distortion improvement or equivalent performance to MLP and COIN models with higher memory footprints.