Onboard Deep Lossless and Near-lossless Predictive Coding of Hyperspectral Images with Line-based Attention

Diego Valsesia, *Member, IEEE*, Tiziano Bianchi, *Member, IEEE* and Enrico Magli, *Fellow, IEEE*

***Abstract*—Deep learning methods have traditionally been difficult to apply to compression of hyperspectral images onboard of spacecrafts, due to the large computational complexity needed to achieve adequate representational power, as well as the lack of suitable datasets for training and testing. In this paper, we depart from the traditional autoencoder approach and we design a predictive neural network, called LineRWKV, that works recursively line-by-line to limit memory consumption. In order to achieve that, we adopt a novel hybrid attentive-recursive operation that combines the representational advantages of Transformers with the linear complexity and recursive implementation of recurrent neural networks. The compression algorithm performs prediction of each pixel using LineRWKV, followed by entropy coding of the residual. Experiments on multiple datasets show that LineRWKV is highly memory-efficient, significantly outperforms state-of-the-art deep learning methods and is the first deep learning approach to outperform CCSDS-123.0-B-2 at lossless and near-lossless compression. Promising throughput results are also evaluated on a 7W embedded system.**

***Index Terms*—Hyperspectral image compression, deep learning, self-attention, RWKV, predictive coding**

I. Introduction

HYPERSPECTRAL images acquired by spacecrafts are an essential tool for Earth observation, playing pivotal roles in environmental monitoring, urban planning, tackling climate change and much more [1]–[6]. The spatial and spectral resolution of instruments keeps growing to satisfy the demands of final users. However, this poses significant challenges for the transmission and management of huge amounts of spatial and spectral information, making compression a topic of paramount importance. Even modest gains in rate-distortion performance may result in significant bandwidth reduction. This is critical for many tasks where downlink bandwidth is a limiting factor in the timeliness of the delivery of the products or their resolution [7]. The challenge of onboard compression of hyperspectral images is far from trivial, as it requires methods that are efficient in processing massive amounts of data with the limited computational resources that are available on a satellite, and, at the same time, are able to capture complex spatial-spectral redundancy patterns.

The authors are with Politecnico di Torino – Department of Electronics and Telecommunications, Italy. email: {name.surname}@polito.it. This study was carried out within the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

Deep learning has shown great promise for the task of RGB image and video compression [8]–[13]. However, the literature on deep learning for hyperspectral image compression, particularly for usage onboard of spacecrafts, is still in its infancy, due to the complexity of this data type. In fact, effective compression of hyperspectral images requires to capture both spatial and spectral correlation patterns. When a neural network is to be designed for this task, it requires processing of the entire 3D data cube to extract spatio-spectral features that are relevant for the compression task. But in doing so, complexity can grow very rapidly both in the amount of operations to be performed by neural layers, and, even more so, in memory requirements. This is clearly at odds with the limited computational resources available onboard.

The literature on deep hyperspectral image compression is currently focused on designs based on autoencoders, following seminal works for RGB images [8]. In this kind of approach, the neural network generally consists of an encoder which maps the image into a compact latent space and a decoder which reconstructs the image. Several designs are possible depending on the focus on spatial or spectral features, complexity tradeoffs, etc. For instance, some works [14], [15] limit complexity by just focusing on spectral redundancy, while others seek complex spatio-spectral representations [16]–[20] with various operations and training objectives to enhance the quality of the features. Several of these works do not attempt to carefully control complexity for onboard usage. The work by Verdú et al. [21] started posing the question of a model with a complexity that would be suitable for onboard usage and presented a design accordingly.

Common to the entire existing literature on deep hyperspectral image compression is the low-bitrate setting in which all these autoencoder methods are tested. Results are typically presented at rates lower than 1 bit per pixel per channel (bpppc), corresponding to very high compression ratios, and quality levels that would generally be unacceptable for real remote sensing missions. This is the regime where such methods are most competitive, due to an intrinsic limitation of the autoencoder design. Indeed, reconstructing an image from a latent space with very high fidelity requires neural networks with large representational power, both in the encoder and decoder parts. Due to this, existing designs do not scale well to a high-rate, high-quality regime, presenting a plateau in rate-distortion performance as rate is increased, or exploding complexity. In fact, as we show in this paper, the state-of-art in deep learning compression of hyperspectral images is

significantly outperformed by classic approaches, such as the low-complexity CCSDS-123.0-B-2 standard [22] for onboard lossless and near-lossless compression, when targeting the high-quality regime. The lack of lossless compression methods is another limitation of current literature that we address in this work. Hence, it is clear that in order to address the weaknesses of the existing literature in the high-quality regime, both in terms of rate-distortion performance and low-complexity designs, we need to depart from the autoencoder approach.

In this paper, we present a novel neural network design, called LineRWKV, capable of lossless and lossy compression. The design departs from existing works in several ways. First, it follows a predictive coding approach, where the neural network predicts a pixel value from a causal context and only the prediction residual is entropy-coded. This is to address the aforementioned limitation of autoencoders where reconstruction from the bottleneck is hard to scale with limited complexity, since a predictor can be simpler to design. This also allows to readily target lossless compression, which is currently neglected by the literature, as well as lossy compression with a bounded error (so-called near-lossless). Moreover, the proposed architecture has a radical novel design to constrain complexity, particularly regarding memory usage. We propose a neural network architecture that can work recursively in the along-track direction. This limits memory usage, as it is only tied to one image line with all its spectral channels, while exploiting a memory of past lines and enabling continuous pushbroom operation. Critically, we use a novel hybrid attentive-recursive operation [23] which avoids pitfalls of older recurrent neural networks (RNNs), enabling parallelized training, and having the representational power of Transformers [24], with the advantage of linear complexity instead of quadratic. While [23] has been proposed for natural language processing tasks, we are the first to adapt this architecture to image processing. We conduct extensive experiments on three datasets to show that, for the first time for a deep-learning method, we can significantly outperform CCSDS-123.0-B-2 in rate-distortion performance. We also conduct tests on a 7W embedded system to validate promising throughput results, matching our low-complexity design goals.

In summary, the main contributions presented in this paper are:

- the first scheme for deep hyperspectral image compression departing from autoencoders by adopting predictive coding;
- a novel low-complexity architecture that constrains memory and computational requirements by working line-by-line with a novel hybrid-recurrent scheme;
- the first deep hyperspectral image compression for lossless compression;
- the first deep learning method to outperform CCSDS-123.0-B-2 in the high-rate, high-quality regime.

A preliminary version of this work [25] introduced the general idea, while this paper significantly expands the methodology and experimental results. The manuscript is organized as follows. Sec. II introduces some background material on methods for compression of remote sensing hyperspectral images, both traditional and based on deep learning, as well as background on deep sequence processing which is useful to understand the proposed method. Sec. III presents the technical description of the proposed method. Sec. IV reports compression results on multiple datasets. Finally, Sec. V draws some conclusions and discusses the main limitations of the method for future improvement.

II. Background

A. Compression onboard of spacecrafts

Onboard hyperspectral image compression faces a challenging balance between rate-distortion performance and computational complexity. Over the years, CCSDS standards have gained popularity due to their low computational demands, enabling high throughput on dedicated FPGAs [26], and good rate-distortion performance. In particular, CCSDS 122.0-B-2 [27] together with CCSDS 122.1-B-1 [28] employ a transform coding approach to lossless and lossy compression, where a 2D discrete wavelet transform is coupled with a spectral transform. The more recent CCSDS 123.0-B-2 [22] follows instead a predictive coding approach to lossless and near-lossless (i.e., lossy with bounded error) compression with a spatial-spectral predictor based on an adaptive filter. CCSDS 123.0-B-2 is regarded as the state-of-the-art for onboard hyperspectral image compression and, to the best of our knowledge, remains unchallenged by current deep-learning methods. Finally, it worth mentioning that Regression Wavelet Analysis (RWA) [29], [30] combines a spectral transform with 2D JPEG2000 to achieve compression performance competitive with CCSDS 123.0-B-2.

B. Image compression with deep learning

Most of the works in the literature of image compression with deep learning stem from the computer vision community, focusing on 8-bit RGB or grayscale images [8]–[13].

Few works [14]–[21], [31] have addressed the topic of hyperspectral images, especially while keeping complexity in mind for possible onboard usage. The general approach mainly follows the early work of Ballé et al. [8], in the design of autoencoder neural networks where an encoder subnetwork creates a latent representation of the input image in a low-dimensional space which is then quantized and entropy-coded. Sometimes encoder features are also used to derive a hyperprior on the distribution of the low-dimensional latent representation. A decoder subnetwork maps the latent representation back to the image space. These networks are trained end-to-end with rate-distortion objectives, balancing, according to the desired tradeoff, the reconstruction error and the entropy of the latent representation. Recent advancements for RGB images focused on improving encoder architectures and context models for the entropy coder [12]. Regarding hyperspectral images, the main challenge lies in the extremely high memory requirements to capture 3D spatial-spectral features for a compressed latent representation. The works by Kuester et al. [14] and La Grassa et al. [15] focus on designing a spectral autoencoder. This limits the computational complexity and captures a significant amount

of correlation, as it is known that hyperspectral images are more spectrally than spatially correlated, but does not exploit spatial redundancy whcih might be significant. Most other works fully exploit spatio-spectral features, e.g., via the simple convolutional autoencoders designed in [16] and [31]. Guo et al. [18] propose a variational autoencoder with an edge extractor to provide guidance to better represent edges. In subsequent work, Guo et al. [19] use cross-channel contrastive learning to enhance spectral features, but not do not attempt to optimize complexity or memory usage. In a recent work targeting onboard compression, Verdú et al. [21] adopt a channel clusterization strategy as well as novel normalization strategies needed for images with high dynamic range. While all these methods based on autoencoders work well at low rates, typically being tested below 1 bpppc, they are difficult to scale to high-rate, high-quality regimes without compromising efficiency. In fact, the autoencoding approach tends to struggle when very low distortion is desired, if the network capacity in terms of number of features, latent space size and total number of parameters is not adequately scaled.

Finally, it is worth mentioning that some autoregressive approaches [32], [33] have been studied on RGB images. They generally rely on modeling the image as a sequence of all pixels, and use a causal context for prediction within the mechanism of either causal convolution, RNNs or Transformers. As detailed in the next section, until recently, each of these approaches had a critical limitation, consisting in either limited representational power, or inefficient training and encoding, or high complexity. We also remark that the proposed method is not purely autoregressive on all image pixels, since the across-track dimension can be effectively processed in parallel, as detailed in Sec. III-G.

