

On Quantizing Implicit Neural Representations

Cameron Gordon
University of Adelaide

Shin-Fang Chng
University of Adelaide

Lachlan MacDonald
University of Adelaide

Simon Lucey
University of Adelaide

Abstract

The role of quantization within implicit/coordinate neural networks is still not fully understood. We note that using a canonical fixed quantization scheme during training produces poor performance at low bit-rates due to the network weight distributions changing over the course of training. In this work, we show that a non-uniform quantization of neural weights can lead to significant improvements. Specifically, we demonstrate that a clustered quantization enables improved reconstruction. Finally, by characterising a trade-off between quantization and network capacity, we demonstrate that it is possible (while memory inefficient) to reconstruct signals using binary neural networks. We demonstrate our findings experimentally on 2D image reconstruction and 3D radiance fields; and show that simple quantization methods and architecture search can achieve compression of NeRF to less than 16kb with minimal loss in performance (323x smaller than the original NeRF).

1. Introduction

There is increasing interest in the compression of implicit neural functions [9, 10, 40, 47]. While existing works have examined the use of quantization as part of a neural compression pipeline, there remain a number of classical quantization methods that have been less applied to these problems [14, 15, 13]. In particular, within compression of implicit neural functions the usual method is to apply *uniform* quantization [9, 10, 40], and to use a *fixed* quantization scheme which does not change over the course of training. While simple and efficient, this can introduce quantization error if the underlying distribution varies across training.

In this work, we apply a cluster quantization method to more closely represent the weights of implicit neural representations. In quantization literature a key idea is to match the *distribution* of the quantized and original signals as closely as possible to prevent reconstruction error; this is

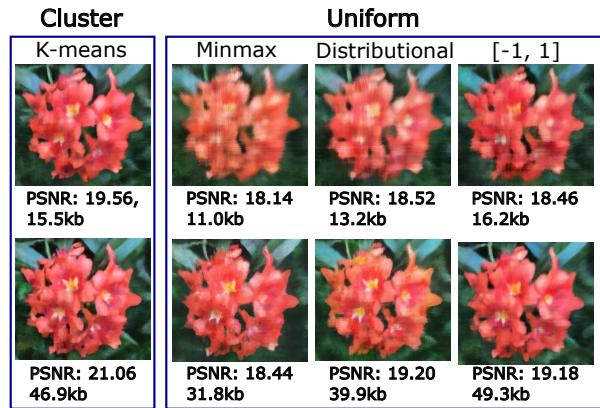


Figure 1. Comparison of cluster and uniform quantization on a small NeRF model. Top: 4-layers, 64 neurons per layer. Bottom: 4-layers, 128 neurons per layer. Quantization to 3-bits-per-weight.

achievable through the use of clustered partitioning of signals [13, 15]. In addition, it is well-known that the distribution of weight values in a network changes over the course of training - as such the distributional assumption at one epoch - may not be valid over the entire training cycle.

Furthermore while it is known that uniform quantization methods can enable representation of signals with a high degree of fidelity, the trade-off between network capacity and the level of feasible quantization has been less explored [9, 10, 40]. Intuitively it would ordinarily be expected that increasing the amount of network quantization should be beneficial in reducing the rate (i.e. size in bits) of the network. However, to maintain the same reconstruction quality one needs to dramatically increase the size of the network to offset the loss of fidelity in the weights. Surprisingly, we show that a *higher* quantization level with a *restricted* network structure can be more memory efficient than a *low* quantization level with a more expressive network structure.

The broad focus of this paper is a comparative analysis of cluster and uniform quantization in implicit neural representations at low bits-per-weight. In particular, our contri-

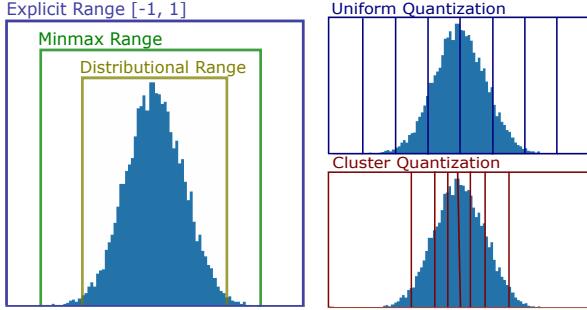


Figure 2. Left: Different uniform quantization ranges. Right: Comparison between decision boundaries for the examined uniform and cluster quantizations. A uniform quantization scheme equally divides the quantization range. A cluster quantization divides the quantization range such that each partition contains equal mass of the distribution.

butions are as follows:

- We introduce an adaptive clustering strategy for quantization-aware training applied to implicit neural networks, showing improved performance at lower quantization levels than uniform methods for multiple modalities (images and neural radiance fields).
- We demonstrate that a performance trade-off between the level of quantization and the expressivity of network architectures is required to adequately reconstruct a signal, with high-fidelity reconstructions able to be achieved even with a binary quantization.
- As an application of the analysis, we demonstrate that substantial compression of neural radiance fields (323x smaller than the original NeRF [27] and 58x smaller than cNeRF [2]) can be obtained with even simple quantization methods and architecture search with minimal collapse in performance.

2. Background

2.1. Quantization

We define a quantization scheme $Q : \mathcal{A} \rightarrow \mathcal{B}$ as a map between two sets \mathcal{A}, \mathcal{B} such that the cardinalities $|\mathcal{A}| \geq |\mathcal{B}|$ (e.g. $Q : \mathbb{R} \rightarrow \mathbb{Z}_5$). While other alphabets are possible, typically the codomain has elements in \mathbb{R} or \mathbb{Z} . For example, a uniform quantization scheme has a codomain with an equal spacing within the range (a, b) . In contrast a non-uniform quantization scheme has non-equal spacing between its elements. Examples of non-uniform quantization schemes include *logarithmic* quantization, in which elements are logarithmically spaced; and cluster quantization, in which elements are partitioned by *decision boundaries* separating clusters of points [14, 15, 13]. The Lloyd Algorithm and K-means are examples of a cluster quantization in which decision boundaries are determined to have equal

data mass, with the values in each partition mapped to the *centroid* (mass centre) of the partition [13, 15].

