CS766 Project: Neural Image Compression*

Megh Doshi megh.doshi
Varun Sundar vsundar4
Zachary Huemann zhuemann

varun19299.github.io/implicit-image-compression
April 13, 2021

Implicit Neural Networks, being a continuous mapping, can serve as a compelling choice for representing a variety of commonly encountered 2D and 3D signals. In this proposal, we specifically consider the task of image compression via implicit networks. Unfortunately, owing to the over-parameterized nature of deep networks, a naive approach may require more parameters than samples present in the original signal. Furthermore, the capacity of such networks often saturates with increasing width or depth. We seek to explore two related directions: (a) efficiently increasing the capacity of implicit MLPs to fit natural images, and (b) reducing the storage requirement of such networks through a combination of structured hashing, quantization and entropy coding.

1 Introduction

A large proportion of recent success in a variety of computer vision (and graphics) problems has been attributed to implicitly defined representations parameterized by neural networks (typically a MLP). These include works on novel view-point rendering (Mildenhall et al., 2020; Martin-Brualla et al., 2020), image stabililization (Liu et al., 2021) and view-consistent image generation (Schwarz et al., 2020; Chan et al., 2020). Such MLPs replace traditional grid-based representations, and map low-dimensional coordinates to output quantities such as pixel intensities or densities. Their inherent continuous and differentiable nature makes these representations a compelling choice. Additionally, in the particular case of 3D points, such networks are often much more compact than grid-based representations. Following Tancik et al. (2020), we shall refer to such neural networks as "coordinate MLPs".

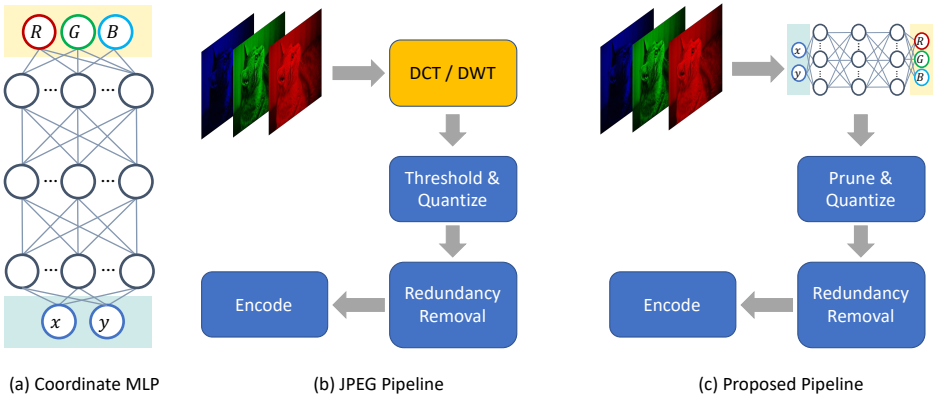
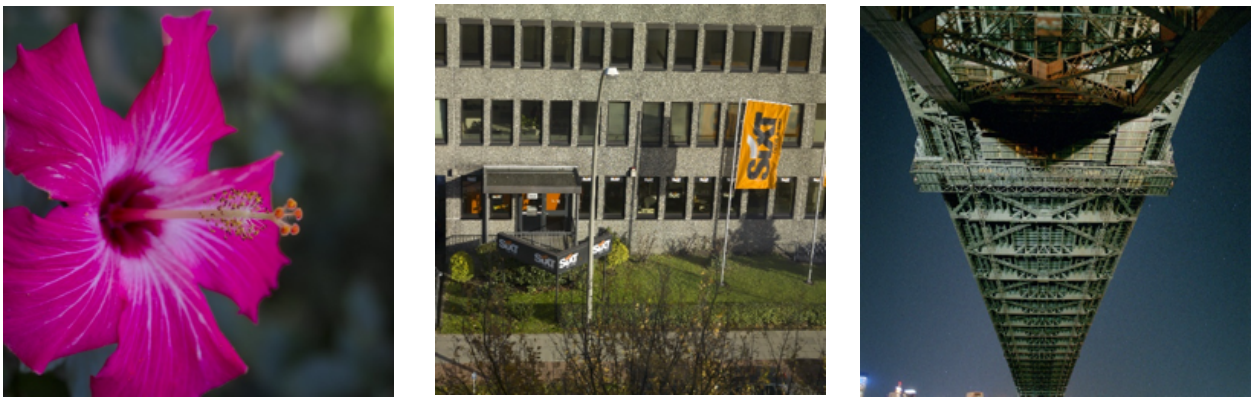


Figure 1: Can coordinate MLPs efficiently represent 2D image signals? We propose to use a pipeline similar to JPEG with two major differences: (a) instead of a discrete cosine or wavelet transform (DCT or DWT), we use a MLP (b) we efficiently store the network's weights as opposed to storing the corresponding DCT or DWT coefficients.

