Onboard Hyperspectral Super-Resolution with Deep Pushbroom Neural Network †

remote sensing

MDPI

Article

Onboard Hyperspectral Super-Resolution with Deep Pushbroom Neural Network †

Davide Piccinini, Diego Valsesia * and Enrico Magli

Department of Electronics and Telecommunications, Politecnico di Torino, 10129 Turin, Italy; davide.piccinini@polito.it (D.P.); enrico.magli@polito.it (E.M.)

* Correspondence: diego.valsesia@polito.it

† This article is an extended version of our paper published in the Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS) 2025, Brisbane, Australia, 3–8 August 2025, entitled "Towards Deep Line-based Architectures for Onboard Hyperspectral Image Super-Resolution".

Highlights

What are the main findings?

- We propose DPSR, a causal line-by-line HSI super-resolution network that matches pushbroom acquisition, achieving image quality comparable to the state of the art while being more than an order of magnitude less complex in terms of FLOPs/pixel and memory requirements.
- The proposed design is the first to run in real time on low-power hardware (15 W Jetson Orin Nano), requiring 4.25 ms to process a line, thus meeting the 4.34 ms/line acquisition time of the recent PRISMA satellite.

What are the implications of the main findings?

- DPSR validates that the line-by-line processing principle enables onboard, real-time, and high-quality HSI super-resolution by following the sensor acquisition system.
- Given the substantial complexity benefits at similar quality, the DPSR line-by-line approach should serve as a strong design principle for future works on onboard processing.

Abstract

Hyperspectral imagers on satellites obtain the fine spectral signatures that are essential in distinguishing one material from another but at the expense of a limited spatial resolution. Enhancing the latter is thus a desirable preprocessing step in order to further improve the detection capabilities offered by hyperspectral images for downstream tasks. At the same time, there is growing interest in deploying inference methods directly onboard satellites, which calls for lightweight image super-resolution methods that can be run on the payload in real time. In this paper, we present a novel neural network design, called Deep Pushbroom Super-Resolution (DPSR), which matches the pushbroom acquisition of hyperspectral sensors by processing an image line by line in the along-track direction with a causal memory mechanism to exploit previously acquired lines. This design greatly limits the memory requirements and computational complexity, achieving onboard real-time performance, i.e., the ability to super-resolve a line in the time that it takes to acquire the next one, on low-power hardware. Experiments show that the quality of the super-resolved images is competitive with or even surpasses that of state-of-the-art methods that are significantly more complex.

Keywords: super-resolution; hyperspectral images; onboard processing; line-based processing; Mamba

Academic Editor: Arturo Sanchez-Azofeifa

Received: 25 September 2025
Revised: 28 October 2025
Accepted: 1 November 2025
Published: 3 November 2025

Citation: Piccinini, D.; Valsesia, D.; Magli, E. Onboard Hyperspectral Super-Resolution with Deep Pushbroom Neural Network. *Remote Sens.* **2025**, *17*, 3634. https://doi.org/10.3390/rs17213634

1. Introduction

Hyperspectral imagery has emerged as a vital tool in the field of remote sensing because it samples a scene across a vast number of narrow spectral bands. This fine spectral resolution allows analysts to precisely identify and characterize the unique spectral signatures of different materials. Consequently, hyperspectral images are proving invaluable in a broad range of applications, including environmental conservation, agricultural management, pollution control, and mineral exploration. By examining the subtle reflectance properties of surfaces, e.g., computing vegetation health indicators, soil composition, or building materials, hyperspectral imaging supports enhanced data-driven decision-making and accurate assessment.

Despite these distinct advantages, the benefits of hyperspectral imaging come with certain trade-offs. One of the most significant drawbacks lies in its typically coarser spatial resolution when compared to multispectral sensors. While these easily provide spatial resolutions in the order of a few meters per pixel, hyperspectral payloads are limited to tens of meters per pixel or worse. For instance, the recent PRISMA [1], EnMAP [2], and HySIS [3] missions all provided a resolution of 30 m/pixel. For this reason, spatial super-resolution in hyperspectral images is a topic of great interest. The recent literature [4–7], particularly regarding deep neural networks, has shown that powerful data priors can be used to accurately enhance spatial resolutions, even from a single image.

At the same time, there is growing interest in the remote sensing community in shifting image analysis procedures from ground-based operations to onboard satellite platforms [8–11]. By processing data directly in space, as they are acquired, satellites can rapidly identify and respond to critical occurrences such as natural disasters or sudden environmental changes. This approach significantly reduces the latency associated with transferring large volumes of raw data to Earth, followed by subsequent processing at ground stations. Ultimately, real-time or near-real-time analytics could provide timely information to decision-makers and emergency response units, potentially saving lives and mitigating damage. However, onboard processing faces significant constraints in terms of the available computational resources, which calls for the development of efficient and low-complexity models. Hyperspectral imagers represent a good case study since they generate very large volumes of spatial–spectral data, which may be difficult to process efficiently, particularly in terms of memory requirements.

Since the super-resolution of hyperspectral images is highly desirable for more accurate image processing in downstream applications, it is natural to ask whether efficient models could be developed so that a satellite payload could perform super-resolution in real time as it acquires images. On the other hand, the current literature on hyperspectral super-resolution [4,12] predominantly emphasizes maximizing the output quality, at the expense of complexity, often relying on sophisticated and computationally expensive deep learning architectures. While these models achieve impressive results, they are poorly suited for potential onboard usage, where constraints related to power, memory, and processing speeds are present. Specifically, the current high-performance models are orders of magnitude heavier than what a small, low-power accelerator can sustain. For example, on the HySpecNet-11k dataset, the state-of-the-art methods MSDformer [4] and CST [12] require 714 K and 245 K FLOPs per pixel (FLOPs/px), respectively, for 4× super-resolution and 528 K and 121 K FLOPs/px for 2× super-resolution, which are excessive for existing low-power accelerators. Under a realistic input of $1000 \times 1000 \times 66$, such as the one of the PRISMA VNIR instrument, they also exceed 24 GB of required memory, again precluding onboard use. Balancing the need for high-fidelity enhancements with the practical limitations of satellite hardware remains a challenging area of study.

In this paper, we depart from the literature and present a highly efficient model for hyperspectral image super-resolution for onboard usage, called Deep Pushbroom Super-Resolution (DPSR), which can run in real time on low-power hardware and with limited memory requirements, while providing performance comparable to that of state-of-the-art methods exploiting one order of magnitude more floating point operations (FLOPs) per pixel. On HySpecNet-11k with a super-resolution factor of 4×, DPSR only requires 31 K FLOPs/px, while, with a factor 2×, it requires 20 K FLOPs/px. Compared to other lightweight models, such as EUNet [13] and SNLSR [7], the core of our contribution is a neural network that sustains the acquisition dynamics of a pushbroom sensor by processing one line at a time as it is acquired. A memory mechanism based on selective state space models (SSMs, e.g., Mamba [14]) processes the image as a sequence of lines (as opposed to a 2D tile) and ensures that information from previously acquired lines is exploited to super-resolve the current line. This design minimizes the memory requirement since only the feature maps for the current line (and a small memory state) need to fit in the accelerator memory, instead of buffers of hundreds of lines and respective features needed for state-of-the-art methods. We show that DPSR can process an entire PRISMA VNIR frame of size 1000 × 1000 × 66 with less than 1 GB of memory—at least an order of magnitude less than in other existing methods. Note that this line-based paradigm naturally matches the pushbroom imaging method commonly used in hyperspectral sensors on satellites, which acquires one line with all its across-track pixels and spectral channels at a given time, and it uses the movement of the satellite in the along-track direction to capture successive lines. Consequently, our super-resolution module could be directly pipelined to the sensor output, enabling real-time image enhancement as a line is super-resolved in the acquisition time of the next one, without the need for extra line buffering. As an example, the line acquisition time for the PRISMA satellite is 4.34 ms [15], and our experimental results show that a largely unoptimized implementation of DPSR on a 15 W system-on-chip super-resolves a line (2× scaling factor, thus producing two output lines with twice as many across-track pixels; the line size is the one of the VNIR PRISMA instrument, i.e., 1 × 1000 × 66) in 4.25 ms, demonstrating, for the first time, real-time performance. On the other hand, we show that existing state-of-the-art models in the literature, including those that are more efficiency-oriented, such as EUNet [13] and SNLSR [7], all run far slower than the real-time threshold of 4.34 ms.

A preliminary version of this work appeared in [16]. The present paper substantially extends and improves upon the earlier version regarding both the methodology and evaluation. Specifically, the model architecture has been redesigned in several ways. First, we now treat the spectral dimension as a feature axis rather than as part of a 3D spatial cube, leading to a more efficient and expressive representation. Moreover, the new design of the basic neural block (the SFE block in this paper) uses attention operations rather than simple convolutions, and a residual connection with bilinear interpolation is introduced. Following these modifications, substantial improvements have been obtained, with DPSR delivering a 43.17 dB MPSNR on HySpecNet-11k (4× SR factor), compared to the 41.82 dB achieved by LineSR. This paper also significantly expands the discussion and the experimental validation, providing results under two super-resolution factors (2× and 4×) across four different datasets, comparing the method against seven baselines, and presenting extensive ablation studies that dissect the contributions of individual architectural components and design choices. Finally, experiments on low-power hardware are also given to support the low complexity claim and suitability for onboard deployment.

We can summarize our contributions as follows:

- We introduce a lightweight deep learning architecture tailored to hyperspectral single-image super-resolution onboard satellites.

- Drawing inspiration from the operational principles of pushbroom sensors, our framework performs super-resolution in a causal, line-by-line fashion. To accomplish this, we leverage deep SSMs, which maintain effective memory of previously processed lines. This design choice allows the network to access relevant historical features without storing or reprocessing the entire image, leading to substantial efficiency gains.
- Experimental results on multiple datasets show image quality comparable to that of state-of-the-art models at a fraction of the complexity, with significantly lower runtime and memory requirements.

2. Related Works

2.1. Super-Resolution of Hyperspectral Images

Hyperspectral image super-resolution stands out as a pivotal challenge in the remote sensing community. The natural trade-off between spectral and spatial resolution often leaves hyperspectral data with relatively coarse pixel sizes. Enhancing the spatial detail of hyperspectral data opens the door to more accurate image analysis in downstream tasks such as land cover mapping, vegetation health monitoring, mineral deposits detection, and many more. In this paper, we focus on single-image super-resolution, where only one image at the lower resolution (LR) is available. Multi-image methods [17] exploiting multi-temporal data are known to better regularize the problem and further enhance the quality. However, our scenario is that of super-resolving the image immediately as it is acquired by the instrument, which is inherently a single-image task.

Historically, researchers have explored a wide array of approaches to tackle this task. Early solutions relied on classical signal processing and machine learning techniques, such as low-rank and sparse coding [18], spectral–spatial sub-pixel mapping [19], and low-rank tensor decomposition integrated with total variation regularization [20]. While these methods demonstrated good results, they generally depended on handcrafted priors, which could be challenging to adapt across different sensors or environmental conditions.

The advent of larger hyperspectral datasets and the parallel rise of deep learning have significantly reshaped the field. In particular, convolutional neural networks (CNNs) have become a popular tool [21,22] for super-resolution because of their powerful feature extraction capabilities and their inherent capacity to learn both spectral and spatial correlations directly from training data. These CNN-based models have substantially outperformed traditional methods, spurring further research to refine network architectures and training strategies. In [23] (GDRRN), the authors proposed a deep architecture based on the recursive use of residual connections and blocks composed of grouped convolutions and non-linearities to obtain super-resolved images.

Subsequently, attention-based mechanisms have introduced new ways to capture dependencies across spatial and spectral dimensions. Recent hybrid models [4,6,13,24–26] have sought to combine the local feature extraction strengths of CNNs with the global context modeling capabilities of Transformers. By leveraging attention to establish relationships between distant spatial locations or widely separated spectral bands, these models have enhanced the performance even further. In [24] (SSPSR), the authors proposed a deep architecture based on progressive upsampling, with parameters shared between branches of the network, and the use of channel attention to fully capture spectral information. In [13] (EUNet), a multi-stage network that uses both a degradation model constraint and a deep learning-based super-resolution model is proposed, aiming at iteratively refining the SR image obtained; their strategy is based on splitting the MAP optimization problem into two sub-problems, i.e., data consistency and prior terms, and then jointly optimizing them. At the same time, in [4] (MSDformer), the authors proposed a Transformer-based architecture comprising two modules to capture both spectral and spatial information: a

multi-scale spectral attention module that makes use of a novel spectral grouping strategy and a combination of convolutions and channel attention and a deformable convolution-based Transformer module that uses a modified self-attention block that has feature maps obtained with convolutions.