C. Deep sequence processing

Since the proposed method is based on a causal prediction approach, it is worth reviewing the main ways in which deep neural networks are used to process sequences and their respective tradeoffs. For simplicity we are going to discuss processing of a 1D sequence.

First, a simple design would be causal convolution, such as that of PixelCNN [34], where the convolution kernel is masked so that the receptive field only expands in the past samples. While this approach enjoys fairly efficient implementations, it has limited representational power as it is not an input-dependent operation (like Attention is), and the size of the receptive field, i.e., how far past samples affect the current prediction, might be limited.

An alternative approach would be the use of RNNs, such as the LSTM [35], to have ideally infinite memory. This approach is limited by the inefficiency of training which requires serial computations over the entire sequence length, so that it does not scale to the large datasets required to train even moderately complex models.

Transformers [24] are currently the state of the art for sequence processing thanks to the high representational power of the input-dependent attention operation, which is capable of creating operations that are adaptive to the sequence under processing. Unlike RNNs, Transformers can also be parallelized during training. However, they suffer from quadratic complexity in the sequence length and require to keep the entire sequence and its features in memory, resulting in high memory and computational cost. This renders them infeasible to process an image in an autoregressive pixelwise fashion.

Very recently, the natural language processing literature has focused its attention towards sequence processing with hybrid designs between RNNs and Transformers which simultaneously have linear complexity, admit parallel training and can be written in a recursive manner for low-memory inference [23], [36]. RWKV [23] is a recent model with such properties, which is at the basis of the proposed work, as detailed in Sec. III.

Fig. 1: Overview of compression scheme for near-lossless compression. A raw image is prequantized by means of a uniform scalar quantizer, then fed to a causal prediction model implemented by the LineRWKV neural network, producing prediction residuals that are entropy-coded.

III. Proposed method

A. Overview

In this section, we present the full design of the proposed hyperspectral compression method, shown in Fig. 1. This is a predictive coding method, where a pixel is predicted from a causal spatial and spectral context, and only the prediction error is encoded. The method comprises a prequantizer to introduce losses, followed by a deep-learning based predictor, called LineRWKV, and an entropy encoder to code the prediction residuals.

Regarding the prequantizer module, lossy compression, in the context of predictive coding algorithms, can be achieved in two ways: in-loop quantization of prediction errors or image prequantization. In-loop quantization predicts a pixel value, computes a prediction residual and quantizes it. However, in order to complete prediction for the next pixel values, it is necessary to reconstruct the pixel value from the quantized residual. This is to ensure that the decoder can perform the same prediction operation. However, it introduces serial data dependencies. On the other hand, prequantization consists in quantizing the input image and then using lossless prediction. This avoids data dependencies at the compressor since there is no discrepancy between the original pixel value and the value to be used for prediction, so it leads to higher compression throughput. While in-loop quantization is theoretically superior, this leads to inefficient implementations and, at the high rates desirable for usage in real missions, which are the focus of this paper, the rate-distortion penalty of prequantization with respect to in-loop quantization is minimal, as shown in [37]. Therefore, this work adopts a prequantization block using a uniform scalar quantizer with odd-integer step size Q to obtain the prequantized image $I^q = \mathrm{round}(I/Q)$.

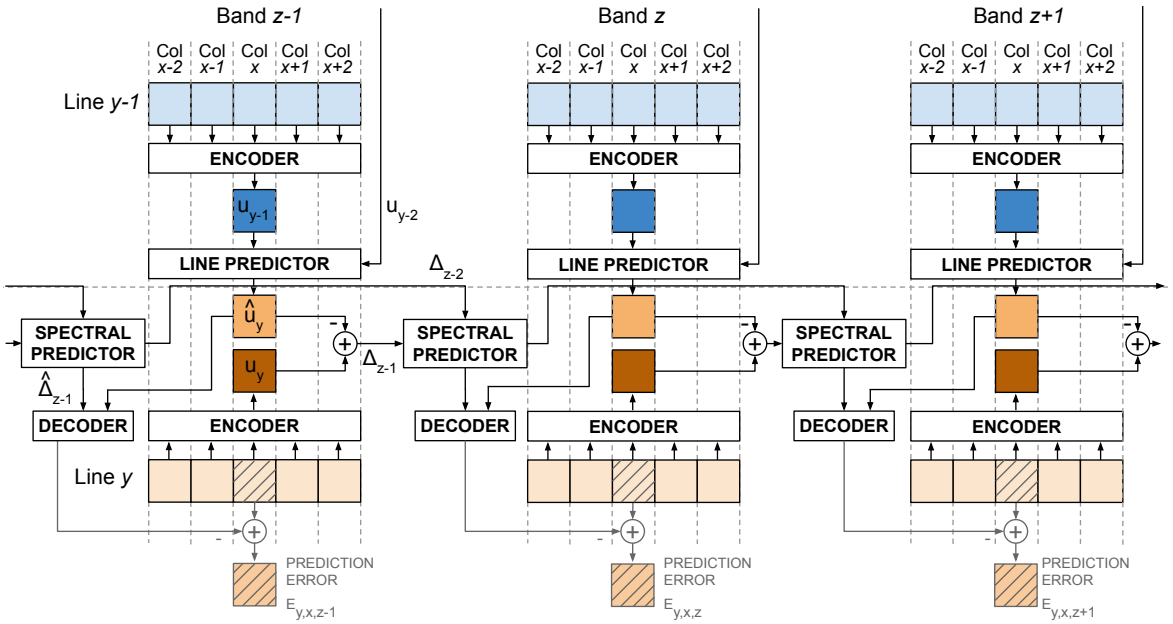


Fig. 2: Overview of LineRWKV predictor for $z > 1$ and $y > 0$. Encoder, decoder, line and spectral predictors are neural networks trained end-to-end to minimize the prediction error for all pixels. Each square represents a pixel in the corresponding spatial-spectral position, as it is processed by the neural modules. Rounded prediction errors are then entropy-coded with a standard code.

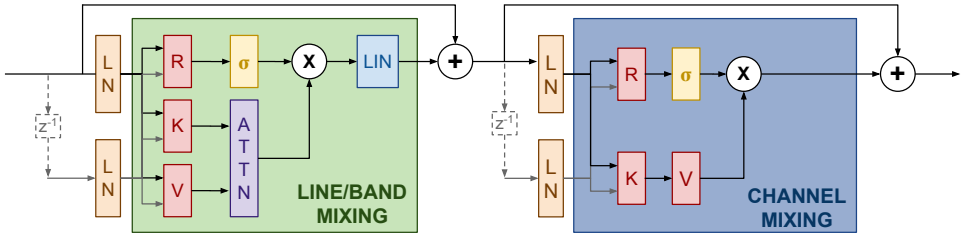


Fig. 3: Line and spectral predictors are made of the repetition of line/band mixers and channel mixers according the depicted architecture. z^{-1} denotes unit delay in the sequence dimension (lines or bands). LN denotes LayerNorm and LIN denotes linear projection of features.

The entropy encoder module can use any method with suitable complexity. In this work, we use the sample-adaptive Golomb encoder defined by the CCSDS-123.0-B-2 standard [22] as entropy coder for the positive-mapped prediction residuals to ensure fair comparisons with the CCSDS predictor.

LineRWKV is a neural network implementing a predictor, i.e., it estimates the value of a pixel based on a context (network receptive field) of past spatial and spectral pixels. For clarity of explanation, we will describe its behavior in inference mode for the compression stage. Discrepancies with respect to the training process are discussed in Sec. III-G. A high-level overview of LineRWKV is presented in Fig. 2 and Alg. 1 recaps the operations performed by the modules presented in the next sections. We denote sampling of the along-track direction of the satellite with the term "lines" and variable $y = 0, 1, 2, \ldots$; the across-track direction is denoted as columns and variable $x = 0, 1, 2, \ldots, N_x - 1$; the spectral direction is denoted as $z = 0, 1, 2, \ldots, N_z - 1$. LineRWKV does not pose any restrictions on the extent of the three dimensions, which may be different between training and testing. In particular, the maximum number of columns and bands is dictated by available memory, while the number of lines can be infinite, supporting continuous

Algorithm 1 LineRWKV predictor

Require: Prequantized image $\mathbf{I}^q = \text{round}(\mathbf{I}/Q)$
Require: Init $\mathbf{u}_{-1,x,z} = \mathbf{0}$ $\forall x, z$
Require: Init states $\mathbf{a}_{-1,x,z}, \mathbf{b}_{-1,x,z} = \mathbf{0}$ $\forall x, z$
  Compress first line (Sec. III-F) to get $E_{0,x,z}$ $\forall x, z$
  for $y = 1, \ldots$ **do**
    $\mathbf{u}_{y-1,x,z} \leftarrow \text{Encoder}(\mathbf{I}^q_{y-1,:,z})$ $\forall x, z$
    $\hat{\mathbf{u}}_{y,x,z}, (\mathbf{a}, \mathbf{b})_{y-1,x,z} \leftarrow \text{LinePredictor}(\mathbf{u}_{y-1,x,z}, \mathbf{u}_{y-2,x,z}, \mathbf{a}_{y-2,x,z}, \mathbf{b}_{y-2,x,z})$ $\forall x, z$
    $E_{y,x,0} = \text{Decoder}_0(\hat{\mathbf{u}}_{y,x,0}) - I^q_{y,x,0}$ $\forall x$
    $\boldsymbol{\Delta}_{y,x,z-1} = \mathbf{u}_{y,x,z-1} - \hat{\mathbf{u}}_{y,x,z-1}$ $\forall x, z \geq 1$
    $\hat{\boldsymbol{\Delta}}_{y,x,z} = \text{SpectralPredictor}(\{\boldsymbol{\Delta}_{y,x,\zeta}\}_{\zeta<z})$ $\forall x, z \geq 1$
    $\hat{I}^q_{y,x,z} = \text{Decoder}(\hat{\boldsymbol{\Delta}}_{y,x,z} + \hat{\mathbf{u}}_{y,x,z})$ $\forall x, z \geq 1$
    $E_{y,x,z} = \hat{I}^q_{y,x,z} - I^q_{y,x,z}$ $\forall x, z \geq 1$
  end for

pushbroom acquisition. The prediction for pixel $I^q_{y=Y,x=X,z=Z}$ depends on all pixels with $y \leq Y$, $z < Z$, $\forall x$. We note that the context used by LineRWKV for prediction is slightly different from conventional non-neural designs. Specifically, the predictor employs all lines above the current pixel, and all previous bands for the current pixel, but it does not use information from the previous pixels on the same line or future bands in past lines. This is due to the LineRWKV design, which employs separate blocks in the spectral, along-track and across-track dimensions. This choice is slightly suboptimal; it is similar in principle to the "narrow" mode of the CCSDS-123.0-B-2 predictor, which incurs a small loss of compression efficiency [37], but it is necessary in order to avoid a much more complex structure of the neural predictor. The predicted value is rounded and subtracted from the original pixel value to compute the prediction error. Finally, the prediction error is losslessly encoded with any entropy encoder. At any time during inference, the last two lines of the image with all their spectral channels (i.e., a tensor of size $2 \times N_x \times N_z$) are supposed to be under processing to predict the next line, which is also available for the computation of the residual.

