Hyperspectral Image Compression Using Sampling and Implicit Neural Representations

Shima Rezasoltani and Faisal Z. Qureshi

***Abstract*—Hyperspectral images record electromagnetic spectrum, and each hyperspectral pixel often stores hundreds of channels. Consequently, a hyperspectral image contains an order of magnitude more information than a similarly-sized RGB color image. Concomitant with the decreasing cost of capturing these images, there is a need to develop efficient techniques for storing, transmitting, and analyzing hyperspectral images. This paper develops a method for hyperspectral image compression using implicit neural representations where a multilayer perceptron network with sinusoidal activation functions "learns" to map pixel locations to pixel spectrum for a given hyperspectral image. This representation thus acts as a compressed encoding of this image, and the original image is reconstructed by evaluating this network at each pixel location. We introduce a sampling scheme to achieve better compression times while keeping decoding errors low. The proposed method is evaluated on four benchmarks against sixteen other schemes for hyperspectral compression, and according to the PSNR and SSIM metrics, the method developed in this paper achieves state-of-the-art compression rates at low-bit rates. Additionally, we show that the proposed sampling technique reduces encoding times.**

***Index Terms*—hyperspectral image compression, implicit neural representations.**

I. Introduction

Unlike a grayscale image that records a single intensity value or a color image that often records three values at each pixel, hyperspectral images capture electromagnetic spectrum [1], [2]. Therefore each pixel in a hyperspectral image contains tens or hundreds of values, representing recorded reflectance at various frequency bands. As a result, hyperspectral images offer greater possibilities for object detection, material identification, and scene analysis than those provided by grayscale or color images. The costs associated with capturing high-resolution (both spatial and spectral) hyperspectral images continue to decrease, and it is no surprise that hyperspectral images have found widespread use in areas such as remote sensing, biotechnology, crop analysis, environmental monitoring, food production, medical diagnosis, pharmaceutical industry, mining, and oil & gas exploration, etc. [3]–[16]. Hyperspectral images require many orders of magnitude more space than what is needed to store a similarly sized grayscale or color image. Consequently, there is a need to develop efficient schemes for capturing, storing, transmitting, and analyzing hyperspectral images. This work studies the problem of hyperspectral image compression, noting that hyperspectral image compression plays an important role in the storage and transmission of these images.

Visual Computing Lab, Faculty of Science, University of Ontario Institute of Technology. 2000 Simcoe Street North, Oshawa, ON L1G OC5, Canada
Corresponding author: faisal.qureshiontariotechu.ca
Manuscript received January 2024

Recently there has been a surge in interest in learning-based compression schemes. For example, autoencoders [17] and rate-distortion autoencoders [18], [19] have been used to learn compact representations of the input signals. Here, network weights together with the signal signature—latent representation in the case of autoencoders—serve as the compressed representation of the input signal. Other concurrent works are exploring the use of Implicit Neural Representations (INRs) for signal compression. INRs are particularly well-suited to describe data that lives on an underlying grid. These are able to capture complex patterns and connections between the data without the need for explicit parameterization of the grid structure. Specifically, INRs learn a mapping between the grid coordinates and the related data values (e.g., a mapping between pixel location to its spectrum) [20]. Mathematically, consider a hyperspectral image $\mathbf{I} \in \mathbb{R}^{\mathrm{h}\times\mathrm{w}\times\mathrm{c}}$, where h, w, and c denote the height, width, and the number of channels, respectively. The goal is to learn a function $\mathrm{f}_\Theta : (\mathrm{x},\mathrm{y}) \mapsto \mathbf{I}\left[\mathrm{x},\mathrm{y}\right]$, where (x,y) denote pixel coordinates and $\mathbf{I}\left[\mathrm{x},\mathrm{y}\right] \in \mathbb{R}^{\mathrm{c}}$ denotes the pixel spectrum. Θ, which serves as an encoding for the image $\mathbf{I}$, denotes the function parameters. The original image can be reconstructed given Θ by evaluating f_Θ at $[1,\mathrm{h}]\times[1,\mathrm{w}]$.

Using INRs to "encode" hyperspectral images offer the following advantages. (1) Flexibility: implicit neural representations can accurately represent complex and irregular grids or surfaces. Contrary to explicit grid-based representations, which require the explicit definition of the grid structure, implicit representations can adjust to the data without imposing strict grid limitations. (2) Generalization ability: implicit neural representations exhibit strong generalization capabilities to previously unknown data items. They have the ability to catch complex patterns in the data, enabling extrapolation beyond the recorded grid points. This is especially advantageous when working with data that is sparsely or irregularly collected. (3) Computational efficiency and scalability: implicit representations can offer computational efficiency, particularly when dealing with high-dimensional data or large grid sizes. Instead of explicitly storing or computing values for each grid point, they can generate values on-demand by evaluating neural networks. Additionally, implicit representations provide excellent scalability in high-dimensional spaces. The complexity of explicit grid-based approaches frequently increases dramatically as the dimensionality of the underlying grid increases. Implicit representations, on the other hand, are capable of properly managing data with a large number of dimensions. (4) Smoothness and continuity assumptions: implicit neural representations have the ability to depict smooth and uninterrupted changes in the data accurately. Describing phenomena that demonstrate progressive variations throughout

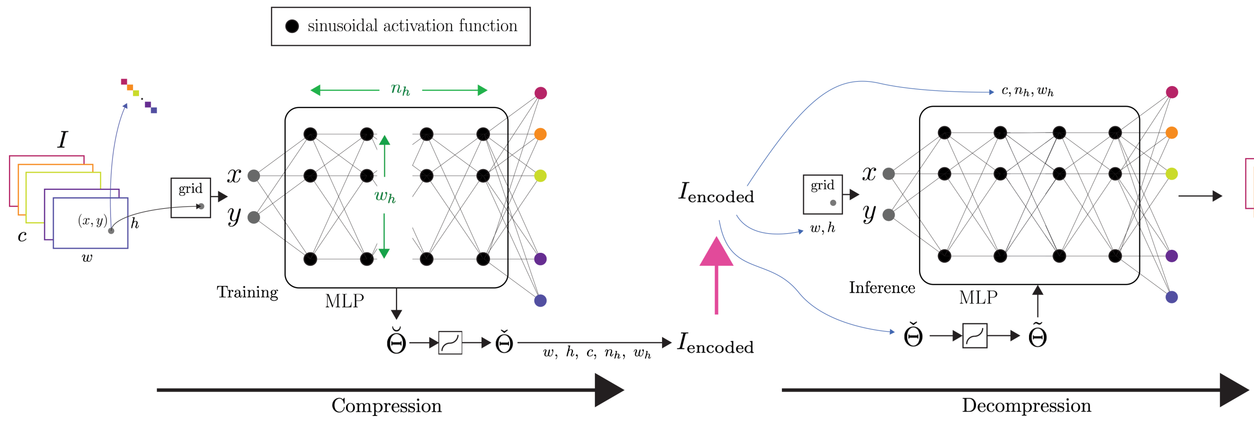


Fig. 1. **Compression and Decompression Pipeline**. (left) An MLP with a periodic activation function is *trained* to map pixel locations to the pixel's spectral signature. (right) Once fitted, MLP is used to reconstruct the hyperspectral image by performing *inference* at various pixel locations.

the grid is advantageous because neural networks naturally smooth out the representations they learn. Lastly, (6) learnable representations: implicit representations can be learned directly from data. Neural networks can be trained using optimization approaches to accurately capture the underlying patterns in the grid-based data, enabling the identification of intricate relationships and features. In a nutshell, implicit neural representations provide a robust framework for encoding data that lies on a grid, offering flexibility, the ability to be generalized, computing efficiency, scalability, and the capacity to capture smooth and continuous variations in the data.

We investigate the use of INRs for hyperspectral image compression and show that it is possible to achieve high rates of compression while maintaining acceptable Peak Signal-to-Noise Ratio (PSNR) values. Figure 1 provides an overview of the proposed compression and decompression pipeline. We evaluate the proposed approach on four benchmarks (Figure 2)—(1) Indian Pines, (2) Jasper Ridge, (3) Pavia University, and (4) Cuprite—and show that at comparable bits-per-pixel-per-band (bpppb) values, our method achieves better PSNR values than those posted by existing hyperspectral compression methods. Specifically, we evaluate the proposed approach against (1) JPEG [21], [22], (2) JPEG2000 [23], (3) PCA-DCT [24], (4) PCA-JPEG2000 [25]–[27] (5) MPEG [28], (6) X264 [25]–[27], (7) X265 [25]–[27], (8) PCA-X264 [25]–[27], (9) PCA-X265 [25]–[27], (10) FPCA-JPEG2000 [29], (11) 3D-DCT [30], (12) 3D-DWT-SVR [31], (13) WSRC [32], (14) HEVC [33], (15) RPM [34], (16) 3D-SPECK [35], and (17) an auto-encoder based scheme proposed in [36]. This list includes both the so-called classical approaches and the more recent learning-based methods. Additionally, we propose a sampling technique to speed up the encoding process and show that sampling achieves faster compression times while achieving PSNR values similar to those obtained when the image is encoded without sampling.[1]

[1]This paper includes five variations of the proposed scheme. We use the following naming convention to refer these methods: "ours" obviously refers to the methods developed in this work; "sampling" denotes whether image coordinates were sampled during the training procedure, and "8-, 16-, or 32-bit" refer to the number of bits used to store individual weights of the neural network.

