Compression with Bayesian Implicit Neural Representations

Zongyu Guo*, Gergely Flamich*, Jiajun He,
Zhibo Chen, José Miguel Hernández Lobato

* equal contribution

Motivation & Background

Implicit Neural Representations (INR)

$$= \quad \begin{aligned} f &: \mathbb{R}^2 \to \mathbb{R}^3 \\ (x, y) &\mapsto (r, g, b) \end{aligned}$$

$$f(x, y) \approx g(x, y \mid \mathbf{w}) \text{ - NN with weights } \mathbf{w}$$

Lossy Compression with INRs

👩‍🍳 Usual recipe:

- Fit INR to data
- Quantize weights
- Encode quantized weights

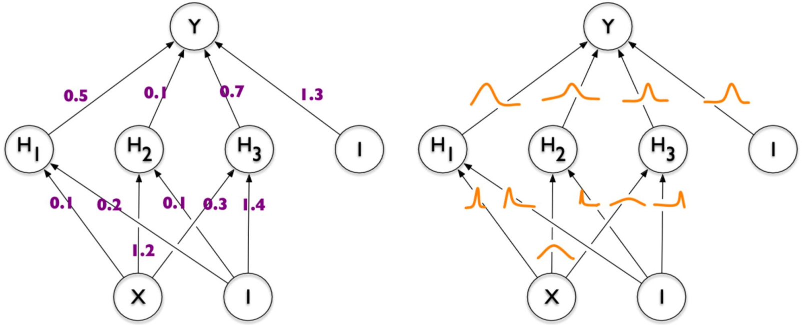
🤔 Issues:

- Weights are brittle, quantization degrades fit
- Have to fix rate before training

💡 Solution: Compression with Bayesian Implicit Neural Representations

Variational INRs + Relative Entropy Coding + Specific Tricks

Variational INRs



$$\mathcal{L}_\beta(\mathcal{D}, q_{\mathbf{w}}, p_{\mathbf{w}}) = \sum_{(x,y),(r,g,b)\in\mathcal{D}} \mathbb{E}_{\mathbf{w}\sim q_{\mathbf{w}}}[\Delta((r,g,b), g(x,y \mid \mathbf{w}))] + \beta \cdot D_{KL}[q_{\mathbf{w}} \,\|\, p_{\mathbf{w}}]$$

distortion

RD trade-off

rate

Image source: C. Blundell et al. "Weight uncertainty in neural networks." ICML 2015.

Relative Entropy Coding to the Rescue!

💡Use A* coding [1] to encode a weight sample

Encoder and decoder share:

