

1. Supplementary Materials

1.1. Activation Ablation

Experiments for on the CIFAR10 and DIV2K datasets were performed using a sine activation function with frequency 30. Motivated by recent work on Gaussian activations we additionally explored the use of alternative activations. The improved performance for sine activations for the tested specifications led to its inclusion in our base experiments. Our NeRF experiments were performed exclusively with the ReLU + positional encoding as per the original NeRF paper.

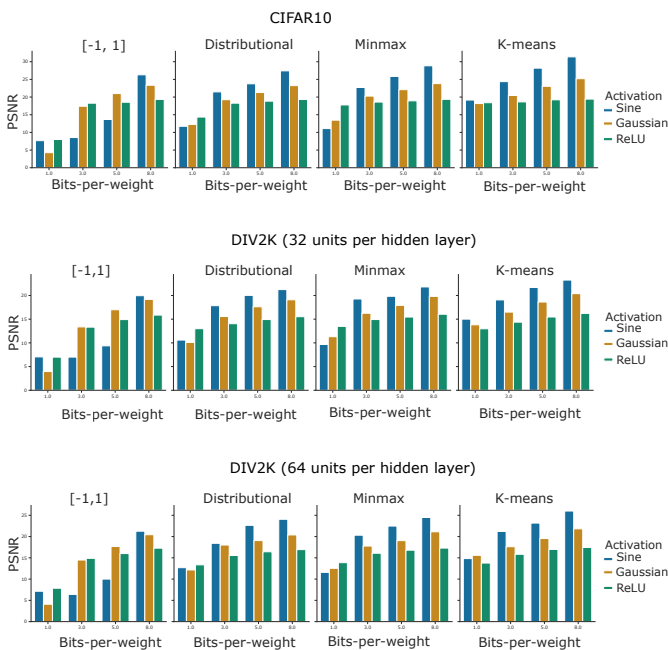


Figure 1: Activation ablation for CIFAR (top), DIV2K with 32 unit neurons (middle), DIV2K with 64 unit neurons (bottom). The sine activation shows improved performance over both a ReLU activation and Gaussian activation for Distributional, Minmax, and K-means quantization. ReLU and Gaussian activations do show some performance at low bit-rates for Explicit [-1,1] quantization, however qualitative evaluation shows little signal is obtained for these instances (the PSNR instead reflecting a colour gradient but no finer details until around 6 bits-per-weight).

1.2. SSIM Evaluation

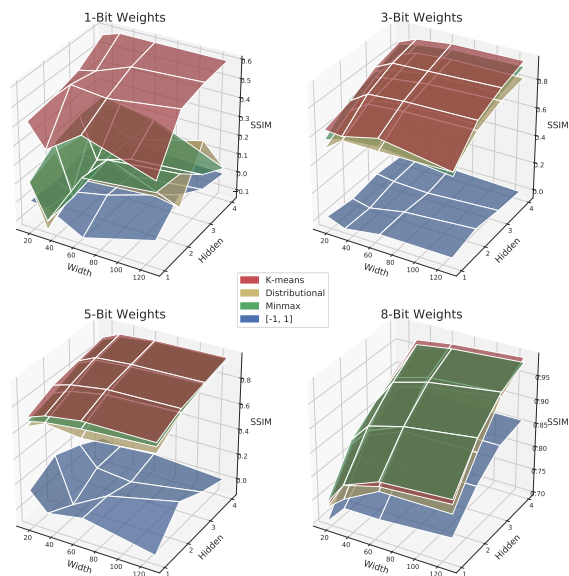


Figure 2: SSIM evaluation for the DIV2K experiments. Results are consistent with the PSNR evaluation presented in Figure 7, with K-means quantization demonstrating higher performance than uniform methods at low bits-per-weight.

1.3. Post-Training Quantization Comparison

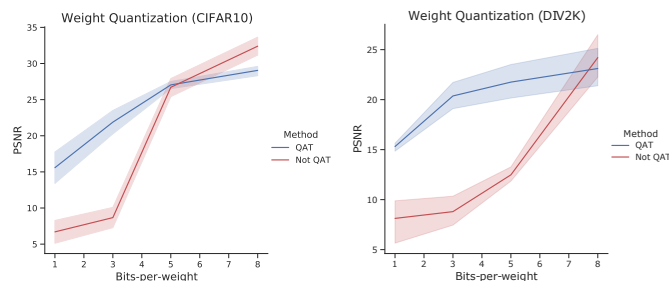


Figure 3: Comparison of quantization aware training (QAT) and post-training quantization. K-means quantization used in both experiments. CIFAR: 1 hidden layer, 20 hidden units. DIV2K: 2 hidden layers, 64 hidden units. QAT outperforms post-training quantization at low bits-per-weight.