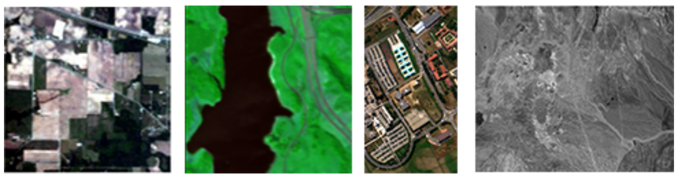


Fig. 2. Datasets used in this study (shown in pseudo-color). (L2R) Indian Pines, Jasper Ridge, Pavia University, and Cuprite.

	w	h	c	n_h	w_h	q	Θ
#bits	16	16	16	8	8	1	bpp $\times$ n_Θ

TABLE I
DISK LAYOUT FOR I_{ENCODED}. HERE q DENOTES IF PARAMETERS Θ WERE QUANTIZED AT COMPRESSION TIME. bpp (OR #BITS-PER-PARAMETER) IS EITHER 32 OR 16.

The rest of the paper is organized as follows. We discuss the related work in the next section. Section III describes the proposed method along with evaluation metrics. Datasets, experimental setup, and compression results are discussed in the following section. Section VI concludes the paper with a summary and possible directions for future work.

II. Related Work

Hyperspectral images exhibit both spatial and spectral redundancies that can be exploited to achieve compression. Lossless compression schemes—e.g., those that use quantization or rely upon entropy-coding—where it is possible to recover the original signal precisely often do not yield large savings in terms of memory required to store or transmit a hyperspectral image [37], [38]. Lossy compression schemes, on the other hand, promise large savings while maintaining acceptable reconstruction quality. Inter-band compression techniques aim to eliminate spectral redundancy [39], while intra-band compression techniques aim to exploit spatial correlations. Intra-band compression techniques often follow the ideas developed for color image compression. [40] exploit the fact that groups of pixels that are around the same location in two adjacent bands are strongly correlated and proposes a scheme that perform both inter-band and intra-band compression. Principal Component Analysis (PCA) is a popular

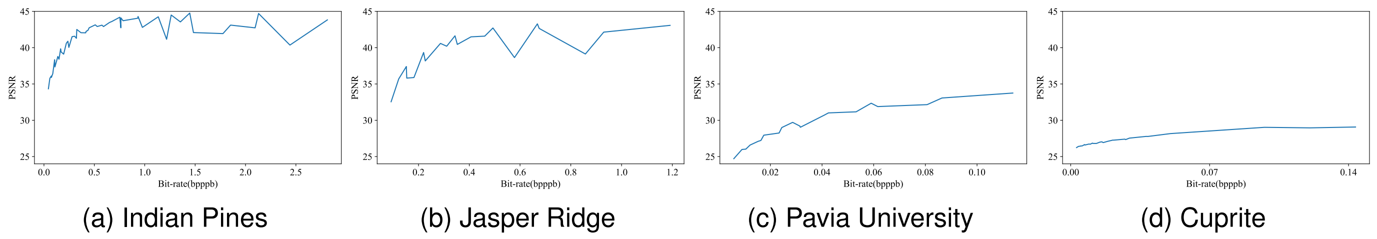


Fig. 3. **Model capacity**. PSNR vs. bpppb. The trend of these plots confirms our intuition that PSNR values increase as bpppb numbers are increased. The plots are not monotonically non-decreasing. This has to do with the stochastic nature of MLP overfitting.

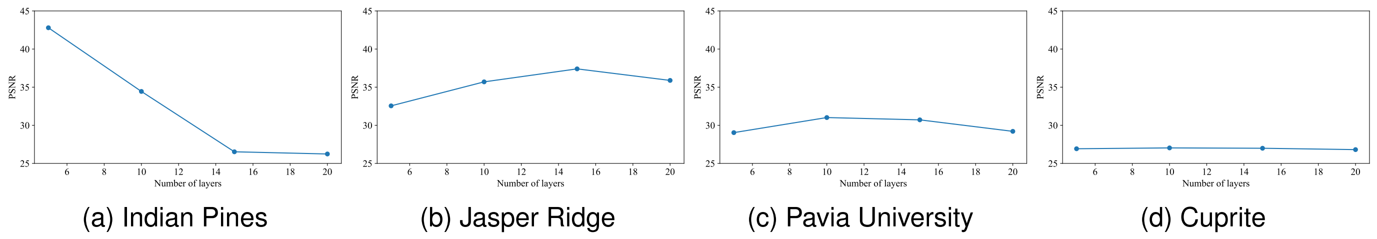


Fig. 4. **Architecture search**. Exploring MLP structure that achieves the best PSNR for different datasets (for a fixed bpppb budget). For our purposes, the MLP structure is defined by the number of hidden layers and the width of these layers. Together the number and width of the hidden layers define network capacity. Note that bpppb is fixed for each dataset. This means that as the number of layers is increased, the width of these layers must decrease accordingly. This figure confirms that architecture search plays an important role. E.g., Indian Pines dataset is best compressed (as measured using PSNR) when using 2 layers; where as the best MLP model for Japster Ridge dataset contains 15 layers.

dimensionality reduction technique that has been used for hyperspectral image compression. PCA offers strong spectral decorrelation, and it is often employed to reduce the number of channels in a hyperspectral image. The remaining channels are subsequently compressed using Joint Picture Expert Group (JPEG) or JPEG2000 standard [41]–[44].

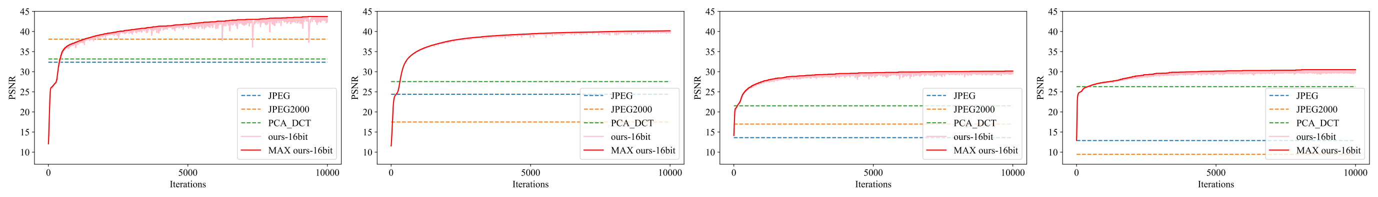
Tensor decomposition methods have also been applied to the problem of hyperspectral image compression [45]. Tensor decomposition achieves dimensionality reduction while maintaining the spatial structure. Transform coding schemes that achieve image compression by reducing spatial correlation have also been used to compress hyperspectral data. Discrete Cosine Transform (DCT) has been used to perform intra-band compression; however, it ignores inter-band (or spectral) redundancy. 3D-DCT that divides a hyperspectral image into $8 \times 8 \times 8$ datacubes is proposed to achieve both inter-band and intra-band compression [30], [46]. Similar to JPEG, which uses 8×8 blocks, 3D-DCT exhibits blocking effects in reconstructed hyperspectral images. The blocking effects can be removed to some extent by using wavelet transform instead [47], [48].

Video coding-based methods that treat each channel of a hyperspectral image as a frame in a video have also been used to perform hyperspectral image compression. Examples of these methods are MPEG, X264, X265, PCA-X264, and PCA-X265. These models rely upon inter-band spectral prediction to compress a hyperspectral image. This is similar to how inter-frame motion prediction is used for video encoding. High-Efficiency Video Coding scheme uses (surface) Reflectance Prediction Modeling to achieve high level of compression [34].

As mentioned earlier, recently, there has been a lot of interest in developing learning-based approaches for signal representation and compression. E.g., autoencoder-based techniques have been proposed to compress hyperspectral images [18], [19], [36]. Hierarchical variational autoencoders have also been used for the purposes of hyperspectral image compression. Here the latent variables are discretized for entropy encoding purposes [49]–[51] to achieve further savings. Our work also employs a similar strategy and uses quantization to reduce the space needed to store the model parameters ϑ. WSRC [32] uses a downsampling scheme that preserves relevant information. Super-resolution is used to maintain high-frequency signal when reconstructing the original image from its compressed, low-resolution version. 3D DWT-SVR [31] scheme represents an image using 3D wavelet coefficients that are compressed via regression using a support-vector machine.