More recently, in [7] (SNLSR), the authors introduced a very efficient deep learning SR method that first unmixes the LR image into its abundance and endmember parts and then super-resolves the LR abundance to obtain its representation in the high-resolution space; finally, the super-resolved abundance is linearly mixed with the same endmembers. Their work is based on the hypothesis that HSI can be efficiently decomposed into a linear spectral combination of its abundance fractions and endmembers [27]. In [28], an effective combination of long short-term memory (LSTM) and multi-head attention, delivering competitive results, was presented. In [12], the authors proposed a new cross-scope spatial–spectral Transformer (CST) model based on two modules that they designed: a cross-scope spectral self-attention module and a cross-scope spatial one, which are built on top of a window-based attention architecture. With these modules, the authors sought to extract both short- and long-range information in the spatial and spectral directions, obtaining richer representations of the input. Recently, diffusion-based generative SR methods have been extended to hyperspectral imagery [29,30], albeit with considerably high computational costs. Lastly, new remote sensing foundation models have also been tested on the hyperspectral SR [31] task, suggesting that foundation-scale pretraining can benefit HSI-SR.

2.2. Sequence Modeling

Since the present work builds upon recent developments in sequence modeling, it is useful to review the main approaches adopted in deep learning for the processing of sequential data and to highlight their respective strengths and limitations. An early classical approach to sequence modeling leverages causal convolutional layers, restricting each kernel to information from preceding time steps in order for the autoregressive flow to remain intact [32]. A more effective method of modeling long sequences is based on recurrent neural networks (RNNs): these models operate by recursively updating a hidden state as new inputs arrive, enabling them to capture temporal dependencies over time. RNN-based architectures have shown their effectiveness across a variety of tasks, including language modeling and time-series forecasting. However, a key limitation arises when dealing with long sequences, where the models may suffer from vanishing or exploding gradients. Making use of gating mechanisms, such as gates in LSTM networks [33] and gated recurrent units [34,35], alleviates but does not completely remove this bottleneck, and the strictly step-by-step computation makes it challenging to harness the full parallelism of modern hardware when training on massive datasets.

Transformers [36] have recently replaced recurrent networks as the go-to architectures for sequence modeling. Instead of a step-by-step recursion mechanism, they make use of attention, which allows the model to weigh every position in a sequence against every other, adapting its computation to the specific input and capturing dependencies no matter how far apart. Because each layer can be trained in parallel, Transformers scale far better than RNNs and are able to excel in tasks as varied as large-scale text [37,38], image [39], and video generation [40] and even protein structure prediction [41].

The main drawback of Transformers is that self-attention grows quadratically in cost and memory with the sequence length—a serious problem for very long inputs or latency-sensitive, memory-constrained applications. Researchers have proposed remedies to tackle this problem, such as sparse [42] and linear attention [43] variants, but fully resolving these scalability limits remains challenging.

More recently, a new class of models based on state space models (SSMs) has emerged as a compelling alternative for sequence modeling. These models are designed to process long sequences efficiently by modeling the temporal dynamics through a latent state vector that evolves over time. Formally, an SSM is defined by a set of equations

$$\mathbf{h}'(t) = \mathbf{A}\mathbf{h}(t) + \mathbf{B}\mathbf{x}(t), \tag{1}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{h}(t) + \mathbf{D}\mathbf{x}(t), \tag{2}$$

where $\mathbf{x}(t) \in \mathbb{R}^M$ denotes the input at time t, $\mathbf{h}(t) \in \mathbb{R}^N$ is the latent state, and $\mathbf{y}(t) \in \mathbb{R}^O$ is the output. The matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ govern the evolution and output generation mechanisms. Early work on SSMs explored constrained parameterizations, such as the diagonal initialization of $\mathbf{A}$ [44], which offered simple and effective control over temporal dynamics. More advanced and recent formulations [45–47] introduced input-dependent and learnable SSM parameters, enabling models to selectively amplify relevant information or attenuate irrelevant patterns within the sequence. This concept was further refined in the popular Mamba architecture [14]. In this design, the selective SSM forms the core of a new modular building block, the Mamba block. This enables linear-time processing with respect to the sequence length, offering a powerful alternative to circumvent the quadratic cost of attention-based models. Encouraged by this success, Mamba blocks have also been used for vision detection tasks, as demonstrated in [48,49], where sequences were formed by scanning flattened image patches from multiple directions. Moreover, Mamba has also been successfully applied in the hyperspectral imaging domain, such as for classification [50,51] and detection [52].

In our setting, the model must process a continuous stream of lines in the along-track direction with the causal propagation of the information and a limited memory and computational budget. Mamba is the most suitable approach compared to attention-based or recurrent methods. Attention-based methods scale quadratically with the sequence length, whereas Mamba scales linearly, thus limiting the complexity. Recurrent designs are causal but they have a limited effective context and do not allow efficient training, while Mamba's selective state updates provide a better way to store and retrieve relevant past information.

2.3. Onboard AI

Over the past few years, onboard AI-based processing has emerged as a powerful method for performing the real-time analysis of satellite data, significantly reducing the latency for tasks requiring an immediate response and optimizing bandwidth usage. The ability to process large volumes of imagery directly in space can indeed enable more responsive satellite operations, facilitate the prompt detection of critical events, and open the door to autonomous decision-making. Nonetheless, the design of such onboard AI solutions faces multiple challenges, including strict power constraints, limited onboard memory, and the need for robust, lightweight architectures that can function reliably in harsh conditions.

Early works proved this concept on narrowly scoped problems. In [53], the authors squeezed a pruned YOLO variant onto a CubeSat for ship spotting, and, in [8], a compact cloud-mask CNN was launched on ESA's Φ-Sat-1 to perform cloud detection. These successes triggered broader surveys: in [10], many networks were benchmarked against flight-qualified FPGAs and CPUs; in [9], distillation was used to produce a lean variational autoencoder for change detection. In [54], the authors report three years of OPS-SAT trials in which an embedded U-Net performed pixel-level cloud segmentation. More recently, a new model was proposed in [11] to perform onboard multi-task inference. Looking toward constellation scenarios, in [55], the authors tried distributing a deep network across

multiple platforms, securing a 40% energy saving. More similarly to the work in this paper, a line processing concept for onboard image compression was designed in [56]. However, they used an RWKV hybrid attention–recurrent mechanism, and their framework relies on a pixel-by-pixel predictive coding approach. Collectively, these efforts show that, while there are still challenging computational and memory constraints, onboard AI is shifting from a curiosity to a mission enabler, edging the satellite community ever closer to fully autonomous operations.

3. Materials and Methods

In this section, we introduce the method that we propose to address the problem of onboard single-image super-resolution and the structure of our line-by-line neural network architecture, called DPSR. A high-level illustration of the concept is given in Figure 1, while the details of the DPSR neural network architecture are shown in Figure 2.

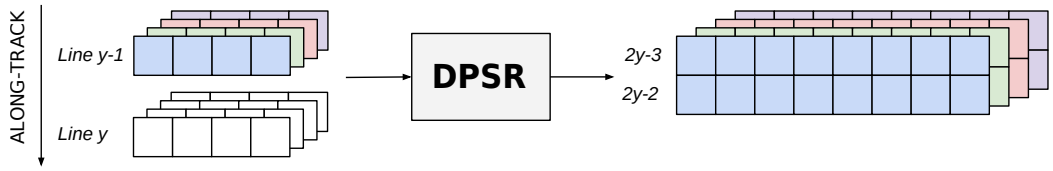


Figure 1. DPSR spatially super-resolves a hyperspectral image line by line, by using line $y-1$ and line y, as well as compact memory of past lines to super-resolve line $y-1$ in the along- and across-track directions. Colors denote spectral channels.

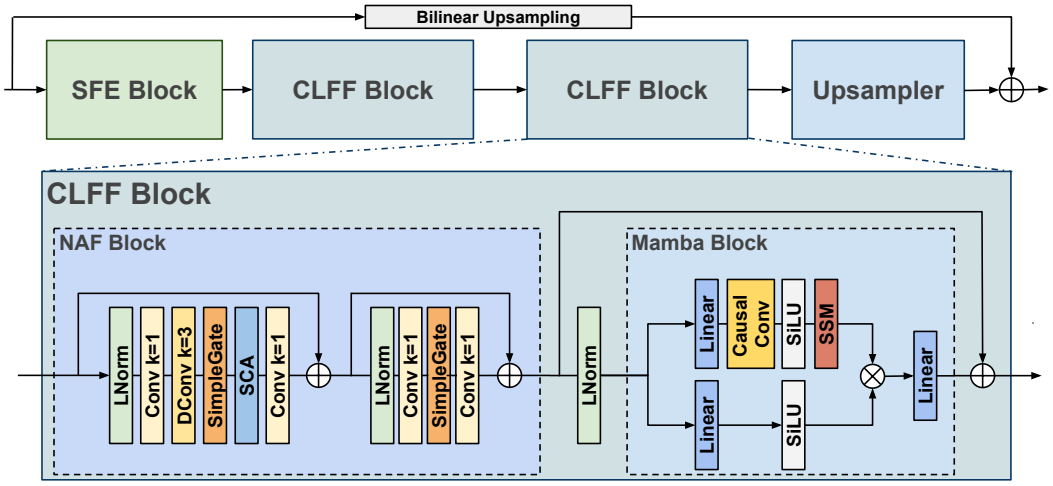


Figure 2. DPSR processes a line with an initial SFE block followed by two CLFF blocks composed of a NAFBlock to extract across-track and spectral features and a Mamba block to provide context from previous lines. The architecture works as a residual correction to bilinear upsampling.

3.1. Preliminaries and Proposed Strategy

The overall design goal is to super-resolve an image line by line in the along-track direction, aiming to match a pushbroom acquisition system and minimize the memory and computational requirements. In the remainder of this paper, we will use H to denote the number of lines in the along-track direction y, W to denote the number of columns in the across-track direction x, and C to denote the number of spectral channels. We also remark that super-resolving a line, with all its spectral channels, of size $1 \times W \times C$ by a factor r produces an output of size $r \times rW \times C$, i.e., r lines in the along-track direction, expanded by a factor of r in the across-track direction.

More formally, we suppose to have acquired the current LR line $\mathbf{x}_y^{\mathrm{LR}} \in \mathbb{R}^{W \times C}$ corresponding to along-track LR line location y and a memory of past lines, which will be

described in Section 3.2. From this, we estimate $\{\mathbf{x}^{SR}_{r(y-1)-(r-1)}, \mathbf{x}^{SR}_{r(y-1)-(r-2)}, \ldots, \mathbf{x}^{SR}_{r(y-1)-1}, \mathbf{x}^{SR}_{r(y-1)}\}$ with $y = 1, \ldots$, i.e., the set of r super-resolved (SR) lines at the high-resolution (HR) along-track spatial locations between the LR spatial locations $y-1$ (included) and y (excluded). It is important to note that we are framing the problem within an *interpolation* setting, where line y is used to predict HR locations strictly preceding it. This is in contrast with an *extrapolation* setting, which would predict spatial locations beyond y and would be more challenging, leading to worse image quality. Moreover, the estimation of the SR lines is performed in a residual fashion, i.e., a neural network only estimates an additive correction to the result of a bilinear interpolator.

We note that this configuration restricts the model to using only previously acquired lines, with no access to future lines. This departs from existing methods, which require a 2D tile of the image. This choice is intentional since such methods would require one to buffer a large number of lines to form a complete image, and their processing would require significantly increased memory and computational resources. Essentially, DPSR trades access to future lines, which may potentially limit quality, for complexity and speed.

3.2. DPSR Architecture

This section introduces the neural network architecture of DPSR, which is shown in Figures 2 and 3.

