Autoencoding Implicit Neural Representations for Image Compression

Tuan Pham [*1] **Yibo Yang** [*1] **Stephan Mandt** [1]

Abstract

Implicit Neural Representations (INRs) are increasingly popular methods for representing a variety of signals (Sitzmann et al., 2020b; Park et al., 2019; Mildenhall et al., 2021). Given their advantages over traditional signal representations, there are strong incentives to leverage them for signal compression. Here we focus on image compression, where recent INR-based approaches learn a base INR network shared across images, and infer/quantize a latent representation for each image in a second stage (Dupont et al., 2022; Schwarz & Teh, 2022; Schwarz et al., 2023). In this work, we view these approaches as special cases of nonlinear transform coding (NTC), and instead propose an end-to-end approach directly optimized for rate-distortion (R-D) performance. We essentially perform NTC with an INR-based decoder, achieving significantly faster training and improved R-D performance, although still falling short of that of state-of-the-art NTC approaches. By viewing an INR base network as a convolutional decoder with 1x1 convolutions, we can also better understand its inferior R-D performance through this inherent architectural constraint.

1. Introduction

Implicit Neural Representations (INR) have gained popularity as a promising approach for representing a variety of data types, including images (Stanley, 2007; Sitzmann et al., 2020b; Chen et al., 2021; Karras et al., 2021), signed distance functions (Park et al., 2019; Sitzmann et al., 2020a), and 3D scenes (Sitzmann et al., 2019; Jiang et al., 2020; Mildenhall et al., 2021). These methods use a neural network to map an input coordinate to the corresponding signal value, and offers advantages over traditional signal representations such as memory efficiency and resolution independence. However, it is still an open question how they can be best used for data *compression*.

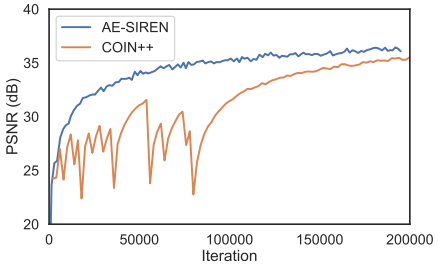

Figure 1. Training curves for AE-SIREN (ours) and COIN++ (Dupont et al., 2022). We optimize only on a distortion loss to be able to compare with COIN++.

Early works using INRs for neural image compression train a separate neural network to represent each image (Dupont et al., 2021; Strümpler et al., 2022), and are quickly surpassed in R-D performance by more recent methods that represent each image by a latent tensor and employ a base network shared across all images (Dupont et al., 2022; Schwarz & Teh, 2022; Schwarz et al., 2023). In most methods, the INR network or latent tensor parameters are optimized without a rate constraint, and are quantized and entropy-coded in a post-processing stage, potentially leaving R-D performance on the table. Moreover, the per-image optimization with gradient descent or meta-learning is not only computationally expensive but can also result in training instability and slow convergence (see Figure 1).

In this work, we view existing INR-based compression approaches through the nonlinear transform coding (NTC) framework, and propose a hybrid approach combining the best of both worlds. We view the INR as a *decoder*, and replace the typical INR-style iterative encoding with a learned encoder network performing amortized inference. This allows end-to-end training with learned quantization and entropy modeling, directly optimized for R-D performance.

More specifically, our contributions are as follows:

- We formulate existing INR-based compression in the NTC framework, allowing us to develop an end-to-end hybrid method and better understand the limitation of INRs for image compression.

[*]Equal contribution [1]Department of Computer Science, University of California, Irvine. Correspondence to: Tuan Pham <tuan.pham@uci.edu>.

Published as a workshop paper at ICML 2023 Neural Compression Workshop.

- We experiment with two autoencoder architectures based on two choices of INR — SIREN (Sitzmann et al., 2020b) and LIIF (Chen et al., 2021). Compared with COIN++ (Dupont et al., 2022), we demonstrate superior R-D performance on the CIFAR and Kodak datasets, as well as faster training (see Fig. 1).