In this work, we examine if these benefits can be carried over to the simpler 2D case of images. We consider applying coordinate MLPs for the task of lossy image compression by mapping 2D grid-locations $(x, y) \in [0, 1]^2$ to RGB intensities. By fitting a MLP, we transfer the task of compressing a grid of pixels to compressing the corresponding

*Mid-Term Report



(a) Flower_foveon (b) Big_building (c) Bridge

Figure 2: Test images from the Image Compression Suite, chosen to include images that are JPEG compressible to varying extents. The Image Compression Benchmark offers both 8-bit and 16-bit—as well as linear and tone-mapped variants of test images—so we prefer it over the standard Kodak dataset (Franzen, 1999) for developing our method. However, when benchmarking the final method, we shall utilize the latter dataset.

network's weights. The representation is no longer limited by the grid-resolution but by the underlying network architecture. This however can be challenging since deep networks often have more parameters than data points itself. However, a wide body of research addresses the efficient storage and inference of deep networks, although generally targeted towards high-dimensional mapping tasks such as image classification.

Another challenge associated with coordinate MLPs is their diminishing increase in capacity with growing layer width and depth. This makes representing signals which are densely sampled (large resolution) or with finer detail difficult. Rebain et al. (2020) tackle this issue for the case of 3D points by decomposing the scene into soft Voronoi diagrams and dedicating smaller networks for each part. For images, frequency domain and wavelet decompositions are potential candidates to achieve similar workarounds. In summary, we focus on the following questions:

- Given a target image to fit, how can we train and efficiently store coordinate MLPs? Since image quality is usually sacrificed for storage, we are interested in exploring the trade-off space for coordinate MLPs and comparing them to conventionally used image compression algorithms such as JPEG.
- For a fixed number of network parameters, can image decomposition help overcome the diminishing returns of naively scaling coordinate MLPs? Valuable insights could include understanding when such decompositions are useful and the range of image resolutions that can be represented.

2 Methodology via Empirical Evaluation

As illustrated in Figure 1, our pipeline first fits a coordinate MLP to an image, either directly or indirectly via a synthesis equation. We then prune and quantize this network, before storing its weights as a compressed sparse array. Although represented sequentially, we may choose to perform some of these steps jointly with training, e.g. directly train sparse MLPs instead of pruning post-training. Given the rich body of literature in training neural networks efficiently, we base our design choices on empirical evaluation presented in this section.

2.1 Image Collection and Training Description

We use three 16-bit, uncompressed images from the Image Compresssion Benchmark[1]: `flower_foveon`, `big_building` and `bridge` (Figure 2). In Figure 3, we present the results of JPEG compression on the three images. We train coordinate MLPs for 10,000 gradient steps each, using the Adam optimizer (Loshchilov and Hutter, 2019) and MSE loss