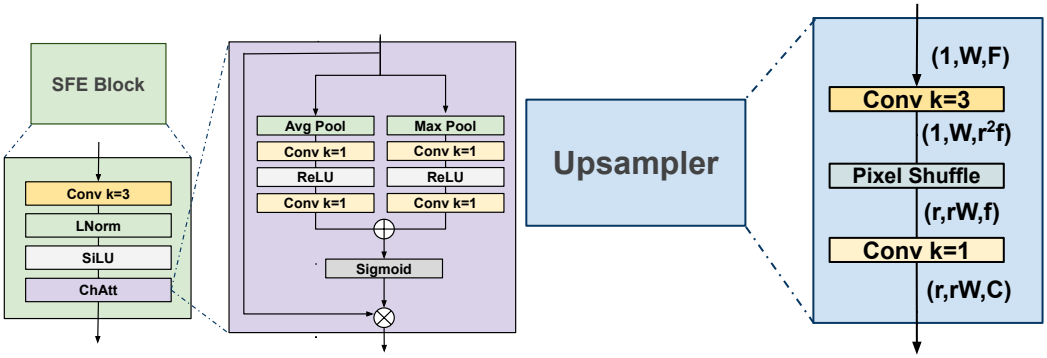


Figure 3. (**a**) Architecture of SFE block. Its main purpose is to filter the initial input in order to remove noise and extract shallow features that will be later refined by the subsequent CLFF block. (**b**) Architecture of upsampler module. It leverages Pixel Shuffle [57] and two convolutions to efficiently manage complexity.

As a first step, we decouple the along-track dimension from the across-track dimension by extracting the line $\mathbf{x}^{LR}_y$ along with all its spectral channels. This leads to the input of the architecture having a size of $1 \times W \times C$. Conceptually, DPSR will first extract features that jointly represent the pixels in the across-track and spectral dimensions of the current line and then use a selective SSM (Mamba [14]) to integrate these features with a memory of those of previous lines.

This results in a design featuring an initial shallow feature extraction block (SFE block) comprising 1D operations operating on across-track pixels and spectral channels. In particular, a 1D convolution layer, with a spatial kernel size of 3, transforms the input channels C into an arbitrary number of features F. This is then followed by layer normalization and non-linear activation, for which we use the sigmoid linear unit (SiLU) function [58]. Subsequently, a channel attention block [59] is introduced. We denote the output of the SFE block as $\mathbf{z}_y \in \mathbb{R}^{W \times F}$, where each pixel in the across-track direction is associated with a feature vector jointly capturing correlations between the columns and the original spectral bands. The SFE block is then followed by a backbone composed of two cross-line feature fusion (CLFF) blocks, each consisting of a NAFBlock followed by a Mamba block. The

design of the CLFF blocks is motivated by the different but complementary roles of the NAFBlock and Mamba block. The tensor $\mathbf{z}_y$ obtained as the output of the SFE block is fed to the CLFF blocks:

$$\mathbf{w}_y^{(2)} = \mathrm{CLFF}_2\big(\mathrm{CLFF}_1(\mathbf{z}_y)\big). \tag{3}$$

In detail, the NAFBlock uses a sequence of layers inspired by the building blocks of the NAFNet architecture [60] to expand the across-track receptive field and extract deeper features in the across-track and spectral dimensions. In particular, this block uses separable convolutions (size-1 1D convolution, followed by depthwise 1D convolution) to achieve lightweight receptive field expansion. Moreover, a SimpleGate and a simplified channel attention operation implement an attention-like mechanism of input-dependent feature extraction, while maintaining a lightweight design employing no specific activation function. We refer the reader to [60] for more details of these operations.

On the other hand, the Mamba block [14] has the crucial task of learning the interactions between successive lines, i.e., the propagation of information in the along-track dimension. The output of the NAFBlock consists of one feature vector for each of the W pixels in the line, which can be regarded as a token whose evolution is modeled over the along-track direction by means of Mamba. In the Mamba block, the input feature dimension F is expanded to EF, where E is an arbitrary constant. Then, a causal convolution operation in the along-track direction further processes the feature sequence. Note that this requires storing the receptive of the causal convolution operator, i.e., the features of K past lines, where K is the convolution kernel size. We remark that K is typically a small value, such as $K = 4$, as we used in the experiments, so that a very modest number of past lines needs to be retained. After non-linear activation with a SiLU function, the line features $\mathbf{z}'_y \in \mathbb{R}^{W\times EF}$ are processed by the Mamba selective SSM formulation introduced in Section 2.2 so that long-term dependencies encoded in the latent state can be exploited. The latent state of the SSM is a vector of length N for each expanded feature of the line pixels, i.e., $\mathbf{h} \in \mathbb{R}^{W\times EF\times N}$, and can be seen as a condensed running memory of all the previous lines scanned by the model, which allows the model to uncover self-similarities in the far past, dropping the requirement of a buffer of a high number of lines in order to perform effective super-resolution. When the SiLU-processed features $\mathbf{z'}_y$ reach the state-space module, they are processed by a discrete-time version of the linear SSM in Section 2.2,

$$\mathbf{h}_y = \mathbf{A}_y\mathbf{h}_{y-1} + \mathbf{B}_y\mathbf{z'}_y, \tag{4}$$

$$\mathbf{z''}_y = \mathbf{C}_y\mathbf{h}_y + \mathbf{D}_y\mathbf{z'}_y, \tag{5}$$

operating in parallel over all W pixels in the current line. The state $\mathbf{h}$ is updated at each time step, i.e., every time a new line is processed, following Equation (4). New context-rich line features are then produced as the output of Equation (5) using the updated latent state. Finally, the SSM-transformed features are gated by the original features input to the Mamba block after feature expansion and SiLU non-linearity and projected back to an F-dimensional space. The Mamba block is the main driver of the memory requirements for the architecture, and, based on the aforementioned design, we can summarize such requirements as follows: (i) a tensor of size $K \times W \times EF$ with the features of the current line and past lines in the receptive field of causal convolution; (ii) a tensor of size $W \times EF \times N$ with the latent state of the SSM. These are quite modest memory requirements; for instance, if we consider a realistic $W = 1000$, $F = 128$, $E = 1$, $N = 16$ setting, then only about 10 MB of memory is required to store the aforementioned tensors as single-precision floating point values.

The final part of the architecture is the upsampler module, which is responsible for the output tensor having the appropriate spatial dimensions, according to the desired super-resolution scale factor. The upsampler first expands the input tensor channels with a 1D convolution to a value fr^2, where r is the SR factor and f is an arbitrary number of features that will remain after the subsequent Pixel Shuffle operation [57]. The last layer is a 1D convolution that restores the number of channels of the input image. We note that choosing a suitable value for f involves a trade-off between limiting the amount of floating point operations performed by the upsampler module and properly representing all the spectral channels. Indeed, it could be argued that the optimal value f is tied to how many of the spectral channels are redundant and thus their dimensionality could be reduced. Hence, the final super-resolved lines are obtained as

$$\mathbf{x}^{\mathrm{SR}}_{r(y-1)}, \ldots, \mathbf{x}^{\mathrm{SR}}_{r(y-1)-(r-1)} = \mathrm{Upsampler}(\mathbf{w}^{(2)}_y) + \mathrm{Bilinear}(\mathbf{x}^{\mathrm{LR}}_y, \mathbf{x}^{\mathrm{LR}}_{y-1}). \tag{6}$$

3.3. Implementation Details and Experimental Settings

The proposed architecture makes use of an internal feature dimension F. This hyperparameter was set to $F = 280$ for all experiments, except those where we considered scaling down the model to $F = 128$ (DPSR-S model). The expansion factor of the Mamba block was set to $E = 1$, whereas the internal state of the SSM used was $N = 16$, following [14] and common practice in the literature that builds on the Mamba model. The reduced features factor in the upsampler, which controls how much we compress the feature information while recovering the appropriate spatial resolution, was set to $f = 64$ throughout all experiments. For the compact DPSR-S variant, our target was real-time operation on low-power hardware; this requirement drove the choice of internal parameters. For the full DPSR model, both the internal feature width and the Mamba expansion factor were selected to match a predefined complexity envelope (30 K FLOPs/px on HySpecNet-11k with an SR factor of 4 and 20 K FLOPs/px on the same dataset with an SR factor of 2). For the training procedure, we used the Adam optimizer with a fixed learning rate of 10^{-4} and a variable number of epochs to let the model converge until overfitting was observed. The optimal number of training epochs depends on the dataset, and it amounted to 300 for Pavia, 1600 for Houston, and 2000 for HySpecNet-11k and Chikusei. This value was determined empirically by monitoring convergence and stopping training once the improvement in the validation MPSNR fell below 0.01 dB over the preceding 50 epochs. No weight decay was used. We used the same loss as in [12], setting α_s to 0.3 and α_g to 0.1, as is common practice for other methods in the literature.

We present results for four different hyperspectral datasets. Besides the widely used Chikusei [61], Houston [62], and Pavia [63], we add the HySpecNet-11k dataset [64]. This is motivated by the significantly larger number of images present in HySpecNet-11k, which makes the results less sensitive to overfitting. See Section 4.1 for details regarding the preprocessing of each of the four datasets. Since our model deploys an interpolation strategy, i.e., it super-resolves the previous line, in order to train and test with SR factor r, we discard the first r lines of the output of the network (as they estimate lines before the start of the image) and the last r lines of the ground truth (as these cannot be estimated by DPSR as it would require a line after the end of the image). Note that, in a real-world scenario, we would not have to discard any line since the stream of input lines would be continuous. For the sake of fairness, we use the same discarding approach for all other baselines that we compare against in the evaluation of the quality metrics.

We compare DPSR against the following state-of-the-art and baseline methods: CST [12], MSDformer [4], SNLSR [7], EUNet [13], SSPSR [24], GDRRN [23]. The official code released by the authors has been used to retrain and test the models, with all

hyperparameters set as specified in the published reference papers. We note that we do not report results for SNLSR at a 2× SR factor since the model has been intrinsically designed and proposed for the 4× factor. In particular, the architecture uses a two-stage spatial expansion, each building its upsampling module with a scale equal to 2. For the method to support a true 2× SR factor, major modifications to the model architecture and size would be required. Additionally, we report baseline results obtained using bicubic interpolation when processing the entire image.

To quantify the fidelity of the super-resolved hyperspectral images, we report four commonly used metrics: the peak signal-to-noise ratio (PSNR) [65], structural similarity (SSIM) [66], spectral angle mapper (SAM) [67], and root mean squared error (RMSE). Together, these four metrics provide a balanced view of the spatial quality (PSNR, SSIM), spectral faithfulness (SAM), and overall error magnitude (RMSE). Following [4,12], for a hyperspectral image with C spectral channels, we compute each metric channelwise, accumulate over all channels, and then take the average as the metric value.

For model training and testing, we used one NVIDIA TITAN RTX GPU, whereas, for the low-power inference speed experiments, we used an NVIDIA Jetson Orin Nano in 15 W mode.

We remark that all reported results were obtained with a fixed random seed for every model and dataset, for both 2× and 4× settings and for all ablation studies, thus representing a typical run, as is commonly done in the literature.

The code and pretrained models will be available at https://github.com/DavidePiccinini98/dpsr (Accessed on 27 October 2025).

4. Results

In this section, we analyze the performance of DPSR in terms of the quality of the super-resolved images as well as the computational complexity with respect to selected state-of-the-art hyperspectral SR models across the chosen datasets.

4.1. Experimental Results on Hyperspectral Datasets

We first conducted experiments on the HySpecNet-11k dataset [64], a large-scale hyperspectral benchmark dataset consisting of 11,483 non-overlapping image patches. Each patch has 128 × 128 pixels, with 202 usable spectral bands and a ground sampling distance of 30 m. The standard training set comprises 70% of the patches, the validation set includes 20% of the patches, and the test set contains the remaining 10%. Moreover, the dataset provides two types of splits: the easy split, where patches from the same tile can appear in different sets (patchwise splitting), and the hard split, where all patches from a single tile are restricted to the same set (tilewise splitting).

All experiments in this work were conducted using the hard split configuration. The use of this new dataset with a sensor with a larger number of spectral channels, as well as a significantly larger number of images compared to traditionally used datasets, such as Pavia, allowed for more reliable model testing, avoiding the risk of overfitting, which afflicts the vast majority of architectures on smaller datasets.