2. Background

2.1. Nonlinear Transform Coding

In nonlinear transform coding (Ballé et al., 2021), an encoder f maps a given image $\mathbf{x}$ to a continuous latent representation $\mathbf{y}$; $\mathbf{y}$ is then quantized to $\hat{\mathbf{y}}$, entropy-coded, and sent to the decoder g, which computes a reconstruction $\hat{\mathbf{x}} = g(\hat{\mathbf{y}})$. The various components are jointly optimized on an R-D objective. See (Yang et al., 2023) for background.

2.2. Implicit Neural Representation

Implicit Neural Representation utilizes a neural network to map the input coordinates to desired output features. NeRF (Mildenhall et al., 2021) employs MLP with ReLU activation and positional embedding to accurately represent 3D scenes. SIREN (Sitzmann et al., 2020b) uses a MLP with sine activation to capture complex, high-frequency patterns, resulting in high-quality reconstructions.

However, a single neural network may not have sufficient capacity to represent a large or complex image, and training each network from scratch can be computationally expensive. Therefore, Mehta et al. (2021); Chen et al. (2021); Müller et al. (2022) proposed to represent each image by a latent tensor $\mathbf{y}$ and a base network g_θ shared across all images. Given a coordinate c (e.g., a pixel location $(12, 50)$), the reconstruction at coordinate c is computed as

$$\hat{\mathbf{x}}_c = g_\theta(\mathbf{y}, c). \tag{1}$$

The base network g_θ in (Mehta et al., 2021; Dupont et al., 2022) consists of a (1). *modulator network* g_{mod}, which computes modulation tensors $\boldsymbol{\alpha} = g_{\text{mod}}(\mathbf{y})$ intended to capture the image content; and (2). a SIREN network whose hidden activations are modulated by $\boldsymbol{\alpha}$ (similarly to FiLM (Perez et al., 2018)), and computes $\hat{\mathbf{x}}_c = g_{\text{SIREN}}(\boldsymbol{\alpha}, c)$.

Some INR methods also employ explicit spatial structures in the latent feature $\mathbf{y}$ to represent data. Those structures could be 2D grid for images (Chen et al., 2021), or 3D voxels for 3D scenes (Jiang et al., 2020; Chan et al., 2022; Müller et al., 2022; Fridovich-Keil et al., 2022). Interpolation techniques are then used to compute the feature value for any queried output coordinate using nearby features on the grid.

For images, Local Implicit Image Function (LIIF) (Chen et al., 2021) has proved effective for tasks that require spatial consistency and coherence, such as image super-resolution. In LIIF, each continuous image is represented as a 2D feature map $\mathbf{y} \in \mathbb{R}^{H \times W \times D}$. Then a decoding MLP takes the 2D coordinates and queries the local latent codes around the coordinates as inputs and predicts the colors.

2.3. INR-based Neural Image Compression

At a high level, INRs essentially allow signals to be *reconstructed* on the basis of network parameters, and INR-based *compression* approaches additionally quantize and entropy code these parameters to achieve bit-rate reduction.

Early work using INR for image compression (Dupont et al., 2021) trains and quantizes a separate SIREN network for each image. Parallel to INR research (Mehta et al., 2021), more recent methods (Dupont et al., 2022; Schwarz & Teh, 2022; Schwarz et al., 2023) also adopt the paradigm of a per-instance latent representation $\mathbf{y}$ and a base network g_θ. Here, only the per-instance latent is quantized and communicated, and the compression cost of the base network is considered negligible when amortized over the instances.

Furthermore, Dupont et al. (2022); Strümpler et al. (2022) propose to reduce the computation burden of INR-style iterative encoding (Dupont et al., 2021) with meta-learning, and Schwarz et al. (2023) apply NTC on the continuous representations $\mathbf{y}$ in a second stage to enable learned quantization and entropy coding.

3. Method

3.1. Autoencoding INR

In this section, we propose to hybridize Nonlinear Transform Coding (NTC) and existing INR-based compression methods in an end-to-end fashion. Our method essentially performs NTC with an INR-based decoder.