The LineRWKV predictor has a modular architecture design composed of the following blocks: i) an *encoder* to map each pixel in a line and given band into a feature space by exploiting across-track spatial correlation; ii) a *line predictor* that predicts the features of the co-located (same x, z) pixel in the next line; iii) a *spectral predictor* that predicts the features of the co-located (same x, y) pixel in the next band; iv) a *decoder* that estimates a pixel value based on its features. The following sections explain each of the blocks in detail. In the following, we suppose that we are predicting the values for line y from current line $y-1$ and past line $y-2$, as depicted in Fig. 2.

B. Encoder

The goal of the encoder is to capture the correlation that exists across image columns and encode it into a feature space for further processing. The encoder function is thus a 1D operation that is shared for all bands and for all lines. A simple design for the encoder is a sequence of blocks composed of 1D convolution, layer normalization [38], and non-linear activation. This design should be modulated according to the expected complexity of the across-track correlation patterns, possibly considering larger receptive fields and attention operations, if complexity allows. For our experiments, we choose the simple convolutional blocks and we denote the number of such blocks with N_{enc}. The encoding of the input line will result in a feature vector $\mathbf{u}$ for each spatial-spectral pixel (i.e., a tensor of size $1 \times N_x \times N_z \times F$).

C. Line Predictor

The line predictor is the core operation of LineRWKV, which enables the recursion over the image lines, constraining memory usage and enabling continuous operation. The line predictor should exploit the features of a number of past lines in order to predict the next line. As discussed in Sec. II-C, it is desirable to have an attention-based operation such as the scaled dot-product attention of the Transformer to process the sequence of lines, both to parallelize training and to exploit the power of attention to adapt to self-similar patterns. However, Transformers require keeping several lines and their features in memory, which is prohibitive, and have quadratic computational cost in the number of lines, which is also prohibitive. On the other hand, training a traditional RNN on large-scales would be extremely slow and would not scale well. Our line predictor is therefore based on the RWKV neural network [23] for sequence processing, and its recurrent implementation is used for inference.

More in detail, each of the $N_x N_z$ pixels in the current line, represented as a feature vector by the across-track encoder, is processed in parallel by the line predictor. The RWKV-based line predictor is composed of the repetition of two fundamental blocks: line mixing and channel mixing, both depicted in Fig. 3. Overall, N_{lp} line and channel mixing blocks are used in our architecture. For brevity of notation, we denote the feature vector of a pixel input to the line mixing operation as $\mathbf{u}_{y-1} \in \mathbb{R}^F$ and its corresponding output as $\mathbf{o}_{y-1} \in \mathbb{R}^F$, omitting subscripts x, z. In particular, line mixing performs the following operations:

$$\mathbf{r}_{y-1} = \mathbf{W}_r(\mu_r \mathbf{u}_{y-1} + (1-\mu_r)\mathbf{u}_{y-2}) \tag{1}$$
$$\mathbf{k}_{y-1} = \mathbf{W}_k(\mu_k \mathbf{u}_{y-1} + (1-\mu_k)\mathbf{u}_{y-2}) \tag{2}$$
$$\mathbf{v}_{y-1} = \mathbf{W}_v(\mu_v \mathbf{u}_{y-1} + (1-\mu_v)\mathbf{u}_{y-2}) \tag{3}$$

$$\mathbf{a}_{y-1} = e^{-\boldsymbol{\alpha}} \odot \mathbf{a}_{y-2} + e^{\mathbf{k}_{y-1}} \odot \mathbf{v}_{y-1} \tag{4}$$
$$\mathbf{b}_{y-1} = e^{-\boldsymbol{\alpha}} \odot \mathbf{b}_{y-2} + e^{\mathbf{k}_{y-1}} \tag{5}$$
$$\mathbf{p}_{y-1} = \frac{\mathbf{a}_{y-2} + e^{\boldsymbol{\beta}+\mathbf{k}_{y-1}} \odot \mathbf{v}_{y-1}}{\mathbf{b}_{y-2} + e^{\boldsymbol{\beta}+\mathbf{k}_{y-1}}} \tag{6}$$
$$\mathbf{o}_{y-1} = \mathbf{W}_o(\sigma(\mathbf{r}_{y-1}) \odot \mathbf{p}_{y-1}) \tag{7}$$

where $\mathbf{W}_*$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ are trainable parameters, μ_* some hyperparameters and σ the sigmoid function. The symbol $\odot$ denotes elementwise product and exponentiation to a vector is intended elementwise. The states in the recursion are initialized as $\mathbf{a} = \mathbf{b} = \mathbf{0}$.

With some abuse of notation, also denoting with $\mathbf{u}_{y-1} \in \mathbb{R}^F$ the input to the channel mixing module and its corresponding output as $\mathbf{o}_{y-1} \in \mathbb{R}^F$, the module performs the following operations:

$$\mathbf{r}_{y-1} = \mathbf{W}'_r(\mu'_r \mathbf{u}_{y-1} + (1-\mu'_r)\mathbf{u}_{y-2}) \tag{8}$$
$$\mathbf{k}_{y-1} = \mathbf{W}'_k(\mu'_k \mathbf{u}_{y-1} + (1-\mu'_k)\mathbf{u}_{y-2}) \tag{9}$$
$$\mathbf{o}_{y-1} = \sigma(\mathbf{r}_{y-1}) \odot \left(\mathbf{W}'_v \max(\mathbf{k}_{y-1}, 0)^2\right) \tag{10}$$

where $\mathbf{W}'_*$, are trainable parameters, μ'_* some hyperparameters, and σ the sigmoid function.

It can be noticed that the recursion requires a limited amount of memory to be implemented. In particular, besides the features of the current line, the only feature vectors that need to be stored are the states $\mathbf{a}_{y-2}$ and $\mathbf{b}_{y-2}$ and features of the previous line $\mathbf{u}_{y-2}$. This needs to be done for each layer of line and channel mixing that is used. However, this operation allows to keep a memory of past lines and use the attention mechanism for the prediction of the next line and is significantly advantageous in terms of memory consumption compared to keeping the features of a large number of lines.

We refer the reader to [23] for details about a numerically-stable implementation.

The output $\mathbf{o}_{y-1}$ of the last channel mixing module in the Line Predictor is summed with the input of the module in a residual fashion, as depicted in Fig. 3, to generate $\hat{\mathbf{u}}_y \in \mathbb{R}^F$, i.e., a feature-space prediction of the co-located pixel in line y, as shown in Alg. 1.

D. Spectral Predictor

The output $\hat{\mathbf{u}}_y$ of the line predictor is a feature vector for each spatial-spectral pixel that can be thought to be a prediction of the features of line y. While this captures the spatial correlation in the across- and along-track directions, the model has not yet exploited correlation in the spectral dimension. We propose to first compute a feature-domain spatial prediction residual and then model its spectral correlation. In particular, the pixels in line y are encoded with the encoder in Sec. III-B and the difference between them and the output of the line predictor applied to line $y-1$ forms the spectral sequence of feature-domain spatial residuals:

$$\boldsymbol{\Delta}_{y,x,z} = \mathbf{u}_{y,x,z} - \hat{\mathbf{u}}_{y,x,z}, \tag{11}$$

which in the following denote in short as $\boldsymbol{\Delta}_z$, omitting subscripts x, y. Notice that $\boldsymbol{\Delta}_{y,x,z}$ requires pixel $I^q_{y,x,z}$ for its computation. Therefore, this needs to have been already decoded and its availability is ensured by treating the first band as a special case, discussed in Sec. III-F.

A causal model over the $\boldsymbol{\Delta}_z$ sequence is needed to obtain features describing the pixel to be predicted. For this spectral prediction model, we propose to also use RWKV blocks. However, for a lossless (or prequantized) compressor, the parallel implementation typically reserved for training is used rather than the recurrent implementation described in Sec. III-C in order to ensure high throughput. This consists in a sequence of N_{sp} band mixing and channel mixing blocks, with the same architecture previously depicted in Fig. 3. The parallel implementation keeps the features for all the bands in memory and computes[1]:

$$\mathbf{r}_{z-1} = \mathbf{W}_r(\mu_r \boldsymbol{\Delta}_{z-1} + (1-\mu_r)\boldsymbol{\Delta}_{z-2}) \tag{12}$$

$$\mathbf{k}_{z-1} = \mathbf{W}_k(\mu_k \boldsymbol{\Delta}_{z-1} + (1-\mu_k)\boldsymbol{\Delta}_{z-2}) \tag{13}$$

$$\mathbf{v}_{z-1} = \mathbf{W}_v(\mu_v \boldsymbol{\Delta}_{z-1} + (1-\mu_v)\boldsymbol{\Delta}_{z-2}) \tag{14}$$

$$\mathbf{p}_{z-1} = \frac{\sum_{i=1}^{z-2} e^{-(z-1-i)\boldsymbol{\alpha}+\mathbf{k}_i} \odot \mathbf{v}_i + e^{\boldsymbol{\beta}+\mathbf{k}_{z-1}} \odot \mathbf{v}_{z-1}}{\sum_{i=1}^{z-2} e^{-(z-1-i)\boldsymbol{\alpha}+\mathbf{k}_i} + e^{\boldsymbol{\beta}+\mathbf{k}_{z-1}}} \tag{15}$$

$$\mathbf{o}_{z-1} = \mathbf{W}_o(\sigma(\mathbf{r}_{z-1}) \odot \mathbf{p}_{z-1}) \tag{16}$$

for the band mixing block, and:

$$\mathbf{r}_{z-1} = \mathbf{W}'_r(\mu'_r \mathbf{u}_{z-1} + (1-\mu'_r)\mathbf{u}_{z-2}) \tag{17}$$

$$\mathbf{k}_{z-1} = \mathbf{W}'_k(\mu'_k \mathbf{u}_{z-1} + (1-\mu'_k)\mathbf{u}_{z-2}) \tag{18}$$

$$\mathbf{o}_{z-1} = \sigma(\mathbf{r}_{z-1}) \odot \left(\mathbf{W}'_v \max(\mathbf{k}_{z-1}, 0)^2\right) \tag{19}$$

for the channel mixing block. The output $\mathbf{o}_{z-1}$ of the last channel mixing module in the Spectral Predictor is summed with the input of the module in a residual fashion, as depicted in Fig. 3, to generate a feature vector $\hat{\boldsymbol{\Delta}}_z$ for each pixel which should ideally represent the spatial-spectral residual in a feature domain. Also notice that spectral prediction is performed on the difference sequence $\boldsymbol{\Delta}_z$ rather than on the outputs of the line predictors as this allows to use the information of the pixel in the same line to be encoded but from previous bands, which is typically the highest source of correlation.