Work in the area of implicit neural representations has shown that it is possible to represent a signal by overfitting an appropriately designed neural network to it. Here the parameters (weights) of the neural network serve as the compact representation of the signal, and it is possible to reconstruct the original signal by sampling the neural network at various input locations [52]–[58]. [59] shows that implicit neural representations with periodic activation functions are able to represent signals, including images, with high-fidelity. This work serves as an inspiration for our work.

Similar to previous research on latent variable models [60]–[63], numerous studies [64]–[66] make an effort to close the amortization gap [67] by combining the usage of amortized inference networks with iterative gradient-based optimization procedures. Using inference time per instance optimization, [66] also identifies and makes an effort to bridge the discretization gap caused by quantizing the latent variables. The concept



(a) Indian Pines (b) Jasper Ridge (c) Pavia University (d) Cuprite

Fig. 5. **Encoding procedure.** Model training on (from left to right) Indian Pines at 0.2 bpppb, Jasper Ridge at 0.15 bpppb, Pavia University at 0.025 bpppb, and Cuprite at 0.02 bpppb. At around 2000 iteration mark, our method is already achieving better PSNR values than those for JPEG, JPEG2000, and PCA-DCT. Furthermore, the PSNR value for our methods continues to improve with more iterations (up to a point).

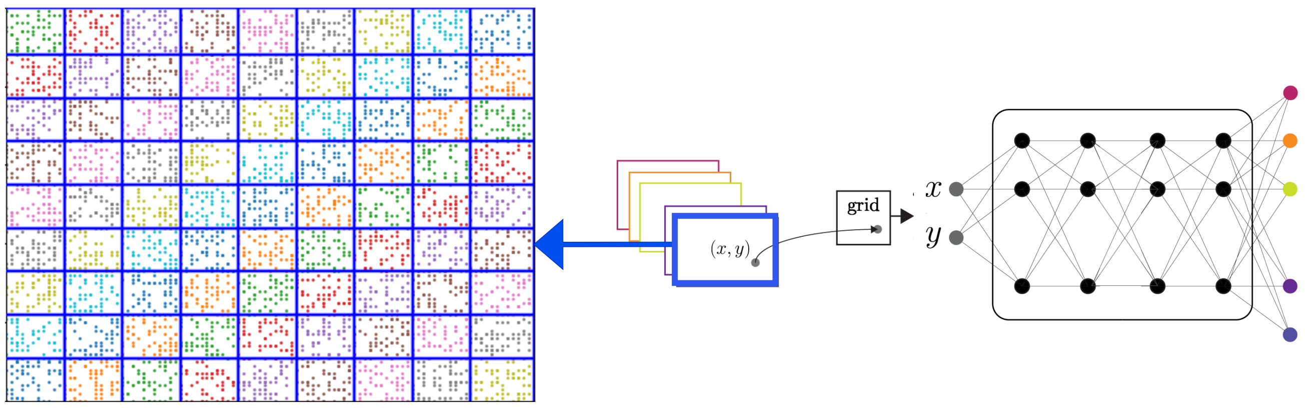


Fig. 6. Compression with random sampling. Pixel locations are uniformly sampled during model training. This is a departure from the traditional approach of using all pixel locations when fitting INRs.

of per-instance model optimization is expanded upon in [68], which fine-tunes the decoder for each instance and transmits updates to the quantized decoder's parameters together with the latent code to provide better rate-distortion performance.

III. Image Compression using INRs

Let us consider a w-by-h grayscale image. We can represent this image as a function

$$\mathrm{I}_{\mathrm{grayscale}} : \mathrm{U} \mapsto [0, 1],$$

where

$$\mathrm{U} = \{(1, 1), \cdots, (\mathrm{w}, 1), \cdots, (1, \mathrm{h}), \cdots, (\mathrm{w}, \mathrm{h})\}.$$

This notation captures the intuition that an image is a function over a 2d grid that defines the pixel locations. The intensity at each pixel is then $\mathrm{I}_{\mathrm{grayscale}}(\mathrm{x}, \mathrm{y})$. It is straightforward to extend this notation to hyperspectral image I as follows

$$\mathrm{I} = \begin{pmatrix} \mathrm{I}_1 : \mathrm{U} \mapsto [0, 1] \\ \vdots \\ \mathrm{I}_\mathrm{c} : \mathrm{U} \mapsto [0, 1] \end{pmatrix}. \tag{1}$$

Here for the sake of simplicity, we assume a w-by-h hyperspectral image comprising c channels. Using this notation, we can find the spectral signature of the pixel at location $(\mathrm{x}, \mathrm{y}) \in \mathrm{U}$ as follows:

$$\{\mathrm{I}_1(\mathrm{x}, \mathrm{y}), \cdots, \mathrm{I}_\mathrm{c}(\mathrm{x}, \mathrm{y})\}.$$

Given this setup, it is possible to imagine a neural network that models the functions $\mathrm{I}_1, \cdots, \mathrm{I}_\mathrm{c}$. Specifically, work on implicit neural representations suggests using a multilayer perceptron (MLP) with periodic activation functions to represent functions of the form shown in Equation 1. Consider an MLP f_Θ with parameters Θ that maps locations in U to pixel spectral signatures:

$$\mathrm{f}_\Theta : \mathrm{U} \mapsto \{\mathrm{I}_1, \cdots, \mathrm{I}_\mathrm{c}\}.$$

Under this regime, training can be defined as

$$\breve{\Theta} = \arg\min_{\Theta} \mathcal{L}(\mathrm{I}, \mathrm{f}_\Theta),$$

where $\mathcal{L}$ is a loss function that is differentiable and that captures the error between the original hyperspectral image and the decompressed hyperspectral image. We use the mean-squared error to compute this loss. Given $\breve{\Theta}$, it is possible to reconstruct the original image I by sampling $\mathrm{f}_{\breve{\Theta}}$ at the relevant locations. Parameters $\breve{\Theta}$, along with w, h, and c, plus the structure of the MLP, represent an encoding $\mathrm{I}_{\mathrm{encoded}}$ of the hyperspectral image I that was used to train the MLP f_Θ. It is expected that the memory required to store $\mathrm{I}_{\mathrm{encoded}}$ is an order of magnitude less than the memory needed to store the hyperspectral image. Table I shows information contained in $\mathrm{I}_{\mathrm{encoded}}$. The decompression process requires constructing the sampling locations U, setting up MLP f_Θ and initializing its weights to $\breve{\Theta}$, and evaluating $\mathrm{f}_{\breve{\Theta}}$ at locations in U.

A. *Metrics for Measuring Compression Quality*

Similar to previous studies, we use Peak Signal-to-Noise Ratio (PSNR) to compare the proposed method with current schemes for hyperspectral image compression. PSNR,

Dataset	Method	bppppb	compression time (Sec)	decompression time (Sec)	PSNR ↑
Indian Pines	JPEG+ [21], [22]	0.1	7.353	3.27	27.47
	JPEG2000+ [23]	0.1	0.1455	0.3115	33.58
	PCA-DCT+ [24]	0.1	1.66	0.04	32.28
	ours-32bit	0.1	243.64	0	36.98
	ours-16bit	0.05	243.64	0	36.95
	ours-sampling-32bit	0.1	282.08	0.0005	**40.1**
	ours-sampling-16bit	0.05	282.08	0.0005	28.40
Jasper Ridge	JPEG+ [21], [22]	0.1	3.71	1.62	21.13
	JPEG2000+ [23]	0.1	0.138	0.395	17.49
	PCA-DCT+ [24]	0.1	1.029	0.027	26.82
	ours-32bit	0.1	312.38	0.0005	32.54
	ours-16bit	0.06	312.38	0.0005	32.51
	ours-sampling-32bit	0.1	75.91	0.0005	**34.77**
	ours-sampling-16bit	0.06	75.91	0.0005	22.07
Pavia University	JPEG+ [21], [22]	0.1	33.86	14.61	20.25
	JPEG2000+ [23]	0.1	0.408	0.628	17.75
	PCA-DCT+ [24]	0.1	6.525	0.235	25.43
	ours-32bit	0.1	780.16	0.0009	34.46
	ours-16bit	0.05	780.16	0.0009	34.17
	ours-sampling-32bit	0.1	72.512	0.0004	**38.08**
	ours-sampling-16bit	0.05	72.512	0.0004	27.02
Cuprite	JPEG+ [21], [22]	0.06	101.195	45.02	12.88
	JPEG2000+ [23]	0.06	1.193	2.476	15.16
	PCA-DCT+ [24]	0.06	11.67	0.754	26.75
	ours-32bit	0.06	1565.97	0.001	28.02
	ours-16bit	0.03	1565.97	0.001	27.90
	ours-sampling-32bit	0.06	664.87	0.001	**37.27**
	ours-sampling-16bit	0.03	664.87	0.001	24.85

TABLE II
The effect of sampling on compression times. Prefix "ours" denotes the methods proposed here. Ours are learning-based methods and as expected these post some of the highest compression times. We address this issue somewhat through "sampling," which halves the compression times. For these results the sampling rate is chosen to be 20%. The good news is that "ours" methods achieve decompression times that are faster than those for JPEG, JPEG2000, and PCA-DCT schemes. These results confirm that (1) sampling helps and (2) the proposed method is well-suited to "compress-once" sort of applications.

measured in *decibels*, is a frequently used metric in image compression. It measures the difference in "quality" between the original image and its compressed version. Higher PSNR values suggest that the compressed image is more similar to the original image, i.e., the compressed image preserves more of the information present in the original image and that it has higher quality. In addition, we also compare the compressed image using Mean Squared Error (MSE), which computes the commulative squared error between the original image and its compressed version. Lower values of MSE mean better reconstruction quality.