A *data-dependent* quantization scheme can be defined as a quantization map determined with respect to the data to be quantized. A *data-agnostic* quantization scheme has a mapping determined a priori. For a uniform quantization scheme a data-dependent approach may be to set the range to be the (min, max) values of the data, or be a clipped range incorporating distributional information such as standard deviations of the data [15, 13]. Figure 2 shows the difference between these approaches. A *fixed* quantization scheme uses the same quantization scheme for every epoch t . That is, $Q_t = Q_{t+1}; \forall t$. An *adaptive* quantization scheme allows Q_t to vary over the course of training.

The *quantization error* of a scalar x is given by [13]:

$$\epsilon = x - Q(x). \quad (1)$$

For a matrix of values we can use the L2-norm of quantization errors as a distance metric:

$$e = \|X - Q(X)\|_2^2. \quad (2)$$

For an L-layer perceptron we can define the *total layer-wise quantization error* (TLQE) to be given by:

$$TLQE = \sum_{l \in L} \|W_l - Q_l(W_l)\|_2^2, \quad (3)$$

where W_l refers to the full precision weights at layer l and Q_l is the quantization mapping applied to layer l .

2.1.1 Fixed and Adaptive Quantization

Consider Q_t as depending on the weights at a given epoch t . Under a fixed data-dependent cluster quantization, we have Q_t defined as a mapping that minimises the distance between quantized and unquantized values at epoch t :

$$Q_t := \underset{Q}{\operatorname{argmin}} \|W_t - Q(W_t)\|_2^2. \quad (4)$$

Note that typically the distribution of weight values changes over training. Consider applying the same Q_t to a matrix of weights at time t' after several epochs of training has occurred. By assumption from Equation 4, we have that Q_t is the optimal mapping for epoch t and $Q_{t'}$ the optimal mapping for epoch t' . As a result, we know that the quantization error that occurs from using Q_t at t' is greater than or equal to that of applying an optimal $Q_{t'}$:

$$\|W_{t'} - Q_t(W_{t'})\|_2^2 \geq \|W_{t'} - Q_{t'}(W_{t'})\|_2^2. \quad (5)$$

This motivates the use of an adaptive quantization scheme. As repartitioning with a K-means algorithm is computationally costly it is possible to set an adaptive quantization rule

based on an interval of epochs, or based on the quantization error such that repartition occurs if for a chosen δ :

$$\|W_{t'} - Q_t(W_{t'})\|_2^2 \geq \|W_t - Q_t(W_t)\|_2^2 + \delta. \quad (6)$$

In practice, we find it sufficient to repartition periodically based only on the number of epochs.

2.2. Implicit Neural Representations (INR)

An INR is a function mapping a coordinate input vector \mathbf{x} to an output feature vector \mathbf{y} parameterised by neural network weights [38, 9, 40, 47] as

$$f_\theta(\mathbf{x}) \rightarrow \mathbf{y}. \quad (7)$$

Examples of INRs include coordinate networks [32, 31], NeRF and its many variants [45, 47, 26, 29], audio [38, 10], video [3, 50, 21], topological representations [48], light-field representations [11], implicit geometry [8, 28], novel view synthesis, volumetric scalar fields [24], and gigapixel image fitting [25]. A special case of INRs are image regression problems [38, 40, 9]. An image regression learns a representation function mapping: $f_\theta(x, y) \rightarrow (r, g, b)$ for a single image. The network weights θ provide an encoding that predicts the approximate pixel value for a given coordinate. A forward pass across the original set of coordinates approximately reconstructs the original image. A sufficiently small or quantized network can therefore be treated as a form of lossy image compression [40, 9, 10].

2.3. Neural Radiance Fields (NeRF)

A NeRF is an implicit neural representation of the form

$$f_\theta(x, y, z, \theta, \phi) \rightarrow (r, g, b, \sigma), \quad (8)$$

where spatial location is provided by the coordinates (x, y, z) and viewing direction (θ, ϕ) [27]. When trained on a set of camera poses for a given scene, the implicit representation enables the interpolation and generation of novel pose estimates. A wide number of technical variations have extended the original model for purposes of improved fidelity or rendering speed, such as [34], [23] and [5]. While the implicit nature of NeRF is itself a compression form, only a few works have investigated further compressing this model. Methods have included a mixture of rank-residual decomposition, quantization, entropy-penalisation, pruning, and distillation techniques [18, 2, 43, 44, 37].

2.4. Related Works

2.4.1 Quantization in Deep Learning

How to quantize a signal to preserve its relevant information content has been of fundamental interest in signal processing, information theory, compression, and other fields since

at least the time of Shannon [36]. Within image processing, quantization is a fundamental element of image compression algorithms such as JPEG [35, 42]. While the term quantization can refer to the discretisation of any continuous signal, a distinction should be made to its most common usage in computer vision: the reduced-precision quantization involved in representing a floating point value in a reduced number of bits [7, 41]. Within deep learning, reduced-precision quantization has been widely applied to the *weight* and/or *activation* values in a deep feedforward network [16, 33, 14, 30, 19]. As deep learning libraries such as PyTorch and Tensorflow typically represent weights in 32-bit or 64-bit format, a lower-precision quantization can lead to memory and speed improvements. At the extreme case, this involves *binary* neural networks whose weights are quantized to take binary values of $\{-1, 1\}$ [6]. This was examined in Rastegari *et al.* [33] which introduces a method of training full-precision networks which is robust to a quantization transformation, known as *quantization-aware training*. A variety of quantization methods for deep neural networks including non-uniform mappings, global and layer-wise mappings, integer quantization [19], and mixed-precision quantization are also widely described in the literature [16, 14].