[1] `https://imagecompression.info`

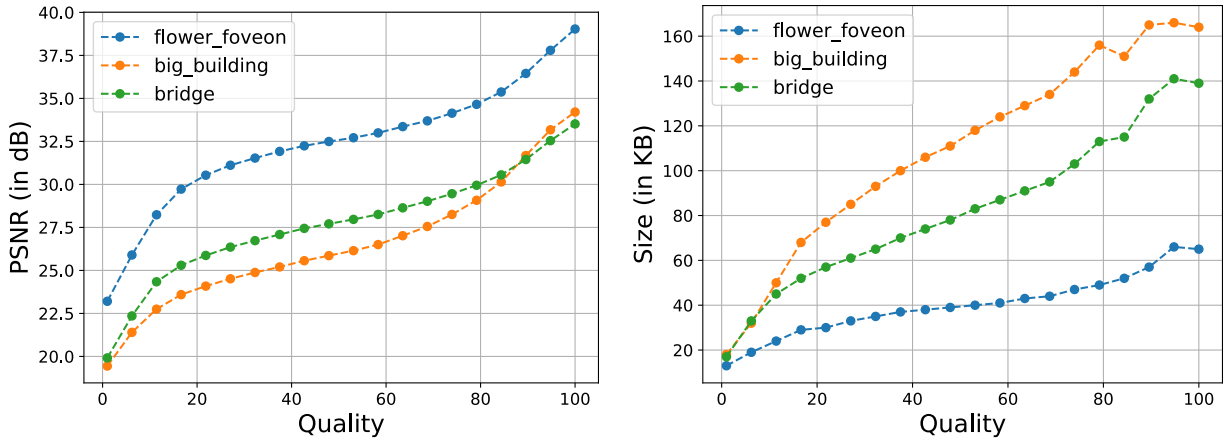


Figure 3: JPEG performance in terms of PSNR and storage space versus quality, evaluated on test images from the Image Compression Benchmark. PSNR is computed between the 8-bit uncompressed image and the JPEG encoded image. Storage space is the actual space on disk in kilobytes, encoded using the *OpenCV* library (Bradski, 2000).

as the objective. Where possible, we set the batch size equal to the total number of pixels—hence corresponding to full-batch gradient descent. All our experiments are conducted on a NVIDIA GTX-1080 GPU with 8GB of VRAM.

2.2 Baseline Network

Network Architecture We compare two recently proposed architectures for enabling MLPs to better represent high-frequency detail in low-dimensional problems: SIREN (Sitzmann et al., 2020) and Fourier Features (Tancik et al., 2020). While SIREN uses sinusoidal activation functions with a particular weight initialization, Fourier Features—abreviated here as FFNet—uses a random Fourier embeddings (Rahimi and Recht, 2008) to increase input dimensions prior to a ReLU MLP. As seen in Figure 4, for a given number of parameters, SIREN significantly outperforms FFNet in image fitting.

By considering the maximum PSNR (equivalently PSNR at quality 100) obtained by JPEG, we choose a SIREN network with depth 8 (or 6 hidden layers) and width 128 as our baseline MLP. We also observe that the minimal architecture that outperforms JPEG in PSNR can differ across images—a hidden layer width of 256 units is better suited for the `big_building` and `bridge` images. Table 1 summarises the architecture, performance and storage space for the chosen baselines. The encoding time required per image is around 20 minutes (10,000 steps), while decoding time is much smaller, around 30 milliseconds, all reported with a GPU device.

Table 1: Performance metrics of baseline SIREN networks, tabulated for all 3 test images. While we find a larger hidden-width necessary for `big_building` and `bridge` images, we instead intend to use a single architecture and make it more sparse to match storage requirements. We also list the compression ratio(s) needed to match JPEG.

Image	**Architecture** (width, depth)	**PSNR** (in dB)	**#Param** (weights, bias)	**Storage** (in KB)	**Compression** (ratio req.)
`Flower_foveon`	128, 8	42.7	99,843	390	6.5–39×
`Big_building`	256, 8	40.9	396,291	1,548	11–80×
`Bridge`	256, 8	38.6	396,291	1,548	9.7–77×

2.3 Reducing Parameter Count

We compare four different techniques to reduce parameter count, viz., *Small-Dense*, *Feathermap*, *RigL* and *Pruning*. *Small-Dense* involves reducing the hidden-layer width commensurately to achieve a target parameter count. *Feathermap*