1.4. Model and Dictionary Memory Usage

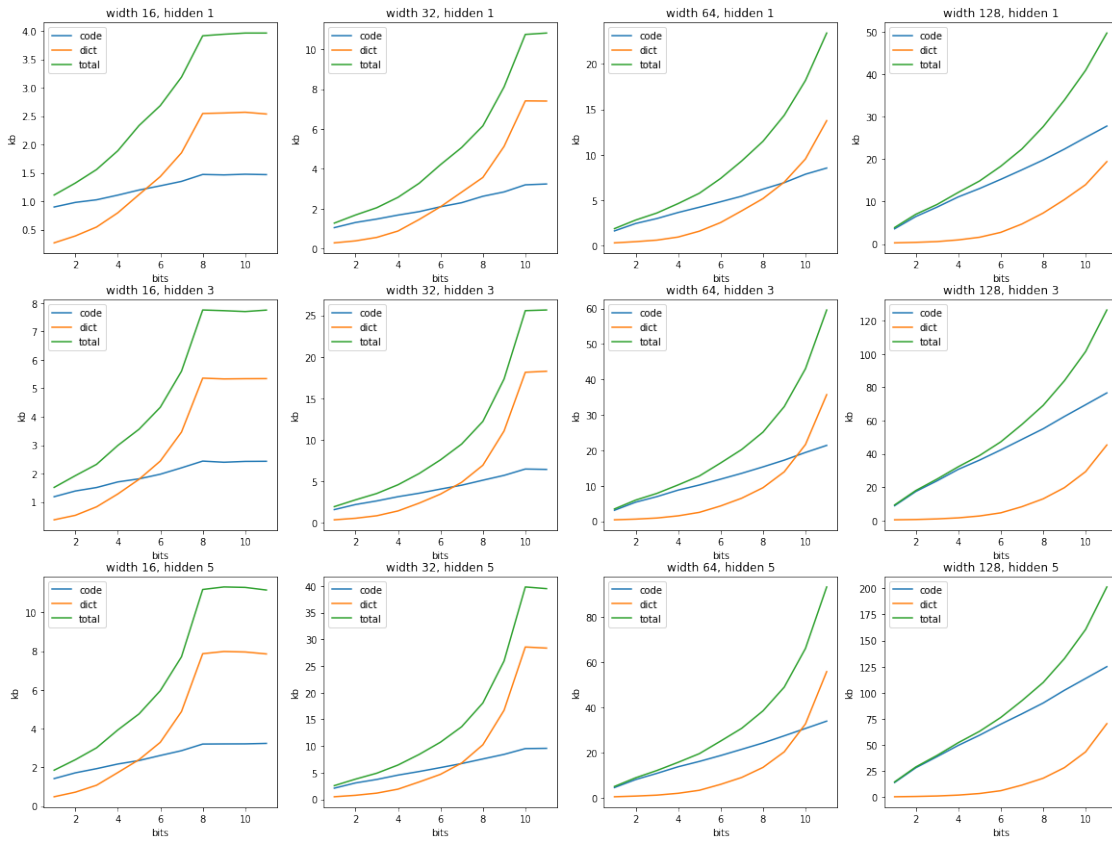


Figure 4: Trade-offs for compressed model (code) and dictionary sizes. For small network architectures the storage of the quantization dictionary exceeds that of the compressed model. This issue diminishes as the network size increases. Note for small models with a higher number of bits-per-weight, if the number of weight elements in the layer is $\leq 2^{bits}$ the dictionary will map a value exactly to each weight. Generated data.