To motivate our method, we view existing INR-based approaches (Dupont et al., 2022; Schwarz & Teh, 2022) as special cases of nonlinear transform coding. The base network g_θ (1) shared by all image instances can be naturally viewed as a *decoder* in NTC. This is often a learned neural network, but can also be meta-learned initialization (Dupont et al., 2022; Schwarz & Teh, 2022; Strümpler et al., 2022), or more generally any computation instruction shared between the sending and receiving parties ahead of time (thus, the "decoder" in COIN (Dupont et al., 2021) is the neural network library). Given a decoder g, and coordinates c viewed as side information, the *encoder* $(\mathbf{x}, c) \to \mathbf{y}$ is implemented by iterative optimization, typically minimizing a distortion via $\arg\min_{\mathbf{y}} \|\mathbf{x}_c - g(\mathbf{y}, c)\|^2$. The entropy model is then typically estimated in the second stage by quantizing the resulting latent representations of training instances and fitting an entropy model on them (Dupont et al., 2022; Schwarz & Teh, 2022).

Based on this perspective, we propose to adopt the whole NTC framework for INR-based compression, replacing the INR-style iterative encoder by an amortized inference network f, and learn to quantize and compress the latents with a learned entropy model. All components are trained end-to-end on a rate-distortion loss. As we will show, this drastically speeds up inference (encoding) and training, and results in improved rate-distortion performance. Further, an INR-based decoder g allows us to retain the advantage of the INR representation, and enables tasks such as image super-resolution and computing the image gradient or Laplacian; see Fig. 2.

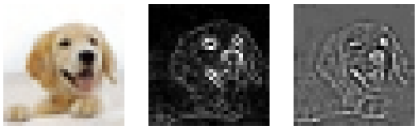

Figure 2. Image gradient and Laplacian computed from our autoencoding INR. Note we take the *partial* derivative $\frac{\partial g(c,\hat{\mathbf{y}})}{\partial c}$.

COIN++ (Dupont et al., 2022) proposes to speed up iterative encoding with meta-learning. We note that this precisely corresponds to semi-amortized variational inference (SVI) (Kim et al., 2018), where the encoder simply maps every instance to the same meta-learned initialization of $\mathbf{y}$. We experimented with SVI in our hybrid approach as well, where we further updated the encoder prediction with additional iterative inference steps, but did not find significant improvement.

3.2. INR as 1x1 convolutions

Viewing the INR base network as a *convolutional* decoder, we also propose implementing its MLP operations via 1x1 convolutions. Specifically, given an image, if we group the coordinate vectors/embeddings of all the pixel locations into a 3D tensor C, such that $C_{(i,j)}$ indexes the coordinate vector/embedding at pixel (i, j), then the required MLP operations on each $C_{(i,j)}$ (such as in SIREN) can be done in parallel by applying 1x1 convolutions on the tensor C. This allows us to efficiently render all the pixels of a image reconstruction in parallel.

Conceptually, formulating an INR network in terms of 1x1 convolutions also illustrates its potential strengths and weaknesses for compression. Essentially, it affords us the flexibility of decoding the reconstruction at each pixel location independently of others. However, images and other natural signals tend to locally exhibit high degrees of correlation and redundancy, and enforcing an independent computation path when reconstructing/rendering each pixel can be more computationally expensive and sub-optimal in compression performance, compared to if we allow larger kernel sizes like in the convolutional decoders of NTC methods.

3.3. Choice of encoder

To avoid bottlenecks in reconstruction quality, an encoder should ideally be capable of inverting the generative procedure computed by the decoder. For convolutional decoders in popular NTC approaches (Ballé et al., 2016; Cheng et al., 2020), the encoders are often also convolutional neural networks (CNN) with similar capacity to the decoder. In our case of an INR based decoder, it is unclear a priori what choice of an encoder can best amortize the iterative optimization in a standard INR encoding procedure.

We experimented with various encoder architectures, such as image transformers (Dosovitskiy et al., 2020; Liu et al., 2021) and a set transformer (Nguyen & Grover, 2022), but found a basic CNN with GDN activation from Ballé et al. (2016) to give the best performance.

3.4. Architectures

Below we describe two variants of our hybrid NTC-INR architectures, AE-SIREN and AE-LIIF, depending on whether a SIREN decoder or a LIIF decoder is used. We use a factorized entropy model (Ballé et al., 2016) for simplicity, but note that more expressive prior (Ballé et al., 2018; Minnen et al., 2018) could be easily used to improve performance further.