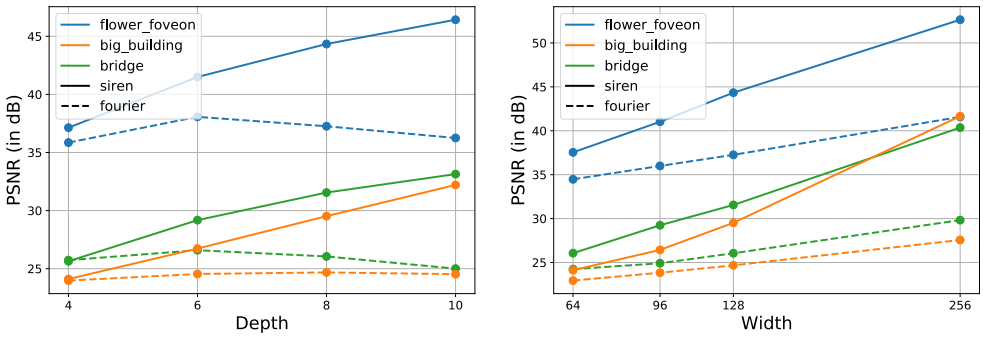


Figure 4: SIREN and FFNet compared across various width, depth configurations. We find SIREN (Sitzmann et al., 2020) to consistently outperform Fourier-Feature networks (FFNet, Tancik et al. (2020)) for all images and configurations. Increasing width and depth consistently improves performance for SIREN and less so for FFNet, but requires greater compression ratios. We set the map size of the random Fourier features as 256, with frequencies sampled from $\mathcal{N}(0, \sigma = 16)$.

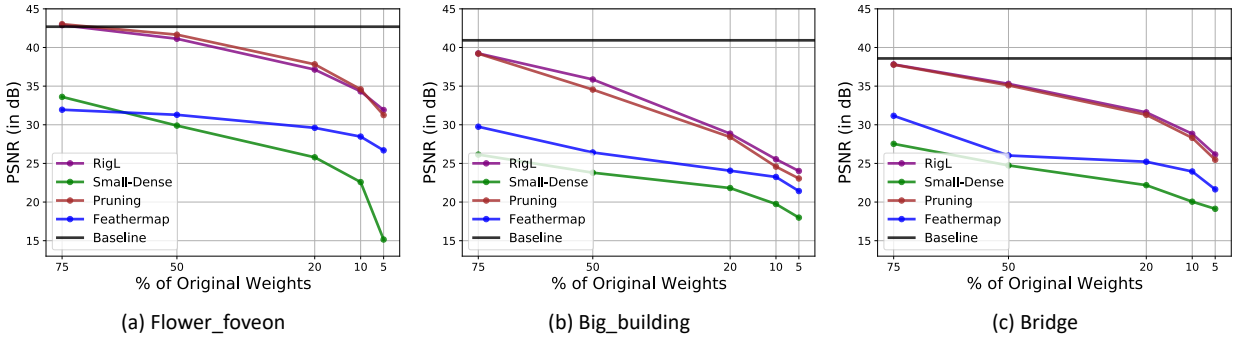


Figure 5: Weight compression techniques evaluated on test images. We compare using a narrow hidden_layer (aka *Small-Dense*), structured hashing (*Feathermap*, Eban et al. (2020)), iterative pruning (Pruning, Zhu and Gupta (2018)) and dynamic-sparse training (*RigL*, Evci et al. (2020)). *RigL* outperforms all other approaches, especially at higher sparsities. Interestingly we can cut up to 80% of the original weights while incurring just a moderate drop in PSNR.

is a recently proposed structured hashing technique that represents the entire weights and biases of the MLP by a single matrix and then stores it via low-rank decomposition. Particularly, we find *Feathermap* to drastically hurt the representation power of the underlying SIREN network. *Pruning* here refers to iterative pruning (Zhu and Gupta, 2018), where low-magnitude weights are gradually removed from a fully-connected MLP till the desired sparsity is achieved. *RigL* (Evci et al., 2020) instead directly trains sparse networks from scratch, with periodic growth, pruning and redistribution steps. Overall, we find *RigL* to be the best approach for lowering parameter-count without significant PSNR loss (Figure 5).

Low bit-rate via extreme sparsity We qualitative illustrate the benefit of using sparse coordinate MLPs to achieve compression. As before, we use *RigL* to directly train sparse networks, but at much higher sparsity rates: 90% and 95%, corresponding to a parameter count reduction of $10\times$ and $20\times$ respectively. By shifting to `int8` quantization, which reduces bits required by $4x$ and storing these weights in the Compressed Sparse Column (CSC) format, we can evaluate the theoretical bit-rates. As seen in Figure 6, even at high compression ratios, our approach retains most of the visual structure and does not suffer from block artefacts.