We remark that approximately 8% of the images that are provided in the HySpecNet-11k dataset present some zeroed-out channels, with variability in which channels present this behavior. In our experiments, these channels were not excluded during training and were used as they were, but they were discarded in the computation of the quality metrics during testing as they represented unreliable and unrealistic predictions. We also used classic augmentations in order to increase the number of training samples by a factor of 8. We employed bicubic downsampling on the images before feeding them into the models,

resulting in inputs with dimensions $32 \times 32 \times 202$ for SR factor $r = 4$ and $64 \times 64 \times 202$ for $r = 2$.

We also conducted experiments on the Chikusei [61], Houston [62], and Pavia [63] datasets. Regarding the Chikusei and Houston datasets, we followed exactly the same pipeline as employed in [4] to crop each original image and obtain the train, validation, and test splits for each of the two. With respect to the Pavia dataset, we followed the cropping and splitting strategy introduced in [12]. It is important to note that this choice was due to the fact that CST requires specific spatial sizes to work: in [4], the test images utilized had spatial sizes 224×224, incompatible with the CST architecture, whereas, in [12], the authors used 256×256 test crops. For all four datasets, no band selection or denoising was applied; however, images were normalized to the $[0, 1]$ range for both training and testing.

Table 1 presents the results on HySpecNet-11k in the $4\times$ SR setting and also summarizes the complexity of all methods in terms of the number of trainable parameters and the number of floating point operations per image pixel. We first note that the 31 K FLOPs/pixel values of DPSR are significantly smaller than those of the state-of-the-art methods, often by an order of magnitude.

Table 1. HySpecNet-11k dataset. $4\times$ super-resolution. Parameters and FLOPs/pixel for input size $32 \times 32 \times 202$.

Method	MPSNR (dB) ↑	MSSIM ↑	SAM ↓	RMSE ↓	Params	FLOPs/px
Bicubic	41.00	0.9004	4.1767	0.0147	–	–
GDRRN [23]	41.42	0.9167	4.0681	0.0132	613 K	284 K
SSPSR [24]	42.90	0.9413	3.6765	0.0108	14.07 M	2569 K
SNLSR [7]	42.94	0.9389	3.6062	0.0110	1.81 M	35 K
EUNet [13]	42.96	0.9386	3.6011	0.0110	1.01 M	186 K
MSDformer [4]	43.45	0.9469	3.3702	0.0103	17.49 M	714 K
CST [12]	43.61	0.9495	3.3599	0.0100	11.48 M	245 K
DPSR	**43.17**	**0.9432**	**3.5010**	**0.0106**	**2.71 M**	**31 K**

↑ means higher is better; ↓ means lower is better.

Despite this, we notice that DPSR delivers image quality close to that of significantly more complex methods. Indeed, it is only 0.28 dB and 0.44 dB below that of the complex Transformer-based MSDformer and CST models, respectively, while outperforming all other recent state-of-the-art models. Similar considerations hold for the $2\times$ SR factor on HySpecNet-11k presented in Table 2. For the Houston urban scene imagery, Tables 3 and 4 show that the reconstruction fidelity of DPSR is again very close to that of the leading models for both SR factors. The Pavia benchmarks presented in Tables 5 and 6 also confirm the trends observed on the other datasets, highlighting the excellent complexity–quality tradeoff offered by DPSR. Finally, regarding the results obtained on Chikusei, Tables 7 and 8 show that DPSR essentially matches MSDformer and comes within 0.1 dB of the CST PSNR.

Table 2. HySpecNet-11k dataset. $2\times$ super-resolution. Parameters and FLOPs/pixel for input size $32 \times 32 \times 202$.

Method	MPSNR (dB) ↑	MSSIM ↑	SAM ↓	RMSE ↓	Params	FLOPs/px
Bicubic	45.68	0.9654	2.8981	0.0083	–	–
GDRRN [23]	47.30	0.9802	2.6726	0.0063	613 K	71 K
SSPSR [24]	48.56	0.9859	2.3735	0.0053	11.72 M	1430 K
SNLSR [7]	–	–	–	–	–	–
EUNet [13]	48.62	0.9850	2.3411	0.0055	944 K	65 K

Table 2. *Cont.*

Method	MPSNR (dB) ↑	MSSIM ↑	SAM ↓	RMSE ↓	Params	FLOPs/px
MSDformer [4]	49.44	0.9879	2.0866	0.0049	15.42 M	528 K
CST [12]	49.77	0.9894	2.0900	0.0045	10.31 M	121 K
DPSR	**48.85**	**0.9862**	**2.2644**	**0.0052**	**2.07 M**	**20 K**

↑ means higher is better; ↓ means lower is better.

Table 3. Houston dataset. 4× super-resolution.

Method	MPSNR (dB) ↑	MSSIM ↑	SAM ↓	RMSE ↓
Bicubic	44.21	0.9657	2.4038	0.0074
GDRRN [23]	45.80	0.9772	2.0724	0.0059
SSPSR [24]	46.93	0.9814	1.7369	0.0053
SNLSR [7]	46.84	0.9808	1.7581	0.0054
EUNet [13]	46.81	0.9808	1.7227	0.0054
MSDformer [4]	46.85	0.9808	1.7244	0.0054
CST [12]	47.63	0.9840	1.6358	0.0049
DPSR	**47.53**	**0.9834**	**1.6451**	**0.0050**

↑ means higher is better; ↓ means lower is better.

Table 4. Houston dataset. 2× super-resolution.

Method	MPSNR (dB) ↑	MSSIM ↑	SAM ↓	RMSE ↓
Bicubic	50.75	0.9927	1.2085	0.0034
GDRRN [23]	52.68	0.9954	1.0355	0.0027
SSPSR [24]	53.24	0.9959	0.9531	0.0025
SNLSR [7]	–	–	–	–
EUNet [13]	53.51	0.9961	0.9089	0.0024
MSDformer [4]	53.25	0.9959	0.9437	0.0025
CST [12]	53.84	0.9964	0.8866	0.0023
DPSR	**53.57**	**0.9961**	**0.9103**	**0.0024**

↑ means higher is better; ↓ means lower is better.

Table 5. Pavia dataset. 4× super-resolution.

Method	MPSNR (dB) ↑	MSSIM ↑	SAM ↓	RMSE ↓
Bicubic	27.69	0.7332	6.5587	0.0429
GDRRN [23]	28.70	0.7981	6.4347	0.0378
SSPSR [24]	28.99	0.8116	5.7788	0.0367
SNLSR [7]	28.78	0.8038	5.8878	0.0374
EUNet [13]	28.80	0.8037	5.7460	0.0374
MSDformer [4]	28.95	0.8088	5.7940	0.0368
CST [12]	29.13	0.8163	5.7399	0.0360
DPSR	**29.05**	**0.8151**	**5.9737**	**0.0363**

↑ means higher is better; ↓ means lower is better.

Table 6. Pavia dataset. 2× super-resolution.

Method	MPSNR (dB) ↑	MSSIM ↑	SAM ↓	RMSE ↓
Bicubic	32.39	0.9187	4.6856	0.0245
GDRRN [23]	34.89	0.9513	4.0743	0.0183
SSPSR [24]	35.73	0.9598	3.6963	0.0166
SNLSR [7]	–	–	–	–
EUNet [13]	35.80	0.9604	3.6056	0.0165

Table 6. *Cont.*

Method	MPSNR (dB) ↑	MSSIM ↑	SAM ↓	RMSE ↓
MSDformer [4]	35.86	0.9610	3.6323	0.0164
CST [12]	36.17	0.9634	3.5850	0.0158
DPSR	**35.87**	**0.9612**	**3.6160**	**0.0164**

↑ means higher is better; ↓ means lower is better.

Table 7. Chikusei dataset. 4× super-resolution.

Method	MPSNR (dB) ↑	MSSIM ↑	SAM ↓	RMSE ↓
Bicubic	37.64	0.8954	3.4040	0.0156
GDRRN [23]	39.09	0.9265	3.0536	0.0130
SSPSR [24]	39.98	0.9393	2.4864	0.0119
SNLSR [7]	39.90	0.9387	2.4722	0.0119
EUNet [13]	39.94	0.9398	2.3923	0.0121
MSDformer [4]	40.09	0.9405	2.3981	0.0118
CST [12]	40.24	0.9431	2.3453	0.0116
DPSR	**40.07**	**0.9410**	**2.3695**	**0.0118**

↑ means higher is better; ↓ means lower is better.

Table 8. Chikusei dataset. 2× super-resolution.

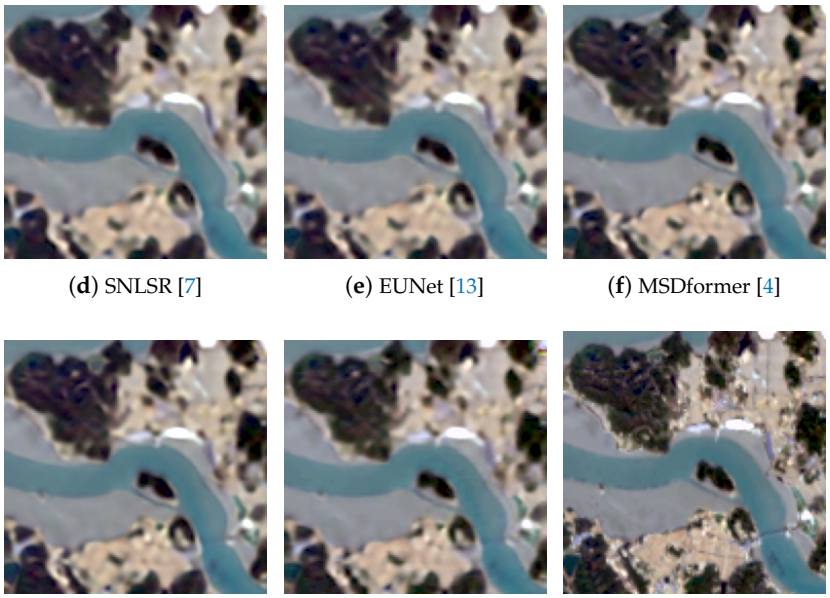
Method	MPSNR (dB) ↑	MSSIM ↑	SAM ↓	RMSE ↓
Bicubic	43.21	0.9721	1.7880	0.0082
GDRRN [23]	46.43	0.9869	1.3911	0.0056
SSPSR [24]	47.41	0.9893	1.2035	0.0051
SNLSR [7]	–	–	–	–
EUNet [13]	47.82	0.9903	1.1143	0.0048
MSDformer [4]	47.69	0.9899	1.1457	0.0049
CST [12]	48.09	0.9908	1.1057	0.0047
DPSR	**47.49**	**0.9894**	**1.1461**	**0.0050**

↑ means higher is better; ↓ means lower is better.

Figures 4–7 show a qualitative comparison of the methods and visualization of the average errors on two test scenes from the HySpecNet-11k dataset. They confirm that no visual artifacts are introduced by the line-based approach. Figure 8 shows a scene from the Houston dataset and Figure 9 reports the per-band mean absolute error on the test set of the Houston dataset. It is clear that DPSR is almost as accurate as CST in every band.

(**a**) Bicubic (**b**) GDRRN [23] (**c**) SSPSR [24]

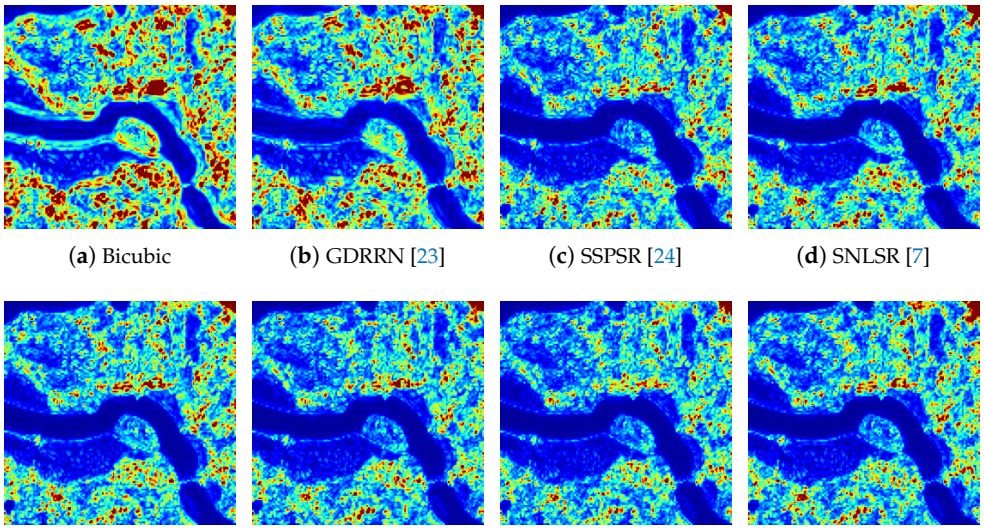
Figure 4. *Cont.*



(**d**) SNLSR [7] (**e**) EUNet [13] (**f**) MSDformer [4]

(**g**) CST [12] (**h**) DPSR (**i**) Ground Truth

Figure 4. Qualitative comparison of methods (GDRRN [23], SSPSR [24], SNLSR [7], EUNet [13], MSDformer [4], CST [12]) on HySpecNet-11k test image 247 with SR factor 4×. Bands 43, 28, 10 were used as R, G, B.