[1]With some abuse of notation we reuse symbols from the line predictor, but they represent different activations and weights.

E. Decoder

A decoder neural network produces a prediction of the raw pixel value from the feature-domain representation of the spatial prediction $\hat{\mathbf{u}}_{y,x,z}$ and spatial-spectral error $\hat{\boldsymbol{\Delta}}_{y,x,z}$, as shown in Alg. 1. This is obtained by feeding their sum to a sequence of N_{dec} blocks composed of 1×1 convolution, LayerNorm and non-linear activation.

The predicted pixel value is then denormalized, rounded to the nearest integer and the difference with respect to the original pixel value forms the prediction error, which is entropy-coded with a suitable technique. Notice that this approach differs somewhat from works addressing lossless compression of natural images with deep learning [13]. In those works, it is more typical to let the neural network produce the prediction as a probability distribution over the set of possible symbols (typically 256 for 8-bit images). However, the large number of symbols (e.g., 2^{16}) of satellite images poses efficiency challenges in terms of computation and memory requirements and incurs in the "softmax bottleneck" issue due to the number of classes exceeding the number of features [39]. The proposed floating-point regression with rounding solves these issues but incurs in subtle numerical conditions that must be managed to ensure decodability. Whenever there is a mismatch in the hardware-software stack between the compressor and the decompressor, the prediction will only be accurate down to numerical precision, i.e., about 7 significant digits on the normalized value, for computations in FP32. This means that a numerical perturbation might cause the predicted value to cross the rounding threshold, causing a decoding error. In order to avoid this, whenever the prediction is closer to the rounding threshold than 10^{-3} (in unnormalized integer digital numbers), we signal the side of the threshold as extra information. This choice has been experimentally verified to ensure correct decoding on the entire HySpecNet-11k test set. The side information is entropy-coded and in our tests incurred a small penalty of 0.02 bpppc, already included in all experimental results.

F. Special cases

In order to ensure causality of the model, the first line for all bands and the entire first band need to be compressed with a separate method. For the first line, we use a simple DPCM encoder where the prediction error is computed as:

$$E_{0,x,z} = \begin{cases} I^q_{0,0,0}, & \text{for } x = 0, z = 0 \\ I^q_{0,x,z} - I^q_{0,x-1,0}, & \text{for } x > 0, z = 0 \\ I^q_{0,x,z} - I^q_{0,x,z-1} & \text{for } x \geq 0, z > 0. \end{cases}$$

TABLE I: LineRWKV architecture configurations.

Model Size	Params	N_{enc}	N_{lp}	N_{sp}	N_{dec}	F	FLOPS/sample
XS	30k	1	2	2	1	32	120k
S	135k	2	2	2	2	64	508k
M	286k	4	4	4	4	64	1M
L	900k	4	6	6	4	96	3.2M

For the first band, we employ spatial prediction only, by feeding the output of the line predictor to a dedicated decoder neural network with architecture comparable to that described in Sec. III-E; the prediction error is then entropy coded.

G. Training vs. Inference

The description in the previous section follows the inference behavior. A few important differences are present for training in order to ensure scalability when using large datasets. First, the recurrent implementation of the line predictor is replaced with the parallel implementation already described in Sec. III-D. This improves efficiency as it enables processing several lines at the same time, and, in fact, it is one of the main advantages over traditional recurrent neural networks. Moreover, we remark an interesting property of the overall design, which is useful for efficient training: all operations performed after the line encoder are columnwise independent. In fact, the predictors work on lines and bands but do not mix columns, and the use of 1×1 convolution in the decoder also keeps this property. This is useful to reduce memory consumption during training by subsampling random subsets of columns after encoding, if desired.

LineRWKV is trained by minimizing the ℓ_1 loss between the predicted and true pixel values. Notice that this means that we only optimize for lossless compression, while never accounting for quantization during training. Nevertheless, lossy rate distortion tests still show excellent performance, as reported in Sec. IV.

IV. Experimental results

In this section, we analyze the performance of LineRWKV in terms of compression efficiency with respect to state-of-the-art approaches to hyperspectral image compression. Moreover, we show how LineRWKV behaves when a different satellite is targeted compared to its original training. Finally, we report throughput and memory usage on a 7W low-power device as a proof of concept for potential onboard implementation on embedded devices. Code and pretrained models are available at https://github.com/diegovalsesia/linerwkv.

A. Experimental setting

Our main experimental results are based on training and testing LineRWKV on the recently introduced HySpecNet-11k dataset [40]. This is the largest curated dataset of hyperspectral images currently available, composed of 11,483 non-overlapping patches of size $128 \times 128 \times 224$ acquired by the EnMAP satellite with a ground sampling distance of 30m. The authors provide standard train-test splits for benchmarking compression algorithms. In particular, we use the "hard" split where patches in the test set belong to entirely separate tiles with respect to the training patches. Preprocessing discards some bands, resulting in a total of 202, and clips values between 0 and 10000. We also test the method on two additional datasets: a collection of images from the PRISMA satellite we assembled and the dataset released for the DFC 2018 challenge [41]. For the former dataset, we collected 110 images of size $1000 \times 1000 \times 239$ from the PRISMA satellite [42], partitioned into 100 for training and 10 for testing. These images have been collected from all over the world to have a highly varied set of scenes, and the test locations are strictly disjoint from the train locations. For the experiments, we used the 66 VNIR bands. For the latter dataset, an aerial scene of Houston of size $4172 \times 1202 \times 50$ has been partitioned into $128 \times 128 \times 50$ sub-images. HyspecNet-11k data are preprocessed with clipping to 10000 maximum intensity, thus having a dynamic range of roughly 14 bits, while the PRISMA and DFC2018 images are over 16 bits.

We consider several baseline methods for hyperspectral image compression, focusing on low-complexity methods. In particular, we choose the state-of-the-art CCSDS standard for onboard lossless and near-lossless hyperspectral predictive compression, i.e., CCSDS-123.0-B-2 [22], and RWA [29] for an alternative state-of-the-art approach to lossless compression based on transforms. As for state-of-the-art deep learning approaches, we evaluate the 1D-CAE [14], SSCNet [15] methods as well as the recent method by Verdú et al. [21]; we remark that these methods only perform lossy compression. All methods have been retrained, using the reference implementation by the HySpecNet-11k authors for the former two and the authors' code for the latter. The 1D-CAE [14] is a convolutional autoencoder which limits computational complexity by working only in the spectral dimension, as it is known that most of the redundancy lies in inter-band correlation. SSCNet [15] also follows a spectral autoencoding approach. The work by Verdú et al. [21] manages spatial-spectral autoencoding at low complexity by introducing a new design for a variational autoencoder with a hyperprior network, operating on groups of bands and with ad-hoc normalization schemes.

We use the sample-adaptive Golomb encoder defined by the CCSDS-123.0-B-2 standard [22] as entropy coder for the positive-mapped prediction residuals, both for our method and the CCSDS method to ensure a fair comparison; it is obvious that the performance could be improved employing an arithmetic coder, especially at rates below 2 bpppc, and as a matter of fact, the Golomb coder is not capable of producing rates below 1 bpppc. The other methods use the same entropy encoder as proposed in their respective works. The CCSDS predictor is used in its full mode with wide neighbor-oriented local sums, and 3 prediction bands, a configuration which typically provides the best tradeoff of compression ratio and complexity.

Several configurations for LineRWKV have been tested in order to validate its scaling potential as function of number of encoder, decoder, predictors layers and number of features. Table I reports the details for four configurations of choice, ranging from extra small (XS) to large (L). Training LineR-

TABLE II: Lossless rate (bpppc) on HySpecNet-11k hard test set.

Model size	LineRWKV	CCSDS [22]	RWA [29]	Diff. CCSDS	Diff. RWA
XS	**5.647**	5.801	5.772	-0.154	-0.125
S	**5.521**	5.801	5.772	-0.280	-0.251
M	**5.510**	5.801	5.772	-0.291	-0.262
L	**5.370**	5.801	5.772	-0.431	-0.402

TABLE III: Transfer Learning - Lossless rate.

Dataset	Model size	LineRWKV (zero-shot)	LineRWKV (finetune)	CCSDS [22]	RWA [29]
PRISMA	XS	8.431	8.214	8.135	**8.105**
	S	8.312	**8.076**	8.135	8.105
	M	8.188	**7.974**	8.135	8.105
	L	8.093	**7.919**	8.135	8.105
DFC2018	XS	7.484	**7.315**	7.364	7.403
	S	7.378	**7.216**	7.364	7.403
	M	7.353	**7.175**	7.364	7.403
	L	7.334	**7.107**	7.364	7.403

WKV has been done over four Nvidia A100 40GB GPUs. In order to limit training memory requirements and speed up convergence, a random subset of 16 contiguous bands has been used in the initial training phases, followed by finetuning with all the available bands and a smaller batch size. For the M configuration, the initial batch size was 8 without any column sampling; in the finetuning phase will all bands, the batch size is 4 with a subsampling of 16 columns. Learning rate was linearly decreased from 10^{-4} to 10^{-6} over 4000 epochs. We remark that memory requirements during training are large due to the desire to process as much of the hyperspectral cube as possible concurrently with a large enough batch size. This is not reflected in the inference phase, which is memory-efficient as shown in Sec. IV-C.