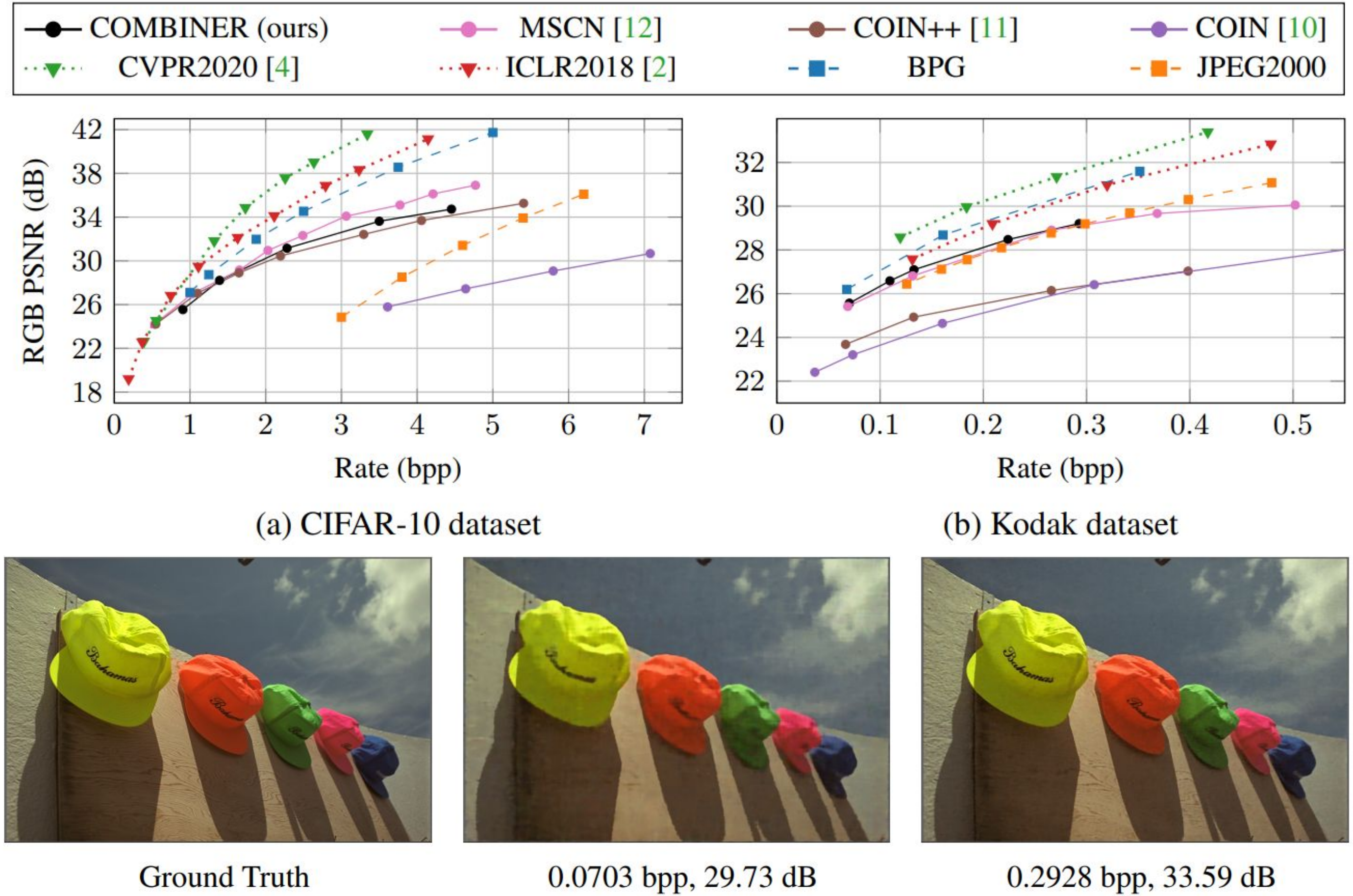
- weight prior
- PRNG seed

Encode approximate posterior sample using KL-many bits

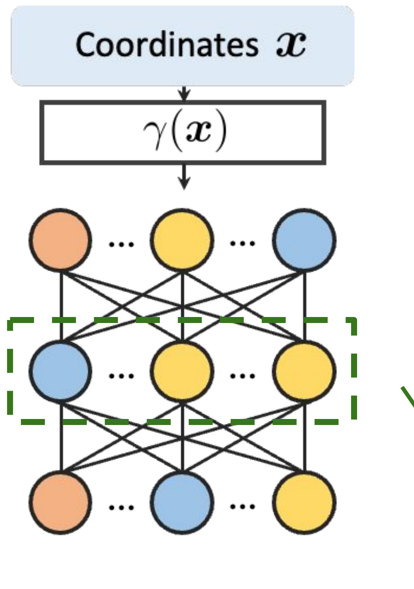
[1] G. Flamich et al. “Fast relative entropy coding with A* coding.” ICML 2022

📈 Results

Results


COMBINER (ours)
MSCN [12]
COIN++ [11]
COIN [10]
CVPR2020 [4]
ICLR2018 [2]
BPG
JPEG2000
RGB PSNR (dB)
Rate (bpp)
(a) CIFAR-10 dataset
(b) Kodak dataset
Ground Truth
0.0703 bpp, 29.73 dB
0.2928 bpp, 33.59 dB