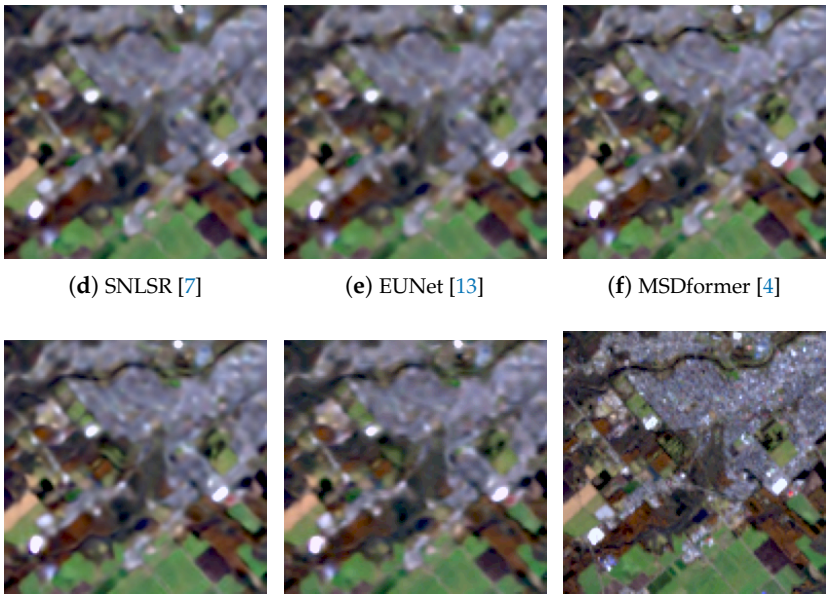
(**a**) Bicubic (**b**) GDRRN [23] (**c**) SSPSR [24] (**d**) SNLSR [7]

(**e**) EUNet [13] (**f**) MSDformer [4] (**g**) CST [12] (**h**) DPSR

Figure 5. Mean (across bands) error heatmaps of methods (GDRRN [23], SSPSR [24], SNLSR [7], EUNet [13], MSDformer [4], CST [12]) on HySpecNet-11k test image 247 with SR factor 4×. Red denotes higher absolute error.

(**a**) Bicubic (**b**) GDRRN [23] (**c**) SSPSR [24]

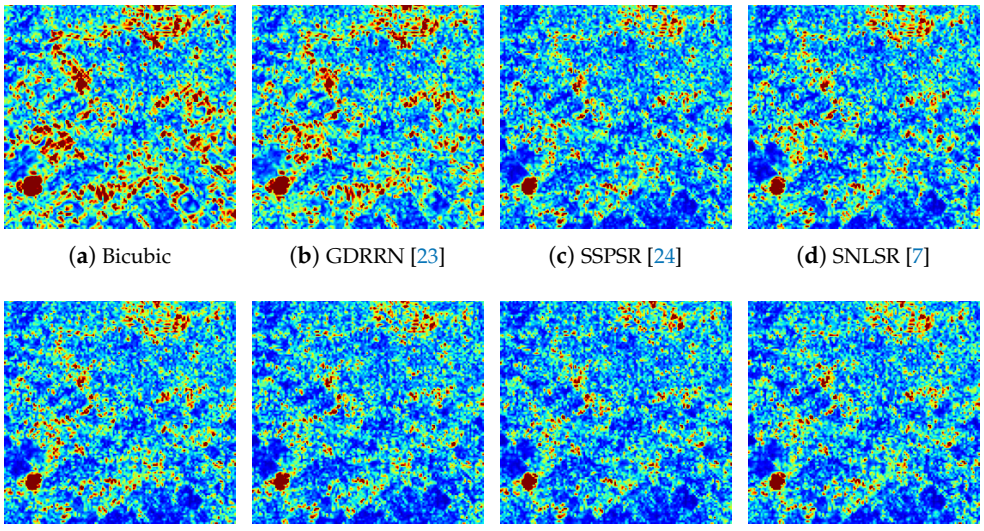
Figure 6. *Cont.*



(**d**) SNLSR [7] (**e**) EUNet [13] (**f**) MSDformer [4]

(**g**) CST [12] (**h**) DPSR (**i**) Ground Truth

Figure 6. Qualitative comparison of methods (GDRRN [23], SSPSR [24], SNLSR [7], EUNet [13], MSDformer [4], CST [12]) on HySpecNet-11k test image 628 with SR factor 4×. Bands 43, 28, 10 were used as R, G, B.



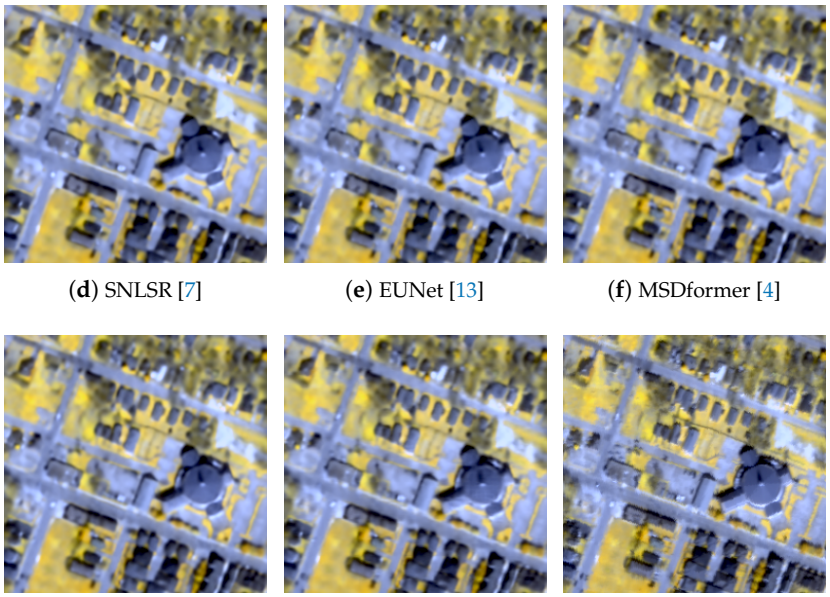
(**a**) Bicubic (**b**) GDRRN [23] (**c**) SSPSR [24] (**d**) SNLSR [7]

(**e**) EUNet [13] (**f**) MSDformer [4] (**g**) CST [12] (**h**) DPSR

Figure 7. Mean (across bands) error heatmaps of methods (GDRRN [23], SSPSR [24], SNLSR [7], EUNet [13], MSDformer [4], CST [12]) on HySpecNet-11k test image 628 with SR factor 4×. Red denotes higher absolute error.

(**a**) Bicubic (**b**) GDRRN [23] (**c**) SSPSR [24]

Figure 8. *Cont.*



(**d**) SNLSR [7] (**e**) EUNet [13] (**f**) MSDformer [4]

(**g**) CST [12] (**h**) DPSR (**i**) Ground Truth

Figure 8. Qualitative comparison of methods (GDRRN [23], SSPSR [24], SNLSR [7], EUNet [13], MSDformer [4], CST [12]) on a Houston test image with SR factor 4×. Bands 29, 26, 19 were used as R, G, B following [4].

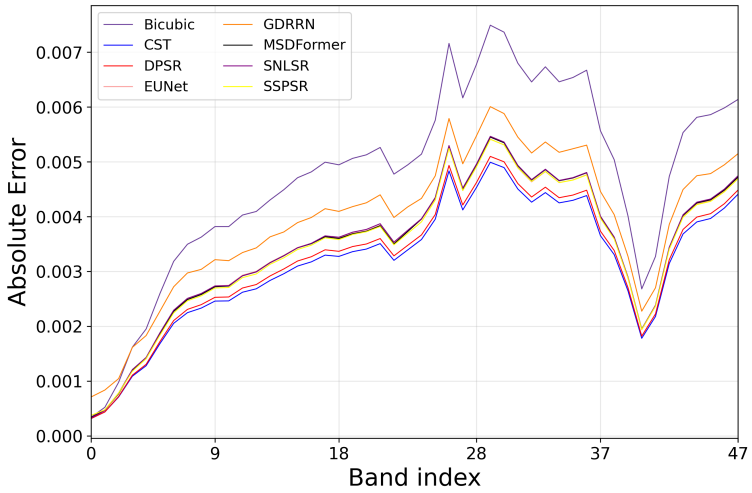


Figure 9. Per-band mean absolute error across the SR images of the Houston test set.

4.2. Analysis of Memory and Computational Efficiency

We now focus on better investigating the efficiency of the proposed DPSR model with respect to the state-of-the-art baselines. In particular, we first analyze the memory requirements of each method and how they scale with the image size. Figures 10 and 11 report the GPU memory usage as a function of each image dimension, while the other two are fixed to values that are representative of real images acquired by satellites. The results clearly illustrate the efficiency of our model. First, we notice that DPSR consistently requires four to eight times less memory than the other methods. In absolute terms, an image of size $1000 \times 1000 \times 66$, which constitutes a frame of the PRISMA VNIR instrument, can be processed by DPSR with less than 1GB of memory, a size compatible with current low-power computing platforms. On the contrary, the state-of-the-art models require more than the 24 GB of VRAM available on the GPU used for this experiment; hence, this point is not reflected in the previous figures. We also remark that the memory usage is constant with the number of lines by design. While it grows linearly with the number of columns,

this dependency is very weak, and we do not see this trend in the figures due to the value being very small and superseded by other aspects of the implementation overhead.

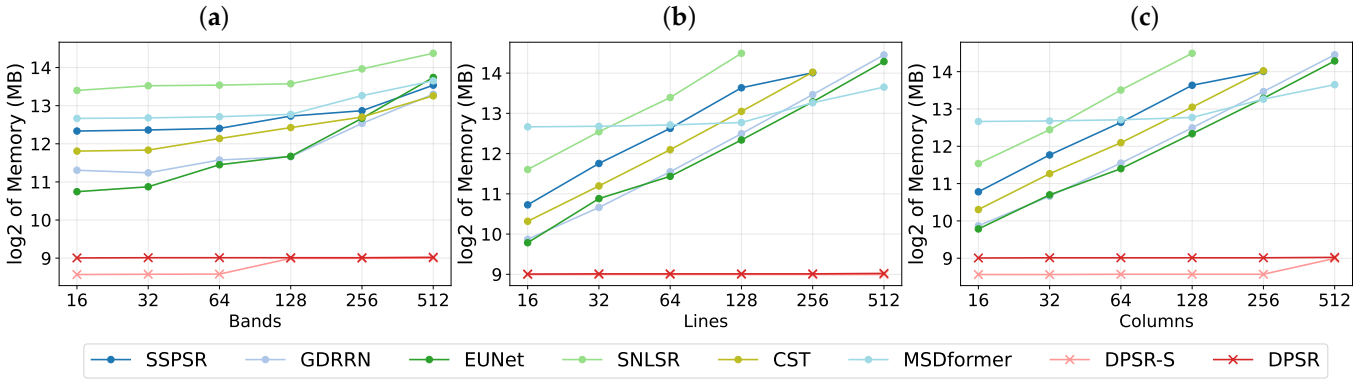


Figure 10. Memory usage comparison for 4× SR. Input image size is (leftmost to rightmost figure) (**a**) 256 × 256 × C, (**b**) H × 1000 × 66, (**c**) 1000 × W × 66. SNLSR runs out of memory (>24 GB) when H, W exceed 128 pixels, and CST and SSPSR when H, W exceed 256 pixels. Note that CST does not accept a spatial size of 1000 as input, so we set it to 992.