B. Results on HySpecNet-11k and transfer learning to other sensors

Our main experiment assesses the rate-distortion performance of all methods when tested on the hard test split of HySpecNet-11k, as well as on the PRISMA and DFC2018 datasets.

We first start with lossless compression. Table II reports a comparison between the rates obtained by LineRWKV, CCSDS-123.0-B-2 [22] and RWA [29] on HySpecNet-11k. RWA is sometimes slightly better than CCSDS 123.0-B-2 at lossless compression, but both are significantly outperformed by the proposed LineRWKV.

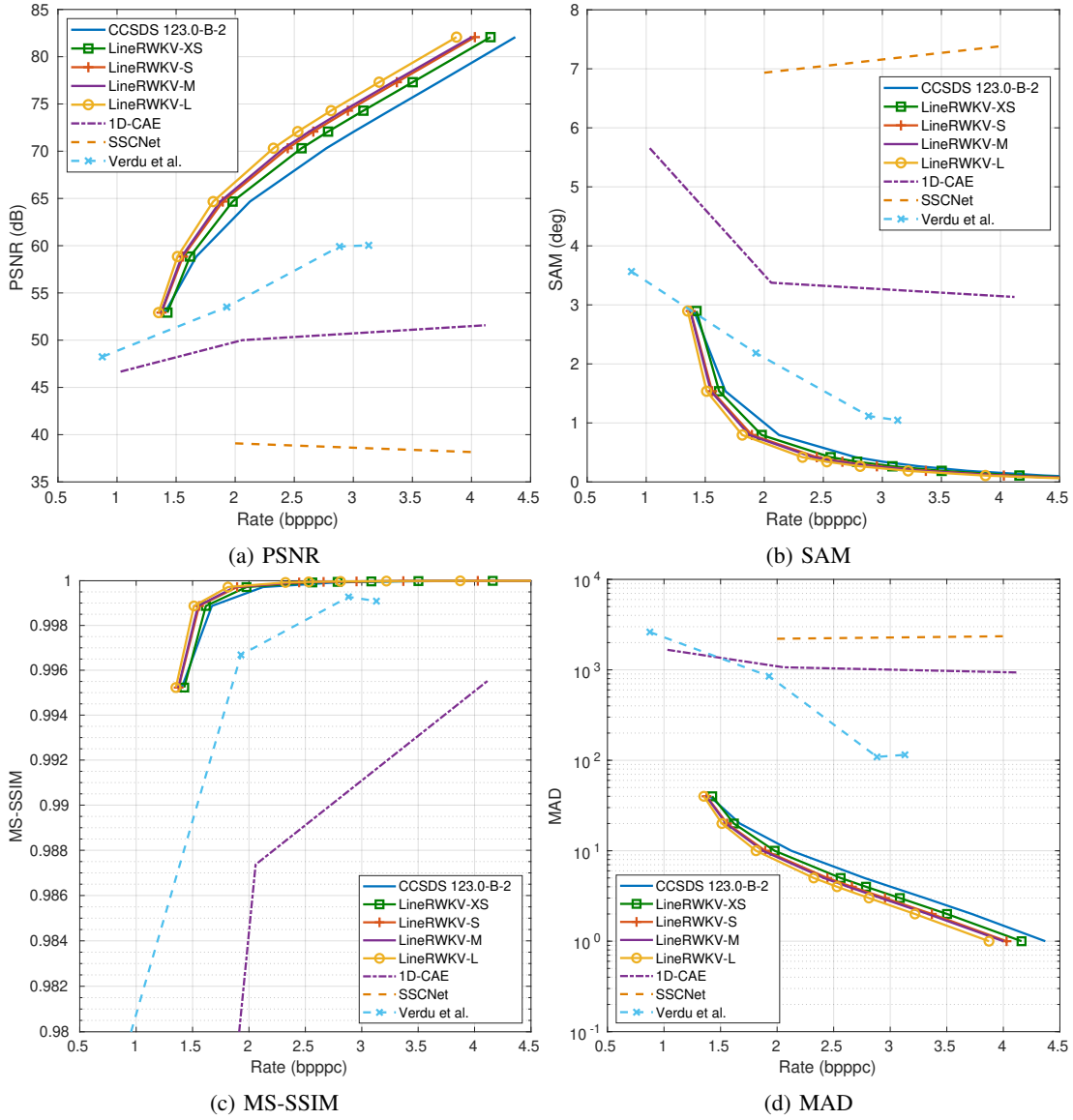
We are also interested in studying how well a LineRWKV model trained on a dataset from a specific satellite behaves when it is applied to a different satellite. This is important since it is likely that a large-scale dataset of images representative of a new satellite is not available before launch. However, it is also conceivable that a certain number of images can be losslessly transmitted with a suboptimal compressor in the pre-operational phase in order to finetune and update the compressor model itself. For this experiment, we use the PRISMA and DFC2018 datasets. Table III reports the lossless rate compared to CCSDS-123.0-B-2 and RWA on the PRISMA and DFC2018 datasets in two settings of interest: *zero-shot transfer*, i.e., the model trained on HySpecNet-11k is used as is on DFC2018 and PRISMA, and after *finetuning*. It can be noticed that the domain gap between EnMAP and the PRISMA and DFC2018 sensors affects compression performance in the zero-shot case which is now either slightly outperformed by or close to the CCSDS algorithm. However, finetuning with a modest number of images allows to recover superior performance.

TABLE IV: BD-Rate of LineRWKV with respect to CCSDS-123.0-B-2 for lossy compression.

Model size	HyspecNet-11k	PRISMA	DFC2018
XS	-5.27%	-0.89%	-4.53%
S	-9.05%	-4.36%	-7.13%
M	-9.94%	-5.11%	-7.84%
L	-12.67%	-6.37%	-8.45%

For lossy compression, we are mostly interested in the high-quality and high-rate regimes (typically ≥ 2 bpppc) as these are the most relevant for space missions. We consider the following metrics for quality evaluation: PSNR, MS-SSIM [43], spectral angle SAM score [44], and maximum absolute distortion (MAD). Figs. 4, 5 and 6 show the average values of the evaluation metrics obtained over the entire test set as a function of the rate for all the methods under study, on the HySpecNet-11k, PRISMA and DFC2018 datasets, respectively. For PRISMA and DFC2018, we are reporting the results after finetuning. It can be noticed that existing deep learning approaches are significantly outperformed by the CCSDS standard at high rates. This is due to the current focus of the literature on hyperspectral image compression towards very low rates, which finds its motivation in complex challenges towards scaling autoencoder neural network designs to also work well at high rates while maintaining acceptable complexity. As a sanity check, we remark that the reported PSNR values on the hard split are consistent with published results on the easy split by the HySpecNet-11k authors.

The results on the MAD metric also raise an interesting point of discussion, since the state-of-the-art deep learning approaches are two orders of magnitude worse than LineRWKV and CCSDS-123.0-B-2. This is expected since the traditional autoencoder design does not constrain the maximum error in any way, possibly resulting in localized very high errors. On the other hand, predictive compressors naturally limit maximum distortion to half of the quantization step size, both for the prequantized approach followed by LineRWKV and the in-loop quantization of CCSDS-123.0-B-2. Fig. 7 shows a visualization of the maximum absolute error and reports histograms of the full absolute error distribution, at a rate point close to 4bpppc for all methods. LineRWKV and CCSDS-123.0-B-2 set the quantization step size to $Q = 21$, so the maximum error in constrained to a value of 10, and we can see it shows a roughly uniform distribution. On the other hand, the deep autoencoding approaches exhibit an exponential error distribution with tails with significantly larger errors. A guarantee on the maximum error is critical to ensure image reliability in real missions, thus showing an important advantage of the proposed LineRWKV method.



(a) PSNR (b) SAM

(c) MS-SSIM (d) MAD

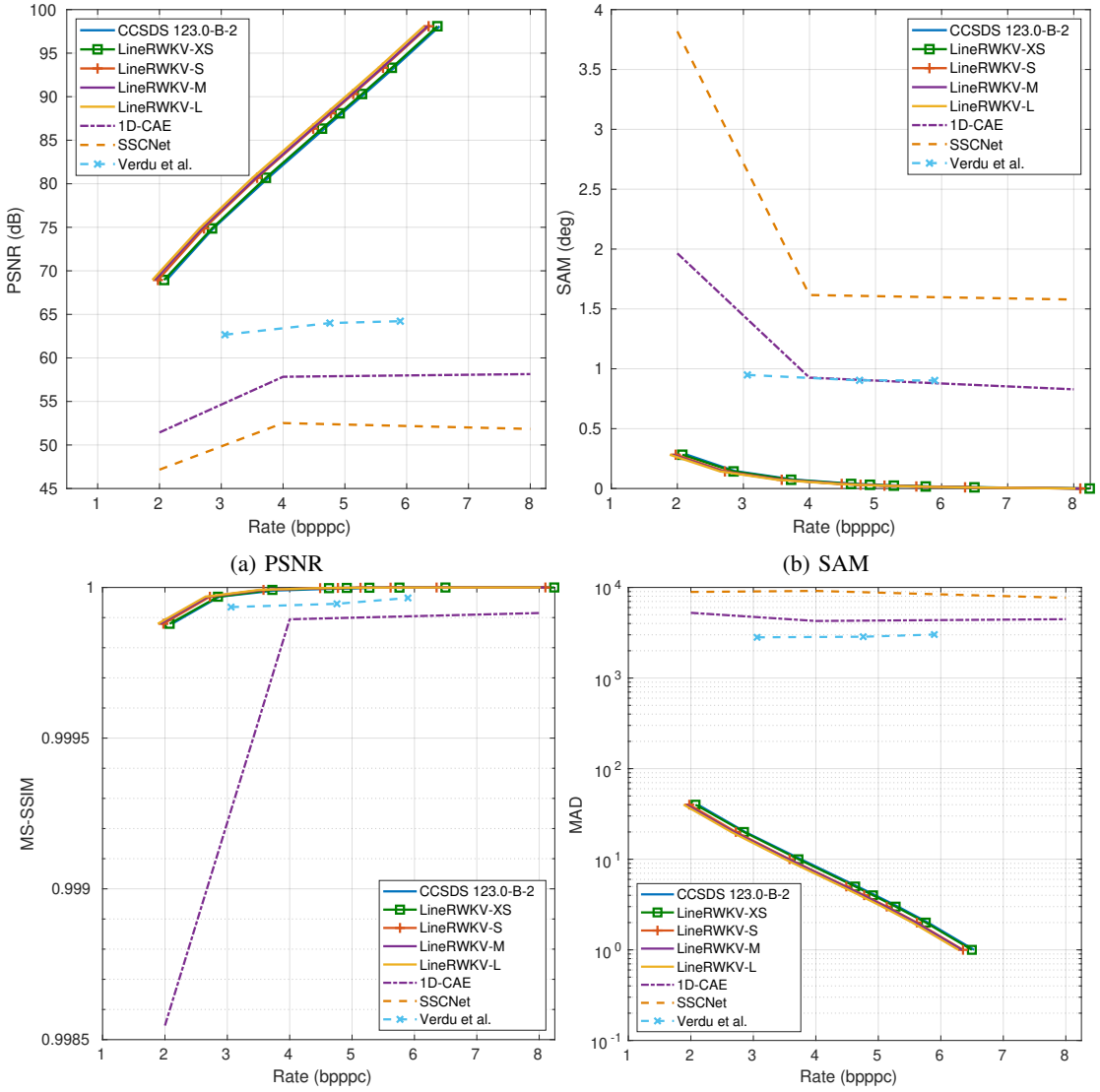
Fig. 4: Rate-distortion performance comparison on HySpecNet-11k hard test set. SSCNet MS-SSIM is outside the figure scale at 0.914 at 2bpppc and 0.891 at 4bpppc.