MSE is computed as follows

$$\mathrm{MSE} = \|\mathrm{I}[\mathrm{i}] - \tilde{\mathrm{I}}[\mathrm{i}]\|, \tag{2}$$

where $\tilde{\mathrm{I}}$ denotes the reconstructed image. MSE is used to calculate PSNR

$$\mathrm{PSNR} = 10\ \log_{10}\left(\frac{\mathrm{R}^2}{\mathrm{MSE}}\right), \tag{3}$$

where R is the largest variation in the input image in the previous equation. For instance, R is 1 if the input image is of the double-precision floating-point data. R is 255, for instance, if pixels values are stored as 8-bit unsigned integers.

Another metric that is commonly used for image compression studies is the structural similarity index (SSIM) [69]. SSIM accounts for the structural information that the human visual system is naturally attuned to. Therefore, SSIM is perceptually more meaningful then MSE, which treats pixels independent of each other. SSIM metric combines luminance, contrast, and structure, and the channel-wise SSIM is computed as follows:

$$\mathrm{SSIM}_{\text{channel-wise}} = \frac{(2\ \mu_{\mathrm{I}}\ \mu_{\tilde{\mathrm{I}}}\ +\ \mathrm{C}_1)\ (2\ \sigma_{\mathrm{I}\tilde{\mathrm{I}}}\ +\ \mathrm{C}_2)}{(\mu_{\mathrm{I}}^2\ +\ \mu_{\tilde{\mathrm{I}}}^2\ +\ \mathrm{C}_1)(\sigma_{\mathrm{I}}^2\ +\ \sigma_{\tilde{\mathrm{I}}}^2\ +\ \mathrm{C}_2)}, \tag{4}$$

where μ_{I} and $\mu_{\tilde{\mathrm{I}}}$ are mean intensity values for the original image and its reconstruction, respectively. Similarly, σ_{I}^2 is the variance of the original image and $\sigma_{\mathrm{I}\tilde{\mathrm{I}}}$ is the covariance value for the original image and its reconstruction. $\mathrm{C}_1 = (\mathrm{k}_1.\mathrm{L})^2$ and $\mathrm{C}_2 = (\mathrm{k}_2.\mathrm{L})^2$ are there to address division by a weak denominator. L, here, denotes the dynamic range of a pixel and $\mathrm{K}_1 = 0.01$ and $\mathrm{K}_2 = 0.03$. Dynamic range L is typically expressed as $2^{(\text{bits per pixel})}$. Mean SSIM value is computed by averaging channel-wise SSIM values. SSIM values range between 0 and 1, and larger SSIM values indicate a better reconstruction.

Method	Dataset	Size (KB)	PSNR	bpppb	n_h, w_h	Dataset	Size (KB)	PSNR	bpppb	n_h, w_h
-	Indian Pines	9251	∞	16	-,-	Jasper Ridge	4800	∞	16	-,-
JPEG$^+$ [21], [22]		115.6	34.085	0.2	-,-		30	21.130	0.1	-,-
JPEG2000$^+$ [23]		115.6	35.84	0.2	-,-		30	17.494	0.1	-,-
PCA-DCT$^+$ [24]		115.6	33.173	0.2	-,-		30	26.821	0.1	-,-
3D-SPECK$^+$ [35]		115.6	-	0.2	-,-		30	-	0.1	-,-
PCA-JPEG2000† [25]–[27]		115.6	39.5	0.2	-,-		30	**44.26**	0.1	-,-
FPCA-JPEG2000† [29]		115.6	40.5	0.2	-.-		30	-	0.1	-,-
3D-DCT† [30]		115.6	-	0.2	-,-		30	-	0.1	-,-
3D-DWT-SVR† [31]		115.6	-	0.2	-,-		30	-	0.1	-,-
WSRC† [32]		115.6	-	0.2	-,-		30	-	0.1	-,-
ours-32bit		115.6	39.10	0.2	5,60		30	32.54	0.1	5,20
ours-16bit		57.5	38.99	0.1	5,60		15	32.51	0.06	5,20
ours-sampling-32bit		115.6	**42.22**	0.2	5,60		30	34.77	0.1	5,20
ours-sampling-16bit		57.5	29.68	0.1	5,60		15	22.07	0.06	5,20
ours-sampling-8bit		28.7	32.97	0.05	5,60		7.5	24.32	0.03	5,20
-	Pavia University	42724	∞	16	-,-	Cuprite	140836	∞	16	-,-
JPEG$^+$ [21], [22]		267	20.253	0.1	-,-		880.2	24.274	0.1	-,-
JPEG2000$^+$ [23]		267	17.752	0.1	-,-		880.2	20.889	0.1	-,-
PCA-DCT$^+$ [24]		267	25.436	0.1	-,-		880.2	27.302	0.1	-,-
3D-SPECK$^+$ [35]		267	-	0.1	-,-		880.2	27.1	0.1	-,-
PCA-JPEG2000† [25]–[27]		267	33	0.1	-,-		880.2	**40.90**	0.1	-,-
FPCA-JPEG2000† [29]		267	-	0.1	-,-		880.2	-	0.1	-,-
3D-DCT† [30]		267	-	0.1	-,-		880.2	33.4	0.1	-,-
3D-DWT-SVR† [31]		267	-	0.1	-,-		880.2	28.20	0.1	-,-
WSRC† [32]		267	-	0.1	-,-		880.2	35	0.1	-,-
ours-32bit		267	34.46	0.1	10,80		880.2	28.954	0.1	25,100
ours-16bit		133.5	34.17	0.05	10,80		440.1	24.334	0.06	25,100
ours-sampling-32bit		267	**38.08**	0.1	10,80		880.2	36.55	0.1	25,90
ours-sampling-16bit		133.5	27.02	0.05	10,80		440.1	24.91	0.06	25,90
ours-sampling-8bit		66.75	24.02	0.02	10,80		220.05	22.35	0.03	25,90

TABLE III

Results for fixed bpppb. Compression rates (i.e., the desired size of the compressed data) is fixed across methods. The quality of compression is expressed in terms of PSNR. Five variants of the proposed methods are included; these are indicated by the prefix "ours." Classical and learning based methods is denoted using $^+$ and † text decorations. The last column indicates the network structure: n_h and w_h indicates the number of hidden layers and the width of the hidden layers, respectively. This column applies to our methods only. Best PSNR values for each dataset are shown in bold. Quantization to 16-bits or 8-bits reduces the bpppb by one-half and one-fourth, respectively. Results for the video-based and autoencoder-based schemes are missing, since these methods do not provide results for the bpppb values used here. Results for video-based method are available in Table IV and the results for autoencoder-based method are plotted in Figure 7.

Bits-per-pixel-per-band (bpppb) is sometimes used to capture the level of compression. Recall bpppb for an uncompressed hyperspectral image is typically either 8 or 32, depending upon how the pixel values are stored. Recall that hyperspectral pixels are often stored as 32-bit floats. For our model bpppb is calculated as follows:

$$\text{bpppb} = \frac{\#\text{parameters} \times (\text{bits per parameter})}{(\text{pixels per band}) \times \#\text{bands}}. \qquad (5)$$

Figure 3 plots PSNR vs. bpppb for the four datasets that we are using in this work. These plots confirm our intuition that higher bpppb leads to better compression quality as measured by PSNR values. bpppb calculations do not include the storage required to keep network structure (number of layers and the layer width) and the image information (height, width, and the number of channels or bands) needed to decode the image.

B. Compression Pipeline

The proposed compression method consists of two steps. Step 1 performs an architecture search (see Figure 4). The goal here is to find an MLP that achieves the highest reconstruction accuracy for a given bpppb budget. Architecture search is performed by overfitting multiple MLPs having different numbers of hidden layers and hidden layers' widths to the hyperspectral image. Architecture search, however, means longer compression times. Step 2 involves quantizing and storing the parameters of the overfitted MLP to disk. Quantizing network parameters is desireable and results in further savings; however, this may further reduce the quality of the reconstructed image. This paper includes results when network parameters are quantized using 16-bit and 8-bits.