2.4.2 Quantizing Implicit Neural Representations

Within the wider INR literature, several works have examined the use of quantization for compression. Works related to ours are Dupont *et al.* [9] which applied quantization and architecture search to compress images, Strumpler *et al.* [40] which applied quantization-aware training and entropy coding, and Chiarlo [4], which looked at compression methods for INRs (distillation, pruning, quantization, and quantization-aware training). Dupont *et al.* [10] showed impressive results compressing INRs over multiple modalities through the use of quantized weight modulations. Each of these papers apply uniform rather than cluster quantization. Furthermore each of recent [40], [10], and [22] apply a meta-learned preinitialisation (MAML) to improve rate-distortion performance and model performance. They require an additional dataset for the preinitialisation; in contrast we learn on single signal instances to directly compare quantization methods.

In terms of works employing cluster quantization to implicit neural representations, we find Lu *et al.* [24], Takikawa *et al.* [43], and Shi *et al.* [37] to be the most similar to our method. Lu *et al.* [24] applied a K-means clustering to the weights of each layer of a network to compress volumetric scalar fields. We differ from their work by investigating NeRF and 2D image compression; and in addition employ a quantization-aware training and entropy compression. The second is the recently released Takikawa *et al.*

[43], who applied a learned quantization to feature grids to compress NeRF and signed-distance fields. While they used K-means clustering as a post-processing benchmark comparison, they did not use it as part of quantization-aware training as in our method. More recently, Shi *et al.* [37] applied a low-rank decomposition and distillation to a trained model based on the original NeRF (8 layers, 256 channels), before a global K-means quantization map is used iteratively to quantize each layer with other layers then retrained. In contrast, we apply layer-wise quantization-aware training throughout our entire training procedure. Furthermore, our work directly investigates the impact of quantization methods, network architectures, and quantization levels have on compression across different modalities.

3. Method

For a loss function L , number of hidden layers h , number of hidden units per layer w , quantization function Q , bits-per-weight k , compression function C and target memory constraint D , we can view the compression of implicit neural representations through as a constrained optimization across architectures and quantization levels as

$$\begin{aligned} \min \quad & L(\cdot) \\ \text{s.t.} \quad & C(Q, w, h, k) \leq D. \end{aligned} \quad (9)$$

3.1. Quantization Methods

Our experiments are performed on 4 quantization methods (*Explicit [-1, 1]*, *Distributional*, *Minmax*, and *K-means*). The first three are uniform quantizations with different ranges (see Figure 2): an explicit range between $[-1, 1]$; a *distributional* range calculated as a function of the standard deviation of the weight distribution; and a *Minmax* quantization range determined by the minimum and maximum value of the distribution. These are compared with a clustered calculated using the K-means algorithm.

The Explicit $[-1, 1]$ uniform quantization is calculated using a k -bit formula found in Rastegari *et al.* [33]:

$$q_k(x) = 2\left(\frac{\text{round}((2^k - 1)\left(\frac{x+1}{2}\right))}{2^k - 1} - \frac{1}{2}\right), \quad (10)$$

where $x \in [-1, 1]$. The Distributional quantization uniformly quantizes within d standard deviations of the weight distribution, where d is calculated according to a k -bit formula found in Dupont *et al.* [10]:

$$d = 3 + \frac{3(k-1)}{15}. \quad (11)$$

3.2. Quantization-Aware Training (QAT)

We employ the quantization-aware training introduced by Rastegari *et al.* [33]. The algorithm is briefly recalled

Algorithm 1 Adaptive Cluster Quantization

Input: bits-per-weight k , model, repartition_epoch

```

1: for epoch 1 to  $N$  do
2:   Train model (quantization-aware training)
3:   if repartition_epoch then
4:     for  $l$  in layers do
5:       Recalculate quantization map (1D K-means)
6:     end for
7:   end if
8: end for
9: Convert Quantized Model to Codebook
10: BZIP2 Codebook Model + Cluster Dictionary

```

in Figure 3. This procedure trains the model robustly to quantization through the use of a straight-through estimator [1, 33]. We adapt this procedure to include *periodic repartitioning* in which the quantization mappings for the Distributional, Minmax, and K-means are recalculated periodically over training. The Explicit $[-1, 1]$ quantization is fixed for all epochs. Repartitioning every F epochs in a N epoch training cycle introduces a computational overhead of $\frac{NG(q,w,h,k)}{F}$, where G is a cost of repartitioning which depends on the network architecture (w, h) , quantization function (q) , and bits-per-weight (k) . Periodic repartitioning therefore strikes a balance between accumulating quantization error and the introduced computational overhead of recomputing partitions as weight distributions change over training; see Figure 4 and Sec. 5. Following training, the quantized weights are converted to an integer representation (a dictionary mapping of the quantized float and the integer) as [40]. Both the integer representation and the mapping are then compressed using BZIP2, an entropy encoding compression library. Algorithm 1 describes our approach.

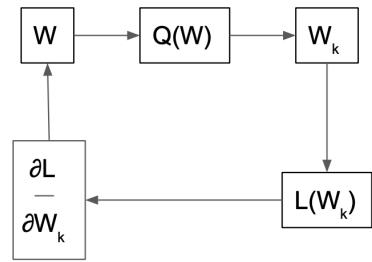


Figure 3. QAT Per epoch training cycle: The current full-precision weight matrix W is quantized $Q(W) = W_k$. The loss function $L(W_k)$ is calculated for the input data, and the error derivatives $\frac{\partial L(W_k)}{\partial W_k}$ calculated. The full-precision weight matrix W is then updated using backpropagation.