Figure 6: Qualitative comparison of JPEG versus our method in the low bit-rate regime. Unlike JPEG, the proposed approach does not lead to any block artefacts under severe compression. Compression ratios for our method are estimated, assuming weight reduction using *RigL* and `int8` quantization—without any entropy coding. Quantization produces a fixed compression of $4\times$, while sparsity is used as a toggle to achieve the target bit-rates. Zoom in to see details.

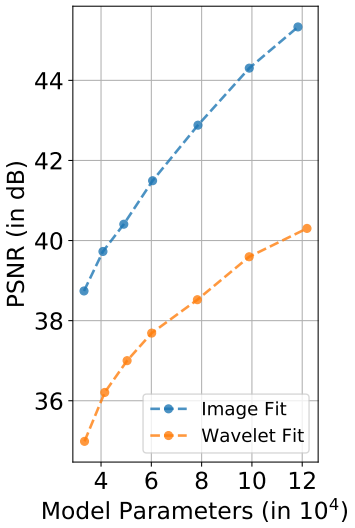


Figure 7: Wavelet decomposition does not improve performance. For a fair comparison, we use the same overall parameter count in either approach. At the parameter counts considered, directly fitting the image performs better.

2.4 Wavelet Fitting

We attempt to increase the representation capacity of coordinate MLPs via wavelet decomposition. We use the Daubechies-3 wavelet to perform decompose an image and fit a MLP each to low-frequency and high-frequency components—both jointly optimized from scratch. For RGB images, we predict low-frequency outputs in the YCbCr space and then simply upsample the chroma components. Unfortunately, this approach does not confer any benefit over directly fitting an image (Figure 7).

2.5 Weight Quantization †

Training with half precision (`float16`) or with surrogate quantization modules that simulate `int8` precision can reduce the possible performance drop due to post-training quantization, while still maintaining full precision (`float32`) PSNR. Amongst post-training quantization techniques, we shall consider k-means or centroid based clustering, range based quantization and distribution based quantization. Han et al. (2015) finds that in the image-classification domain, fully-connected layers can be represented with just 5-bits—although we expect more bits (6-8) required for the significantly harder image-fitting problem. Yet another alternative is to use the SZ lossy algorithm (Di and Cappello, 2016), although this can be harder to implement (lots of carefully crafted stages, not widely supported).

2.6 Entropy Coding †

Post pruning and quantization, we are left with a bunch of sparse matrices that need to be efficiently stored. We shall use the Compressed Sparse Columns (CSC) format, which stores $2\eta(n \times m) + m + 1$ numbers for a matrix of size $n \times m$ and sparsity η. We note that since no pruning is performed on the bias vectors, these can be represented as dense arrays. The stored matrices can now be compressed by a combination of common entropy coding techniques such as huffman encoding, LZ77 or the more recent proposed ZStandard [3].

†*This section includes preliminary observations and work currently in progress.*

[3] http://facebook.github.io/zstd/

References

G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

E. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *arXiv*, 2020.

S. Di and F. Cappello. Fast error-bounded lossy hpc data compression with sz. In *2016 ieee international parallel and distributed processing symposium (ipdps)*, pages 730–739. IEEE, 2016.

E. Eban, Y. Movshovitz-Attias, H. Wu, M. Sandler, A. Poon, Y. Idelbayev, and M. A. Carreira-Perpinan. Structured multi-hashing for model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen. Rigging the lottery: Making all tickets winners. In *Proceedings of Machine Learning and Systems (ICML)*, July 2020.

R. Franzen. Kodak lossless true color image suite. 1999.

S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *Proceedings of the International Conference on Learning Representations (ICLR)*, April 2015.

Y.-L. Liu, W.-S. Lai, M.-H. Yang, Y.-Y. Chuang, and J.-B. Huang. Neural re-rendering for full-frame video stabilization. In *arXiv*, 2021.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, April 2019.

R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. 2020.

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 *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2020.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, December 2008.

D. Rebain, W. Jiang, S. Yazdani, K. Li, K. M. Yi, and A. Tagliasacchi. Derf: Decomposed radiance fields. 2020.

K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. 2020.

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.

M. Zhu and S. Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *Proceedings of the International Conference on Learning Representations (ICLR)*, April 2018.