Adaptive Parameter Activation



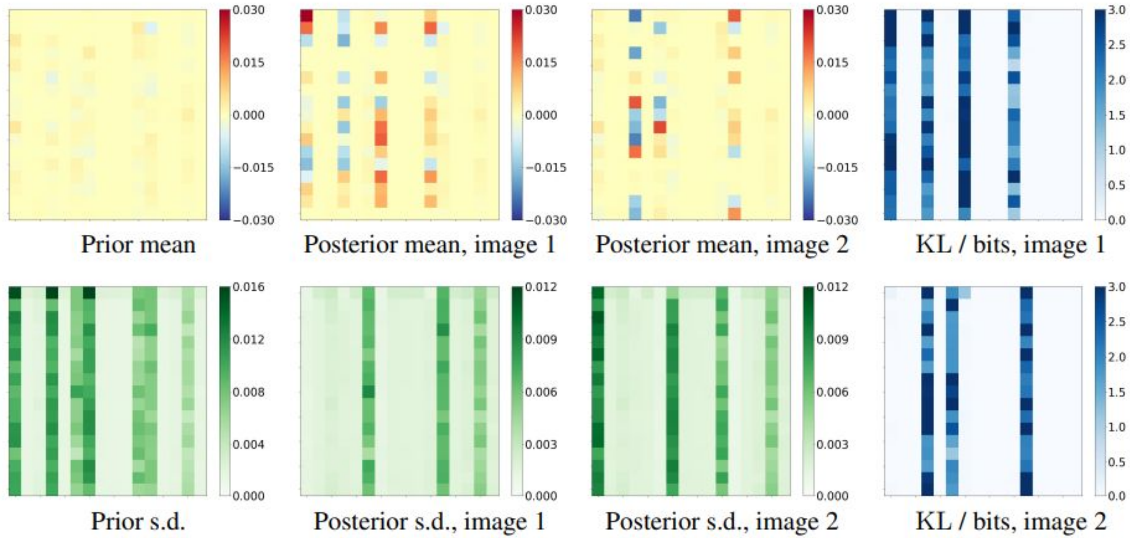
Parameter Group

KL Budget = 16 bits

Visualizations

Model prior:
7 activated hidden units.

Posterior - image 1 :
4 activated hidden units.
Posterior - image 2 :
3 activated hidden units.



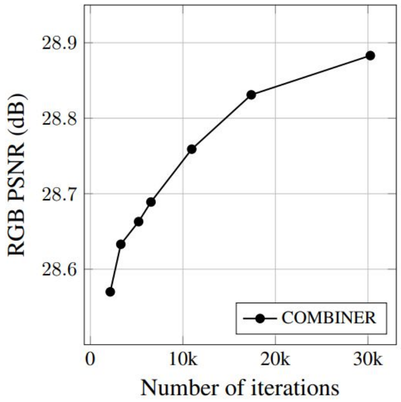
Complexity

bit-rate	Encoding (500 images, GPU A100 80G)			Decoding (1 image, CPU)
	Learning Posterior	REC + Fine-tuning	Total	
0.91 bpp	∼7 min	∼6 min	∼13 min	2.06 ms
1.39 bpp		∼9 min	∼16 min	2.09 ms
2.28 bpp		∼14 min 30 s	∼21 min 30 s	2.86 ms
3.50 bpp		∼21 min 30 s	∼28 min 30 s	3.82 ms
4.45 bpp		∼27 min	∼34 min	3.88 ms

Table 1: The encoding time and decoding time of COMBINER on CIFAR-10 dataset.

bit-rate	Encoding (1 image, GPU A100 80G)			Decoding (1 image, CPU)
	Learning Posterior	REC + Fine-tuning	Total	
0.07 bpp	∼9 min	∼12 min 30 s	∼21 min 30 s	348.42 ms
0.11 bpp		∼18 mins	∼27 min	381.53 ms
0.13 bpp		∼22 min	∼31 min	405.38 ms
0.22 bpp	∼11 min	∼50 min	∼61 min	597.39 ms
0.29 bpp		∼68 min	∼79 min	602.32 ms

Table 2: The encoding time and decoding time of COMBINER on Kodak dataset.