1) Overfitting a SIREN network: The compression procedure comprises overfitting a SIREN network f_Θ to a hyperspectral image I [59]. The width w and height h of the hyperspectral image are used to set up an input location grid on $[-1,+1] \times [-1,+1]$, and the MLP is trained to reconstruct a pixel's spectral signature given its location. The parameters $\breve{\Theta}$ of this overfitted MLP are quantized $\breve{\Theta}$. MLP structure that contains the number of hidden layers n_h, widths of these layers w_h, and the width w, height h, and the number of channels c of the original hyperspectral image I along with $\breve{\Theta}$

serve as a compressed encoding I_{encoded} of the hyperspectral image I (Table I). Parameters $\breve{\Theta}$ are typically stored as 32-bit floats. However, further savings are possible by quantizing these parameters and storing them using fewer bits. We include compression results for when these parameters are stored as 16-bit and 8-bit values. The overfitted MLP contains

$$(w_h \times 2) + (w_h \times w_h)^{(n_h - 1)} + (w_h \times c) \tag{6}$$

parameters.

C. Decompressing $I_{encoded}$

The hyperspectral image is reconstructed from its compressed encoding I_{encoded} as follows: 1) use n_h, w_h, and c to reconstruct f_Θ, 2) dequantize $\breve{\Theta}$ to $\tilde{\Theta}$ and use it to initialize the parameters of f_Θ, 3) use the width w and height h to set up the input grid between $[-1, +1] \times [-1, +1]$, and 4) evaluate $f_{\tilde{\Theta}}$ at each location in the input grid to reconstruct the image $\tilde{I}$ (see Figure 1).

IV. Experimental Setup & Ablative Study

This section discusses the datasets that we have used to study and evaluate the proposed method. It also discusses the various design aspects—architecture search, compression and decompression times, model fitting, and the effects of random sampling—of the proposed model. We evaluate the proposed method against other schemes in the next section.

We have used four datasets to evaluate our approach. We have selected these datasets since others have used these previously to study hyperspectral image compression. Furthermore, these datasets have been extensively used in classification and analysis of hyperspectral imagery. Indian Pines, Cuprite, and Jasper Ridge datasets have been collected using NASA's Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) sensor. The AVIRIS sensor gathers geometrically coherent spectroradiometric data that can be used to characterize the Earth's surface. Pavia University dataset was captured using the ROSIS (Reflective Optics System Imaging Spectrometer).

Indian Pines is a $145 \times 145 \times 220$ hyperspectral image. It was collected using the AVIRIS sensor in 1992, and it spans a region over NW Indiana. The hyperspectral image contains a mix of farms and wooded areas. Additionally, the image contains low-density built-up regions, houses, a number of secondary roads, a rail line, and two dual-lane motorways. **Jasper Ridge** is a $100 \times 100 \times 224$ hyperspectral image. It was also captured using the AVIRIS sensor. **Pavia University** is a $610 \times 340 \times 103$ hyperspectral image. This dataset was captured by the ROSIS-03 aerial instrument in 2001. The dataset was acquired over the University of Pavia. **Cuprite** dataset contains one $614 \times 512 \times 224$ hyperspectral image.

A. Architecture Search

Given an image and our (MLP) parameter budget, which is measured in bits per pixel per band, or bpppb for short), the first goal is to select the MLP structure—i.e., the number of hidden layers and their widths—that is able to represent this image with an acceptable PSNR value. Figure 4 shows PSNR values achieved for different architectures for the four datasets having a fixed bpppb budget. This suggests that network structure, in addition to network capacity, affects how well a network represents the hyperspectral image.

MLP structure is chosen via hyperparameter search, which involves training feasible designs containing the right number of hidden layers having the correct width on a given hyperspectral image. The result of this process is a single MLP that is able to reconstruct the hyperspectral image with the desired PSNR value. The parameters of the final MLP are then quantized to 16-bit (or 8-bit) precision, which leads to further savings in terms of the storage needed to represent the hyperspectral image. Our experiments suggest that reducing the MLP parameters from 32-bit to 16-bit precision did not increase distortion and that it had little effect on the signal-to-noise ratio. However, reducing these parameters to 8-bit precision had an adverse effect.

B. Encoding Considerations

Our method belongs to the class of "slow-encoding-fast-decoding" compression methods. The method needs to train, actually *overfit*, multiple MLPs at encoding (compression) time. This is needed to find the MLP structure that best represents the hyperspectral image given a particular storage budget. Decoding, however, only requires evaluating this MLP at various pixel locations. Decoding is fast. It can be made even faster by exploiting the parallelism inherent to this procedure. The "slow-encoding-fast-decoding" nature of this method makes it well-suited for applications where the hyperspectral image is compressed once only, say at capture time. Obviously, encoding is also compute heavy and that is something we need to keep in mind as we imagine hyperspectral sensors capable of compressing hyperspectral images at capture time using the method proposed here.

We show an example of the overfitting procedure in Figure 5. These plots show the encoding procedure on the four datasets: (1) Indian Pines at 0.2 bpppb; (2) Jasper Ridge at 0.15 bpppb; (3) Pavia University at 0.025 bpppb; and (4) Cuprite at 0.02 bpppb. JPEG, JPEG2000, and PCA-DCT methods do not require iterations. Consequently, their respective PSNR values are denoted with the horizontal dashed lines. The method proposed in this paper is iterative. Note that PSNR values for *ours-16bit* continue to increase with the number of iterations (up to a point). Improvement in PSNR values saturates at around 10K, 15K, 10K, and 5K iterations for Indian Pines, Jasper Ridge, Pavia University, and Cuprite datasets, respectively. This hints at the upper bound on encoding, or compression, time for our method. Note also that at around 2K iteration mark *ours-16bit* method starts to obtain better PSNR values than the other three methods. As stated earlier, our method involves model fitting, which is inherently stochastic. Therefore, throughout the iterative process, we store the model parameters that obtained the highest value for PSNR thus far. In these plots *MAX ours-16bit* denote these PSNR scores. This guarantees that the model does not get worse over time.

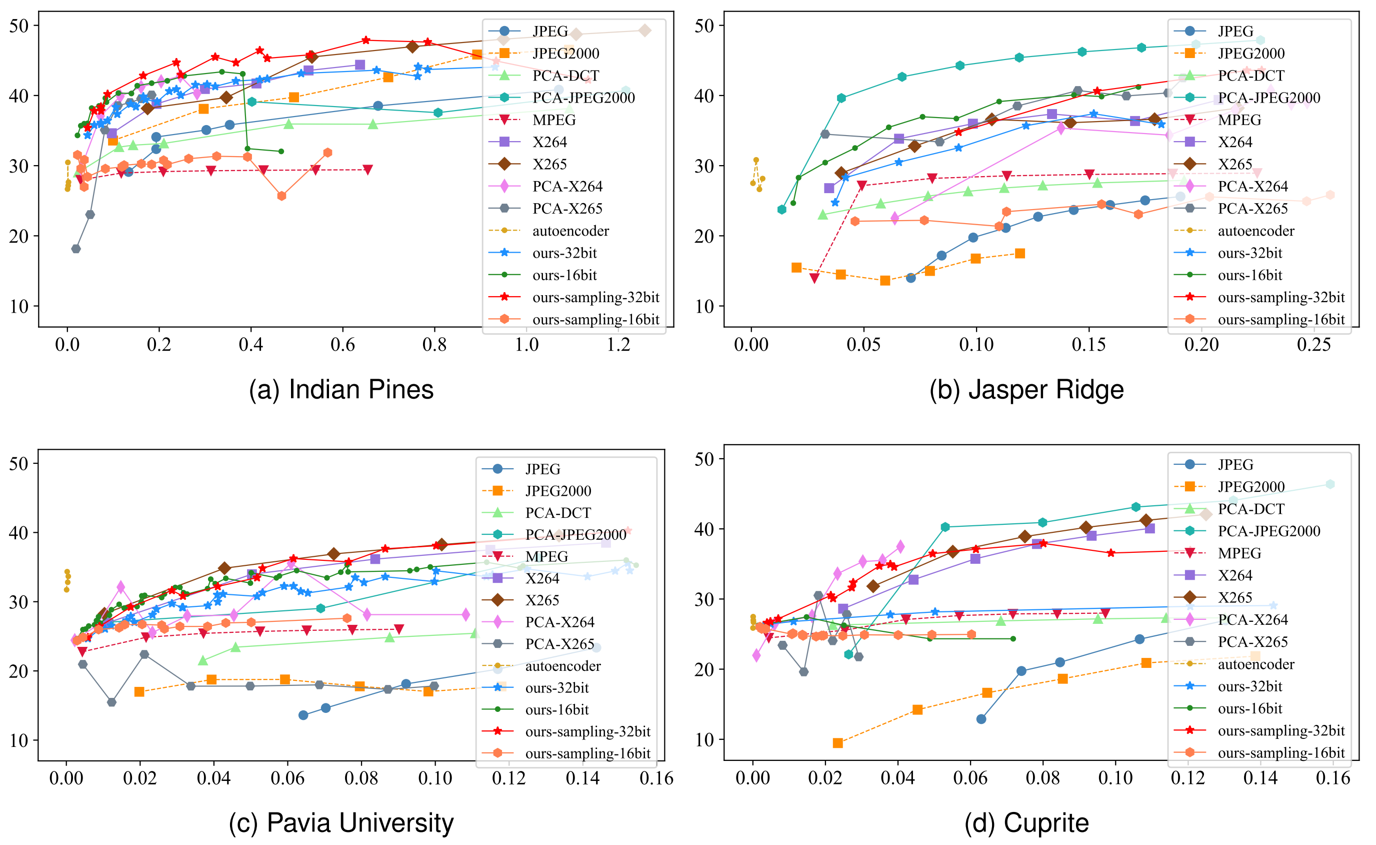


Fig. 7. **PSNR values vs.** bpppb. The x-axis represents bpppb values, and the y-axis represents PSNR values. The uncompressed hyperspectral images have a bpppb value of 16. This figure is best viewed in color.