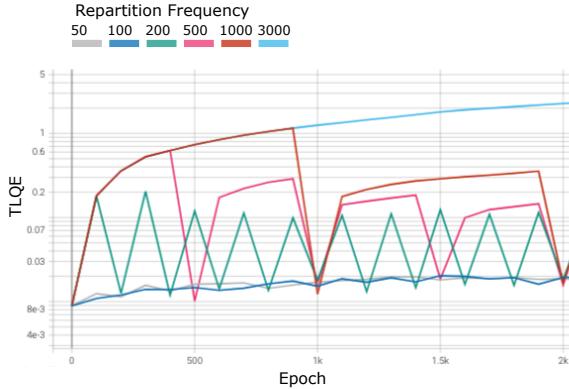


Figure 4. Effect of repartitioning on Total Layerwise Quantization Error (TLQE) [log scale]. Recalculating partitions reduces quantization error at the cost of increased computation. Architecture: 1 hidden layer, 18 neurons, 5-bit weights, K-means quantization.

3.3. Implementation Details

Training was conducted with the Adam optimizer with the hyperparameters $1e^{-4}, \beta = (0.99, 0.999)$ and weight decay $= 1e^{-8}$ [20]. For image regression we use a sine activation with frequency 30 as described by [9, 40], but compare with Gaussian and ReLU activations as supplemental ablation [9, 38]. For our 2D image regression experiments, we experimented with both the MSE loss and its negative base-10 logarithm (i.e. an unscaled peak signal-to-noise ratio (PSNR)). For the NeRF experiments, we use the MSE loss [27]; see Sec. 5 for more details. Our experiments are evaluated quantitatively using standard perceptual metrics, including the PSNR, the structural similarity index measure (SSIM), and LPIPS Alex and VGG (two learned perceptual metrics) [46, 49, 35, 42, 13]. We additionally evaluate the *gradient PSNR*, which we define by

$$PSNR_{\Delta} = -\log_{10}(\|\Delta(f_{\theta}(x, y)) - \Delta(X)\|_2^2), \quad (12)$$

where $\Delta(\cdot)$ is an approximation of the image gradient generated through the Sobel operator [42]. The gradient PSNR is used to determine the quality of preserved image gradients, as these are often important for downstream tasks such as image classification and segmentation [42].

4. Results

In this section we present our comparison of different quantization schemes. First, we demonstrate a performance difference between cluster and uniform quantization for image regression of the CIFAR10 Dataset. This is followed by a deeper architectural analysis on an instance of the DIV2K Dataset. As an example application of the analysis we apply our method to the more complicated modality of NeRF.

4.1. CIFAR10

Experiments for 2D image reconstruction were conducted on the CIFAR10 dataset. We use a base network of 1 hidden layer, sine activation, 20 neuron layers replicating the architecture used for COIN compression experiments in [10]. Repartitioning was applied at every epoch. As can be seen in Figure 5, the K-means cluster quantization has improved reconstruction relative to the other methods as evaluated across perceptual metrics. Interestingly, it is found to have substantially higher accuracy according to the Gradient PSNR even at high bits-per-weight. Comparing the uniform approaches, Distributional and Minmax quantization gives broadly similar results. Quantizing uniformly between a fixed $[-1, 1]$ gives the worst performance with little signal reconstructed at low bits-per-weight.

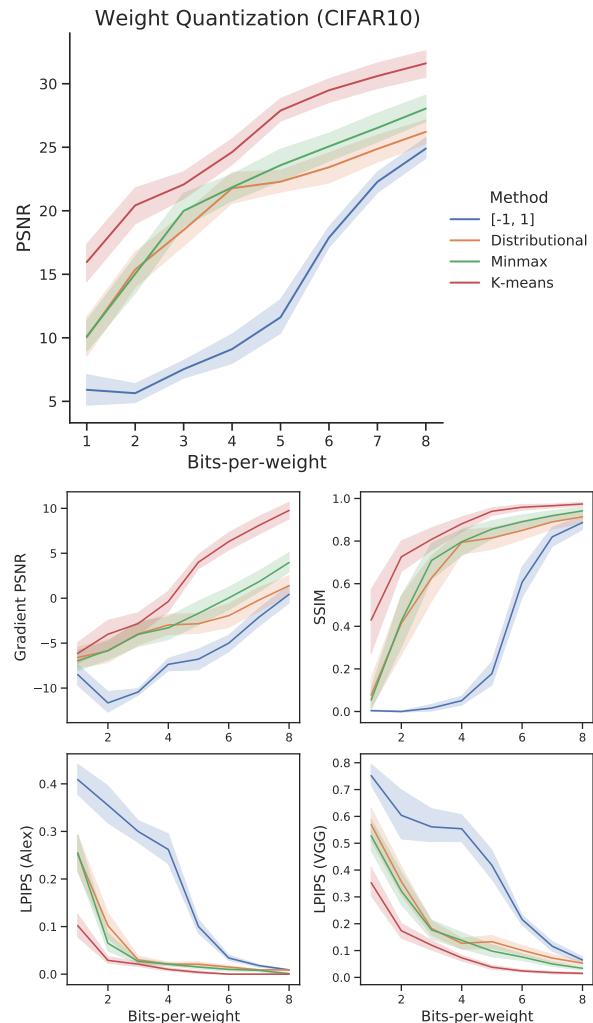


Figure 5. Perceptual evaluation (CIFAR10)

4.2. DIV2K

We conducted the experiments for 2D image reconstructions on the DIV2K image test suite, which consists images with at least 2000 pixels on at least one axis. Figure 7 compares the quantization methods at different architectures and rates of quantization. Repartitioning was applied every 50 epochs. As expected, the K-means quantization outperforms the uniform quantization methods at low bit-rates. At 8 bits-per-weight the difference between K-means, Distributional, and Minmax quantization is negligible. This is consistent with the well-known result that high resolution uniform scalar quantizers closely approximate an optimal quantization (to within 2.82dB for a Gaussian source), a result first noted by Koshelev in 1963 and since rediscovered by multiple authors [15, 13].