Regarding the overall comparison with CCSDS-123.0-B-2, we can notice that it is closer in performance than deep learning approaches, but LineRWKV always outperforms it, even in the XS configuration. This is also shown in Table IV which reports the Bjontegaard Delta metric (BD-Rate) with respect to PSNR. We remark that since both LineRWKV and CCSDS-123.0-B-2 are predictive schemes working at high rates, their quality metrics are essentially the same, differing only in rate.

Fig. 8 shows a qualitative comparison of the positive-mapped prediction residuals of LineRWKV-XS and CCSDS-123.0-B-2 for a band in a test image. It can be noticed that some complex correlation patterns are discovered by the neural network and removed, leading to smaller residuals. On the other hand, Fig. 9 reports the median positive-mapped prediction residual as a function of band index for the entire HySpecNet-11k hard test set. We can see that LineRWKV-XS provides smaller residuals across basically all the 202 bands, with the largest gains observed for bands 130 to 202.

We also report an example of performance of a downstream application, where compressed images are used to address land cover classification. The Pavia dataset is used for this task with models trained on HyspecNet-11k. Table V reports the accuracy as a function of rate, when images compressed by LineRWKV and 1D-CAE are used for training and testing a simple model based on support vector machines (SVMs). Although this particular application is fairly robust to compression, we can see that LineRWKV provides a smaller degradation of accuracy as bitrate decreases. In general, performance of downstream applications might be more or less sensitive to loss of information due to compression, and should be carefully assessed on a task-by-task basis.

It is also worth noticing that there is a clear performance scaling, with increasing rate-distortion gains both in lossless



(a) PSNR (b) SAM

(c) MS-SSIM (d) MAD

Fig. 5: Rate-distortion performance comparison on PRISMA test set. SSCNet MS-SSIM is outside the figure scale at 0.988 at 4bpppc and 0.989 at 8bpppc.

TABLE V: Land cover classification from compressed images. SVM Accuracy on Pavia dataset.

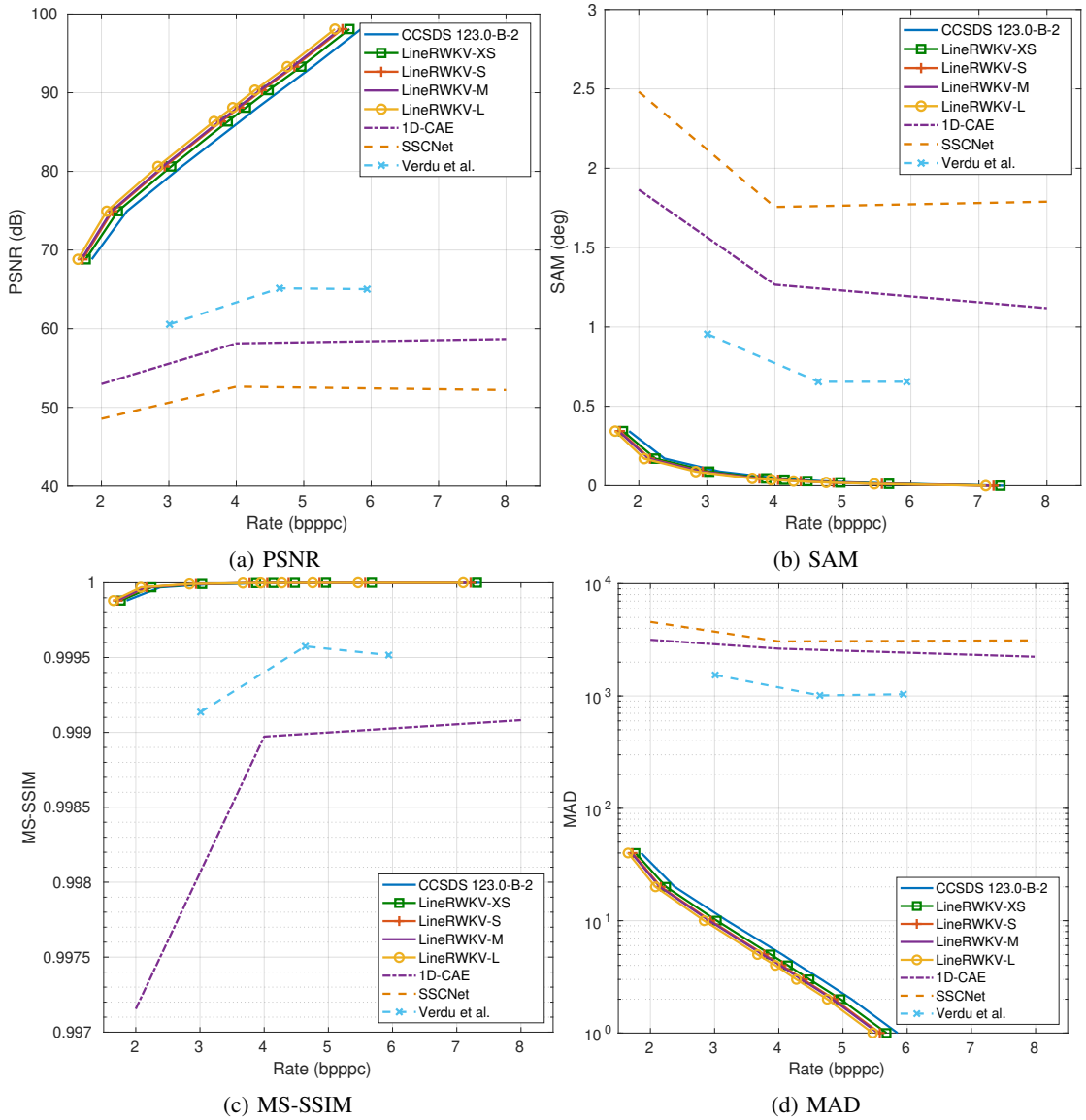
Model	**Rate**			
	Orig.	**8 bpppc**	**4 bpppc**	**2 bpppc**
LineRWKV-XS	**91.15%**	**91.15%**	**91.14%**	**90.94%**
1D-CAE	**91.15%**	88.92%	87.85%	85.33%

and lossy compression, in the LineRWKV model, as model complexity is increased from the XS to L configuration. For example, on HySpecNet-11k, the XS configuration reports an already substantial 0.154 bpppc gain over lossless CCSDS-123.0-B-2, while the L configuration achieves a massive 0.431 bpppc gain. Although the L configuration is too complex for practical onboard usage, it serves as a validation that LineRWKV has potential scaling and is not significantly bottlenecked in its design.

As a term of comparison regarding complexity, the work by Verdú et al. [21] reports a complexity of about 17k FLOPs/sample compared to the 120k FLOPs/sample of LineRWKV-XS. However, we also tested a variant of the method where we increased the number of hidden and latent features to approximately match 120k FLOPs/sample. This is the point with rate equal to 3.130 bpppc reported in Fig. 4, which however does not seem to provide significant improvements, highlighting that the design has some bottleneck preventing it from reaching low distortion values at higher rates.

C. Performance on low-power hardware

In order to validate the low-complexity inference of LineRWKV, we test its performance on a low-power embedded system, namely the Nvidia Jetson Orin Nano. The platform has 8GB of shared CPU-GPU memory and a maximum power of 7W. We test the XS configuration in single-precision



(a) PSNR (b) SAM

(c) MS-SSIM (d) MAD

Fig. 6: Rate-distortion performance comparison on DFC2018 test set. SSCNet MS-SSIM is outside the figure scale at 0.995 at 4bpppc and 0.994 at 8bpppc.

floating point (FP32). For the HySpecNet-11k images ($128 \times 128 \times 202$), the peak memory usage due to LineRWKV is 673 MB, which is a quite modest value in the realm of deep learning-based processing of hyperspectral images. The measured latency is 6.472 seconds, resulting in a throughput of 511,345 samples/sec. Fig. 10 shows how memory scales as a function of number of lines, columns and bands. First, we can notice that the recurrent property over the along-track direction allows constant memory usage regardless of the number of lines. For what concerns columns and bands, we can see that scaling is fairly linear, in contrast with the quadratic dependence in traditional Transformers, with a floor for small image size. Throughput is roughly constant around 500k samples/s for all image sizes except for the very small ones, which also confirms linear scaling of complexity.

It must be remarked that the current implementation is far from optimized. Indeed, several optimizations are possible that would substantially raise the throughput and might be worth investigating in future works. Few examples include model compilation for the target accelerator (e.g., via TensorRT), mixed-precision inference where parts of the model run integer-quantized or in floating point half precision. Our proposed design also does not exploit any spatial downsampling: if the line and spectral predictors could be run on a downsampled cube, throughput would be increased by a factor close to the downsampling factor. However, this necessitates careful decoder design not to reduce rate-distortion performance.