C. Model Fitting

The number of inputs for all our models is 2, and the number of outputs is equal to the number of channels (or bands) of the hyperspectral image. The activation functions for hidden layers is sinusoidal. We initialized the MLP using the guidelines provided in [57]. Adam optimizer was used during training, and the learning rate was set to $2e-4$. All experiments were conducted on an Intel i7 desktop with Nvidia RTX 2080 GPU.

D. Random Sampling

During compression, the INR model is trained by iterating over $w \times h$ pixel locations for each epoch. For hyperspectral images with high-spatial resolution this leads to longer compression times. Hyperspectral images, similar to color images, exhibit a spatial coherence. We use this assumption and explore the effect of pixel location sampling, hereafter referred to as sampling, during INR model training. The idea is to sample a fraction of pixel locations at training time. Figure 6 illustrates the sampling procedure. First, the image is divided into non-overlapping tiles (in the spatial domain). This is done in order to ensure that the same fraction of pixel locations are sampled from each region of the image. The colored dots indicate the sampled locations within each tile. The sampling is performed with rejection to prevent the case where the same pixel location is sampled multiple times during each epoch. Furthermore, there is no restriction that the same pixel locations are sampled across epochs. The sampling procedure is controlled by two parameters: (1) tiles and (2) the fraction of locations to be sampled within each tile. For the results presented in Table II, the image is divided into 9-by-9 tiles and the sampling fraction is set to 20%. This table shows compression and decompression times for four datasets. As expected sampling reduces the compression times in half for all datasets. What is more intriguing, however, is that for comparable bpppb values, sampling results in higher PSNR values. We do not yet fully understand why this is so. *ours-sampling-32bit* and *ours-sampling-16bit* denote models that used sampling during training. As before prefix *HP* denotes that model parameters are stored using 16-bit precision.

V. Results

We compare our method with a number of (1) classical, (2) learning-based, and (3) video-based schemes. Table III lists compression results in terms of PSNR with a number of classical and learning-based hyperspectral image compression approaches. In order to compare the performance of different methods, the target bpppb is fixed across methods. For the Indian Pines dataset, the target bpppb is set to 0.2, and for the other three datasets, it is set to 0.1. These values are selected since results were available for other methods for these values. Note that the uncompressed bpppb for all datasets is 16. Results for other methods are taken from their respective publications—Cuprite is a popular dataset since most of the

Dataset	Method	bpppb	PSNR ↑
Indian Pines	X264‡ [25]–[27]	0.1	34.61
	X265‡ [25]–[27]	0.1	38.5
	PCA-X264‡ [25]–[27]	0.1	39.8
	PCA-X265‡ [25]–[27]	0.1	38.1
	MPEG‡ [28]	0.1	28.9
	HEVC‡ [33]	0.1	30
	RPM‡ [34]	0.1	31
	ours-sampling-32bit	0.1	**40.1**
Jasper Ridge	X264‡ [25]–[27]	0.15	37.35
	X265‡ [25]–[27]	0.15	36.12
	PCA-X264‡ [25]–[27]	0.15	35.35
	PCA-X265‡ [25]–[27]	0.15	39.94
	MPEG‡ [28]	0.15	28.75
	HEVC‡ [33]	0.15	-
	RPM‡ [34]	0.15	-
	ours-sampling-32bit	0.15	**41.15**
Pavia University	X264‡ [25]–[27]	0.1	37.17
	X265‡ [25]–[27]	0.1	37.90
	PCA-X264‡ [25]–[27]	0.1	28.13
	PCA-X265‡ [25]–[27]	0.1	17.82
	MPEG‡ [28]	0.1	26.01
	HEVC‡ [33]	0.1	-
	RPM‡ [34]	0.1	-
	ours-sampling-32bit	0.1	**38.08**
Cuprite	X264‡ [25]–[27]	0.03	28.6
	X265‡ [25]–[27]	0.03	31.8
	PCA-X264‡ [25]–[27]	0.03	**35.5**
	PCA-X265‡ [25]–[27]	0.03	21.7
	MPEG‡ [28]	0.03	25.5
	HEVC‡ [33]	0.03	25
	RPM‡ [34]	0.03	29
	ours-sampling-32bit	0.03	34.9

TABLE IV
COMPARING THE PROPOSED METHOD (WITH SAMPLING RATE OF 20%) AGAINST VIDEO-BASED SCHEMES. OUR METHOD ACHIEVES BETTER RESULTS THEN ALL OTHER METHODS ON INDIAN PINES, JASPER RIDGE, AND PAVIA UNIVERSITY DATASETS. ADDITIONALLY, OUR METHOD ACHIEVES THE SECOND BEST PSNR VALUE AFTER PCA-X264 ON CUPRITE DATASET. TEXT DECORATION ‡ INDICATES A VIDEO-BASED METHODS.

other methods have results availaBle for this dataset. The last column describes the network structure used for INR learning: $\mathrm{n_h}$ and $\mathrm{w_h}$ refer to the number of hidden layers and the width of these layers, respectively. The proposed method achieves better PSNR than other methods for Indian Pines and Pavia University dataset. However, PCA-JPEG2000 method posts better PSNR than our methods for Jasper Ridge and Cuprite datasets.

A. Compression Rates

Figure 7 shows PSNR values at various compression rates for different methods. Specifically, we compare our method, denoted by the prefix “ours” against JPEG, JPEG2000, PCA-DCT, PCA-JPEG2000, MPEG, X264, X265, PCA-X264, PCA-X265, and an autoencoder-based method [36]. This figure does not include result for *ours-*-8bit*, since our primary focus has been on 32- and 16-bit precision. This is in part due to the fact that storing network parameters in 8-bit precision adversely effects the compression results. The plots suggest that th eproposed method achieves higher compression quality at lower bpppb values (i.e., higher compression rates).

For the Indian Pines dataset, *our-sampling-32bit* achieves better PSNR up to around 0.7 bpppb, at which point X265 obtains better PSNR. What is curious is that the PSNR for*ours-16bit* drops drastically at around 0.4 bpppb. This merits further investigation. *ours-sampling-32bit* method got better PSNR values than all other methods at various bpppb levels. For Jasper Ridge, *ours-sampling-32bit* performs better than other methods except PCA-JPEG2000, which achieves better PSNR for all bpppb values. For Pavia University, *ours-sampling-32bit* method achieves PSNR that is better than every method except X265, whose results are comparable to the proposed method. For Cuprite, *ours-sampling-32bit* is in the top four methods. It achieves the PSNR at lower bpppb values. These results confirm that the proposed model is among the top methods across the four datasets at various bpppb. Additionally, that the proposed method seems well-suited for lower values of bpppb.

B. Video-based Methods

Table IV evaluates *ours-sampling-8bit* against seven video-based methods that treat various channels of a hyperspectral image as frames of a video and employ video coding techniques to achieve compression.[2] The bpppb is fixed at 0.1 for Indian Pines, 0.15 for Jasper Ridge, 0.1 for Pavia Univeristy, and 0.03 for Cuprite dataset. Best results are shown in bold. Our method achieves outperforms all other methods on Indian Pines, Jasper Ridge, and Pavia University datasets, and our method achieves the second best PSNR value on Cuprite dataset. These results also confirm that “sampling,” which not only reduces encoding times, achieves top PSNR scores.

C. SSIM Comparison

Table V uses SSIM metric to compare our method on Cuprite and Pavia University datasets against other schemes for fixed bpppb. The results reported for other methods are taken from their respective publications. SSIM scores for other methods for Indian Pines and Jasper Ridge datasets are not available; therefore, this table does not include Indian Pines and Jasper Ridge datasets. Nevertheless these results suggest that the proposed scheme is able to preserve “perceptual quality” in the compressed signal.