3.4.1. AE-SIREN ARCHITECTURE

Encoder. As SIREN uses a dense latent representation, we use an extra linear layer to flatten the output of our CNN encoder. To improve reconstruction quality, the encoder also receives the positional embedding computed by the SIREN network g_{SIREN}, concatenated with the usual image input $\mathbf{x}$.

Decoder. The decoder g follows a similar design to Mehta et al. (2021), using a SIREN network g_{SIREN} modulated with a modulation network g_{mod}. We modulate the activation of the i-th hidden layer of the SIREN network by multiplying it with a modulation vector α_i, i.e., $\phi_i(\mathbf{h}_i) = \alpha_i \odot \sin(\mathbf{W}_i\mathbf{h}_i + \mathbf{b}_i)$. This provides an efficient method for conditioning the output of the SIREN network on the instance-specific latent modulation.

To generate modulation vectors α_i, we use a modulation network with Sine activation. This network takes a quantized latent vector $\hat{\mathbf{y}}$ as input, and generates a sequence of modulation vectors $\{\alpha_1, ..., \alpha_L\}$ for L hidden layers of the SIREN network. It is defined as:

$$\begin{aligned} \mathbf{h}'_0 &= \sin\left(\mathbf{W}'_0\hat{\mathbf{y}} + \mathbf{b}'_0\right), \\ \alpha_{i+1} = \mathbf{h}'_{i+1} &= \sin\left(\mathbf{W}'_{i+1}\left[\mathbf{h}'_i\ \hat{\mathbf{y}}\right]^T + \mathbf{b}'_{i+1}\right) \end{aligned} \tag{2}$$

Our modulated SIREN decoder has several differences com-

pared to COIN++. Rather than using the FiLM layers shift and scale vectors, we choose to modulate the hidden features of the SIREN network through element-wise multiplication. Additionally, we replace the linear modulation network used in COIN++ with a deep MLP with sine activation. We found this combination to perform better in the NTC framework.

3.4.2. AE-LIIF ARCHITECTURE

We use the CNN encoder (Ballé et al., 2016) to directly encode an image $\mathbf{x}$ to its representation $\mathbf{y}$. For the decoder, we adopt the LIIF architecture (Chen et al., 2021) unchanged, consisting of a feature aggregation module followed by an MLP. We give a more detailed description in the Appendix.

It should be noted that our training process does not incorporate multi-scale training within the LIIF representation. However, it is straightforward to incorporate this into our training, thereby developing a model capable of concurrently performing image compression and super-resolution.

4. Experiments

We evaluate our AE-SIREN and AE-LIIF architectures on CIFAR-10 (Krizhevsky et al., 2009) and Kodak datasets (Kodak, 1992). For Kodak dataset, we follow COIN++ (Dupont et al., 2022) to train our models with random 32×32 crop from the Vimeo-90k (Xue et al., 2019) dataset, and evaluate by splitting each Kodak image into 32×32 patches.

We compare our model against INR-based compression methods COIN (Dupont et al., 2021) and COIN++ (Dupont et al., 2022); convolutional autoencoder based methods BMS (Ballé et al., 2018) and CST (Cheng et al., 2020); and traditional codecs JPEG (Wallace, 1992), JPEG2000 (Skodras et al., 2001), BPG (Bellard, 2014).

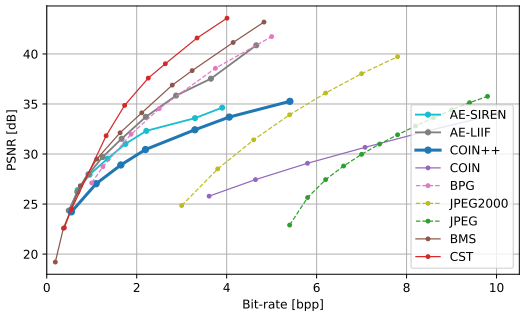

Figure 3. Rate distortion performance on CIFAR-10.

The rate-distortion curves of our models, alongside those of the benchmark models, are showed in Figure 3 and Figure 4. We additionally show our qualitative results on Kodak dataset in the appendix. Both AE-SIREN and AE-LIIF demonstrate superior performance over INR-based compression techniques such as COIN and COIN++. In the CIFAR-10 dataset, AE-LIIF's performance closely mirrors that of the Hyperprior model (Ballé et al., 2018), despite that we are using a factorized prior. However, this performance margin is worse on the Kodak dataset. We postulate that this is due to our use of a smaller patch size (32×32), in contrast to the 256×256 patch size employed by the baseline convolutional autoencoder methods.