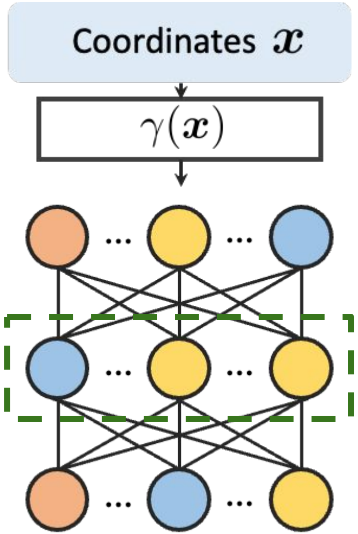
Finetuning steps: 30260 -> 2184
PSNR: only decreased by ~0.3 dB

Encoding time = Time of learning posterior + Time of progressive finetuning

Thanks!

Code: https://github.com/cambridge-mlg/combiner/

Adaptive Parameter Activation



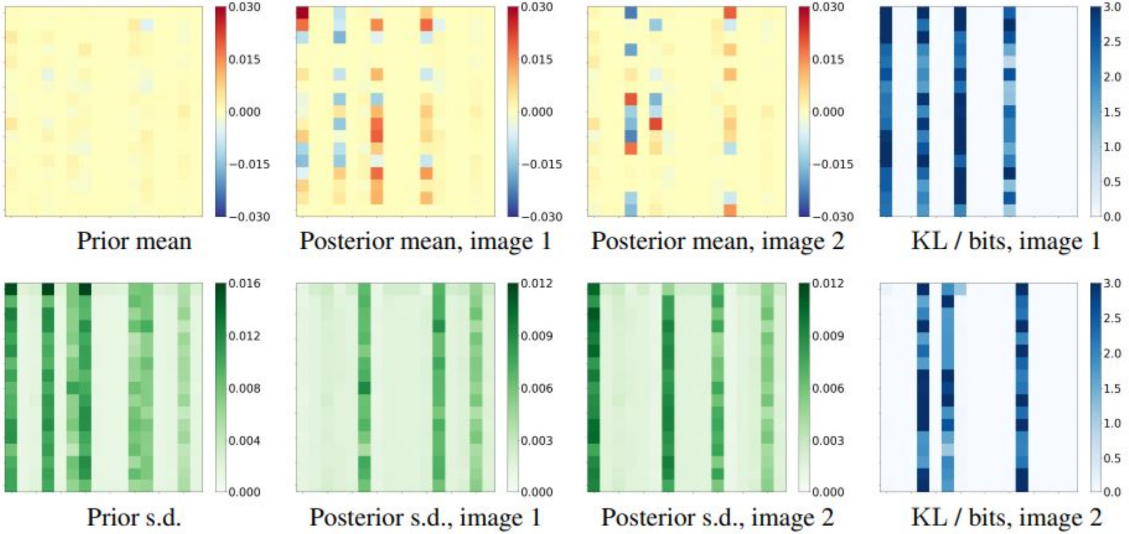
Parameter Group

KL Budget = 16 bits

Visualizations

Model prior:
7 activated hidden units.

Posterior - image 1 :
4 activated hidden units.
Posterior - image 2 :
3 activated hidden units.



Relative Entropy Coding to the Rescue!

💡 Use A* coding [1] to encode a weight sample

Encoder shares PRNG seed with decoder. Then:

$$\mathbf{w}_1, \ldots, \mathbf{w}_N \sim p_{\mathbf{w}} \quad N = 2^{D_{KL}[q_{\mathbf{w}} \| p_{\mathbf{w}}]}$$

$$G_0 = \infty, \quad G_i \mid G_{i-1} \sim \mathrm{TruncGumbel}(-\infty, G_{i-1})$$

$$I = \mathrm{argmax}_{i \in [1:N]} \{q_{\mathbf{w}}(\mathbf{w}_i)/p_{\mathbf{w}}(\mathbf{w}_i) + G_i\}$$

$$q_{\mathbf{w}_I} \approx q_{\mathbf{w}}, \quad \text{encode } I$$

[1] G. Flamich et al. “Fast relative entropy coding with A* coding.” ICML 2022