For Cuprite dataset, we compare our method against WSRC [32], 3D-SPECK [35], 3D-SPHIT [70], 3D-WBTC [71], 3D-LSK [35], 3D-NLS [72], 3D-LMBTC [73], and 3D-ZM-SPECK [74]. WSRC is evalauted at bpppb = 0.1 and *ours-sampling-32bit* method outperforms WSRC at this bpppb on Cuprite dataset. 3D-SPECK, 3D-SPHIT, 3D-WBTC, 3D-LSK, 3D-NLS, 3D-LMBTC, and 3D-ZM-SPECK are evaluated at bpppb = 0.01, and *ours-32bit* method achieves better SSIM scores than these methods at this bpppb.

On Pavia University dataset, we compare our method against 3D-SPHIT and 3D-DCT [30] methods. These methods are evaluated at bpppb = 0.1, and *ours-sampling-32bit* method achieves the highest SSIM score for this bpppb value.

[2]It is not possible to include the results presented in Table IV in Table III since the bpppb values used by video-based methods do not match those used by methods listed in Table III.

Dataset	bpppb	Method	SSIM ↑
Cuprite	0.1	WSRC [32]	0.75
		ours-sampling-32bit	**0.9798**
	0.01	3D-SPECK [35]	0.142
		3D-SPIHT [70]	0.136
		3D-WBTC [71]	0.141
		3D-LSK [35]	0.138
		3D-NLS [72]	0.135
		3D-LMBTC [73]	0.140
		3D-ZM-SPECK [74]	0.141
		ours-32bit	**0.9565**
		ours-16bit	0.9514
		ours-sampling-32bit	0.9527
		ours-sampling-16bit	0.9390
Pavia University	0.1	3D-SPHIT [70]	0.4
		3D-DCT [30]	0.85
		ours-32bit	0.9564
		ours-16bit	0.9527
		ours-sampling-32bit	**0.9901**
		ours-sampling-16bit	0.8518

TABLE V
SSIM COMPARISON FOR THE CUPRITE AND PAVIA UNIVERSITY DATASETS AT FIXED bpppb VALUES. VALUES FOR OTHER METHODS ARE TAKEN FROM THEIR RESPECTIVE PUBLICATIONS. CONSEQUENTLY, WSRC IS EVALUATED AT bpppb = 0.1 FOR CUPRITE DATASET, AND ALL OTHER METHODS ARE EVALUATED FOR bpppb = 0.1. AT THE TIME OF WRITING THE SSIM SCORES WERE NOT AVAILABLE FOR OTHER METHODS ON INDIAN PINES AND JASPER RIDGE DATASETS.

VI. Conclusion

We employ implicit neural representations to compress hyperspectral images. Multi-layer perceptron neural networks with sinusoidal activation layers are overfitted to a hyperspectral image. The network is trained to map pixel locations to pixels' spectral signatures. The parameters of the network, along with its structure, represent a compressed encoding of the original hyperspectral image. We also use a sampling method that is controlled by (1) window size and (2) sampling rate to reduce the compression time. We have evaluated our approach on four benchmarks, and we draw the following conclusions: 1) the proposed method (with/without sampling; 32-/16-bit quantization), perform high-quality compression at high compression rates; 2) it is beneficial to perform architecture search plus examine the effects of quantization at compression time since on some datasets *ours-*-16bit* outperforms *ours-*-32bit*; 3) the compression quality obtained by the proposed method is better or in some cases similar to that achieved by other schemes; and 4) sampling improves compression times while maintaining acceptable reconstruction quality as measured by PSNR. Being a learning-based method that requires to overfit an MLP to data at compression time, the proposed method falls into the category of "slow-encoding-fast-decoding" schemes. In the future we plan to investigate techniques to speed up the encoding process and to reduce the computational requirements.

References

[1] M. L EH, "Lightness and retinex theory," *J. Opt. Soc. Am.*, vol. 61, no. 1, pp. 1–11, 1971.

[2] A. F. Goetz, G. Vane, J. E. Solomon, and B. N. Rock, "Imaging spectrometry for earth remote sensing," *science*, vol. 228, no. 4704, pp. 1147–1153, 1985.

[3] P. Ghamisi, N. Yokoya, J. Li, W. Liao, S. Liu, J. Plaza, B. Rasti, and A. Plaza, "Advances in hyperspectral image and signal processing: A comprehensive overview of the state of the art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 37–78, 2017.

[4] M. Govender, K. Chetty, and H. Bulcock, "A review of hyperspectral remote sensing and its application in vegetation and water resource studies," *Water Sa*, vol. 33, no. 2, pp. 145–151, 2007.

[5] E. Adam, O. Mutanga, and D. Rugege, "Multispectral and hyperspectral remote sensing for identification and mapping of wetland vegetation: a review," *Wetlands Ecology and Management*, vol. 18, no. 3, pp. 281–296, 2010.

[6] C. Fischer and I. Kakoulli, "Multispectral and hyperspectral imaging technologies in conservation: current research and potential applications," *Studies in Conservation*, vol. 51, no. sup1, pp. 3–16, 2006.

[7] H. Liang, "Advances in multispectral and hyperspectral imaging for archaeology and art conservation," *Applied Physics A*, vol. 106, no. 2, pp. 309–323, 2012.

[8] O. Carrasco, R. B. Gomez, A. Chainani, and W. E. Roper, "Hyperspectral imaging applied to medical diagnoses and food safety," in *Geo-Spatial and Temporal Image and Data Exploitation III*, vol. 5097. SPIE, 2003, pp. 215–221.

[9] M. A. Afromowitz, J. B. Callis, D. M. Heimbach, L. A. DeSoto, and M. K. Norton, "Multispectral imaging of burn wounds," in *Medical Imaging II*, vol. 914. SPIE, 1988, pp. 500–504.

[10] J. Kuula, I. Pölönen, H.-H. Puupponen, T. Selander, T. Reinikainen, T. Kalenius, and H. Saari, "Using vis/nir and ir spectral cameras for detecting and separating crime scene details," in *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense XI*, vol. 8359. International Society for Optics and Photonics, 2012, p. 83590P.

[11] R. L. Schuler, P. E. Kish, and C. A. Plese, "Preliminary observations on the ability of hyperspectral imaging to provide detection and visualization of bloodstain patterns on black fabrics," *Journal of forensic sciences*, vol. 57, no. 6, pp. 1562–1569, 2012.

[12] R. Padoan, T. Steemers, M. Klein, B. Aalderink, and G. De Bruin, "Quantitative hyperspectral imaging of historical documents: technique and applications," *Art Proceedings*, pp. 25–30, 2008.

[13] G. J. Edelman, E. Gaston, T. G. Van Leeuwen, P. Cullen, and M. C. Aalders, "Hyperspectral imaging for non-contact analysis of forensic traces," *Forensic science international*, vol. 223, no. 1-3, pp. 28–39, 2012.

[14] A. A. Gowen, C. P. O'Donnell, P. J. Cullen, G. Downey, and J. M. Frias, "Hyperspectral imaging–an emerging process analytical tool for food quality and safety control," *Trends in food science & technology*, vol. 18, no. 12, pp. 590–598, 2007.

[15] Y.-Z. Feng and D.-W. Sun, "Application of hyperspectral imaging in food safety inspection and control: a review," *Critical reviews in food science and nutrition*, vol. 52, no. 11, pp. 1039–1058, 2012.

[16] R. N. Clark and G. A. Swayze, "Mapping minerals, amorphous materials, environmental materials, vegetation, water, ice and snow, and other materials: the usgs tricorder algorithm," in *JPL, Summaries of the Fifth Annual JPL Airborne Earth Science Workshop. Volume 1: AVIRIS Workshop*, 1995.

[17] "Autoencoders, minimum description length, and helmholtz free energy," in *Advances in neural information processing systems*, 1994, pp. 3–10.

[18] A. Alemi, B. Poole, I. Fischer, J. Dillon, R. Saurous, and K. Murphy, "Fixing a broken elbo," in *Proc. 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 159–168.

[19] J. Balle', V. Laparra, and E. Simoncelli, "End-to-end optimized image compression," in *Proc. International Conference on Learning Representations (ICLR)*, June 2017.

[20] Y. Strümpler, J. Postels, R. Yang, L. V. Gool, and F. Tombari, "Implicit neural representations for image compression," in *European Conference on Computer Vision*. Springer, 2022, pp. 74–91.

[21] W. F. Good, G. S. Maitz, and D. Gur, "Joint photographic experts group (jpeg) compatible data compression of mammograms," *Journal of Digital Imaging*, vol. 7, no. 3, pp. 123–132, 1994.

[22] T. Qiao, J. Ren, M. Sun, J. Zheng, and S. Marshall, "Effective compression of hyperspectral imagery using an improved 3d dct approach for land-cover analysis in remote-sensing applications," *International Journal of Remote Sensing*, vol. 35, no. 20, pp. 7316–7337, 2014.