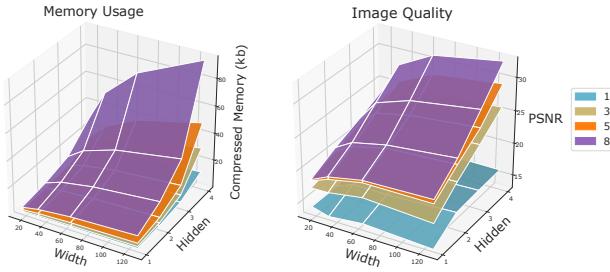


Figure 6. Architecture search over the number of hidden layers, and the number of units per hidden layer. The colours correspond to different quantization bits-per-weight (1, 3, 5, 8) using K-means quantization. Div2K index 3, evaluated at 3000 epochs.

4.3. NeRF Experiments

For evaluations on neural radiance fields, we used a 4-layer NeRF [27] with 64 hidden units per layer without hierarchical sampling trained for 200,000 epochs with repartitioning every 100 epochs. The highest performing epoch was selected for evaluation. A ReLU activation with positional encoding is used as per [27]. Note that our analysis is agnostic to activations; see supplemental ablation. Initial experiments were performed on the LLFF ‘flower’ instance with bits-per-weight (1, 3, 5, 8). Following this an evaluation was conducted on the full LLFF and Blender Datasets using with weights quantized to 3 bits-per-weight.

LLFF Dataset Figure 9 shows our results in compressing the NeRF on a ‘flower’ instance (LLFF) using uniform and cluster quantization. Significant compression of NeRF is observed without a catastrophic degradation of accuracy (e.g. 20.77 PSNR with a model size of 25.6kb). In terms of memory usage, this compares favourably to that obtained by both the original NeRF (27.42 PSNR, 5169kb) and cNeRF (27.39 PSNR at 938kb) [27, 2]. Furthermore, we note

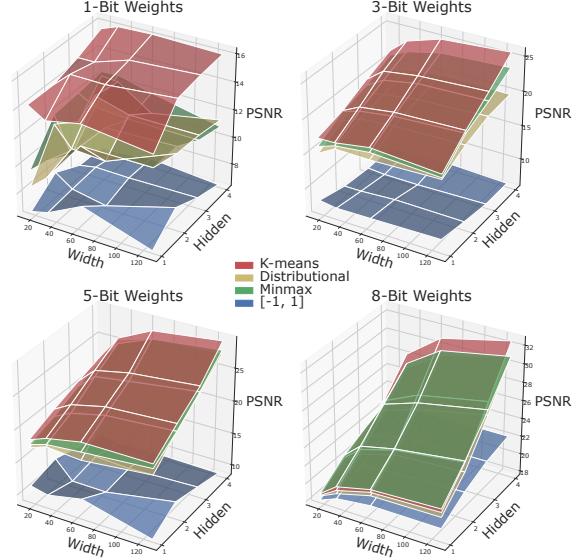


Figure 7. Comparison between quantization methods under different architectures at different rates on DIV2K index 3. The clustered quantization outperforms uniform methods at low bits-per-weight and the difference becomes marginal at higher resolutions.

the improvement of K-means quantization at low bits-per-weight. At 3 bits-per-weight K-means quantization obtains a test PSNR of 18.93 with an architecture of 4-layers 64 neurons, compared to 17.82 under Explicit [-1,1] quantization, 17.56 under Minmax quantization, and 18.02 under a Distributional quantization. Moreover K-means quantization enables the signal to be obtained under 1-bit quantization (16.21 PSNR); a restriction that causes the signal to collapse under both Explicit and Distributional quantization. At higher bits-per-weight this benefit is diminished. This is expected and consistent with quantization theory, as increasing partitions reduces the uncovered distributional support [13, 15]. Architecture choice has a large impact on memory footprint (e.g. increasing hidden layer neurons to 128 approximately triples the used memory from 25.6kb to 76.9kb), with a positive but marginal improvement in PSNR (20.77 to 21.61; see Supplementary Materials).

Blender Dataset Table 1 shows comparison of the evaluated methods on Blender test NeRF instances. Results show clear improvement on perceptual metrics for K-means quantization. The size of these models following compression with BZIP2 is noticeably larger for the K-means quantization. As BZIP2 compresses most effectively for repeated information, it is possible that the distribution of quantized weights is more uniform under the K-means clustering (and is therefore being used more completely). Note that the compressed size obtained under this process is very small, with the NeRF model compressed to a size of approx-

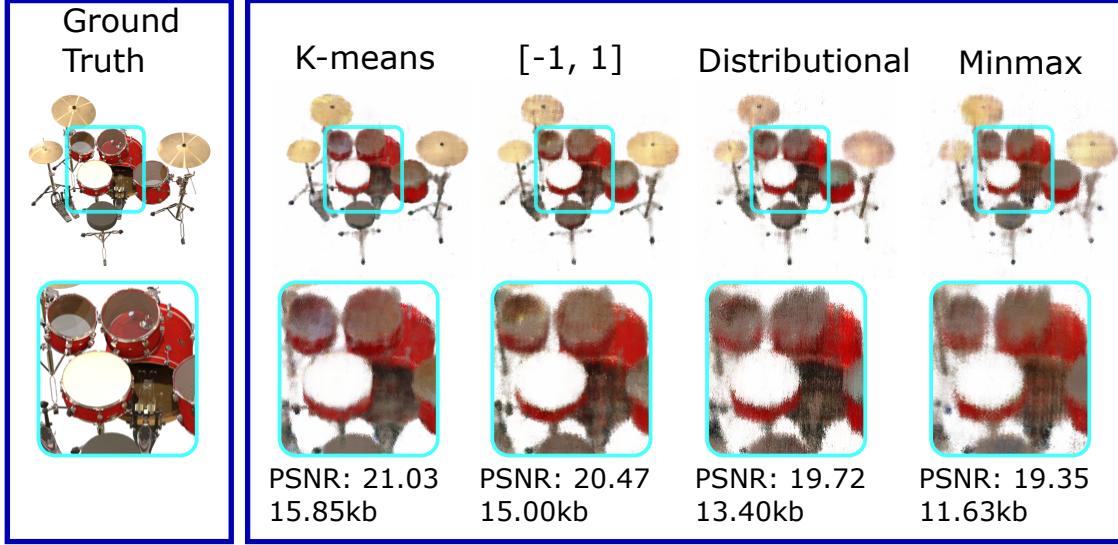


Figure 8. Synthetic NeRF Qualitative Result on “drum” instance. Less distortion is visible under the K-means quantization. Each model is less than 16kb (4 hidden layers of 64 neurons, with 3-bit quantization).