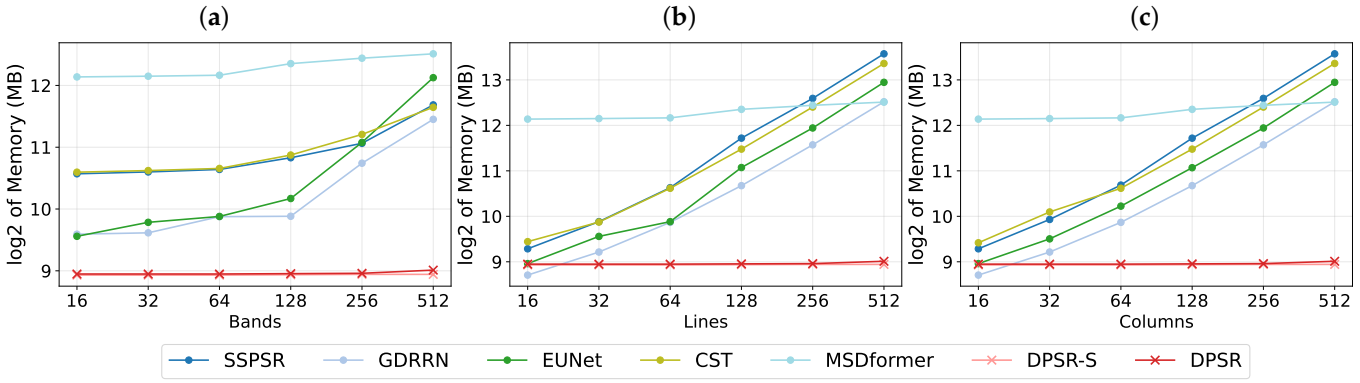
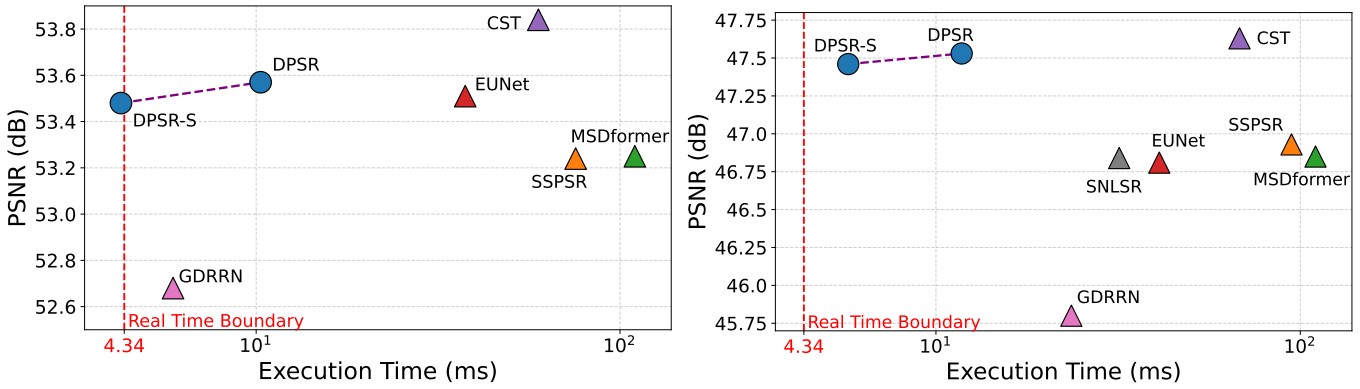


Figure 11. Memory usage comparison for 2× SR. Input image size is (leftmost to rightmost figure) (**a**) 256 × 256 × C, (**b**) H × 1000 × 66, (**c**) 1000 × W × 66. Note that CST does not accept a spatial size of 1000 as input, so we set it to 992. SNLSR does not support 2× SR.

In addition to testing the memory requirements, we also characterize the performance of the method in terms of the inference speed on a low-power platform, namely the NVIDIA Jetson Orin Nano, which requires only 15 W of power for the full system. We benchmark every model in terms of half-precision floating points, but without further optimization. We use a test input with 1000 columns and 66 bands. We present the results for two variants of DPSR, namely the one with $F = 280$ features reported in the previous experiments (DPSR) and a smaller version with $F = 128$ (DPSR-S). Figure 12 presents the resulting runtime normalized by the number of lines processed to have an equivalent line processing time. This allows a comparison with the actual system time needed to acquire a line by existing pushbroom sensors to determine whether a method can perform super-resolution in real time. For example, the PRISMA VNIR instrument acquires 1 × 1000 × 66 every 4.34 ms [15]. We can see that DPSR-S requires 4.25 ms for 2× SR and 5.70 ms for 4× SR, thus satisfying or being marginally above the real-time requirements. DPSR requires a slightly longer duration of 10.28 ms and 11.76 ms, respectively, but it is still substantially faster than the state-of-the-art methods. We remark that our current implementation is not yet hardware-optimized. While the unoptimized DPSR-S already satisfies the real-time speed requirements, the base model falls slightly short, although it still outperforms the state-of-the-art methods by a significant margin. Future work will explore hardware-aware

optimizations, such as kernel-level tuning and FPGA deployment, to further enhance the throughput. At the same time, we note that the state-of-the-art methods exhibit significantly higher runtimes, which do not allow them to support real-time processing.



(**a**) (**b**)

Figure 12. Line processing time ($W = 1000$, $C = 66$) on NVIDIA Jetson Orin Nano and PSNR on Houston. (**a**): 2× SR. (**b**): 4× SR. Notice the dashed line representing the line acquisition time of 4.34 ms.

4.3. Ablation Experiments

In this section, we validate some design choices of our model through ablation experiments. Throughout the tables, the parameters, FLOPs/px, and memory are reported for inputs of size 1 × 1000 × 66 to emulate the onboard sizes of the PRISMA satellite, in order to provide consistency across different experiments and to simulate onboard deployment metrics.

We first assess the effectiveness of the memory mechanism implemented by Mamba by comparing it against alternative approaches. In particular, we consider alternatives based on simple causal convolution, an LSTM recurrent neural network [33], and a recently proposed improvement over Mamba (Mamba-2) [68] with an expansion factor of 1, head dimension of 10, number of heads of 28, and number of groups of 1, using its multi-input SSM version.

These approaches replace the Mamba block in our architecture while retaining the rest of the architecture as a way to measure only the effectiveness of the line memory mechanism. Table 9 shows the results of this experiment, which confirms that Mamba outperforms other approaches. Mamba-2 does not seem to provide significant improvements over the original Mamba for our setting, so we retain the simpler Mamba design.

Table 9. Architecture ablation. MPSNR is computed on the test set of the HySpecNet-11k dataset with an SR factor of 4. Parameters and FLOPs/px are computed on inputs of size 1 × 1000 × 66, reflecting the acquisition dimensions of the PRISMA VNIR instrument, in order to simulate onboard deployment.

Model	MPSNR (dB)	Params	FLOPs/px
CausalConv	42.96	2.68 M	138.78 K
LSTM	43.05	3.31 M	100.90 K
Mamba	**43.17**	2.57 M	78.18 K
Mamba-2	43.12	**2.55** M	**77.72 K**

Best result is highlighted in bold if applicable.

Moreover, Table 10 reports how the the proposed architecture scales with respect to the number of features F across all datasets. We compare the DPSR-S ($F = 128$) model,

whose unoptimized implementation already reaches real-time performance, and the base DPSR ($F = 280$) model, which we conjecture could reach real-time speeds with some optimized implementation. Overall, we notice that the smaller model is still quite close to the state-of-the-art baselines in terms of image quality.

Table 10. Model size ablation. The numbers reported represent the MPSNR (dB) under two super-resolution factors across four datasets.

SR Factor	Features	HySpecNet-11k	Chikusei	Houston	Pavia
2×	$F = 128$	48.35	47.36	53.48	35.80
	$F = 280$	**48.85**	**47.49**	**53.57**	**35.87**
4×	$F = 128$	42.80	40.07	47.46	29.00
	$F = 280$	**43.17**	**40.07**	**47.53**	**29.05**

Best MPSNR per SR factor and dataset is highlighted in bold.

Table 11 shows a comparison between DPSR and a preliminary design (LineSR) that we proposed in [16]. The results clearly show that DPSR yields significantly better results at reduced computational and memory requirements.

Table 11. Comparison with LineSR [16]. MPSNR is computed on the test set of the HySpecNet-11k dataset with an SR factor of 4. Parameters, FLOPs/px, and memory are computed on inputs of size $1 \times 1000 \times 66$, reflecting the acquisition dimensions of the PRISMA VNIR instrument, in order to simulate onboard deployment.

Model	MPSNR (dB)	Params	FLOPs/px	Memory
LineSR	41.82	**95.11 K**	173.78 K	2684.62 MiB
DPSR	**43.17**	2.57 M	**78.18 K**	**204.58 MiB**

Bold numbers indicate improvements over LineSR.

Then, we examine the effect of the internal expansion factor of the Mamba block in Table 12. We chose to fix $E = 1$ in order to optimize the memory and power requirements and also in order not to degrade the inference speed, given the trade-off between the performance gain and efficiency drop. In particular, an expansion factor $E = 2$ would require 16.5% more FLOPs/px and 18.1% more memory while delivering a negligible quality improvement in terms of the MPSNR on the test set of the Houston dataset—47.55 dB instead of 47.53 dB.

Table 12. Impact of Mamba internal expansion factor E on the results of DPSR. MPSNR is computed on the test set of the Houston dataset with an SR factor of 4. Parameters, FLOPs/px, and memory are computed on inputs of size $1 \times 1000 \times 66$, reflecting the acquisition dimensions of the PRISMA VNIR instrument, in order to simulate onboard deployment. Note that, throughout the paper, DPSR makes use of $E = 1$.

Model	MPSNR (dB)	Params	FLOPs/px	Memory
E = 1	47.53	**2.57 M**	**78.18 K**	**204.58 MiB**
E = 2	**47.55**	3.09 M	93.63 K	249.91 MiB

Bold numbers indicate improvements over DPSR with an expansion factor of 1.

We show in Table 13 that allowing the network to learn a residual correction from bilinear upsampling delivers better performance, i.e., around 0.1 dB, with a negligible parameter and memory overhead.

Finally, in Table 14, we show the impact of bilinear and bicubic upsampling. Note that the bilinear interpolator operates using only the current LR line y and the previous line

$y-1$, whereas bicubic interpolation requires the buffering of at least three earlier lines. We note that, in this experiment, bilinear and bicubic residuals achieve the same MPSNR. Since there are no substantial quality benefits, we opt for the simpler and lower-memory bilinear upsampling.

Table 13. Impact of bilinear residual on the results. MPSNR is computed on the test set of the Houston dataset with an SR factor of 4. Parameters, FLOPs/px, and memory are computed on inputs of size $1 \times 1000 \times 66$, reflecting the acquisition dimensions of the PRISMA VNIR instrument, in order to simulate onboard deployment.

Model	MPSNR (dB)	Params	FLOPs/px	Memory
No-Res DPSR	47.44	2.57 M	78.18 K	**204.50 MiB**
DPSR	**47.53**	2.57 M	78.18 K	204.58 MiB

Bold numbers indicate improvements over DPSR without bilinear upsampling.

Table 14. Comparison between the usage of bilinear and bicubic residuals. MPSNR is computed on the test set of the Pavia dataset with an SR factor of 4. Parameters, FLOPs/px, and memory are computed on inputs of size $1 \times 1000 \times 66$, reflecting the acquisition dimensions of the PRISMA VNIR instrument, in order to simulate onboard deployment. Note that, throughout the paper, DPSR makes use of bilinear upsampling.

Model	MPSNR (dB)	Params	FLOPs/px	Memory
Bilinear	29.05	2.57 M	78.18 K	**204.58 MiB**
Bicubic	29.05	2.57 M	78.18 K	205.91 MiB

Bold numbers indicate an improvement for one upsampling strategy over the other.