[23] Q. Du and J. E. Fowler, "Hyperspectral image compression using jpeg2000 and principal component analysis," *IEEE Geoscience and Remote sensing letters*, vol. 4, no. 2, pp. 201–205, 2007.

[24] Y. Nian, Y. Liu, and Z. Ye, "Pairwise klt-based compression for multispectral images," *Sensing and Imaging*, vol. 17, no. 1, pp. 1–15, 2016.

[25] C. Kwan and J. Larkin, "New results in perceptually lossless compression of hyperspectral images," *Journal of Signal and Information Processing*, vol. 10, no. 3, pp. 96–124, 2019.

[26] C. Kwan, J. Larkin *et al.*, "Perceptually lossless compression for mastcam multispectral images: A comparative study," *Journal of Signal and Information Processing*, vol. 10, no. 04, p. 139, 2019.

[27] C. Kwan, J. Larkin, B. Budavari, and B. Chou, "Compression algorithm selection for multispectral mastcam images," *Signal & Image Processing: An International Journal (SIPIJ)*, vol. 10, no. 1, 2019.

[28] D. Le Gall, "Mpeg: A video compression standard for multimedia applications," *Communications of the ACM*, vol. 34, no. 4, pp. 46–58, 1991.

[29] S. Mei, B. M. Khan, Y. Zhang, and Q. Du, "Low-complexity hyperspectral image compression using folded pca and jpeg2000," in *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2018, pp. 4756–4759.

[30] R. J. Yadav and M. Nagmode, "Compression of hyperspectral image using pca–dct technology," in *Innovations in Electronics and Communication Engineering: Proceedings of the Fifth ICIECE 2016*. Springer, 2018, pp. 269–277.

[31] N. Zikiou, M. Lahdir, and D. Helbert, "Support vector regression-based 3d-wavelet texture learning for hyperspectral image compression," *The Visual Computer*, vol. 36, no. 7, pp. 1473–1490, 2020.

[32] M. Ouahioune, S. Ameur, and M. Lahdir, "Enhancing hyperspectral image compression using learning-based super-resolution technique," *Earth Science Informatics*, vol. 14, no. 3, pp. 1173–1183, 2021.

[33] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[34] M. Paul, R. Xiao, J. Gao, and T. Bossomaier, "Reflectance prediction modelling for residual-based hyperspectral image coding," *PloS one*, vol. 11, no. 10, p. e0161212, 2016.

[35] R. Ngadiran, S. Boussakta, B. Sharif, and A. Bouridane, "Efficient implementation of 3d listless speck," in *International Conference on Computer and Communication Engineering (ICCCE'10)*. IEEE, 2010, pp. 1–4.

[36] R. La Grassa, C. Re, G. Cremonese, and I. Gallo, "Hyperspectral data compression using fully convolutional autoencoder," *Remote Sensing*, vol. 14, no. 10, p. 2472, 2022.

[37] J. Mielikainen and B. Huang, "Lossless compression of hyperspectral images using clustered linear prediction with adaptive prediction length," *IEEE Geosci. Remote Sens. Lett.*, no. 9, p. 1118–1121, 2012.

[38] N. R. M. Noor and T. Vladimirova, "Investigation into lossless hyperspectral image compression for satellite remote sensing," *International Journal of Remote Sensing*, vol. 34, p. 5072–5104, 2013.

[39] C. I. Chang, *Hyperspectral Data Processing: Algorithm Design and Analysis*. Wiley, 2013.

[40] D. Zhao, S. Zhu, and F. Wang, "Lossy hyperspectral image compression based on intra-band prediction and inter-band fractal encoding," *Computers & Electrical Engineering*, vol. 54, pp. 494–505, 2016.

[41] I. Blanes and J. Serra-Sagristà, "Cost and scalability improvements to the karhunen–loêve transform for remote-sensing image coding," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 7, pp. 2854–2863, 2010.

[42] Q. Du and J. E. Fowler, "Hyperspectral image compression using jpeg2000 and principal component analysis," *IEEE Geoscience and Remote sensing letters*, vol. 4, no. 2, pp. 201–205, 2007.

[43] L. Wang, J. Wu, L. Jiao, and G. Shi, "Lossy-to-lossless hyperspectral image compression based on multiplierless reversible integer tdlt/klt," *IEEE Geoscience and remote sensing letters*, vol. 6, no. 3, pp. 587–591, 2009.

[44] B. Penna, T. Tillo, E. Magli, and G. Olmo, "Transform coding techniques for lossy hyperspectral data compression," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 5, pp. 1408–1421, 2007.

[45] L. Zhang, L. Zhang, D. Tao, X. Huang, and B. Du, "Compression of hyperspectral remote sensing images by tensor approach," *Neurocomputing*, no. 147, p. 358–363, 2015.

[46] T. Qiao, J. Ren, M. Sun, J. Zheng, and S. Marshall, "Effective compression of hyperspectral imagery using an improved 3d dct approach for land-cover analysis in remote-sensing applications," *International Journal of Remote Sensing*, vol. 35, no. 20, pp. 7316–7337, 2014.

[47] B. Rasti, J. R. Sveinsson, M. O. Ulfarsson, and J. A. Benediktsson, "Hyperspectral image denoising using 3d wavelets," in *2012 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2012, pp. 1349–1352.

[48] J. González-Conejero, J. Bartrina-Rapesta, and J. Serra-Sagrista, "Jpeg2000 encoding of remote sensing multispectral images with no-data regions," *IEEE Geoscience and Remote Sensing Letters*, vol. 7, no. 2, pp. 251–255, 2009.

[49] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *International Conference on Learning Representations*, 2018.

[50] J. Lee, S. Cho, and S.-K. Beack, "Context-adaptive entropy model for end-to-end optimized image compression," in *International Conference on Learning Representations*, 2018.

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

[52] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Occupancy flow: 4d reconstruction by learning particle dynamics," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 5379–5389.

[53] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 165–174.

[54] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5939–5948.

[55] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic programming and evolvable machines*, vol. 8, no. 2, pp. 131–162, 2007.

[56] 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 *European conference on computer vision*. Springer, 2020, pp. 405–421.

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

[58] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7537–7547, 2020.

[59] E. Dupont, A. Golinski, M. Alizadeh, Y. W. Teh, and A. Doucet, "Coin: Compression with implicit neural representations," in *Neural Compression: From Information Theory to Applications–Workshop@ ICLR 2021*, 2021.

[60] D. Hjelm, R. R. Salakhutdinov, K. Cho, N. Jojic, V. Calhoun, and J. Chung, "Iterative refinement of the approximate posterior for directed belief networks," *Advances in neural information processing systems*, vol. 29, 2016.

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

[62] R. Krishnan, D. Liang, and M. Hoffman, "On the challenges of learning with inference networks on sparse, high-dimensional data," in *International conference on artificial intelligence and statistics*. PMLR, 2018, pp. 143–151.

[63] J. Marino, Y. Yue, and S. Mandt, "Iterative amortized inference," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3403–3412.

[64] J. Djelouah and C. Schroers, "Content adaptive optimization for neural image compression," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit*, vol. 2, 2019.

[65] T. Guo, J. Wang, Z. Cui, Y. Feng, Y. Ge, and B. Bai, "Variable rate image compression with content adaptive optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 122–123.

[66] Y. Yang, R. Bamler, and S. Mandt, "Improving inference for neural image compression," *Advances in Neural Information Processing Systems*, vol. 33, pp. 573–584, 2020.

[67] C. Cremer, X. Li, and D. Duvenaud, "Inference suboptimality in variational autoencoders," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1078–1086.

[68] T. van Rozendaal, I. Huijben, and T. S. Cohen, "Overfitting for fun and profit: Instance-adaptive data compression," in *9th International Conference on Learning Representations, ICLR 2021*. ICLR, 2021.

[69] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[70] J. E. Fowler and J. T. Rucker, “Three-dimensional wavelet-based compression of hyperspectral imagery,” *Hyperspectral Data Exploitation: Theory and Applications*, pp. 379–407, 2007.

[71] S. Bajpai, N. R. Kidwai, and H. V. Singh, “3d wavelet block tree coding for hyperspectral images,” *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 6C, pp. 64–68, 2019.

[72] V. Sudha and R. Sudhakar, “3d listless embedded block coding algorithm for compression of volumetric medical images,” 2013.

[73] S. Bajpai, N. R. Kidwai, H. V. Singh, and A. K. Singh, “Low memory block tree coding for hyperspectral images,” *Multimedia Tools and Applications*, vol. 78, pp. 27 193–27 209, 2019.

[74] ——, “A low complexity hyperspectral image compression through 3d set partitioned embedded zero block coding,” *Multimedia Tools and Applications*, vol. 81, no. 1, pp. 841–872, 2022.