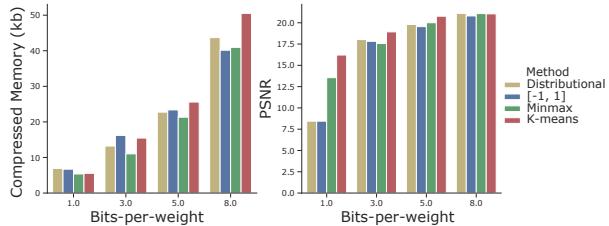


Figure 9. Comparison of PSNR and Compressed Memory Size for NeRF Flower images. Architecture (4-layers, 64 neuron layers).

imately 16kb. The qualitative evaluation shows that even under this extreme compression a clear signal is able to be reconstructed (see Figure 8).

5. Discussion and Limitations

Periodic Repartitioning While recalculating partitions more frequently reduces quantization error (see Figure 4), this overhead may make frequent repartitioning using K-means impractical for large architectures with high quantization resolution. The computational complexity of the implemented K-means algorithm is $O(mn + n \log n)$ where m is the number of partitions and n is the data points [39]. For our case this depends on the bits-per-weight k , (as $m = 2^k$); the number of neurons at each layer w_{in}, w_{out} (as $n = w_{in} * w_{out}$); and the number of layers. As such, the clustering operation scales inefficiently with large architecture sizes and bits-per-weight. In practice, we find that repartitioning every 50 to 100 epochs to balance this cost in our experiments to a manageable overhead. To a lesser extent, the overhead for Distributional and Minmax is also affected by repartitioning frequency. While a balance be-

tween the increased computation cost and perceptual benefit of more frequent repartitioning is a subjective consideration for the experimenter, we note as a societal impact the high global energy consumption in machine learning [12].

Network Capacity and Quantization Trade-Off One interesting experimental observation is an apparent trade-off between the network capacity and quantization. In Figure 6, we show the effect of weight quantization at different bit rates using K-means quantization. By considering the level-sets of the induced PSNR, we note that multiple configurations of network width, depth, and quantization level can lead to the same PSNR. The architectures differ however on the memory consumption, with exponential memory increase in the number of layers. A consequence of this is architectural mitigation of reconstruction failure even with extreme weight quantization. Figure 10 shows this visually on CIFAR: a) shows a network of 3 hidden layers, 256 neurons per layer, 1-bit weights (PSNR: 22.21, 31.6kb); b) 3 hidden layers 512 neurons, 1-bit weights (PSNR: 33.61; 124.41kb). As compression, this is not very useful: the quantized network in b) is approximately 40x the original CIFAR image - however it clearly demonstrates the performance trade-off between weight quantization and neural architecture. As comparison, we note that improved memory efficiency can be obtained with a higher number of bits-per-weight as shown in: c) where we have 2 hidden layers, 20 neurons, quantized to 5-bit weights (PSNR: 29.69, 3.11kb) due to the architecture memory cost.

Future Work Further improvements can additionally be applied to our chosen quantization schemes. In particular,

Method	PSNR	SSIM	LPIPS	Size (kb)
Chair				
K-means	29.78	0.95	0.08	15.58
Distributional	26.76	0.91	0.16	13.45
Minmax	27.69	0.93	0.14	10.95
Explicit [-1, 1]	28.40	0.94	0.12	14.49
Drums				
K-means	21.03	0.81	0.25	15.85
Distributional	19.72	0.72	0.41	13.40
Minmax	19.35	0.71	0.42	11.63
Explicit [-1, 1]	20.47	0.77	0.32	15.00
Ficus				
K-means	23.85	0.89	0.12	16.17
Distributional	23.03	0.86	0.22	13.42
Minmax	22.71	0.85	0.25	11.85
Explicit [-1, 1]	23.39	0.88	0.16	14.10
Hotdog				
K-means	27.52	0.88	0.19	15.52
Distributional	25.04	0.80	0.36	13.24
Minmax	25.60	0.83	0.29	11.51
Explicit [-1, 1]	26.07	0.83	0.30	14.83
Lego				
K-means	22.88	0.82	0.16	15.69
Distributional	21.37	0.77	0.27	13.33
Minmax	20.79	0.76	0.28	10.92
Explicit [-1, 1]	21.89	0.79	0.21	14.89
Materials				
K-means	21.51	0.82	0.23	15.36
Distributional	19.88	0.76	0.37	13.22
Minmax	19.76	0.75	0.37	10.64
Explicit [-1, 1]	20.77	0.78	0.34	14.03
Mic				
K-means	26.07	0.93	0.13	16.02
Distributional	23.68	0.89	0.24	13.43
Minmax	23.09	0.89	0.24	11.30
Explicit [-1, 1]	24.96	0.92	0.18	14.44
Ship				
K-means	24.49	0.69	0.37	15.87
Distributional	23.45	0.65	0.46	13.35
Minmax	23.48	0.66	0.45	11.08
Explicit [-1, 1]	23.93	0.66	0.44	14.52