D. Spectral Predictor Ablation

In this section, we evaluate the choice of RWKV blocks as a spectral prediction mechanism against an alternative approach, presented in [25]. The alternative approach adopts causal convolution operations such that the receptive field

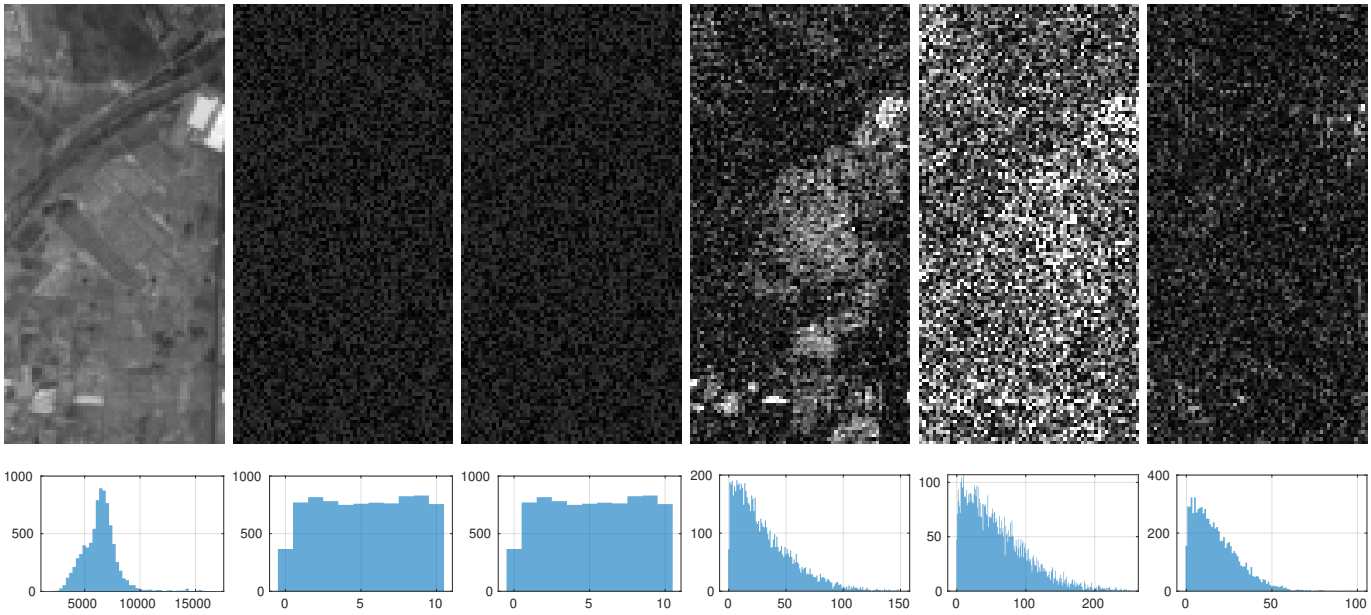


Fig. 7: Absolute error at 4bpppc. Left to right: ground truth image crop from PRISMA test set, LineRWKV-XS, CCSDS-123.0-B-2, 1D-CAE, SSCNet, Verdú et al.. Note that the white level is 50 for LineRWKV and CCSDS and 150 for 1D-CAE, SSCNet, Verdú et al. to improve visibility.

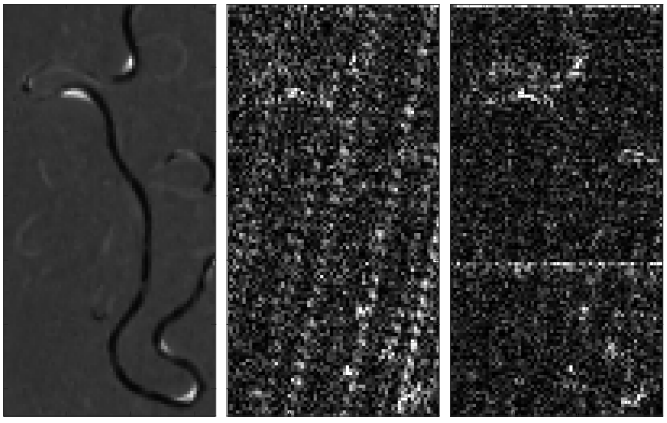


Fig. 8: Left to right: band 187 of a test image (colormap $[0, 4000]$, positive-mapped prediction residuals (colormap $[0, 100]$) for CCSDS-123.0-B-2 and LineRWKV-XS.

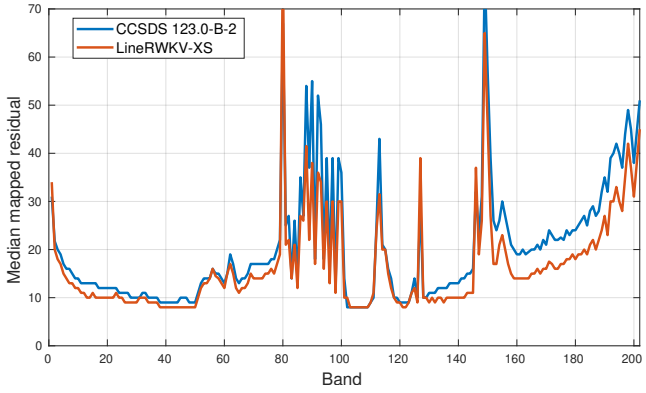


Fig. 9: Positive-mapped prediction residual for CCSDS-123.0-B-2 and LineRWKV-XS. Median, over the entire HySpecNet-11k hard test set, of spatial medians.

expands only towards past bands. In particular, blocks are repeated to form the spectral predictor; each block is composed of 1D causal convolution with kernel length of 3 bands, Layer Normalization, GeLU non-linearity and an attention operation. This operation computes an attention mask from the output of the GeLU non-linearity with two size-1 convolutions interleaved by GeLU and a final sigmoid activation. The resulting mask is multiplied elementwise to the input of the attention operation and the result summed to it. Comparisons are made to have a similar number of trainable parameters with respect to the design with RWKV blocks.

Table VI reports a comparison between the lossless rate and throughput on the Nvidia Jetson Orin Nano of some configurations of the proposed LineRWKV model with RWKV-based spectral predictor, and those of the model with causal convolution-based spectral predictor. We can see that the causal convolution method is faster but not as effective in terms of compression ratio.

TABLE VI: Ablation of the design of spectral predictor.

Model size	RWKV		Causal conv.	
	Rate	Throughput	Rate	Throughput
XS	5.647	511,345	5.764	595,962
M	5.510	173,078	5.593	334,476

V. Conclusions and Limitations

We presented a novel design of a low-complexity neural network for compression of hyperspectral images with the goal of usage onboard of satellites. In particular, we showed that the memory and complexity bottleneck of current autoencoder designs can be overcome by adopting a predictive coding

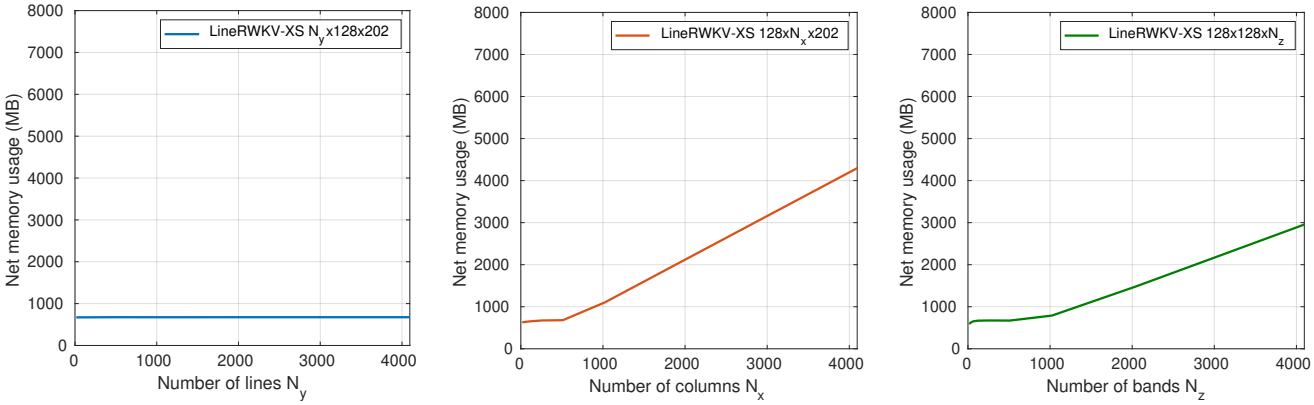


Fig. 10: Scaling of LineRWKV-XS memory usage on the Nvidia Jetson Orin Nano 8GB as a function of number of lines, columns and bands.

approach with a neural network working line-by-line based on a hybrid recurrent-attentive operations. This leads to a substantial improvement of rate-distortion performance, resulting in the first deep-learning approach capable of outperforming CCSDS-123.0-B-2, and having modest memory requirements.

While LineRWKV represent a substantial improvement in the performance of low-complexity deep learning compressors, some limitations need to be addressed in future work. In particular, complexity needs to be further reduced to achieve throughput comparable with current in-flight implementations. We conjecture that a possible avenue could be reducing the columns and bands into a more compact latent space to be used for prediction, essentially creating a hybrid method between autoencoding and prediction. The numerical stability of the prediction, while managed with a small overhead in the current scheme, also deserves further investigation. Moreover, optimal training may require a substantial amount of GPU memory for sensors with hundreds of bands. The solution to this does not seem trivial, as we have shown that lighter methods for spectral prediction, such as causal convolution, do not seem as effective as RWKV. Finally, our scaling experiments show that increasing the number of parameters of LineRWKV brings improvements in rate-distortion performance for the same architectural design. However, at the moment, it is unclear is this is an intrinsic limitation of neural networks, requiring larger scales to be more effective, or if the design could be made more parameter efficient.

REFERENCES

[1] M. Jiang, Y. Su, L. Gao, A. Plaza, X.-L. Zhao, X. Sun, and G. Liu, "Graphgst: Graph generative structure-aware transformer for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, 2024.

[2] S. K. Roy, A. Deria, C. Shah, J. M. Haut, Q. Du, and A. Plaza, "Spectral–spatial morphological attention transformer for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–15, 2023.

