COIN: COMPRESSION WITH IMPLICIT NEURAL REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a new simple approach for image compression: instead of storing the RGB values for each pixel of an image, we store the weights of a neural network overfitted to the image. Specifically, to encode an image, we fit it with an MLP which maps pixel locations to RGB values. We then quantize and store the weights of this MLP as a code for the image. To decode the image, we simply evaluate the MLP at every pixel location. We found that this simple approach outperforms JPEG at low bit-rates, *even without entropy coding or learning a distribution over weights*. While our framework is not yet competitive with state of the art compression methods, we show that it has various attractive properties which could make it a viable alternative to other neural data compression approaches.

1 INTRODUCTION

Neural image compression methods typically operate in an autoencoder setup [2, 24, 20]. The sender uses an encoder to map the input data to a discretized latent code, which is then entropy coded into a bitstream according to a learned latent distribution. The bitstream is transmitted to the receiver that decodes it into a latent code, which is finally passed through the decoder to reconstruct the image.

In this paper, we take a different approach: we encode an image by overfitting it with a small MLP mapping pixel locations to RGB values and then transmit the weights θ of this MLP as a code for the image (see Figure 1). While overfitting such MLPs, referred to as implicit neural representations, is difficult due to the high frequency information contained in natural images [3, 34], recent research has shown that this can be mitigated by using sinusoidal encodings and activations [23, 34, 30]. In this work, we show that using MLPs with sine activations, often referred to as SIRENs [30], we can fit large images (393k pixels) with surprisingly small networks (8k parameters).

We evaluate our method on standard image compression tasks and show that we outperform JPEG at low bit-rates, even without entropy coding or learning a distribution over weights. As implicit representations have successfully been applied in the context of generative modeling [10], it is likely that combining our approach with a learned weight distribution could lead to promising new approaches for neural data compression. Further, by treating our image as a function from pixel locations to RGB values, we can perform progressive decoding simply by evaluating our function at progressively higher resolutions, which is particularly attractive for resource constrained receiving devices.

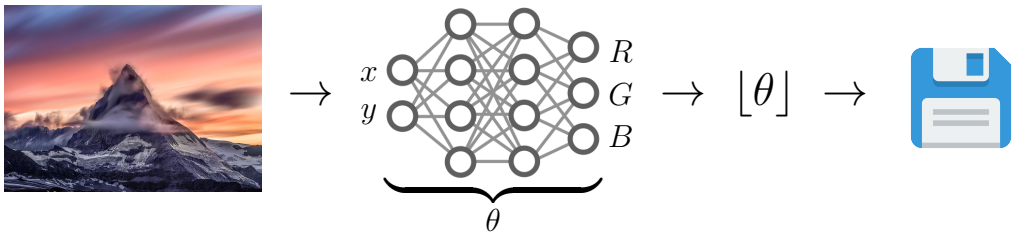

Figure 1: Compressed implicit neural representations. We overfit an image with a neural network mapping pixel locations (x, y) to RGB values (often referred to as an implicit neural representation). We then quantize the weights θ of this neural network to a lower bit-width and transmit them.

2 METHOD

In this section, we describe COmpressed Implicit Neural representations (COIN), our proposed method for image compression. The encoding step consists in overfitting an MLP to the image, quantizing its weights and transmitting these. At decoding time, the transmitted MLP is evaluated at all pixel locations to reconstruct the image.

2.1 ENCODING

Let I denote the image we wish to encode, such that $I[x, y]$ returns the RGB values at pixel location (x, y). We define a function $f_\theta : \mathbb{R}^2 \to \mathbb{R}^3$ with parameters θ mapping pixel locations to RGB values in the image, i.e., $f_\theta(x, y) = (r, g, b)$. We can then encode the image by overfitting f_θ to the image under some distortion measure. In this paper, we use the mean squared error, resulting in the following optimization problem

$$\min_\theta \sum_{x,y} \|f_\theta(x, y) - I[x, y]\|_2^2, \tag{1}$$

where the sum is over all pixel locations.

Choosing the parameterization of f_θ is crucial. Indeed, parameterizing f_θ by an MLP with standard activation functions results in underfitting, even when using a large number of parameters [34, 30]. This problem can be overcome in multiple ways, e.g. by encoding pixel coordinates with Fourier features [34] or by using sine activation functions [30]. Empirically, we found that the latter option yielded better results for a given parameter budget.

Minimizing equation (1) is trivial given a large enough MLP. However, we store the parameters θ of the MLP as the compressed description of the image, so restricting the number of weights will improve the compression rate. The goal is therefore to fit f_θ to I (i.e., minimize distortion) using the fewest parameters possible (i.e., maximizing rate). Our approach then effectively converts a data compression problem to a model compression problem.

To reduce the model size, we consider two approaches: architecture search and weight quantization. More specifically, we perform a hyperparameter sweep over the width and number of layers of the MLP and quantize the weights from 32-bit to 16-bit precision, which was sufficient to outperform the JPEG standard for low bit-rates. However, we believe that more sophisticated approaches to architecture search [11] and especially model compression [35, 13, 36] will further improve results.

2.2 DECODING

Given the stored quantized weights θ, decoding simply consists in evaluating the function f_θ at every pixel location to reconstruct the image. This decoding approach gives us extra flexibility: we can progressively decode the image, e.g. by decoding parts of the image or a low resolution image first, simply by evaluating the function at various pixel locations. Partially decoding images in this way is difficult with autoencoder based methods, showing a further advantage of the COIN approach.

3 RELATED WORK

Implicit neural representations. Representing data with neural networks was originally proposed by [32] but has seen a recent surge in interest in the 3D vision community [27, 26, 6]. Motivated by the fact that memory requirements of deep voxel representations grow cubically [25, 31, 9], implicit representations were proposed to compactly encode high resolution signals [23, 34, 30]. While the MLPs used to represent images typically have a relatively small number of parameters [10], we take this even further and show that by carefully choosing the architecture of the MLP and quantizing the weights, we can fit images with MLPs that take up significantly less space than storing RGB values.

Neural data compression. Learned image compression methods are commonly based on hierarchical variational autoencoders [2, 24, 20] with a learned prior and latent variables being discretized for the purpose of entropy coding. In a similar vein to work in the latent variable model literature [14, 19, 16, 22], several works [5, 12, 38] attempt to close the amortization gap [8] by performing

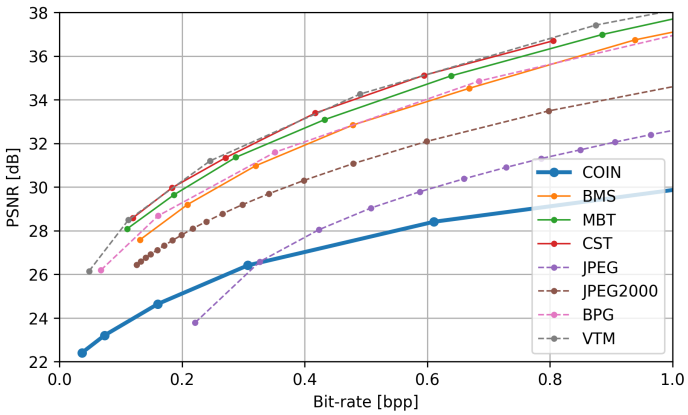

Figure 2: Rate distortion plots on the Kodak dataset.

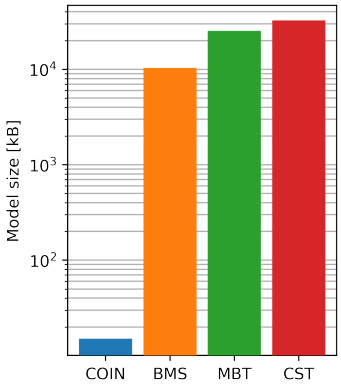

Figure 3: Model sizes at 0.3bpp.

iterative gradient-based optimization steps on top of the use of amortized inference networks. [38] additionally identify and attempt to close the discretization gap stemming from the quantization of the latent variables, also by inference time per-instance optimization. [37] take the idea of per-instance optimization of the model further: they perform per-instance finetuning of the decoder and transmit the quantized decoder parameter updates along with the latent code, leading to improved rate-distortion performance. In this paper, we take a different, and even more extreme from the per-instance optimization perspective, approach: we optimize an MLP to overfit a single image and transmit its weights as the compressed description of the image.

Model compression. While it is known that the problems of data and model compression are very closely related, COIN explicitly casts the problem of data compression into a problem of model compression. There exists a rich body of literature on model compression, [35, 21, 13, 36, 18, 15], which could likely be used to improve COIN's performance.

4 Experiments

We perform experiments on the Kodak image dataset [17] consisting of 24 images of size 768×512. We compare our model against three autoencoder based neural compression baselines which we refer to as BMS [2], MBT [24], and CST [7]. We also compare against the JPEG, JPEG2000, BPG and VTM image codecs. To benchmark our model, we use the CompressAI library [4] and the pre-trained models provided therein. We implement our model in PyTorch [28] and perform all experiments on a single RTX2080Ti GPU.

Rate-distortion plots. To determine the best model architectures for a given parameter budget (measured in bits per pixel or bpp[1]), we first find valid combinations of depth and width for the MLPs representing an image. For example, for 0.3bpp using 16-bit weights, valid networks include MLPs with 10 layers of width 28, 7 layers of width 34 and so on. We then select the best architecture by running a hyperparameter search over learning rates and valid architectures on a single image using Bayesian optimization (we found that the results of the architecture search transferred well to other images). The resulting model is trained on each image in the dataset at 32-bit precision and converted to 16-bit precision after training. We note that decreasing the weights' precision from 32-bit to 16-bit after training resulted in almost no distortion increase, but decreasing them further to 8-bit incurred significant amount of distortion, outweighing the benefit of halving the bpp.

Results of this procedure for various bpp levels are shown in Figure 2. As can be seen, at low bit-rates our model improves upon JPEG *even without using entropy coding*. While our approach is still far from the state of the art compression methods, we believe the performance of such a simple approach is promising for future work in this direction.

[1] bits-per-pixel $= \frac{\#\text{parameters} \times \text{bits-per-parameter}}{\#\text{pixels}}$

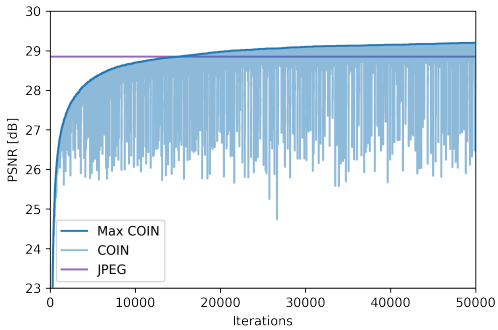

Figure 4: Model training on image 15 in the Kodak dataset. Max COIN represents the max PSNR achieved at any point during training.

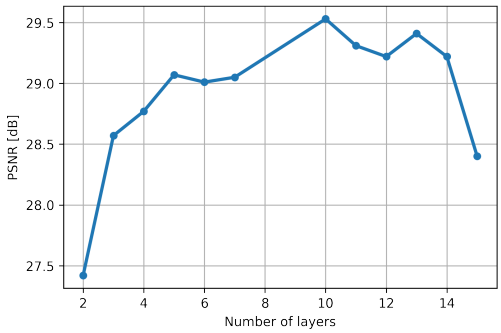

Figure 5: Plot of maximum PSNR for networks of the same size (0.3bpp) with different architectures.

Model size. In contrast to most other neural data compression algorithms, our method does not require a decoder at test time. Indeed, while the latent code representing the compressed image in such methods is small, the decoder model is large (typically much larger than an uncompressed image). As such, the memory required on the decoding device is also large. In our case, we only require the weights of a (very small) MLP on the decoder side, leading to memory requirements that are orders of magnitude smaller. As can be seen in Figure 3, at 0.3bpp, our method requires 14kB, whereas other baselines require between 10MB and 40MB.

Encoding optimization dynamics. We show an example of the overfitting procedure in Figure 4. As can be seen, COIN outperforms JPEG after 15k iterations and continues improving beyond that. While the optimization can be noisy, we simply save the model with the best PSNR.

Architecture choice. In Figure 5, we show the performance of various valid architectures of size 0.3bpp. As can be seen, the quality of compression depends on the architecture choice, with different optimal architectures for different bpp values, see Appendix A for details.

5 Scope, Limitations and Future Work

Limitations. The main limitation of our approach is that encoding is slow, because we have to solve an optimization problem for each encoded image. However, paying a significant computational cost upfront to compress content for delivery to many receivers is a standard practice in the setting of one-to-many media distribution, e.g., at Netflix [1]. Nevertheless, this limitation could likely be sidestepped with meta-learning [29, 33] or amortized inference approaches. Further, at decoding time, we are required to evaluate the network at every pixel location to decode the full image. However, this computation can be embarrassingly parallelized to the point of a single forward pass for all pixels. Finally, our method performs worse than state of the art compression methods. However, we believe there are several promising directions to reduce this gap.

Future work. Recent work in generative modeling of implicit representations [10] suggests that learning a distribution over the function weights could translate to significant compression gains for our approach. In addition, exploring meta-learning or other amortization approaches for faster encoding could be an important direction for future work [29, 33]. Refining the architectures of the functions representing the images (through neural architecture search or pruning for example) is another promising avenue. While we simply converted weights to half-precision in this paper, large gains in performance could likely be made by using more advanced model compression [13, 36, 15]. Finally, as implicit representations map arbitrary coordinates to arbitrary features [34, 30, 10], it would be interesting to apply our method to different types of data, such as video or audio.

6 Conclusion

In this paper, we proposed COIN, a new method for compressing images by fitting neural networks to pixels and storing the weights of the resulting models. We showed through experiments that this simple approach can outperform JPEG at low bit-rates, even without the use of entropy coding. We hope that further work in this area will lead to a novel class of methods for neural data compression.

REFERENCES

[1] Anne Aaron, Zhi Li, Megha Manohara, Jan De Cock, and David Ronca. Per-title encode optimization. https://netflixtechblog.com/per-title-encode-optimization-7e99442b62a2, 2015. [Online; accessed 26-Feb-2021].

[2] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *ICLR*, 2018.

[3] Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman. The Convergence Rate of Neural Networks for Learned Functions of Different Frequencies. *NeurIPS*, 2019.

[4] Jean Bégaint, Fabien Racapé, Simon Feltman, and Akshay Pushparaja. Compressai: a pytorch library and evaluation platform for end-to-end compression research. *arXiv preprint arXiv:2011.03029*, 2020.

[5] Joaquim Campos, Simon Meierhans, Abdelaziz Djelouah, and Christopher Schroers. Content Adaptive Optimization for Neural Image Compression. *CVPR Workshop on Learned Image Compression*, 2019.

[6] Zhiqin Chen and Hao Zhang. Learning Implicit Fields for Generative Shape Modeling. *CVPR*, 2019.

[7] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned Image Compression with Discretized Gaussian Mixture Likelihoods and Attention Modules. *CVPR*, 2020.

[8] Chris Cremer, Xuechen Li, and David Duvenaud. Inference Suboptimality in Variational Autoencoders. *ICML*, 2018.

[9] Emilien Dupont, Miguel Angel Bautista, Alex Colburn, Aditya Sankar, Carlos Guestrin, Josh Susskind, and Qi Shan. Equivariant Neural Rendering. *ICML*, 2020.

[10] Emilien Dupont, Yee Whye Teh, and Arnaud Doucet. Generative Models as Distributions of Functions. *arXiv preprint arXiv:2102.04776*, 2021.

[11] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural Architecture Search: A Survey. *JMLR*, 2019.

[12] Tiansheng Guo, Jing Wang, Ze Cui, Yihui Feng, Yunying Ge, and Bo Bai. Variable Rate Image Compression with Content Adaptive Optimization. *CVPR Workshops*, 2020.

[13] Marton Havasi, Robert Peharz, and José Miguel Hernández-Lobato. Minimal Random Code Learning: Getting Bits Back from Compressed Model Parameters. *ICLR*, 2019.

[14] R. Devon Hjelm, Kyunghyun Cho, Junyoung Chung, Russ Salakhutdinov, Vince Calhoun, and Nebojsa Jojic. Iterative Refinement of the Approximate Posterior for Directed Belief Networks. *NeurIPS*, 2016.

[15] 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. *CVPR*, 2018.

[16] Yoon Kim, Sam Wiseman, Andrew C. Miller, David Sontag, and Alexander M. Rush. Semi-Amortized Variational Autoencoders. *ICML*, 2018.

[17] Kodak. Kodak Dataset. http://r0k.us/graphics/kodak/, 1991.

[18] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

[19] Rahul G. Krishnan, Dawen Liang, and Matthew Hoffman. On the challenges of learning with inference networks on sparse, high-dimensional data. *AISTATS*, 2018.

[20] Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack. Context-adaptive Entropy Model for End-to-end Optimized Image Compression. *ICLR*, 2019.

[21] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian Compression for Deep Learning. *NeurIPS*, 2017.

[22] Joseph Marino, Yisong Yue, and Stephan Mandt. Iterative Amortized Inference. *ICML*, 2018.