5. Discussion

This work explored whether an onboard HSI SR model could approach the reconstruction fidelity of state-of-the-art networks while meeting the latency and memory constraints of pushbroom imagers. Our results quantitatively confirm that DPSR achieves an excellent quality–efficiency trade-off: it trails the best-performing high-complexity Transformer models by only small margins (see Tables 1, 3, 5 and 7), while requiring roughly an order of magnitude fewer FLOPs per pixel and substantially less memory (see Figures 10–12). For instance, on the Houston dataset, DPSR is only 0.10 dB (2× SR) below the best model, CST, yet it runs in 10.28 ms instead of 90 ms and with less than 1 GB of memory instead of 20 GB on realistic input sizes. This is all thanks to the line-by-line design of DPSR, which eliminates the need for large spatial buffers and allows for low-complexity models.

5.1. Relationship with Prior Work and Working Hypotheses

DPSR was designed around two hypotheses: (i) that optimizing the architecture of a neural network in terms of inference speed and minimal memory and power requirements can be achieved without sacrificing reconstruction quality and (ii) that causal, linewise processing that matches pushbroom acquisition can retain sufficient spatial–spectral context via compact memory to yield competitive SR quality while satisfying the constraints in (i). The proposed architecture implements these ideas with across-track and spectral feature extraction followed by cross-line fusion, where a selective SSM (Mamba) stores information in the along-track direction and a lightweight upsampler recovers the target spatial resolution. This departs from 2D tile-based designs that must stage large windows to aggregate the context, thus better aligning with the onboard streaming constraint. The proposed method DPSR stems from the work in [16], which gave promising indications that exploiting the Mamba memory to match the pushbroom sensors physics could be effective. However, limited experiments and comparisons with state-of-the-art methods

were presented. This work changed the strategy to process the input hyperspectral cube, expanded and refined the architecture with insightful modifications, and showed the effectiveness of the method through extensive evaluation across four datasets and several ablations to assess the architectural choices. In particular, the hyperspectral images are no longer treated as cubes, but we decided to use the spectral dimension as a feature dimension throughout the whole architecture, allowing a change from 3D to 2D for all operations involved. In addition, we introduced a new block to firstly refine the input based on channel attention, named the SFE block, which improves the simple first convolution in the LineSR model. Finally, we introduced a way to perform residual learning using buffers, with no tangible increase in memory or computational requirements. These modifications provide an entirely different final model, with significantly improved performance (see Table 11) and even lower complexity.

5.2. Main Findings

On HySpecNet-11k (4× SR), DPSR achieves a 43.17 dB MPSNR with only 31 K FLOPs/px, within 0.28–0.44 dB of MSDformer and CST while being significantly lighter (e.g., 714 K and 245 K FLOPs/px for MSDformer and CST, respectively). A similar pattern holds for 2× SR. On the Chikusei, Houston, and Pavia benchmarks, DPSR is consistently competitive, often within 0.1–0.3 dB of the strongest baselines and occasionally surpassing them, while maintaining a much lower FLOPs/px profile. In urban scenes (Houston), DPSR's gap with CST remains small at both 2× SR and 4× SR, indicating that linewise memory is effective even in highly structured environments with many details. For Pavia, the method remains close to the best scores, delivering the second-best quality metrics, only behind CST, as in Houston. These results demonstrate that the long-range along-track context captured by DPSR through Mamba's selective memory enables the accurate reconstruction of fine spatial and spectral details.

Beyond quality, the efficiency analysis reinforces the practical relevance of the design. Memory scales linearly with columns and bands and is independent of the number of lines processed, so a PRISMA-like frame of 1000 × 1000 × 66 can be super-resolved with less than 1 GB of memory, whereas several 2D counterparts exceed 24 GB for similar sizes. The runtime on an NVIDIA Jetson Orin Nano (15 W) reaches 4.25 ms/line (2× SR) and 5.70 ms/line (4× SR) for the small variant (DPSR-S), crossing the real-time threshold set by representative line times (e.g., PRISMA VNIR 4.3 ms/line). The base model remains substantially faster than state-of-the-art models and, given that our implementation is not yet hardware-optimal, future work will explore hardware-aware optimizations, such as kernel-level tuning and FPGA deployment, to further enhance the throughput.

5.3. Why Does the Line-by-Line Strategy Work?

Two design factors appear central. First, the feature extraction pipeline explicitly separates across-track and spectral mixing from along-track memory. The former is addressed by shallow and NAFNet-style separable convolutions with lightweight gating simulating attention, avoiding heavy 2D context aggregation. This preserves spectral information while keeping a sufficient across-track receptive field. On the other hand, Mamba handles the latter: the selective SSM maintains a compact latent state per across-track position, acting as a content-adaptive accumulator of information from past lines. Together with the residual learning on top of bilinear interpolation, this mitigates hallucinations and stabilizes spectral metrics despite the low compute budget.

5.4. Limitations

Despite its strengths, DPSR is not the state of the art in peak image quality: CST still retains a modest PSNR edge. We attribute this to three factors intrinsic to the line-

causal constraint: (i) the model never accesses future lines, whereas 2D tile-based methods can implicitly fuse both past and future spatial contexts; (ii) the selective memory could sometimes discard or forget important information since it needs to compress many details, causing the network to lose some features, since it cannot actually see lines far in the past (more than four lines ago); and (iii) the residual baseline is not followed by learnable layers, which can underfit high-frequency textures in challenging regions, possibly limiting the top quality reachable by the residual learner.

5.5. Implications and Future Directions

In a possible practical onboard pipeline, SR would not be the end task, but it should be paired with a downstream task. DPSR's causal, streaming nature admits straightforward composition with downstream analytics without increasing buffering or breaking the real-time throughput. We see several promising directions:

- **Designing multi-task architectures** that jointly perform real-time denoising and super-resolution, incorporating the advantages of the two proposed methods while delivering real-time performance without increasing the complexity;
- **Integrating downstream tasks**, such as semantic segmentation, into the pipeline after inverse tasks to build an end-to-end efficient onboard processing model;
- **Developing and incorporating selective compression strategies** that leverage the improved quality of super-resolved and denoised images to optimize the downlink efficiency;
- **Hardware–aware kernels,** implementing the model on FPGA in an attempt to bring the base model within real-time speeds even at $F = 280$.

These extensions would maintain the central constraint of the present work—the single-line causality—so that the complexity and memory remain compatible with small onboard accelerators.

6. Conclusions

Overall, DPSR proves that respecting the acquisition physics, i.e., pushbroom, causal access, could be a successful strategy to design neural networks suited for onboard deployment, making such deep learning models strong candidates to be the future of onboard processing systems. Moreover, exploiting selective SSM memory can recover much of the accuracy of heavier 2D models at a fraction of their cost. While a small fidelity gap remains relative to the best-performing state-of-the-art, the ability to operate within a single line time and with minimal memory budgets strongly supports the viability of onboard, real-time HSI super-resolution.

Author Contributions: Conceptualization, D.P., D.V. and E.M.; methodology, D.P., D.V. and E.M.; software, D.P.; validation, D.P.; formal analysis, D.P. and D.V.; investigation, D.P. and D.V.; resources, E.M.; data curation, D.P.; writing—original draft preparation, D.P.; writing—review and editing, D.V. and E.M.; visualization, D.P.; supervision, D.V. and E.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All datasets used are publicly available. Code for this work will be published at https://github.com/DavidePiccinini98/dpsr (Accessed on 27 October 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Cogliati, S.; Sarti, F.; Chiarantini, L.; Cosi, M.; Lorusso, R.; Lopinto, E.; Miglietta, F.; Genesio, L.; Guanter, L.; Damm, A.; et al. The PRISMA imaging spectroscopy mission: Overview and first performance analysis. *Remote Sens. Environ.* **2021**, *262*, 112499. [CrossRef]
2. Brell, M.; Guanter, L.; Segl, K.; Scheffler, D.; Bohn, N.; Bracher, A.; Soppa, M.A.; Foerster, S.; Storch, T.; Bachmann, M.; et al. The EnMAP Satellite–Data Product Validation Activities. In Proceedings of the 2021 11th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS), Amsterdam, The Netherlands, 24–26 March 2021; pp. 1–5. [CrossRef]
3. eoPortal Directory. HySIS (HyperSpectral Imaging Satellite). 2024. Available online: https://www.eoportal.org/satellite-missions/hysis (accessed on 2 May 2025).
4. Chen, S.; Zhang, L.; Zhang, L. MSDformer: Multiscale Deformable Transformer for Hyperspectral Image Super-Resolution. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 1–14. [CrossRef]
5. Xu, Y.; Hou, J.; Zhu, X.; Wang, C.; Shi, H.; Wang, J.; Li, Y.; Ren, P. Hyperspectral Image Super-Resolution With ConvLSTM Skip-Connections. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 1–16. [CrossRef]
6. Wang, H.; Zhang, Q.; Peng, T.; Xu, Z.; Cheng, X.; Xing, Z.; Li, T. SSAformer: Spatial–Spectral Aggregation Transformer for Hyperspectral Image Super-Resolution. *Remote Sens.* **2024**, *16*, 1766. [CrossRef]
7. Hu, Q.; Wang, X.; Jiang, J.; Zhang, X.P.; Ma, J. Exploring the Spectral Prior for Hyperspectral Image Super-Resolution. *IEEE Trans. Image Process.* **2024**, *33*, 5260–5272. [CrossRef]
8. Giuffrida, G.; Fanucci, L.; Meoni, G.; Batič, M.; Buckley, L.; Dunne, A.; van Dijk, C.; Esposito, M.; Hefele, J.; Vercruyssen, N.; et al. The Φ-Sat-1 Mission: The First On-Board Deep Neural Network Demonstrator for Satellite Earth Observation. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–14. [CrossRef]
9. Ružička, V.; Vaughan, A.; De Martini, D.; Fulton, J.; Salvatelli, V.; Bridges, C.; Mateo-Garcia, G.; Zantedeschi, V. RaVÆn: Unsupervised change detection of extreme events using ML on-board satellites. *Sci. Rep.* **2022**, *12*, 16939. [CrossRef]
10. Ziaja, M.; Bosowski, P.; Myller, M.; Gajoch, G.; Gumiela, M.; Protich, J.; Borda, K.; Jayaraman, D.; Dividino, R.; Nalepa, J. Benchmarking Deep Learning for On-Board Space Applications. *Remote Sens.* **2021**, *13*, 3981. [CrossRef]
11. Inzerillo, G.; Valsesia, D.; Magli, E. Efficient Onboard Multitask AI Architecture Based on Self-Supervised Learning. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2025**, *18*, 828–838. [CrossRef]
12. Chen, S.; Zhang, L.; Zhang, L. Cross-Scope Spatial-Spectral Information Aggregation for Hyperspectral Image Super-Resolution. *IEEE Trans. Image Process.* **2024**, *33*, 5878–5891. [CrossRef] [PubMed]
13. Liu, D.; Li, J.; Yuan, Q.; Zheng, L.; He, J.; Zhao, S.; Xiao, Y. An efficient unfolding network with disentangled spatial-spectral representation for hyperspectral image super-resolution. *Inf. Fusion* **2023**, *94*, 92–111. [CrossRef]
14. Gu, A.; Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv* **2023**, arXiv:2312.00752. [CrossRef]
15. Italiana, A.S. PRISMA Algorithm Theoretical Basis Document (ATBD). 2021. Available online: https://prisma.asi.it/missionselect/docs/PRISMA%20ATBD_v1.pdf (accessed on 7 May 2025).
16. Piccinini, D.; Valsesia, D.; Maglia, E. Towards deep line-based architectures for onboard hyperspectral image super-resolution. In Proceedings of the 2025 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Brisbane, Australia, 3–8 August 2025.
17. Valsesia, D.; Magli, E. Permutation Invariance and Uncertainty in Multitemporal Image Super-Resolution. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–12. [CrossRef]
18. Huang, H.; Christodoulou, A.G.; Sun, W. Super-resolution hyperspectral imaging with unknown blurring by low-rank and group-sparse modeling. In Proceedings of the 2014 IEEE International Conference on Image Processing (ICIP), Paris, France, 27–30 October 2014; pp. 2155–2159. [CrossRef]
19. Xu, X.; Tong, X.; Li, J.; Xie, H.; Zhong, Y.; Zhang, L.; Song, D. Hyperspectral image super resolution reconstruction with a joint spectral-spatial sub-pixel mapping model. In Proceedings of the 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Beijing, China, 10–15 July 2016; pp. 6129–6132. [CrossRef]
20. He, S.; Zhou, H.; Wang, Y.; Cao, W.; Han, Z. Super-resolution reconstruction of hyperspectral images via low rank tensor modeling and total variation regularization. In Proceedings of the 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Beijing, China, 10–15 July 2016; pp. 6962–6965. [CrossRef]
21. Mei, S.; Yuan, X.; Ji, J.; Zhang, Y.; Wan, S.; Du, Q. Hyperspectral Image Spatial Super-Resolution via 3D Full Convolutional Neural Network. *Remote Sens.* **2017**, *9*, 139. [CrossRef]
22. Arun, P.V.; Buddhiraju, K.M.; Porwal, A.; Chanussot, J. CNN-Based Super-Resolution of Hyperspectral Images. *IEEE Trans. Geosci. Remote Sens.* **2020**, *58*, 6106–6121. [CrossRef]
23. Li, Y.; Zhang, L.; Dingl, C.; Wei, W.; Zhang, Y. Single Hyperspectral Image Super-Resolution with Grouped Deep Recursive Residual Network. In Proceedings of the 2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM), Xi'an, China, 13–16 September 2018; pp. 1–4. [CrossRef]