1.5. Blender Qualitative



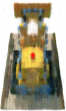
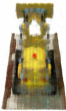
























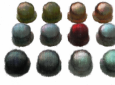








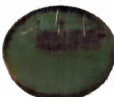


Ground Truth	K-Means	Distributional	Minmax	Explicit [-1, 1]
				
	PSNR: 22.88 15.7kb	PSNR: 21.37 13.3kb	PSNR: 20.79 10.9kb	PSNR: 21.89 14.9kb
				
	PSNR: 29.78 15.6kb	PSNR: 26.76 13.4kb	PSNR: 27.69 11.0kb	PSNR: 28.4 14.5kb
				
	PSNR: 21.03 15.8kb	PSNR: 19.72 13.4kb	PSNR: 19.35 11.6kb	PSNR: 20.47 15.0kb
				
	PSNR: 23.85 16.2kb	PSNR: 23.03 13.4kb	PSNR: 22.71 11.9kb	PSNR: 23.39 14.1kb
				
	PSNR: 27.52 15.5kb	PSNR: 25.04 13.2kb	PSNR: 25.60 11.5kb	PSNR: 26.07 14.8kb
				
	PSNR: 21.51 15.4kb	PSNR: 19.88 13.2kb	PSNR: 19.75 10.6kb	PSNR: 20.77 14.0kb
				
	PSNR: 26.07 16.0kb	PSNR: 21.68 13.4kb	PSNR: 23.09 11.3kb	PSNR: 24.96 14.4kb
				
	PSNR: 24.49 15.9kb	PSNR: 23.45 13.3kb	PSNR: 23.48 11.1kb	PSNR: 23.93 14.5kb

Figure 5: Qualitative NeRF results for Blender (4 hidden layers of 64 neurons, with 3-bit quantization).

1.6. LLFF Qualitative












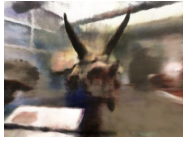
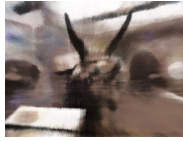

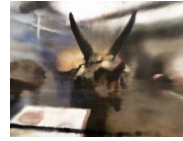











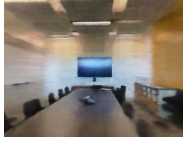
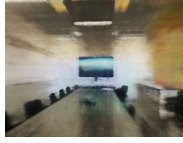
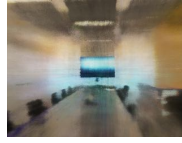
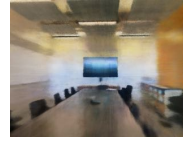





Ground Truth	K-Means	Distributional	Minmax	Explicit [-1. 1]
	 PSNR: 19.36 15.9kb	 PSNR: 18.27 13.3kb	 PSNR: 17.98 11.4kb	 PSNR: 19.43 16.3kb
	 PSNR: 23.46 15.2kb	 PSNR: 22.27 13.2kb	 PSNR: 21.59 10.8kb	 PSNR: 23.04 16.0kb
	 PSNR: 17.69 15.5kb	 PSNR: 16.59 13.3kb	 PSNR: 16.90 11.6kb	 PSNR: 17.93 16.0kb
	 PSNR: 14.54 15.5kb	 PSNR: 14.25 13.4kb	 PSNR: 14.30 11.7kb	 PSNR: 14.41 15.9kb
	 PSNR: 14.50 15.6kb	 PSNR: 14.51 13.4kb	 PSNR: 14.45 11.6kb	 PSNR: 15.22 16.3kb
	 PSNR: 22.91 15.2kb	 PSNR: 19.96 13.2kb	 PSNR: 20.43 11.0kb	 PSNR: 22.64 16.0kb
	 PSNR: 17.94 15.3kb	 PSNR: 16.76 13.2kb	 PSNR: 16.81 11.1kb	 PSNR: 17.46 15.9kb

Figure 6: Qualitative NeRF results for LLFF (4 hidden layers of 64 neurons, with 3-bit quantization).