[23] 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. *ECCV*, 2020.

[24] David Minnen, Johannes Ballé, and George Toderici. Joint Autoregressive and Hierarchical Priors for Learned Image Compression. *NeurIPS*, 2018.

[25] Thu Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yong-Liang Yang. RenderNet: A deep convolutional network for differentiable rendering from 3D shapes. *NeurIPS*, 2018.

[26] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy Flow: 4D Reconstruction by Learning Particle Dynamics. *ICCV*, 2019.

[27] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. *CVPR*, 2019.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *NeurIPS*, 2019.

[29] Vincent Sitzmann, Eric R. Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. MetaSDF: Meta-learning Signed Distance Functions. *NeurIPS*, 2020.

[30] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. *NeurIPS*, 2020.

[31] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. DeepVoxels: Learning Persistent 3D Feature Embeddings. *CVPR*, 2019.

[32] Kenneth O Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007.

[33] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned Initializations for Optimizing Coordinate-Based Neural Representations. *arXiv preprint arXiv:2012.02189*, 2020.

[34] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *NeurIPS*, 2020.

[35] Karen Ullrich, Edward Meeds, and Max Welling. Soft Weight-Sharing for Neural Network Compression. *ICLR*, 2017.

[36] Mart van Baalen, Christos Louizos, Markus Nagel, Rana Ali Amjad, Ying Wang, Tijmen Blankevoort, and Max Welling. Bayesian Bits: Unifying Quantization and Pruning. *NeurIPS*, 2020.

[37] Ties van Rozendaal, Iris A M Huijben, and Taco S Cohen. Overfitting for Fun and Profit: Instance-Adaptive Data Compression. *ICLR*, 2021.

[38] Yibo Yang, Robert Bamler, and Stephan Mandt. Improving Inference for Neural Image Compression. *NeurIPS*, 2020.

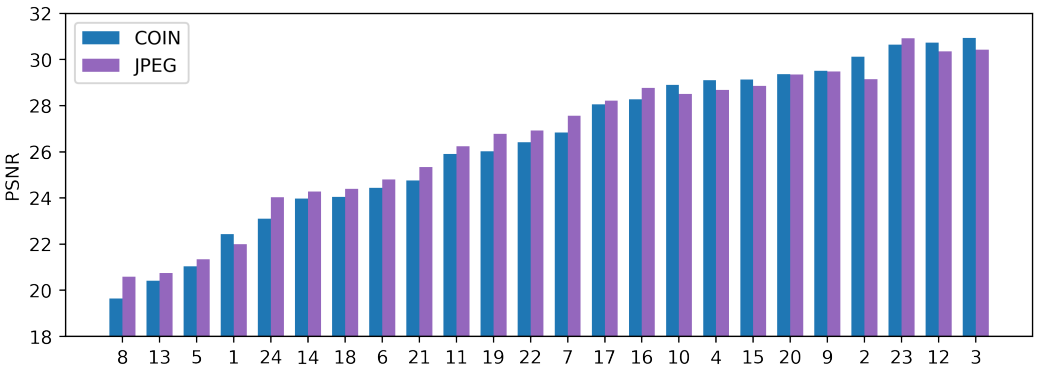

Figure 6: Histogram of PSNR for all images in the Kodak dataset for COIN and JPEG.

A Experimental details

All models were trained using Adam for 50k iterations. We used MLPs with 2 input dimensions (corresponding to (x, y) coordinates) and 3 output dimensions (corresponding to RGB values). The coordinates were normalized to lie in $[-1, 1]$ and the RGB values were normalized to lie in $[0, 1]$. We used sine non-linearities at every layer except the last and used the initialization described in [30]. We used a learning rate of 2e-4. Below we describe the architectures for each bpp level.

- 0.07bpp. Number of layers: 5, width of layers: 20.
- 0.15bpp. Number of layers: 5, width of layers: 30.
- 0.3bpp. Number of layers: 10, width of layers: 28.
- 0.6bpp. Number of layers: 10, width of layers: 40.
- 1.2bpp. Number of layers: 13, width of layers: 49.

B Additional results

In Figure 6, we plot the performance of COIN and JPEG at 0.3bpp (since COIN and JPEG perform similarly at this bit-rate) for all images in the Kodak dataset. As can be seen, the distortion values closely follow each other: images that are difficult for COIN to encode are also difficult for JPEG to encode.