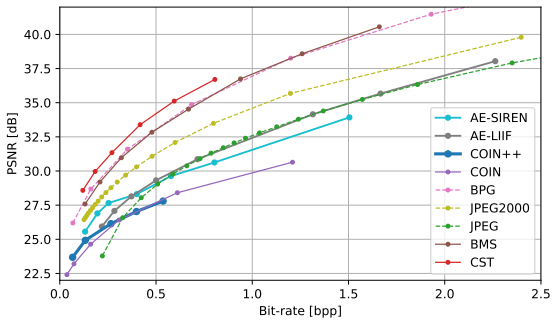

Figure 4. Rate distortion performance on Kodak.

Furthermore, it is notable that AE-LIIF significantly surpasses AE-SIREN in the high bit-rate regime, which indicates the critical role of spatial information. However, we concede that advanced conditional approaches for the SIREN decoder, like the ones proposed by (Schwarz & Teh, 2022; Schwarz et al., 2023), may help improve the performance of AE-SIREN.

5. Discussion

INR-based methods have quickly moved away from representing each signal by a separate network (Dupont et al., 2021; Mildenhall et al., 2021), to much stronger methods employing a base network g that captures common structures among the signals (Dupont et al., 2022; Müller et al., 2022). The latter methods can be understood as implementing a learned, parametric form of vector quantization, a core idea underlying NTC (Ballé et al., 2021). Our work essentially implements the full NTC approach for INR-based image compression, and demonstrates the possibility of replacing INR-style iterative encoding with a learned encoder network. Our interpretation of an INR network as a convolutional decoder blurs the line between NTC and INR-based approaches, and raises interesting questions about the role of INR in compression. Future work may explore similar hybrid approaches and amortized inference methods for non-image data, such as 3D scenes, where a direct application of NTC may no longer be computationally feasible.

References

Ballé, J., Laparra, V., and Simoncelli, E. P. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

Ballé, J., Minnen, D., Singh, S., Hwang, S. J., and Johnston, N. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

Ballé, J., Chou, P. A., Minnen, D., Singh, S., Johnston, N., Agustsson, E., Hwang, S. J., and Toderici, G. Nonlinear transform coding. *IEEE Trans. on Special Topics in Signal Processing*, 15, 2021.

Bellard, F. Bpg image format. https://bellard.org/bpg/, 2014.

Chan, E. R., Lin, C. Z., Chan, M. A., Nagano, K., Pan, B., De Mello, S., Gallo, O., Guibas, L. J., Tremblay, J., Khamis, S., et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16123–16133, 2022.

Chen, Y., Liu, S., and Wang, X. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8628–8638, 2021.

Cheng, Z., Sun, H., Takeuchi, M., and Katto, J. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7939–7948, 2020.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Dupont, E., Goliński, A., Alizadeh, M., Teh, Y. W., and Doucet, A. Coin: Compression with implicit neural representations. *arXiv preprint arXiv:2103.03123*, 2021.

Dupont, E., Loya, H., Alizadeh, M., Goliński, A., Teh, Y. W., and Doucet, A. Coin++: Data agnostic neural compression. *arXiv preprint arXiv:2201.12904*, 2022.

Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., and Kanazawa, A. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5501–5510, 2022.

Jiang, C., Sud, A., Makadia, A., Huang, J., Nießner, M., Funkhouser, T., et al. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6001–6010, 2020.

Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., and Aila, T. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863, 2021.

Kim, Y., Wiseman, S., Miller, A., Sontag, D., and Rush, A. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pp. 2678–2687. PMLR, 2018.

Kodak. Kodak photocd dataset. Eastman Kodak Company, Rochester, NY, 1992. Available at: http://r0k.us/graphics/kodak/.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

Mehta, I., Gharbi, M., Barnes, C., Shechtman, E., Ramamoorthi, R., and Chandraker, M. Modulated periodic activations for generalizable local functional representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14214–14223, 2021.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Minnen, D., Ballé, J., and Toderici, G. D. Joint autoregressive and hierarchical priors for learned image compression. *Advances in neural information processing systems*, 31, 2018.