Table 1. Quantitative Results on Blender (NeRF), Architecture: 4 hidden layers, 64 units per layer, 3-bit quantization). Metrics averaged over sampled 2D views of the reconstruction.

the quantization methods employed are *deterministic* which can introduce patterns and artefacts in the quantized mapping; the well-known method of *dithering* is one way to avoid this issue [13, 35]. To a certain extent recalculating centroids introduces a source of stochasticity which may help obviate this issue, but a formal evaluation remains an area for future work. As a side note, we find that using

a log L2 loss function (i.e. directly optimising for PSNR) produced higher accuracy reconstructions at a given training epoch than the standard L2 loss function more common in the literature (see supplementary material), a result that held consistently across ablations of network architecture and activation function. We note that the logarithm is a monotonic operation, and therefore does not change the theoretical minimum of the converged network. As a result, it is interesting to see that this modification led to consistent improvements in the 2D evaluation. This result did not hold in general for other more expressive experiments, such as NeRF. Formal investigation to the cause of this observation remains a potential avenue of enquiry.

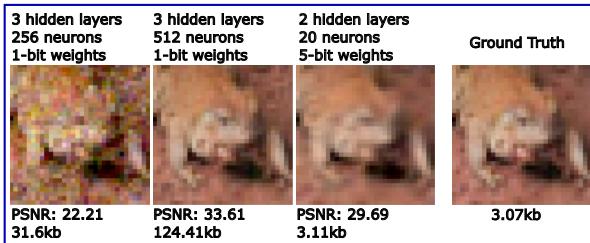


Figure 10. Trading network capacity for quantization levels. While it is possible to use K-means weight quantization to enable 1-bit reconstructions with sufficient network capacity, this comes at a trade-off for memory efficiency.

6. Conclusion

We investigated the use of non-uniform quantization for the compression of implicit neural functions. By accounting for the weight distribution of the neural network layers, we are able to achieve higher performance at lower bits-per-weight than under uniform quantization approaches. We have additionally shown that there exists a trade-off between the network capacity and the weight quantization levels, with an extreme (binary) quantization able to be compensated with sufficient network capacity for simple experiments. Our method enables compression of neural radiance fields to a large degree compared to the original NeRF model. As the strategy involves a modification of the quantization strategy employed to the neural weights, it is possible that this may also be applied to other large implicit neural representations. Of particular interest is the potential to apply it to high resolution methods such as kiloNeRF, which have a large memory footprint of 100MB or SNeRG with 90MB [34, 17]. As these methods are optimised for inference in real-time this would be a step towards lightweight real-time NeRF models.

Acknowledgements

We thank Dr. Sameera Ramasinghe and Dr. Hemanth Saratchandran for their valuable discussions on this work.

References

- [1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. Technical Report arXiv:1308.3432, arXiv, Aug. 2013. arXiv:1308.3432 [cs] type: article.
- [2] Thomas Bird, Johannes Ballé, Saurabh Singh, and Philip A. Chou. 3D Scene Compression through Entropy Penalized Neural Representation Functions. Technical Report arXiv:2104.12456, arXiv, Apr. 2021. arXiv:2104.12456 [cs, eess] type: article.
- [3] Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shrivastava. NeRV: Neural Representations for Videos. In *Advances in Neural Information Processing Systems*, volume 34, pages 21557–21568. Curran Associates, Inc., 2021.
- [4] Francesco Maria Chiarlo. Implicit Neural Representations for Image Compression. Publisher: Politecnico di Torino, July 2021.
- [5] Shin-Fang Chng, Sameera Ramasinghe, Jamie Sherrah, and Simon Lucey. GARF: Gaussian Activated Radiance Fields for High Fidelity Reconstruction and Pose Estimation. Technical Report arXiv:2204.05735, arXiv, Apr. 2022. arXiv:2204.05735 [cs] type: article.
- [6] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. Technical Report arXiv:1602.02830, arXiv, Mar. 2016. arXiv:1602.02830 [cs] type: article.
- [7] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006.
- [8] François Darmon, Bénédicte Basle, Jean-Clément Devaux, Pascal Monasse, and Mathieu Aubry. Improving neural implicit surfaces geometry with patch warping. arXiv:2112.09648 [cs], Dec. 2021. arXiv: 2112.09648.
- [9] Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. COIN: COmpression with Implicit Neural representations. arXiv:2103.03123, 2021.
- [10] Emilien Dupont, Hrushikesh Loya, Milad Alizadeh, Adam Goliński, Yee Whye Teh, and Arnaud Doucet. COIN++: Data Agnostic Neural Compression. arXiv:2201.12904, 2022.
- [11] Brandon Yushan Feng and Amitabh Varshney. SIGNET: Efficient Neural Representation for Light Fields. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14204–14213, Montreal, QC, Canada, Oct. 2021. IEEE.
- [12] Eva García-Martín, Crefeda Faviola Rodrigues, Graham Riley, and Håkan Grahn. Estimation of energy consumption in machine learning. *Journal of Parallel and Distributed Computing*, 134:75–88, Dec. 2019.
- [13] Allen Gersho and Robert Gray. *Vector Quantization and Signal Compression*. The Kluwer International Series In Engineering And Computer Science. Kluwer International Publishers, 1992.
- [14] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A Survey of Quantization Methods for Efficient Neural Network Inference. arXiv:2103.13630 [cs], June 2021. arXiv: 2103.13630.
- [15] R.M. Gray and D.L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, Oct. 1998. Conference Name: IEEE Transactions on Information Theory.
- [16] Song Han, Huizi Mao, and William J. Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. Technical Report arXiv:1510.00149, arXiv, Feb. 2016. arXiv:1510.00149 [cs] type: article.
- [17] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5875–5884, 2021.
- [18] Berivan Isik. Neural 3D Scene Compression via Model Compression. Technical Report arXiv:2105.03120, arXiv, May 2021. arXiv:2105.03120 [cs] type: article.
- [19] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. arXiv:1712.05877 [cs, stat], Dec. 2017. arXiv: 1712.05877.
- [20] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2014. _eprint: 1412.6980.
- [21] Hoang Le, Liang Zhang, Amir Said, Guillaume Sautiere, Yang Yang, Pranav Shrestha, Fei Yin, Reza Pourreza, and Auke Wiggers. MobileCodec: Neural Inter-frame Video Compression on Mobile Devices. Technical Report arXiv:2207.08338, arXiv, July 2022. arXiv:2207.08338 [cs, eess] type: article.
- [22] Jaeho Lee, Jihoon Tack, Namhoon Lee, and Jinwoo Shin. Meta-Learning Sparse Implicit Neural Representations. In *Advances in Neural Information Processing Systems*, volume 34, pages 11769–11780. Curran Associates, Inc., 2021.
- [23] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. BARF: Bundle-Adjusting Neural Radiance Fields. Technical Report arXiv:2104.06405, arXiv, Aug. 2021. arXiv:2104.06405 [cs] type: article.
- [24] Yuzhe Lu, Kairong Jiang, Joshua A. Levine, and Matthew Berger. Compressive Neural Representations of Volumetric Scalar Fields. Technical Report arXiv:2104.04523, arXiv, Apr. 2021. arXiv:2104.04523 [cs] type: article.
- [25] Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. ACORN: Adaptive Coordinate Networks for Neural Scene Representation. arXiv:2105.02788 [cs], May 2021. arXiv: 2105.02788.
- [26] Fabian Mentzer, George Toderici, Michael Tschanne, and Eirikur Agustsson. High-Fidelity Generative Image Compression, 2020. _eprint: 2006.09965.
- [27] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, 2020.