1.7. Log(MSE) Loss

In Section 3.3 we note that we experimented with both the MSE loss function and its negative base-10 logarithm (i.e. an unscaled peak signal-to-noise ratio). Experimentally we find that faster convergence occurred when minimising for $L = \log_{10}(MSE)^1$, and as such this was used for all 2D image regression experiments. We have not found discussion of this within the implicit neural network literature and so include an intuition as to why this may occur. Note that log is a monotonic transformation and so does not change the optimising points for a function (i.e. $\operatorname{argmax} \log(f(x)) = \operatorname{argmax} f(x)$). To understand why the convergence may differ under this change we show that it may be viewed as equivalent to scaling the gradients under an optimiser (such as SGD) for the original loss.

i.e. there is a mapping $h(\cdot)$ such that:

$$\frac{\partial \log(f(x))}{\partial x} = h\left(\frac{\partial f(x)}{\partial x}\right) \quad (1)$$

Note that by the chain rule:

$$\frac{\partial \log(f(x))}{\partial x} = \frac{1}{f(x)} \cdot \frac{\partial f(x)}{\partial x} \quad (2)$$

So:

$$h\left(\frac{\partial f(x)}{\partial x}\right) = \frac{1}{f(x)} \frac{\partial f(x)}{\partial x} \quad (3)$$

Therefore there should see similar convergence behaviour by scaling our gradients by $\frac{1}{f(x)}$.

From this we can modify an optimiser with for the loss function $L = f(x)$ such that it will give equivalent convergence to using a $\log(f(x))$ loss. Note that the update rule for SGD minimising $f(x)$ is:

$$x := x - \eta \Delta f(x) \quad (4)$$

Setting $\eta_2 = \eta \frac{1}{f(x)}$ gives a step-scheduler for a loss function which will be equivalent to optimising the log-loss.

$$x := x - \eta_2 \Delta f(x) \quad (5)$$

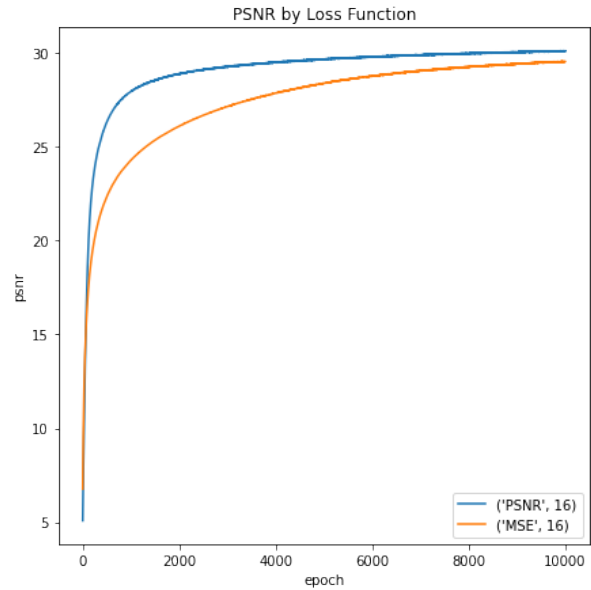


Figure 7: Convergence difference between MSE loss and $\log(\text{MSE})$ loss [labelled as 'PSNR']. Uniform quantization $[-1,1]$ applied with 16 bits. DIV2K.

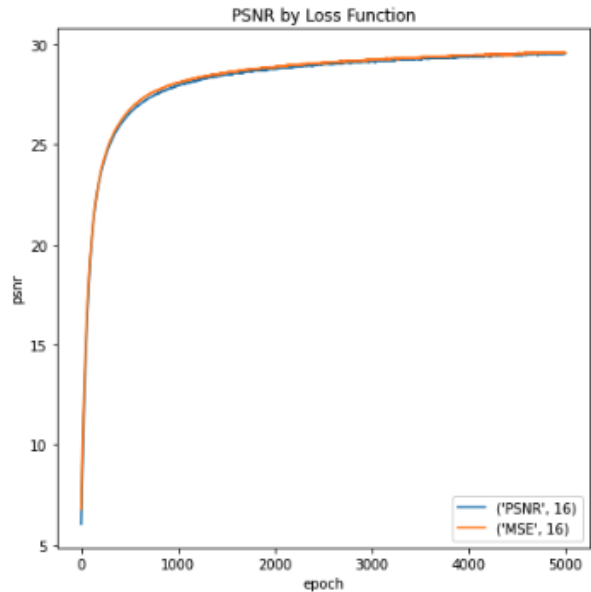


Figure 8: Step-sized scaled by $1/\text{MSE}$ when optimising for a MSE loss. DIV2K.

¹Note that minimising for $\log(MSE)$ is the same as maximising $-\log(MSE)$ (i.e. directly optimising for PSNR).