Müller, T., Evans, A., Schied, C., and Keller, A. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4): 1–15, 2022.

Nguyen, T. and Grover, A. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. *arXiv preprint arXiv:2207.04179*, 2022.

Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 165–174, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Schwarz, J. R. and Teh, Y. W. Meta-learning sparse compression networks. *arXiv preprint arXiv:2205.08957*, 2022.

Schwarz, J. R., Tack, J., Teh, Y. W., Lee, J., and Shin, J. Modality-agnostic variational compression of implicit neural representations. *arXiv preprint arXiv:2301.09479*, 2023.

Sitzmann, V., Zollhöfer, M., and Wetzstein, G. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019.

Sitzmann, V., Chan, E., Tucker, R., Snavely, N., and Wetzstein, G. Metasdf: Meta-learning signed distance functions. *Advances in Neural Information Processing Systems*, 33:10136–10147, 2020a.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020b.

Skodras, A., Christopoulos, C., and Ebrahimi, T. The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001.

Stanley, K. O. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8:131–162, 2007.

Strümpler, Y., Postels, J., Yang, R., Gool, L. V., and Tombari, F. Implicit neural representations for image compression. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVI*, pp. 74–91. Springer, 2022.

Wallace, G. K. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

Xue, T., Chen, B., Wu, J., Wei, D., and Freeman, W. T. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127:1106–1125, 2019.

Yang, Y., Mandt, S., and Theis, L. An introduction to neural data compression. *Foundations and Trends in Computer Graphics and Vision*, 15(2):113–200, 2023. URL http://dx.doi.org/10.1561/0600000107.

A. Architecture details

A.1. AE-SIREN architecture

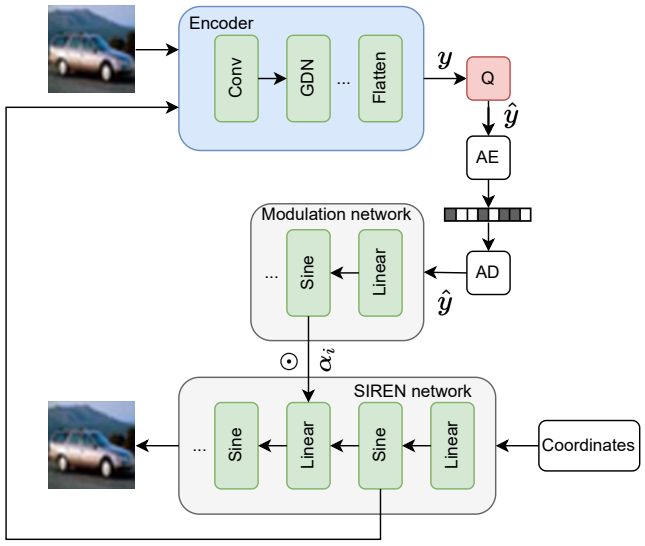

Figure 5. Our AE-SIREN architecture. Representation y from the encoder is quantized (Q) to $\hat{\text{y}}$, which is compressed into a bitstream using an arithmetic encoder AE and decompressed by an arithmetic decoder AD. The modulation network g_{mod} then takes this decompressed $\hat{y}$ as input to calculate the modulations for each linear layer of the SIREN network g_{SIREN}.

A.2. AE-LIIF architecture

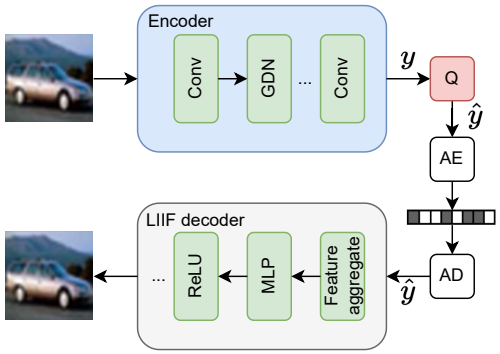

Figure 6. Our AE-LIIF architecture. The spatial LIIF representation y is quantized and compressed. After decompression, we use the LIIF decoder g_{LIIF} to reconstruct the image.