[28] David Palmer, Dmitriy Smirnov, Stephanie Wang, Albert Chern, and Justin Solomon. DeepCurrents: Learning Implicit Representations of Shapes with Boundaries. *arXiv:2111.09383 [cs]*, Mar. 2022. arXiv: 2111.09383.

[29] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable Neural Radiance Fields. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5845–5854, Montreal, QC, Canada, Oct. 2021. IEEE.

[30] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. Binary neural networks: A survey. *Pattern Recognition*, 105:107281, Sept. 2020.

[31] Sameera Ramasinghe and Simon Lucey. Learning Positional Embeddings for Coordinate-MLPs. *arXiv:2112.11577 [cs]*, Dec. 2021. arXiv: 2112.11577.

[32] Sameera Ramasinghe and Simon Lucey. Beyond Periodicity: Towards a Unifying Framework for Activations in Coordinate-MLPs. *arXiv:2111.15135 [cs]*, Mar. 2022. arXiv: 2111.15135.

[33] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. *arXiv:1603.05279 [cs]*, Aug. 2016. arXiv: 1603.05279.

[34] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. Technical Report *arXiv:2103.13744*, arXiv, Aug. 2021. arXiv:2103.13744 [cs] type: article.

[35] David Salomon. *Data Compression: The Complete Reference*. pub-SV, pub-SV:adr, 2007.

[36] C.E. Shannon. Communication in the Presence of Noise. *Proceedings of the IRE*, 37(1):10–21, Jan. 1949. Conference Name: Proceedings of the IRE.

[37] Jinglei Shi and Christine Guillemot. Distilled Low Rank Neural Radiance Field with Quantization for Light Field Compression. Technical Report *arXiv:2208.00164*, arXiv, July 2022. arXiv:2208.00164 [cs] type: article.

[38] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7462–7473. Curran Associates, Inc., 2020.

[39] Daniel Steinberg. kmeans1d: A Python package for optimal 1D k-means clustering.

[40] Yannick Strümpler, Janis Postels, Ren Yang, Luc van Gool, and Federico Tombari. Implicit Neural Representations for Image Compression. *arXiv:2112.04267*, 2021.

[41] Richard Szeliski. *Computer Vision: Algorithms and Applications*, volume 5. Jan. 2021. Publication Title: Computer Vision: Algorithms and Applications, Texts in Computer Science, ISBN 978-1-84882-934-3. Springer-Verlag London Limited, 2011.

[42] Richard Szeliski. *Computer Vision - Algorithms and Applications, Second Edition*. Texts in Computer Science. Springer, 2022.

[43] Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. Variable Bitrate Neural Fields. Technical Report *arXiv:2206.07707*, arXiv, June 2022. arXiv:2206.07707 [cs] type: article.

[44] Jiaxiang Tang, Xiaokang Chen, Jingbo Wang, and Gang Zeng. Compressible-composable NeRF via Rank-residual Decomposition. Technical Report *arXiv:2205.14870*, arXiv, May 2022. arXiv:2205.14870 [cs] type: article.

[45] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Niessner, Jonathan T. Barron, Gordon Wetzstein, Michael Zollhoefer, and Vladislav Golyanik. Advances in Neural Rendering. Technical Report *arXiv:2111.05849*, arXiv, Mar. 2022. arXiv:2111.05849 [cs] type: article.

[46] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[47] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural Fields in Visual Computing and Beyond. Technical Report *arXiv:2111.11426*, arXiv, Apr. 2022. arXiv:2111.11426 [cs] type: article.

[48] Jonas Zehnder, Yue Li, Stelian Coros, and Bernhard Thomaszewski. NTopo: Mesh-free Topology Optimization using Implicit Neural Representations. In *Advances in Neural Information Processing Systems*, volume 34, pages 10368–10381. Curran Associates, Inc., 2021.

[49] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*, 2018.

[50] Yunfan Zhang, Ties van Rozendaal, Johann Brehmer, Markus Nagel, and Taco Cohen. Implicit Neural Video Compression. Technical Report *arXiv:2112.11312*, arXiv, Dec. 2021. arXiv:2112.11312 [cs] type: article.