24. Jiang, J.; Sun, H.; Liu, X.; Ma, J. Learning Spatial-Spectral Prior for Super-Resolution of Hyperspectral Imagery. *IEEE Trans. Comput. Imaging* **2020**, *6*, 1082–1096. [CrossRef]
25. Liu, Y.; Hu, J.; Kang, X.; Luo, J.; Fan, S. Interactformer: Interactive Transformer and CNN for Hyperspectral Image Super-Resolution. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 1–15. [CrossRef]
26. Zhang, M.; Zhang, C.; Zhang, Q.; Guo, J.; Gao, X.; Zhang, J. ESSAformer: Efficient Transformer for Hyperspectral Image Super-resolution. In Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 1–6 October 2023; pp. 23016–23027. [CrossRef]
27. Wang, X.; Ma, J.; Jiang, J.; Zhang, X.P. Dilated projection correction network based on autoencoder for hyperspectral image super-resolution. *Neural Netw.* **2022**, *146*, 107–119. [CrossRef] [PubMed]
28. Dong, Y.; Yang, B.; Liu, C.; Geng, Z.; Wang, T. Hyperspectral image super-resolution via joint network with spectral-spatial strategy. *Geo-Spat. Inf. Sci.* **2025**, 1–19. [CrossRef]
29. Cheng, Y.; Ma, Y.; Fan, F.; Ma, J.; Yao, Y.; Mei, X. Latent spectral-spatial diffusion model for single hyperspectral super-resolution. *Geo-Spat. Inf. Sci.* **2025**, 1–16. [CrossRef]
30. Liu, J.; Wu, Z.; Xiao, L. A Spectral Diffusion Prior for Unsupervised Hyperspectral Image Super-Resolution. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 1–13. [CrossRef]
31. Wang, D.; Hu, M.; Jin, Y.; Miao, Y.; Yang, J.; Xu, Y.; Qin, X.; Ma, J.; Sun, L.; Li, C.; et al. HyperSIGMA: Hyperspectral Intelligence Comprehension Foundation Model. *IEEE Trans. Pattern Anal. Mach. Intell.* **2025**, *47*, 6427–6444. [CrossRef]
32. Van Den Oord, A.; Kalchbrenner, N.; Kavukcuoglu, K. Pixel recurrent neural networks. In Proceedings of the 33rd International Conference on Machine Learning (ICML 2016), New York, NY, USA, 19–24 June 2016; Volume 48, pp. 1747–1756. [CrossRef]
33. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
34. Cho, K.; Van Merriënboer, B.; Bahdanau, D.; Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv* **2014**, arXiv:1409.1259. [CrossRef]
35. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555. [CrossRef]
36. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS 2017), Long Beach, CA, USA, 4–9 December 2017; Volume 30, pp. 5998–6008. [CrossRef]
37. Team, G.; Anil, R.; Borgeaud, S.; Alayrac, J.B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A.M.; Hauth, A.; Millican, K.; et al. Gemini: A family of highly capable multimodal models. *arXiv* **2023**, arXiv:2312.11805. [CrossRef]
38. Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. Llama: Open and efficient foundation language models. *arXiv* **2023**, arXiv:2302.13971. [CrossRef]
39. Hatamizadeh, A.; Song, J.; Liu, G.; Kautz, J.; Vahdat, A. Diffit: Diffusion vision transformers for image generation. In Proceedings of the European Conference on Computer Vision, Milan, Italy, 29 September–4 October 2024; pp. 37–55. [CrossRef]
40. Yu, L.; Cheng, Y.; Sohn, K.; Lezama, J.; Zhang, H.; Chang, H.; Hauptmann, A.G.; Yang, M.H.; Hao, Y.; Essa, I.; et al. Magvit: Masked generative video transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 10459–10469. [CrossRef]
41. Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Ronneberger, O.; Tunyasuvunakool, K.; Bates, R.; Žídek, A.; Potapenko, A.; et al. Highly accurate protein structure prediction with AlphaFold. *Nature* **2021**, *596*, 583–589. [CrossRef]
42. Child, R.; Gray, S.; Radford, A.; Sutskever, I. Generating long sequences with sparse transformers. *arXiv* **2019**, arXiv:1904.10509. [CrossRef]
43. Katharopoulos, A.; Vyas, A.; Pappas, N.; Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In Proceedings of the International Conference on Machine Learning (ICML), PMLR, Vienna, Austria, 12–18 July 2020; pp. 5156–5165. [CrossRef]
44. Gu, A.; Goel, K.; Ré, C. Efficiently modeling long sequences with structured state spaces. *arXiv* **2021**, arXiv:2111.00396. [CrossRef]
45. Dao, T.; Fu, D.Y.; Saab, K.K.; Thomas, A.W.; Rudra, A.; Ré, C. Hungry Hungry Hippos: Towards Language Modeling with State Space Models. In Proceedings of the 11th International Conference on Learning Representations (ICLR), Kigali, Rwanda, 1–5 May 2023. [CrossRef]
46. Peng, B.; Alcaide, E.; Anthony, Q.G.; Albalak, A.; Arcadinho, S.; Biderman, S.; Cao, H.; Cheng, X.; Chung, M.N.; Derczynski, L.; et al. RWKV: Reinventing RNNs for the Transformer Era. In Proceedings of the the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, 6–10 December 2023. [CrossRef]
47. Poli, M.; Massaroli, S.; Nguyen, E.; Fu, D.Y.; Dao, T.; Baccus, S.; Bengio, Y.; Ermon, S.; Ré, C. Hyena hierarchy: Towards larger convolutional language models. In Proceedings of the International Conference on Machine Learning, PMLR, Honolulu, HI, USA, 23–29 July 2023; pp. 28043–28078. [CrossRef]

48. Zhu, L.; Liao, B.; Zhang, Q.; Wang, X.; Liu, W.; Wang, X. Vision mamba: Efficient visual representation learning with bidirectional state space model. In Proceedings of the 41st International Conference on Machine Learning (ICML), Vienna, Austria, 21–27 July 2024; pp. 62429–62442. [CrossRef]
49. Hatamizadeh, A.; Kautz, J. Mambavision: A hybrid mamba-transformer vision backbone. In Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR), Nashville, TN, USA, 11–15 June 2025; pp. 25261–25270. [CrossRef]
50. Yao, J.; Hong, D.; Li, C.; Chanussot, J. Spectralmamba: Efficient mamba for hyperspectral image classification. *arXiv* **2024**, arXiv:2404.08489. [CrossRef]
51. Huang, L.; Chen, Y.; He, X. Spectral-Spatial Mamba for Hyperspectral Image Classification. *Remote Sens.* **2024**, *16*, 2449. [CrossRef]
52. Shen, D.; Zhu, X.; Tian, J.; Liu, J.; Du, Z.; Wang, H.; Ma, X. HTD-Mamba: Efficient Hyperspectral Target Detection With Pyramid State Space Model. *IEEE Trans. Geosci. Remote Sens.* **2025**, *63*, 1–15. [CrossRef]
53. Yao, Y.; Jiang, Z.; Zhang, H.; Zhou, Y. On-Board Ship Detection in Micro-Nano Satellite Based on Deep Learning and COTS Component. *Remote Sens.* **2019**, *11*, 762. [CrossRef]
54. Kervennic, E.; Louis, T.; Benguigui, M.; Bobichon, Y.; Avaro, N.; Grenet, I.; Férésin, F.; Girard, A. Embedded cloud segmentation using AI: Back on years of experiments in orbit on OPS-SAT. In Proceedings of the 2023 European Data Handling & Data Processing Conference (EDHPC), Juan-Les-Pins, France, 2–6 October 2023; pp. 1–8. [CrossRef]
55. Qiao, Y.; Teng, S.; Luo, J.; Sun, P.; Li, F.; Tang, F. On-Orbit DNN Distributed Inference for Remote Sensing Images in Satellite Internet of Things. *IEEE Internet Things J.* **2025**, *12*, 5687–5703. [CrossRef]
56. Valsesia, D.; Bianchi, T.; Magli, E. Onboard Deep Lossless and Near-Lossless Predictive Coding of Hyperspectral Images With Line-Based Attention. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 1–14. [CrossRef]
57. Shi, W.; Caballero, J.; Huszár, F.; Totz, J.; Aitken, A.P.; Bishop, R.; Rueckert, D.; Wang, Z. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 1874–1883. [CrossRef]
58. Hendrycks, D.; Gimpel, K. Gaussian error linear units (gelus). *arXiv* **2016**, arXiv:1606.08415. [CrossRef]
59. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19. [CrossRef]
60. Chen, L.; Chu, X.; Zhang, X.; Sun, J. Simple baselines for image restoration. In Proceedings of the European Conference on Computer Vision (ECCV), Tel Aviv, Israel, 23–27 October 2022; pp. 17–33. [CrossRef]
61. Yokoya, N.; Iwasaki, A. *Airborne Hyperspectral Data over Chikusei*; Technical Report SAL-2016-05-27; Space Application Laboratory, The University of Tokyo: Tokyo, Japan, 2016; Volume 5, p. 5.
62. Le Saux, B.; Yokoya, N.; Hansch, R.; Prasad, S. 2018 IEEE GRSS Data Fusion Contest: Multimodal Land Use Classification [Technical Committees]. *IEEE Geosci. Remote Sens. Mag.* **2018**, *6*, 52–54. [CrossRef]
63. Benediktsson, J.; Palmason, J.; Sveinsson, J. Classification of hyperspectral data from urban areas based on extended morphological profiles. *IEEE Trans. Geosci. Remote Sens.* **2005**, *43*, 480–491. [CrossRef]
64. Fuchs, M.H.P.; Demir, B. HySpecNet-11k: A Large-Scale Hyperspectral Dataset for Benchmarking Learning-Based Hyperspectral Image Compression Methods. In Proceedings of the IGARSS 2023–2023 IEEE International Geoscience and Remote Sensing Symposium, Pasadena, CA, USA, 16–21 July 2023; pp. 1779–1782. [CrossRef]
65. Mannos, J.; Sakrison, D. The effects of a visual fidelity criterion of the encoding of images. *IEEE Trans. Inf. Theory* **1974**, *20*, 525–536. [CrossRef]
66. Wang, Z.; Bovik, A.; Sheikh, H.; Simoncelli, E. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef] [PubMed]
67. Yuhas, R.H.; Goetz, A.F.; Boardman, J.W. Discrimination among semi-arid landscape endmembers using the spectral angle mapper (SAM) algorithm. In *Summaries of the Third Annual JPL Airborne Geoscience Workshop, Volume 1: AVIRIS Workshop, Proceedings of the JPL, Pasadena, CA, USA, 1–5 June 1992*; NASA: Washington, DC, USA, 1992.
68. Dao, T.; Gu, A. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In Proceedings of the 41st International Conference on Machine Learning (ICML), Vienna, Austria, 21–27 July 2024; pp. 10041–10071. [CrossRef]