Our AE-LIIF architecture is shown in Figure 6. Following LIIF, we first perform feature unfolding, which concatenates the 3×3 neighbor latent codes in $\hat{\mathbf{y}}$.

$$\hat{\mathbf{y}}_{jk} = \text{Concat}\left(\{\hat{\mathbf{y}}_{j+l,k+m}\}_{l,m\in\{-1,0,1\}}\right) \tag{3}$$

Given the concatenated features, we calculate the RGB color at coordinate c with local ensemble:

$$\hat{\mathbf{x}}(c) = \sum_{t \in \{00,01,10,11\}} \frac{S_t}{S} \cdot f_\theta(z_t^*, c - c_t^*) \tag{4}$$

where $\hat{\mathbf{x}}(c)$ is the RGB color at coordinate c of the reconstructed image $\hat{\mathbf{x}}$, z_t^* are the nearest latent code in top-left, top-right, bottom-left, bottom-right in $\hat{\mathbf{y}}$, c_t^* is the coordinate of z_t^*, S_t is the area of the rectangle between c and c_t^*, and $S = \sum S_t$ is the weight normalizer.

B. Experiment details

Our model is implemented in Pytorch (Paszke et al., 2019), and trained on NVIDIA A6000 and NVIDIA TITAN RTX gpus. The RGB color values are normalized in the range $[0, 1]$, while the coordinates are normalized in the range $[-1, 1]$ for both SIREN and LIIF.

We measure reconstruction performance using PSNR (in dB), defined as $\mathrm{PSNR} = -10 \log_{10}(\mathrm{MSE})$. For bit-rate, we measure the exact bits per pixel (bpp).

B.1. Hyperparameters

Our hyperparameters used for training are shown in Table 1 and Table 2.

Table 1. Hyperparameters used for AE-SIREN experiments.

	Hyperparameter	Value
Encoder	Number of channels	512
	Number of layers	4
	Filter sizes	5×5
	Strides	2
	Padding	2
	Activation	GDN
Decoder	Number of layers	5
	Number of hidden units	512
	Activation	sin
	Activation ω_0	30.0 for first layer, 1.0 for other layers
Training	Rate-distortion λ for CIFAR-10	$0.01, 0.025, 0.05, 0.1, 1.0, 5.0, 15.0$
	Latent y dimension for CIFAR-10	$512, 512, 512, 512, 1024, 1024, 2048$
	Rate-distortion λ for Kodak	$0.001, 0.0025, 0.005, 0.01, 0.05, 0.1, 0.25$
	Latent y dimension for Kodak	$128, 128, 128, 192, 192, 256, 512$
	Model learning rate	$1e-4$
	Auxiliary learning rate	$1e-4$
	Optimizer	Adam

C. Qualitative results

We show the qualitative results for different models in Figure 7, 8, 9, 10.

Table 2. Hyperparameters used for AE-LIIF experiments.

	Hyperparameter	Value
Encoder	Number of channels	128 or 192
	Number of layers	4
	Filter sizes	5×5
	Strides	2
	Padding	2
	Activation	GDN
Decoder	Number of layers	5
	Number of hidden units	256
	Activation	ReLU
Training	Rate-distortion λ	$0.0018, 0.0035, 0.0067, 0.013, 0.025,$ $0.0483, 0.0932, 0.18, 0.35$
	Latent y dimension	$12, 12, 12, 12, 12, 20, 20, 20, 32$
	Model learning rate	$1e-4$
	Auxiliary learning rate	$1e-3$
	Optimizer	Adam

Figure 7. Qualitative results for AE-SIREN on Kodak images. From left to right: original images, decompressed images, and residuals. The model used has the average PSNR of 33.92 and 1.50 bpp.

Figure 8. Qualitative results for AE-SIREN on Kodak images. From left to right: original images, decompressed images, and residuals. The model used has the average PSNR of 25.56 and 0.13 bpp.

Figure 9. Qualitative results for AE-LIIF on Kodak images. From left to right: original images, decompressed images, and residuals. The model used has the average PSNR of 38.04 and 2.26 bpp.

Figure 10. Qualitative results for AE-LIIF on Kodak images. From left to right: original images, decompressed images, and residuals. The model used has the average PSNR of 26.13 and 0.21 bpp.