[3] S. Mei, C. Song, M. Ma, and F. Xu, "Hyperspectral image classification using group-aware hierarchical transformer," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2022.

[4] Y. Su, Z. Zhu, L. Gao, A. Plaza, P. Li, X. Sun, and X. Xu, "Daan: A deep autoencoder-based augmented network for blind multilinear hyperspectral unmixing," *IEEE Transactions on Geoscience and Remote Sensing*, 2024.

[5] M. E. Paoletti, S. Moreno-Álvarez, Y. Xue, J. M. Haut, and A. Plaza, "Aatt-cnn: Automatic attention-based convolutional neural networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–18, 2023.

[6] Z. Wu and B. Wang, "Transformer-based autoencoder framework for nonlinear hyperspectral anomaly detection," *IEEE Transactions on Geoscience and Remote Sensing*, 2024.

[7] Y. Dua, V. Kumar, and R. S. Singh, "Comprehensive review of hyperspectral image compression algorithms," *Optical Engineering*, vol. 59, no. 9, pp. 090 902–090 902, 2020.

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

[9] L. Zhou, C. Cai, Y. Gao, S. Su, and J. Wu, "Variational autoencoder for low bit-rate image compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2617–2620.

[10] 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.

[11] J. Ballé, P. A. Chou, D. Minnen, S. Singh, N. Johnston, E. Agustsson, S. J. Hwang, and G. Toderici, "Nonlinear transform coding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 339–353, 2020.

[12] W. Jiang, J. Yang, Y. Zhai, P. Ning, F. Gao, and R. Wang, "MLIC: Multi-reference entropy model for learned image compression," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 7618–7627.

[13] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. V. Gool, "Practical full resolution learned lossless image compression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 10 629–10 638.

[14] J. Kuester, W. Gross, and W. Middelmann, "1d-convolutional autoencoder based hyperspectral data compression," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 43, pp. 15–21, 2021.

[15] 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.

[16] V. Alves de Oliveira, M. Chabert, T. Oberlin, C. Poulliat, M. Bruno, C. Latry, M. Carlavan, S. Henrot, F. Falzon, and R. Camarero, "Reduced-complexity end-to-end variational autoencoder for on board satellite image compression," *Remote Sensing*, vol. 13, no. 3, p. 447, 2021.

[17] Y. Dua, R. S. Singh, K. Parwani, S. Lunagariya, and V. Kumar, "Convolution neural network based lossy compression of hyperspectral images," *Signal Processing: Image Communication*, vol. 95, p. 116255, 2021.

[18] Y. Guo, Y. Tao, Y. Chong, S. Pan, and M. Liu, "Edge-guided hyperspectral image compression with interactive dual attention," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–17, 2022.

[19] Y. Guo, Y. Chong, and S. Pan, "Hyperspectral image compression via cross-channel contrastive learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–18, 2023.

[20] J. Kuester, W. Gross, S. Schreiner, W. Middelmann, and M. Heizmann,

"Adaptive two-stage multisensor convolutional autoencoder model for lossy compression of hyperspectral data," *IEEE Transactions on Geoscience and Remote Sensing*, 2023.
[21] S. Mijares i Verdú, J. Ballé, V. Laparra, J. Bartrina-Rapesta, M. Hernández-Cabronero, and J. Serra-Sagristà, "A scalable reduced-complexity compression of hyperspectral remote sensing images using deep learning," *Remote Sensing*, vol. 15, no. 18, p. 4422, 2023.
[22] Consultative Committee for Space Data Systems (CCSDS), "Low-Complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression," *Blue Book*, no. 2, February 2019. [Online]. Available: https://public.ccsds.org/Pubs/123x0b2c3.pdf
[23] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, H. Cao, X. Cheng, M. Chung, M. Grella, K. K. GV *et al.*, "RWKV: Reinventing RNNs for the Transformer Era," *arXiv preprint arXiv:2305.13048*, 2023.
[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
[25] D. Valsesia, T. Bianchi, and E. Magli, "Hybrid recurrent-attentive neural network for onboard predictive hyperspectral image compression," in *IGARSS 2024-2024 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2024.
[26] L. Santos, L. Berrojo, J. Moreno, J. F. López, and R. Sarmiento, "Multispectral and hyperspectral lossless compressor for space applications (hyloc): A low-complexity fpga implementation of the ccsds 123 standard," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 2, pp. 757–770, 2015.
[27] Consultative Committee for Space Data Systems (CCSDS), "Image Data Compression," *Blue Book*, no. 2, September 2017. [Online]. Available: https://public.ccsds.org/Pubs/122x0b2.pdf
[28] ——, "Spectral Preprocessing Transform for Multispectral and Hyperspectral Image Compression," *Blue Book*, no. 1, September 2017. [Online]. Available: https://public.ccsds.org/Pubs/122x1b1.pdf
[29] N. Amrani, J. Serra-Sagristà, V. Laparra, M. W. Marcellin, and J. Malo, "Regression wavelet analysis for lossless coding of remote-sensing data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 9, pp. 5616–5627, 2016.
[30] S. Álvarez-Cortés, J. Serra-Sagristà, J. Bartrina-Rapesta, and M. W. Marcellin, "Regression wavelet analysis for near-lossless remote sensing data compression," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 2, pp. 790–798, 2019.
[31] Y. Chong, L. Chen, and S. Pan, "End-to-end joint spectral–spatial compression and reconstruction of hyperspectral images using a 3d convolutional autoencoder," *Journal of Electronic Imaging*, vol. 30, no. 4, pp. 041 403–041 403, 2021.
[32] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable rate image compression with recurrent neural networks," *arXiv preprint arXiv:1511.06085*, 2015.
[33] K. Islam, L. M. Dang, S. Lee, and H. Moon, "Image compression with recurrent neural network and generalized divisive normalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1875–1879.
[34] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves *et al.*, "Conditional image generation with pixelcnn decoders," *Advances in neural information processing systems*, vol. 29, 2016.
[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[36] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.
[37] D. Valsesia and E. Magli, "High-throughput onboard hyperspectral image compression with ground-based cnn reconstruction," *IEEE transactions on geoscience and remote sensing*, vol. 57, no. 12, pp. 9544–9553, 2019.
[38] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.
[39] Z. Yang, Z. Dai, R. Salakhutdinov, and W. W. Cohen, "Breaking the softmax bottleneck: A high-rank rnn language model," in *International Conference on Learning Representations*, 2018.
[40] M. H. P. Fuchs and B. Demir, "Hyspecnet-11k: A large-scale hyperspectral dataset for benchmarking learning-based hyperspectral image compression methods," *arXiv preprint arXiv:2306.00385*, 2023.
[41] U. of Houston, "Data fusion contest (dfc) 2018 dataset," https://hyperspectral.ee.uh.edu/?page_id=1075, 2018, accessed: 25 June 2024.
[42] S. Cogliati, F. Sarti, L. Chiarantini, M. Cosi, R. Lorusso, E. Lopinto, F. Miglietta, L. Genesio, L. Guanter, A. Damm *et al.*, "The PRISMA imaging spectroscopy mission: overview and first performance analysis," *Remote sensing of environment*, vol. 262, p. 112499, 2021.
[43] Z. Wang, E. Simoncelli, and A. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems and Computers, 2003*, vol. 2, 2003, pp. 1398–1402 Vol.2.
[44] F. Kruse, A. Lefkoff, J. Boardman, K. Heidebrecht, A. Shapiro, P. Barloon, and A. Goetz, "The spectral image processing system (sips)—interactive visualization and analysis of imaging spectrometer data," *Remote Sensing of Environment*, vol. 44, no. 2, pp. 145–163, 1993, airbone Imaging Spectrometry. [Online]. Available: https://www.sciencedirect.com/science/article/pii/003442579390013N

Diego Valsesia (S'13-M'17) received the Ph.D. degree in electronic and communication engineering from the Politecnico di Torino, in 2016. He is currently an Assistant Professor with the Department of Electronics and Telecommunications (DET), Politecnico di Torino. His main research interests include processing of remote sensing images, and deep learning for inverse problems in imaging. He is an Associate Editor for the IEEE Transactions on Image Processing, for which he received the 2023 Outstanding Editorial Board Member Award. He is a member of the EURASIP Technical Area Committee for Signal and Data Analytics for Machine Learning and a member of the ELLIS society. He was the recipient of the IEEE ICIP 2019 Best Paper Award, the IEEE Multimedia 2019 Best Paper Award.

Tiziano Bianchi received the M.Sc. degree (Laurea) in electronic engineering and the Ph.D. degree in information and telecommunication engineering from the University of Florence, Italy, in 2001 and 2005, respectively. From 2005 to 2012, he was a Research Assistant with the Department of Electronics and Telecommunications, University of Florence. He joined Politecnico di Torino in 2012 as an Assistant Professor. He is currently an Associate Professor with the Politecnico di Torino. He has authored over 100 papers in international journals and conference proceedings. His research interests include multimedia security technologies, multimedia forensics, processing of remote sensing images. He is currently Associate Editor for the IEEE Transactions on Information Forensics and Security and Senior Area Editor for the Journal of Visual Communication and Image Representation. He was the recipient of the IEEE Multimedia 2019 Best Paper Award and the 2021 and 2022 Best Associate Editor Award of the Journal of Visual Communication and Image Representation.

Enrico Magli (S'97-M'01-SM'07-F'17) received the M.Sc. and Ph.D. degrees from the Politecnico di Torino, Italy, in 1997 and 2001, respectively. He is currently a Full Professor with Politecnico di Torino, Italy, where he leads the Image Processing and Learning group, performing research in the fields of deep learning for image and video processing, image compression and image forensic for multimedia and remote sensing applicaitons. He is an Associate Editor of the IEEE Transactions on Circuits and Systems for Video Technology and the EURASIP Journal on Image and Video Processing. He is a Fellow of the ELLIS Society for the advancement of artificial intelligence in Europe, and has been an IEEE Distinguished Lecturer from 2015 to 2016. He was the recipient of the IEEE Geoscience and Remote Sensing Society 2011 Transactions Prize Paper Award, the IEEE ICIP 2015 Best Student Paper Award (as senior author), the IEEE ICIP 2019 Best Paper Award, the IEEE Multimedia 2019 Best Paper Award, and the 2010 and 2014 Best Associate Editor Award of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY.