Onboard Real-Time Hyperspectral Subpixel Mapping With Pushbroom Neural Network

Onboard Real-Time Hyperspectral Subpixel Mapping With Pushbroom Neural Network

Matteo Impieri, Davide Piccinini, *Student Member, IEEE*, Diego Valsesia, *Member, IEEE*, and Enrico Magli, *Fellow, IEEE*

(Special Section: Advances in Real-Time Image and Video Processing Techniques for Remote Sensing Applications)

***Abstract*—Satellite hyperspectral images provide fine spectral information but often suffer from coarse spatial resolution. This may result in mixed pixels containing multiple land cover classes, degrading the performance of conventional classification methods. Subpixel mapping (SPM) seeks to assign class labels to multiple subpixels within each coarse pixel, producing a super-resolved classification map. Existing approaches rely on priors that are either handcrafted or obtained via large deep learning architectures for offline processing requiring substantial memory and computational resources. In this work, we focus on the processing setting directly onboard the satellite, where such priors are not available and there are strict constraints on computational complexity. In particular, we propose a line-based deep SPM network tailored for real-time deployment on satellites with a processing strategy that aligns with the pushbroom acquisition of hyperspectral sensors, named deep pushbroom subpixel mapping. The model predicts a super-resolved classification map exploiting spatial–spectral context from neighboring lines, along with long-range information from previous lines through efficient state-space models. The method achieves fast inference and has low memory requirements compatible with resource-constrained satellite platforms. Experiments demonstrate that the proposed method achieves state-of-the-art subpixel hyperspectral mapping accuracy, while supporting real-time inference at the line acquisition rate, as tested on a low-power accelerator.**

***Index Terms*—Hyperspectral segmentation, onboard neural network, pushbroom processing, subpixel mapping (SPM).**

I. Introduction

THE mixed pixel problem [1] is omnipresent in remote sensing images due to the limited spatial resolution of imaging instruments and affects land-use interpretation tasks. A mixed pixel contains the spectral contribution of multiple pure ground cover classes, known as endmembers, effectively mixing information from multiple materials and reducing the effectiveness of classification methods. This particularly affects satellite hyperspectral images, where the system design involves a tradeoff between high spectral resolution and lower spatial resolution. Spectral unmixing [2], [3] is a classic approach to mixed pixels where endmembers and their relative abundances are extracted. This process alleviates the issue to some extent through soft classification, but abundance maps still retain the original spatial resolution.

On the other hand, SubPixel Mapping (SPM) [4], [5], also referred to as super-resolution mapping, aims at finding the location of different materials in the most plausible positions inside a coarse-resolution pixel. By extrapolating multiple subpixels from a single low-resolution (LR) one, this process is therefore equivalent to increasing the spatial resolution of the land cover map by means of a super-resolved semantic segmentation model. In contrast to conventional spectral unmixing, which produces coarse abundance maps, SPM explicitly targets a high-resolution (HR) land-cover map [6], providing sharper object boundaries, more realistic object shapes, and improved support for downstream applications, such as urban monitoring, precision agriculture, and fine-scale environmental analysis.

Classical SPM approaches usually follow an unmixing-then-mapping (UTM) strategy [7], [8]: abundances are first estimated from the LR hyperspectral image, and then a subpixel allocation algorithm reconstructs a fine-grained land-cover map from these fraction images. However, this two-stage pipeline has well-known limitations. The ill-posed nature of SPM implies that many different subpixel configurations are compatible with the same abundance vector, so classical methods must rely on fixed spatial priors that often oversimplify complex and heterogeneous landscapes. Moreover, errors in the spectral unmixing stage propagate directly to the mapping stage, and the sequential nature of the pipeline prevents joint optimization of spectral fidelity and spatial detail. These limitations have motivated a new generation of deep learning-based SPM (DLSPM) methods [9], [10], which learn end-to-end mappings from LR hyperspectral inputs, and sometimes auxiliary soft priors, to HR land cover maps.

DLSPM networks [11], [12] have demonstrated remarkable accuracy and boundary sharpness by exploiting convolutional backbones, attention mechanisms, and multiscale feature hierarchies that implicitly capture spatial–spectral priors.

The focus of our work is to move DLSPM from the ground segment to the satellite payload. This reflects a growing trend in the remote sensing community [13], [14] toward executing inference tasks directly onboard satellites or airborne platforms

Received 19 December 2025; revised 9 February 2026 and 17 April 2026; accepted 11 May 2026. Date of publication 13 May 2026; date of current version 29 May 2026. *(Matteo Impieri and Davide Piccinini contributed equally to this work.) (Corresponding author: Diego Valsesia.)*

The authors are with the Politecnico di Torino,Department of Electronics and Telecommunications, 10129 Torino, Italy (e-mail: matteo.impieri@polito.it; davide.piccinini@polito.it; diego.valsesia@polito.it; enrico.magli@polito.it).

Code and pretrained models will be released at https://github.com/DavidePiccinini98/DeepPushbroomSubPixelMapping.

Digital Object Identifier 10.1109/JSTARS.2026.3692947

for real-time monitoring and decision support. Processing data onboard as they are acquired [15], [16] greatly reduces downlink bandwidth, avoids the latency of transferring massive raw hyperspectral cubes to the ground, and enables time-critical applications, such as disaster response, rapid change detection, or environmental alerts.

Most existing DLSPM architectures are conceived as offline, ground-based systems [10], [17]: they process full images or large 2-D tiles, use heavy encoder–decoder backbones and/or maintain large intermediate feature maps in memory. As a result, their computational and memory footprints are often incompatible with the strict power, memory, and latency constraints of spaceborne platforms, limiting their suitability for truly real-time onboard deployment.

Moreover, several of the most recent DLSPM methods [18], [19] further enhance performance by introducing LR semantic priors derived from ground-truth HR maps, such LR semantic maps obtained by downsampling the HR ground truth. While this strategy has proven highly effective in ground-based settings, it assumes access to an LR version of the very land-cover map the network aims to predict. In realistic onboard processing scenarios, such priors are not available at inference time.

Addressing real-time onboard processing requires to carefully consider the image acquisition pipeline. The most used hyperspectral sensors are pushbroom ones, which acquire the scene line-by-line in the along-track direction. This acquisition geometry naturally suggests line-based processing strategies, in which algorithms operate on a small buffer of consecutive lines instead of full 2-D tiles. Such designs drastically reduce memory usage since the model never needs to hold the full image, and allow real-time streaming operation, where each newly acquired line is processed on-the-fly using only a short along-track context. Recent work on line-based hyperspectral super-resolution [20] has shown that carefully designed neural architectures can match or surpass 2-D methods in quality while using orders of magnitude less memory, making real-time onboard operation feasible on low-power hardware.

For this reason, in this article, we present a line-based deep SPM neural network called DPSPM (Deep Pushbroom SubPixel Mapping) that is specifically designed for real-time, resource-constrained onboard deployment by following the line-based processing philosophy. The proposed model greatly limits memory requirements by operating on a small buffer of three consecutive LR hyperspectral lines: the previous $y-2, y-1$ lines, stored in memory, and y, the newly acquired one. For each triplet, it predicts a super-resolved semantic segmentation map of the middle line $y-1$, producing class labels at a higher spatial resolution both in the across- and along-track directions. This is done by carefully exploiting a spatial–spectral context consisting not only of the neighboring across-track pixels and the immediately neighboring lines, but also of a memory of past processed lines to capture longer range dependencies in a latent space. For this latter aspect, we use the recently proposed Mamba [21] to process the sequence of lines in the along-track direction. Through its selective state space model (SSM), Mamba allows to effectively combine information from past lines in an efficient manner, with a complexity linear in the sequence length and a small memory footprint which is constant regardless of the total number of lines that will eventually be processed. This is in contrast with the popular Transformer [22] alternative, which has quadratic complexity in the sequence length and large memory requirements for the key–value (KV) caching mechanism, making Mamba more suitable for our resource-constrained scenario. Experiments on benchmark hyperspectral datasets show that DPSPM achieves state-of-the-art performance among SPM methods that do not exploit additional side information. Moreover, we present experiments on a low-power embedded platform (Nvidia Jetson Orin Nano), where we show that DPSPM can sustain real-time line-rate inference with low memory requirements, thus enabling continuous SPM as new lines are acquired.

We can summarize our contributions as follows.

1) We introduce the first deep learning-based subpixel mapping framework (DPSPM) explicitly tailored to onboard, real-time operation on pushbroom hyperspectral sensors, directly producing super-resolved semantic segmentation maps.
2) We propose a design based on a triplet-based line processing strategy in which three consecutive LR lines are jointly processed and the middle line is semantically super-resolved at subpixel scale. This exploits multiscale line encoding, across-track refinement (ATR), and lightweight along-track state-space modeling for lightweight memory and computational footprint.
3) We assess that DPSPM outperforms, in terms of accuracy, state-of-the-art methods designed to only exploit inputs realistically available in orbit and that it can process every line faster than its acquisition time on a low-power embedded device.

II. Related Works

A. UTM SPM

Traditional SPM methods follow the UTM approach, using the abundance maps derived from spectral unmixing as input to allocate land cover at the subpixel level. However, SPM is an ill-posed inverse problem: this requires incorporating spatial priors or constraints to regularize the solution space and reduce the number of feasible fine distribution images [10]. Spatial priors used in UTM methods fall into two major categories: external dataset-based priors and algorithm-based fixed priors [11], [18].

The first category leverages auxiliary information, such as sample points, external pairs of LR and HR images, or multitemporal images with different resolutions [11]. These external priors guide subpixel allocation by transferring spatial cues unavailable in the LR hyperspectral image, but they usually provide local information, which suffers from a lack of generalization and is only suitable for regional cases. A few examples of methods in this category are backpropagation-based SPM [8], semivariogram model [23], multipoint simulations [24], and support vector regression SPM [25].

For the second category, a commonly used prior information is the spatial dependence assumption, which encourages smoothness in the reconstructed subpixel distribution. Representative

methods include GA-SPM [7], which uses a genetic algorithm together with spatial dependence to search for subpixel configurations that match abundances under spatial smoothness constraints; [26], which exploits spatial dependence to position subpixels inside each parent pixel; and SASPM [27], which instead models attraction between subpixels and neighboring pixels (or subpixels), balancing the fidelity of abundances with spatial consistency.

B. DLSPM Networks

In recent years, deep learning has significantly advanced SPM by enabling end-to-end, data-driven approaches that model the spatial–spectral complexity of hyperspectral images more flexibly and accurately. An end-to-end DLSPM network typically consists of three main components [10], [11], [18]: feature extraction and transformation, upsampling, and classification. One of the earliest DL-based methods specifically designed for SPM is SRCNN-SPM [28]. This approach first requires abundance maps and then uses a super-resolution convolutional neural network (CNN) to map them to HR subpixel classifications. Among the end-to-end DLSPM approaches, SPMCNN-ESPCN [10] learns spatial–spectral priors directly from paired LR–HR patches, instead of relying on fixed handcrafted priors or abundance maps derived from spectral unmixing. SRM_{CNN} [17] formulates super-resolution mapping as an image super-resolution task and employs an encoder–decoder CNN to directly infer fine-scale land-cover probabilities from coarse imagery. CASNet [29] integrates global context attention modules to model long-range spatial dependencies and a hierarchical detail-preservation mechanism to retain fine structural information.

Beyond these approaches, several recent DL-based methods have been proposed to enhance SPM through explicit prior integration. SCNet [30] introduces a spatial deep-learning framework that incorporates a soft prior derived directly from the hyperspectral image itself, computed from spatial-transition cues similarly to abundance-based UTM methods. In contrast, other recent works exploit independent side information. SIM-Net [11], SFRNet-MLFF [18], and MSMCNet [19] propose to employ side information in the form of LR maps obtained from previous acquisitions to directly inject class-distribution information into the model. As it is not reasonable to possess such side information in the onboard setting, we do not include these methods in our experimental evaluation, as the comparison would not be fair.

C. Onboard Processing

Recent advances in satellite-embedded computation and hardware accelerators [31] now enable the execution of deep learning models in orbit, significantly reducing bandwidth requirements and supporting time-critical applications.

Early demonstrations of onboard deep learning for vision tasks established the feasibility of deploying compact CNNs and pruned models. For instance, a reduced YOLO variant was embedded on a CubeSat for ship detection [32], while ESA's Φ-Sat-1 showed that cloud filtering could be performed directly in orbit through an embedded CNN [33], [34]. Building on these results, the broader Φ-Sat programme has continued to advance in-orbit AI capabilities, and the follow-up Φ-Sat-2 mission supports multiple onboard applications, such as adaptive compression and feature extraction [35], [36]. Together, these developments highlight the growing relevance of onboard intelligence and the need for lightweight architectures that operate within strict resource constraints.

These efforts opened the door to systematic benchmarking of deep networks on flight-qualified CPUs and FPGAs [37], lightweight architectures obtained through distillation for tasks like change detection [38], long-term experimental validation on platforms, such as OPS-SAT [39], and energy-saving tasks or onboard multitask inference. More recent developments have realized onboard segmentation and detection from hyperspectral/optical data: for example, a 1DCNN [40] was deployed on the HYPSO-1 satellite to classify pixels into sea, land, and cloud categories, while an explainable embedded CNN [41] was demonstrated on FPGA for onboard ship detection from optical imagery, achieving detection accuracy up to 95%, while remaining deployable under satellite resource constraints.

Most relevant to our application scenario, novel line processing architectures focusing on pushbroom imaging, have been explored for onboard compression [16], and image super-resolution [20], demonstrating the potential of sequential, memory-efficient models tailored to the acquisition geometry. Real-time onboard processing for hyperspectral pushbroom sensors requires matching the line acquisition time in order to keep up with continuous acquisition. As an example, the PRISMA satellite [42] acquires a VNIR line of size $1 \times 1000 \times 66$ every 4.34 ms.

To the best of our knowledge, no existing work has demonstrated a deep learning model for SPM capable of meeting the real-time processing speed and memory constraints of an onboard satellite platform. The work in this article is the first to address SPM in this scenario, leveraging novel neural network designs [20] based on line-by-line processing.

III. Method

In this section, we describe the proposed DPSPM architecture for SPM. The model operates on small windows of three consecutive LR lines acquired by a pushbroom hyperspectral sensor and predicts a HR semantic segmentation map for the middle LR line in each window. Our design combines a multiscale line encoder (MSLE), an ATR step, which alternates with an along-track selective state space modeling block, and a final module composed of an efficient upsampling layer and a classification head specifically tailored to the super-resolved segmentation task. A graphical overview is given in Fig. 1 and a detailed description in Algorithm 1.

A. Problem Formulation and Line-Based Strategy

Let $\mathbf{X}^{\text{LR}} \in \mathbb{R}^{H\times W\times C}$ denote a LR hyperspectral cube, with H along-track lines, W across-track pixels, and C spectral channels. Let $\mathbf{Y}^{\text{HR}} \in \{1,\ldots,K\}^{rH\times rW}$ be the corresponding HR semantic segmentation mask, with K classes and an integer

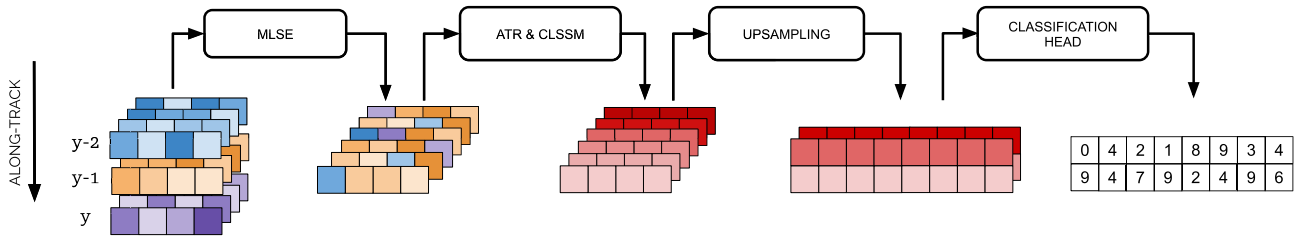

Fig. 1. High-level visual representation of the processing steps carried out by DPSPM: it shows how the three lines in the input window contribute to generating informative features, which are then refined and aligned with past context, upsampled and finally reduced to segmentation labels.

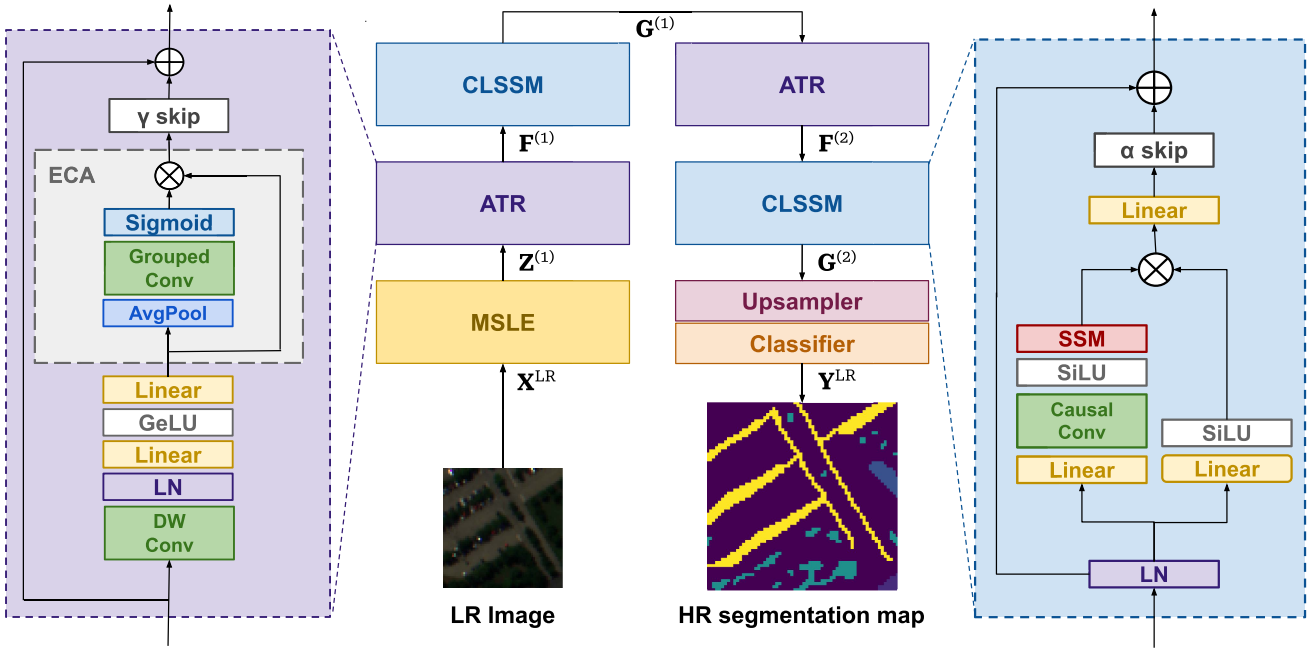

Fig. 2. DPSPM takes as input three LR lines at time and process them with an initial MSLE block, followed by an interleaved sequence of ATR and CLSSM blocks, which, respectively, refine the across-track global dependencies and the long-term along-track context. Then, a lightweight Upsampler paired with a classifier transform the information-dense feature maps into HR semantic segmentation labels.

spatial upsampling factor r. In a pushbroom acquisition system, the sensor captures one line at a time with all the spectral channels in the along-track direction. Rather than processing large 2-D tiles, the proposed model operates on overlapping triplets of consecutive LR lines, i.e., the input of DPSPM after acquiring line y is

$$\mathbf{X}^{\mathrm{LR}}_{y-2,y-1,y} \in \mathbb{R}^{3\times W\times C} \tag{1}$$

centered at along-track index $y-1$. For each triplet, the network f_θ predicts the SR semantic segmentation map corresponding to the middle LR line, namely

$$f_\theta : \mathbf{X}^{\mathrm{LR}}_{y-2,y-1,y} \longmapsto \mathbf{Y}^{\mathrm{HR}}_{y-1} \in \{1,\dots,K\}^{r\times rW} \tag{2}$$

where $\mathbf{Y}^{\mathrm{HR}}_{y-1}$ represents the r HR rows and rW HR columns providing a fine-grained semantic description of the middle LR line $y-1$. The spatial context is therefore restricted to a narrow window of three LR lines, which is sufficient to exploit local along-track correlations, while the longer range context is captured by a memory mechanism based on SSMs that propagates information along track. This strategy minimizes the along-track latency and substantially reduces the memory requirements with respect to the classic 2-D tiling approaches, making the architecture suitable for resource-constrained onboard deployment. We remark that one line look-ahead is required for the model to work appropriately since it is the middle line that gets semantically super-resolved.

B. Network Overview

The proposed network processes the LR triplet through four main stages, depicted in Fig. 2: a *MSLE*, which collapses the three input lines into a single feature line while aggregating multiscale across-track context; an *ATR* stage, based on ConvNeXt-style [43] 1-D blocks and channel attention; a *Cross-Line State Space Model (CLSSM)*, implemented by a Mamba

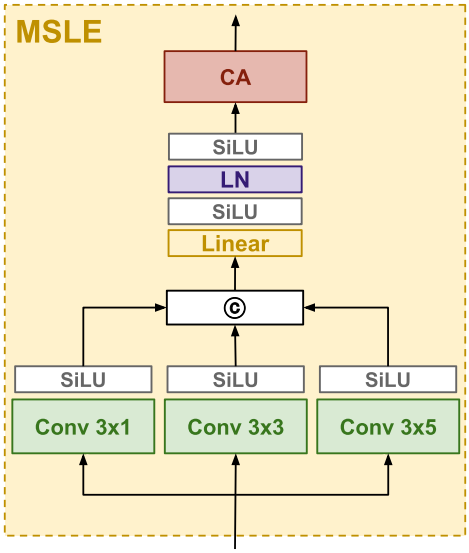

Fig. 3. MSLE compresses information coming from the three-line input window into a single line using an initial multiscale branch followed by a SiLU nonlinearity. After a sequence composed of a linear, a second SiLU and a LayerNorm (LN), a final channel attention module (CA) produces content-rich features.

block, which injects along-track context via a selective SSM and an *Upsampling and Semantic Classification Head*, which jointly upsamples along-track and across-track by a factor r and predicts class logits for the HR pixels. Internally, the spectral dimension is treated as a latent feature dimension rather than an explicit third spatial dimension.

C. MSLE

Given a triplet $\mathbf{X}^{\mathrm{LR}}_{y-2,y-1,y} \in \mathbb{R}^{3\times W\times C}$, a MSLE is first applied. As shown in Fig. 3, the MSLE consists of a set of parallel height-valid 2-D convolutions, each with kernel size $3\times k$, $k\in\{1,3,5\}$, which span exactly the three LR lines in the along-track direction and have different receptive fields in the across-track direction. Since we want the internal representation to collapse into one line that contains multiscale features, no padding is used along-track, while reflection padding is adopted across-track to avoid boundary artifacts.

Each convolutional branch, followed by a SiLU activation function, produces a feature map $\mathbf{H}^{(k)}_{y} \in \mathbb{R}^{1\times W\times F_k}$ with an arbitrary number of channels F_k. These maps are concatenated along the channel dimension and refined through a 1×1 convolution followed by another SiLU, yielding a feature line aligned with the middle LR line but with rich information of the previous and the subsequent lines

$$\mathbf{H}^{(k)}_{y-1} = \sigma\big(\mathrm{Conv}_{3\times k}(\mathbf{X}^{\mathrm{LR}}_{y-2,y-1,y})\big) \tag{3}$$

$$\mathbf{Z}^{(0)}_{y-1} = \sigma\Big(\mathrm{Conv}_{1\times 1}\Big([\mathbf{H}^{(1)}_{y-1}, \mathbf{H}^{(3)}_{y-1}, \mathbf{H}^{(5)}_{y-1}]\Big)\Big) \tag{4}$$

where $[\cdot]$ denotes concatenation along the channel dimension and σ is the SiLU nonlinearity. Subsequently, Layer Normalization is applied over the feature dimension, followed by another SiLU nonlinearity and a lightweight channel-attention module [44]. The latter performs global average and max pooling over the across-track dimension, passes the resulting features through a small multilayer perceptron and a sigmoid gating function, and finally reweights each feature channel. This operation yields a refined feature line $\mathbf{Z}^{(1)}_{y-1} \in \mathbb{R}^{1\times W\times F}$.

D. ATR Stage

The features $\mathbf{Z}^{(1)}_{y-1}$ are further processed to capture richer across-track structures and spectral–spatial interactions. To this end, two ATR blocks are employed, operating on tensors with shape $1\times W\times F$. Each ATR block starts with a depthwise 1-D convolution with a large kernel (e.g., $k=15$) and dilated filters, which act only in the across-track dimension. The output is then normalized by Layer Normalization over the feature dimension. Subsequently, two 1×1 convolutional layers, interleaved by a gaussian error linear unit (GELU) activation, expand the feature dimension by a factor s. An efficient channel attention (ECA) module refines these representations by exploiting channel-wise dependencies without introducing a heavy parameter overhead: it applies average pooling followed by a grouped convolution and a sigmoid function to generate a channel-wise gating vector. Finally, a learnable channel-wise tensor $\gamma\in\mathbb{R}^F$ modulates the residual branch.

Each ATR block produces a refined representation which primarily encodes local and global across-track dependencies, leaving the modeling of long-range along-track context to the subsequent SSMs. In particular, the architecture presents a first ATR block, followed by a CLSSM block, and then a second ATR Block again followed by a CLSSM block.

E. CLSSM

Reasoning one line at a time, it is natural to model the along-track dimension as a sequence of feature lines and design our CLSSM block based on operations that can efficiently process 1-D sequences. While the Transformer [22] is considered the most powerful sequence processor, we opted for Mamba [21], [45], a recent alternative based on SSMs. In particular, Mamba scales linearly with the sequence length and admits a recurrent implementation using a state encoding the memory of past lines in the sequence. By contrast, standard Transformer has self-attention quadratic computational complexity in sequence length and its inference relies on KV caching whose memory requirement can be significant as it grows linearly with the sequence length. These reasons make Mamba a more suitable choice for resource-constrained scenarios, such as the onboard processing one.

We denote by $\mathbf{F}_{y-1}\in\mathbb{R}^{1\times W\times F}$ the output of the previous ATR block. For notational purposes, we denote the feature vector corresponding to the ith across-track position as $\mathbf{f}_{y-1,i}\in\mathbb{R}^F$. The aim of the CLSSM block is to model the evolution of each sequence $\{\mathbf{f}_{t,i}\}_t$ in the along-track dimension, treating the line index t as the sequence axis. We first normalize the input feature vector using a Layer Normalization operator. Let

$$\mathbf{n}_{y-1,i} = \mathrm{LayerNorm}(\mathbf{f}_{y-1,i}) \in \mathbb{R}^F \tag{5}$$

and define the linearly projected representation as

$$\mathbf{g}_{y-1,i} = \text{Linear}(\mathbf{n}_{y-1,i}) \in \mathbb{R}^F. \tag{6}$$

To incorporate very recent context, a causal convolution of kernel size 3 is also applied in the along-track direction. Let $\mathbf{g}_{y-2,i}$ and $\mathbf{g}_{y-3,i}$ denote the two previously stored projected feature lines in the causal buffer. The convolutional activation is then

$$\mathbf{u}_{y-1,i} = \sigma(\text{CausalConv}_{k=3}(\mathbf{g}_{y-3,i}, \mathbf{g}_{y-2,i}, \mathbf{g}_{y-1,i})) \tag{7}$$

where σ denotes a SiLU nonlinearity. This vector $\mathbf{u}_{y-1,i}$ constitutes the token processed by the SSM. For each across–track index i, the CLSSM block maintains a latent recurrent state $\mathbf{h}_t^{(i)} \in \mathbb{R}^{F \times d_{\text{state}}}$. The state update and output equations are

$$\mathbf{h}_{y-1,i} = \mathbf{A}\,\mathbf{h}_{y-2,i} + \mathbf{B}\,\mathbf{u}_{y-1,i} \tag{8}$$

$$\mathbf{s}_{y-1,i} = \mathbf{C}\,\mathbf{h}_{y-1,i} + \mathbf{D}\,\mathbf{u}_{y-1,i} \tag{9}$$

where the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ are input-dependent, allowing the selective retention of past information. We refer the reader to [21] for more details. Equations (8)–(9) evolve the hidden state and produce the context-enriched representation $\mathbf{s}_{y-1,i} \in \mathbb{R}^F$. To combine the SSM output with the current features, the module applies a gated interaction of the form

$$\widetilde{\mathbf{s}}_{y-1,i} = \text{Linear}(\mathbf{s}_{y-1,i} \otimes \sigma(\text{Linear}(\mathbf{n}_{y-1,i}))) \tag{10}$$

where $\otimes$ denotes elementwise product. Reassembling the set $\{\widetilde{\mathbf{s}}_{y-1,i}\}_{i=1}^{W}$ into a tensor $\tilde{\mathbf{S}}_{\mathbf{y-1}}$ of shape $1 \times W \times F$ and introducing a learnable channel-wise gate $\alpha \in \mathbb{R}^F$ yields the final CLSSM block output

$$\mathbf{G}_{y-1} = \mathbf{F}_{y-1} + \alpha \otimes \left(\tilde{\mathbf{S}}_{\mathbf{y-1}}\right). \tag{11}$$

F. Upsampling and Semantic Classification Head

The context-enhanced features $\mathbf{G}_{y-1}^{(2)} \in \mathbb{R}^{1 \times W \times F}$, where $\mathbf{G}_{y-1}^{(2)}$ is the output of the last CLSSM block, are then super-resolved to the HR grid associated with the middle LR line. This is achieved by jointly upsampling the along-track and across-track dimensions by a factor r using a 1×1 convolution followed by a pixel shuffle-based upsampler [46]. First, the channel dimension is expanded to $F_{\text{up}} = F_0 r^2$, where F_0 is a reduced fixed arbitrary feature size through a 1×1 convolution

$$\mathbf{U}_{y-1} = \text{Conv}_{1\times1}\left(\mathbf{G}_{y-1}^{(2)}\right) \in \mathbb{R}^{1 \times W \times F_{\text{up}}}. \tag{12}$$

The resulting tensor is processed by a 2-D pixel shuffling operation with upscaling factor r. This operation rearranges the channel dimension into spatial dimensions, producing a feature map $\mathbf{U}_{y-1}^{\text{PS}}$ of size $r \times rW \times F_0$.

On top of this representation, a lightweight classifier predicts class logits for each HR pixel. The classifier consists of a 1×1 convolution that preserves the feature dimensionality and is followed by Layer Normalization and a nonlinear activation, and a final 3×1 convolution that outputs K channels

$$\mathbf{V}_{y-1} = \text{LayerNorm}\left(\text{Conv}_{1\times1}\left(\mathbf{U}_{y-1}^{\text{PS}}\right)\right) \tag{13}$$

$$\mathbf{O}_{y-1} = \text{softmax}\left(\text{Conv}_{3\times1}\left(\phi(\mathbf{V}_{y-1})\right)\right) \in \mathbb{R}^{r \times rW \times K} \tag{14}$$

where $\mathbf{U}_{y-1}^{\text{PS}}$ denotes the pixel shuffle output and $\phi(\cdot)$ is the ReLU activation function. A complete overview of the proposed method is also provided in Algorithm 1.

IV. Experimental Results

This section presents some experiments to assess the performance of DPSPM in terms of quality of the generated maps as well as computational efficiency on low-power hardware. Design choices are also validated through ablation experiments.

A. Experimental Setting

Datasets: Our experimental analysis is conducted on three widely used hyperspectral benchmarks: Pavia University [47], Houston2018 [48], and WHU-Hi-LongKou [49]. We also used the WHU-OHS dataset [50] for larger super-resolution factors. These datasets differ in spatial resolution, number of spectral bands, and scene characteristics, which allows us to test the robustness and generalization ability of the proposed method across diverse environments. The Pavia dataset consists in an image of size 610×340 with 103 spectral bands and presents 9 land cover classes, excluding background. The Houston2018 dataset consists in an image of size 1202×4768 with 50 spectral bands as provided by the rs_fusion_datasets Python library [48] and presents 21 land cover classes, including background. The WHU-Hi-LongKou dataset consists in an image of size 550×400 with 270 spectral bands and presents 9 land cover classes, excluding background. The WHU-OHS dataset consists of 7795 Orbita hyperspectral satellite (OHS) images of size 512×512 with 32 spectral bands and presents 24 land cover classes, excluding background.

We also note that, in highly imbalanced datasets such as WHU-Hi-LongKou, the severe underrepresentation of some classes (e.g., Narrow-leaf soybean, which accounts for only about 0.76% of the labeled pixels) may hinder their learning and favor dominant categories; this potentially results in very low or even zero Producer's Accuracy for certain rare classes.

Data preprocessing: Each dataset consists of a single large hyperspectral image and an HR classification map. Image patches for training and testing were extracted by random selection of a proportion ρ of the available patches. Both the number of spectral bands and the number of classes vary across the three datasets. To partition the data, a random proportion ρ of the available patches is used for training, while the remaining samples form the test set; the training sample proportion varies for different datasets (Pavia: $\rho = 0.65$, Houston2018: $\rho = 0.65$, WHU-Hi-LongKou: $\rho = 0.50$).

For DPSPM, patch extraction must satisfy an architectural constraint: the line-wise processing requires a minimum spatial extent due to the padding behavior of the depthwise convolution in the second ATR block (kernel size of 15, dilation factor of 2, and reflection padding), which in practice imposes a minimum patch width of at least $W = 14$ to avoid degenerate boundary effects. We stress that this constraint is not a limitation in the intended onboard scenario, where the across-track width is much larger (e.g., for PRISMA data $W = 1000$), and thus the model operates far from the small-patch regime used for benchmarking.

Algorithm 1: DPSPM: Line-Based Deep Subpixel Mapping for Pushbroom Sensors.

Inputs: 3 LR hyperspectral lines $\mathbf{X}^{\mathrm{LR}}_{y-2}, \mathbf{X}^{\mathrm{LR}}_{y-1}, \mathbf{X}^{\mathrm{LR}}_{y}$, scale factor r
Output: HR semantic segmentation maps with r lines and rW columns $\mathbf{Y}^{\mathrm{HR}}_{y-1}$
// Initialization for streaming inference
initialize CLSSM internal states: $\mathbf{h}^1_{y-2} \leftarrow \mathbf{0}$, $\mathbf{h}^2_{y-2} \leftarrow \mathbf{0}$
initialize 2-line causal conv buffer $\mathbf{g}^1_{y-2} \leftarrow \emptyset$, $\mathbf{g}^2_{y-2} \leftarrow \emptyset$.
// Warm-up: store the first two acquired LR lines
for each of the first two incoming lines **do**
    append line to buffer $\mathcal{B}$
end for
// 1. Form the 3-line input window
$\mathbf{X}^{\mathrm{LR}}_{y-2,y-1,y} \leftarrow (\mathbf{X}^{\mathrm{LR}}_{y-2}, \mathbf{X}^{\mathrm{LR}}_{y-1}, \mathbf{X}^{\mathrm{LR}}_{y})$ *(pushbroom-aligned triplet)*
// 2. Multi-Scale Line Encoder (MSLE)
compute multi-scale features:
    $\mathbf{H}^{(k)} \leftarrow \mathrm{Conv}_{(3,k)}(\mathbf{X}^{\mathrm{LR}}_{y-2,y-1,y})$ for $k \in \{1,3,5\}$
collapse to single feature line (middle line):
    $\mathbf{Z}^{(0)}_{y-1} \leftarrow \sigma(\mathrm{Conv}_{1\times1}([\mathbf{H}^{(1)}, \mathbf{H}^{(3)}, \mathbf{H}^{(5)}]))$
apply channel attention:
    $\mathbf{Z}^{(1)}_{y-1} \leftarrow \mathrm{CA}(\mathbf{Z}^{(0)}_{y-1})$
// 3. Across-Track and Along-Track Refinement
$\mathbf{F}^{(1)}_{y-1} \leftarrow \mathrm{ATR}_1(\mathbf{Z}^{(1)}_{y-1})$
update feature line using CLSSM:
    $\mathbf{G}^{(1)}_{y-1}, \mathbf{g}^1_{y-1}, \mathbf{h}^1_{y-1} \leftarrow \mathrm{CLSSM}_1(\mathbf{F}^{(1)}_{y-1}, \mathbf{g}^1_{y-2}, \mathbf{h}^1_{y-2})$
$\mathbf{F}^{(2)}_{y-1} \leftarrow \mathrm{ATR}_2(\mathbf{G}^{(1)}_{y-1})$
update feature line using second CLSSM:
    $\mathbf{G}^{(2)}_{y-1}, \mathbf{g}^2_{y-1}, \mathbf{h}^2_{y-1} \leftarrow \mathrm{CLSSM}_2(\mathbf{F}^{(2)}_{y-1}, \mathbf{g}^2_{y-2}, \mathbf{h}^2_{y-2})$
// 4. HR Upsampling (pixel shuffle)
expand channels: $\mathbf{U}_{y-1} \leftarrow \mathrm{Conv}_{1\times1}(\mathbf{G}^{(2)}_{y-1})$
rearrange to HR grid: $\mathbf{U_{y-1}}^{\mathrm{PS}} \leftarrow \mathrm{PixelShuffle}_r(\mathbf{U_{y-1}})$
// 5. Classification of the HR pixels
$\mathbf{V}_{y-1} \leftarrow \mathrm{LayerNorm}(\mathrm{Conv}_{1\times1}(\mathbf{U}^{\mathrm{PS}}_{y-1}))$
$\mathbf{O}_{y-1} \leftarrow \mathrm{Conv}_{3\times1}(\phi(\mathbf{V}_{y-1}))$
produce HR semantic map for line $y-1$:
    $\mathbf{Y}^{\mathrm{HR}}_{y-1} = \arg\max_c \mathbf{O}_{y-1}[c,\cdot,\cdot]$
return all predicted HR maps
// Streaming pushbroom acquisition loop
for each newly acquired line $\mathbf{X}^{\mathrm{LR}}_{y+1}$ **from the sensor do**
    // Slide the 3-line window to incorporate the new line
    $(\mathbf{X}^{\mathrm{LR}}_{y-2}, \mathbf{X}^{\mathrm{LR}}_{y-1}, \mathbf{X}^{\mathrm{LR}}_{y}) \leftarrow (\mathbf{X}^{\mathrm{LR}}_{y-1}, \mathbf{X}^{\mathrm{LR}}_{y}, \mathbf{X}^{\mathrm{LR}}_{y+1})$
    // Update CLSSM memory for next step
    $(\mathbf{h}^i_{y-2}, \mathbf{h}^i_{y-1}) \leftarrow (\mathbf{h}^i_{y-1}, \mathbf{h}^i_{y})$ $\forall i \in \{1,2\}$
    $(\mathbf{g}^i_{y-2}, \mathbf{g}^i_{y-1}) \leftarrow (\mathbf{g}^i_{y-1}, \mathbf{g}^i_{y})$ $\forall i \in \{1,2\}$
    // Run the forward pass to produce the HR segmentation of the middle line
    Compute $\mathbf{Y}^{\mathrm{HR}}_{y}$ as in lines 7–44.
end for

For the $\times2$ and $\times4$ experiments, we use HR patches of size 64×64, resulting in LR patches of size 32×32 and 16×16, respectively. For the $\times8$ setting, we increase the HR patch size to 128×128 so that the corresponding LR input is 16×16, which provides a stable and meaningful LR support for training and evaluation, while remaining compatible with the model padding requirements. However, because nonoverlapping patches are extracted, this larger patch size reduces the number of available patches and thus training samples; for this reason, we report $\times8$ results only on Houston2018, where the scene extent still provides a sufficient number of patches for a stable evaluation, while Pavia and WHU-Hi-LongKou training sets would become too small under the same protocol and would not give significant results for any method.

Comparison Methods: To thoroughly evaluate the performance of our approach, we benchmark it against deep learning SPM networks and classical UTM-based SPM techniques. For DLSPM we selected $\mathrm{SRM}_{\mathrm{CNN}}$ [17], SPMCNN-ESPCN [10], CASNet [29], and SCNet [30]. Moreover, the SRCNN-SPM [28] method was also selected, and the spatial attraction-based subpixel mapping (SASPM) method [27] was chosen as UTM-based approach. We carefully followed the implementation instructions provided in the original papers to ensure faithful reproduction of each method. For methods based on UTM-SPM, we follow standard workflows from prior literature [10], [11], [18]. Specifically, endmember spectra are extracted from LR images using only the training set. Abundance maps are then computed via spectral unmixing, precisely using the fully constrained least squares (FCLS) algorithm [51] implemented in the Python library *pysptools*, and provided to SPM algorithms as input. This protocol ensures a fair comparison between unmixing-based methods and end-to-end networks that directly operate on LR hyperspectral measurements. Concerning SCNet, its additional input is created using the same process used for the abundance maps of the UTM methods. We remark that SCNet, which incorporates a soft prior as additional input, was heavily influenced by the quality of the abundance maps produced, while CASNet, instead, exhibited a highly unstable loss when trained on limited data. Moreover, to ensure a fair comparison with all the other methods in the $\times2$ upscale scenario, we slightly modified the two final convolutional layers of $\mathrm{SRM}_{\mathrm{CNN}}$, reducing the fixed $\times4$ upscaling to a $\times2$ one, leaving the rest completely unchanged. We remark that we do not report results on SIMNet [11], SFRNet-MLFF [18], and MSMCNet [19] as they propose to use side information in the form of maps from previous acquisitions, which would not be feasible to have onboard.

Hyperparameter Setting: For DPSPM, we set a batch size of 5, a number of epochs of 3000 and the Adam optimizer with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$. The learning rate, fixed throughout the datasets, was set to 5×10^{-5}. The number of internal features F of our model is 88. The depthwise 1-D convolution at the start of each ATR block has a kernel size of 15; in the first ATR block the dilation parameter is equal to 1, in the second it is equal to 2. The feature dimension expand factor is $s = 4$. In CLSSM block the convolution kernel size is equal to 3 and the state dimension is equal to 16. Finally, the parameter F_0 of the upsampler block is equal to 32.

All DLSPM baseline networks are trained using the following standard configuration unless otherwise noted: the batch size, for each dataset, is 4; Adam optimizer is used with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$; the training epochs are 1000, and no pretrained weights are used. The learning rate used for the Pavia University and WHU-Hi-LongKou datasets was 10^{-4},

TABLE I
QUANTITATIVE ASSESSMENT OF EACH ALGORITHM ON PAVIA UNIVERSITY DATASET (UPSCALE FACTORS $\times 2$ AND $\times 4$)

Class	SASPM [27]	SRCNN-SPM [28]	SRM_{CNN} [17]	SPMCNN-ESPCN [10]	CASNet [29]	SCNet [30]	DPSPM
Upscale Factor $\times 2$							
Asphalt	18.23	24.62	94.08	96.54	81.62	62.62	**98.50**
Meadows	49.53	58.99	92.92	81.84	81.28	88.67	**99.65**
Gravel	27.29	34.60	71.87	51.48	55.60	**82.28**	78.94
Trees	89.08	**98.74**	91.30	97.68	81.55	80.68	98.47
Painted metal sheets	92.19	98.36	0.25	99.62	98.36	34.38	**99.86**
Bare soil	31.29	47.09	62.96	45.98	**41.64**	40.92	47.37
Bitumen	71.97	75.16	0	22.45	74.02	4.46	**89.55**
Self-blocking bricks	73.13	80.42	78.02	80.12	74.47	81.47	**86.65**
Shadows	**100**	**100**	96.92	92.92	88.92	48.31	98.96
OA (%)	52.41	61.57	77.69	75.65	73.38	70.96	**88.33**
κ	0.4197	0.5173	0.7016	0.6769	0.6456	0.6105	**0.8410**
Upscale Factor $\times 4$							
Asphalt	16.92	16.77	79.62	**84.31**	53.88	66.77	74.77
Meadows	48.67	62.92	83.90	84.33	65.51	62.02	**94.56**
Gravel	22.22	20.82	31.79	40.65	30.45	40.37	**87.60**
Trees	79.03	**91.98**	83.00	70.72	69.86	75.07	91.68
Painted metal sheets	87.28	**97.86**	92.44	93.83	81.34	7.05	96.11
Bare soil	32.17	50.59	47.58	43.59	42.25	22.84	**65.89**
Bitumen	64.33	**80.25**	57.48	41.88	3.50	9.08	36.43
Self-blocking bricks	66.73	81.42	57.34	61.49	51.54	77.12	**83.63**
Shadows	93.85	**99.69**	92.92	83.69	59.69	20.92	87.83
OA (%)	49.83	62.66	71.98	71.31	67.59	52.52	**84.24**
κ	0.3891	0.5273	0.6244	0.6176	0.5700	0.4003	**0.7898**

Bold indicates best result.

while it was 10^{-5} for the Houston2018 dataset. SCNet always used learning rate equal to 10^{-3}. For SRCNN-SPM, the model is pretrained on BSD300, following its published procedure and parameters.

All models were trained until convergence; we hypothesize that DPSPM requires more epochs to converge because its line-wise mechanism needs additional training iterations to stabilize and fully exploit long-range along-track dependencies while refining across-track features.

Evaluation Metrics: To assess performance quantitatively, we adopt two widely accepted classification and segmentation metrics: the Overall Accuracy (OA) and the Cohen's Kappa coefficient (κ). Moreover, for per-class evaluation, we report Producer's Accuracy (PA). These class-wise values are expressed as percentages and are listed in the result tables in the rows corresponding to each land-cover class.

Computing Environment: All experiments are implemented in PyTorch and executed with one Nvidia TITAN RTX GPU. The real-time tests were performed on the low-power platform Nvidia Jetson Orin Nano with 15 W of power.

B. Main Results

The first experiment we present is about the quality of the land cover maps obtained by DPSPM and how it compared with the selected state-of-the-art algorithms. Quantitative results for the $\times 4$ and $\times 2$ upscaling factors and qualitative results for the $\times 4$ upscaling factor are reported in Table I and Fig. 4 for the Pavia dataset, Table II and Fig. 5 for the Houston2018 dataset and Table III and Fig. 6 for the WHU-Hi-LongKou dataset. We also report quantitative results for the $\times 8$ upscaling factor on the Houston2018 dataset. We remark that evaluating at $\times 8$ requires using larger patches to obtain meaningful downsampled inputs and to satisfy structural requirements of DPSPM. Under this constraint, Pavia and WHU-Hi-LongKou datasets cannot be reliably tested at $\times 8$ because the corresponding increase in patch size would drastically reduce the number of available training samples, making the training set too small for a fair and stable evaluation.

Overall, it can be noticed that DPSPM outperforms the state-of-the-art alternatives in terms of overall accuracy and Kappa coefficient, sometimes by significant margins. At an upscale factor of $\times 4$, classical UTM-based methods, such as SASPM exhibit limited performance, reflecting the difficulty of resolving fine spatial structures from abundance maps alone. CNN-based DLSPM baselines, including SRCNN-SPM, SRM_{CNN}, and SPMCNN-ESPCN, provide a clear improvement, while CASNet and SCNet, despite incorporating contextual modeling, show performances that often do not exceed those of simpler CNN-based architectures. The limited training data, particularly on the Pavia dataset, may hinder the ability of these deeper 2-D architectures to generalize effectively, leading to reduced

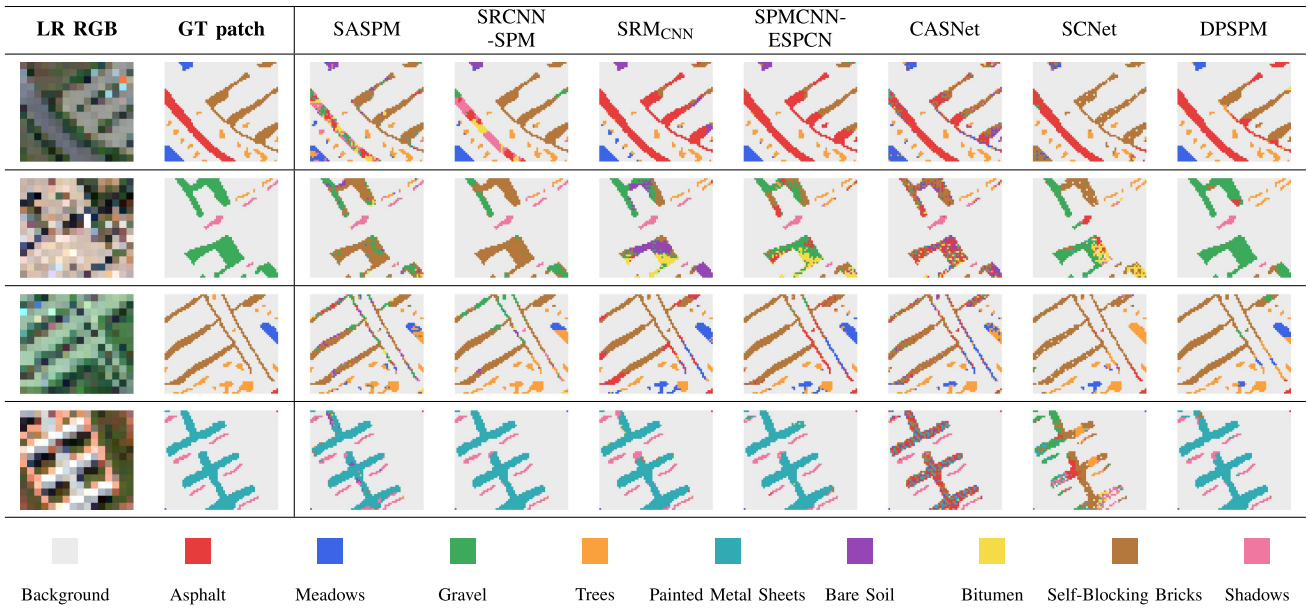

Fig. 4. Visual comparison of different SPM methods on Pavia University patches; upscale factor ×4. For the LR RGB patches, the bands 56 (∼ 650 nm), 31 (∼ 550 nm), and 9 (∼ 460 nm) were selected as R, G, and B, respectively.

TABLE II
QUANTITATIVE ASSESSMENT OF EACH ALGORITHM ON HOUSTON2018 DATASET (UPSCALE FACTORS ×2, ×4 AND ×8)

Metric	**SASPM** [27]	**SRCNN-SPM** [28]	**SRM$_{CNN}$** [17]	**SPMCNN-ESPCN** [10]	**CASNet** [29]	**SCNet** [30]	**DPSPM**
Upscale Factor ×2							
OA (%)	13.51	14.18	71.20	84.53	81.29	76.00	**87.64**
κ	0.1105	0.1217	0.6197	0.8002	0.7587	0.6911	**0.8413**
Upscale Factor ×4							
OA (%)	13.29	14.10	74.30	82.48	84.49	69.49	**86.31**
κ	0.1085	0.1212	0.6555	0.7758	0.8008	0.6032	**0.8250**
Upscale Factor ×8							
OA (%)	15.63	17.14	63.42	69.23	70.40	54.93	**76.67**
κ	0.1302	0.1493	0.5182	0.6042	0.6201	0.4090	**0.7005**

performance compared to the other DLSPM models. Trends observed for the ×4 upsampling factor are also confirmed for the ×2 and ×8 factors, with DPSPM delivering the best metrics. Moreover, the per-class Producer's Accuracy reported in Tables I and III indicates that DPSPM achieves consistently strong and competitive performance across most classes, while performance drops are only observed for a small subset of challenging categories in the WHU-Hi-LongKou dataset (Cotton and Sesame). The qualitative results agree with the quantitative analysis: methods such as SASPM, CASNet, and SCNet exhibit fragmented or oversmoothed boundaries, whereas DPSPM produces cleaner class delineations and sharper spatial structures, especially for narrow urban features. In particular, CASNet and SCNet output a great number of wrong predictions, even in smooth parts of the image. Generally, it is clear that DPSPM yields the best segmentation maps among the tested methods across all four presented images.

C. Real-Time Processing, Memory Scalability, and Feasibility

We used the Nvidia Jetson Orin Nano low-power hardware platform, which has a maximum power of 15 W, to simulate an onboard environment and evaluate the real-time capability of the proposed DPSPM architecture. Although not space-qualified, this platform can be considered representative of onboard edge processors, as its power consumption is suitable for an onboard embedded system. Achieving real-time performance is fundamental for practical onboard deployment, since many operational tasks, including early decision-making, require models that can keep pace with the sensor's acquisition rate.

Table IV reports the inference times of all the evaluated methods normalized as the time taken to process a $1 \times 1000 \times 66$ line. This dimension is chosen as reference since it replicates the dimensions of the VNIR instrument of the PRISMA satellite, which acquires a line in 4.34 ms. For most methods the time

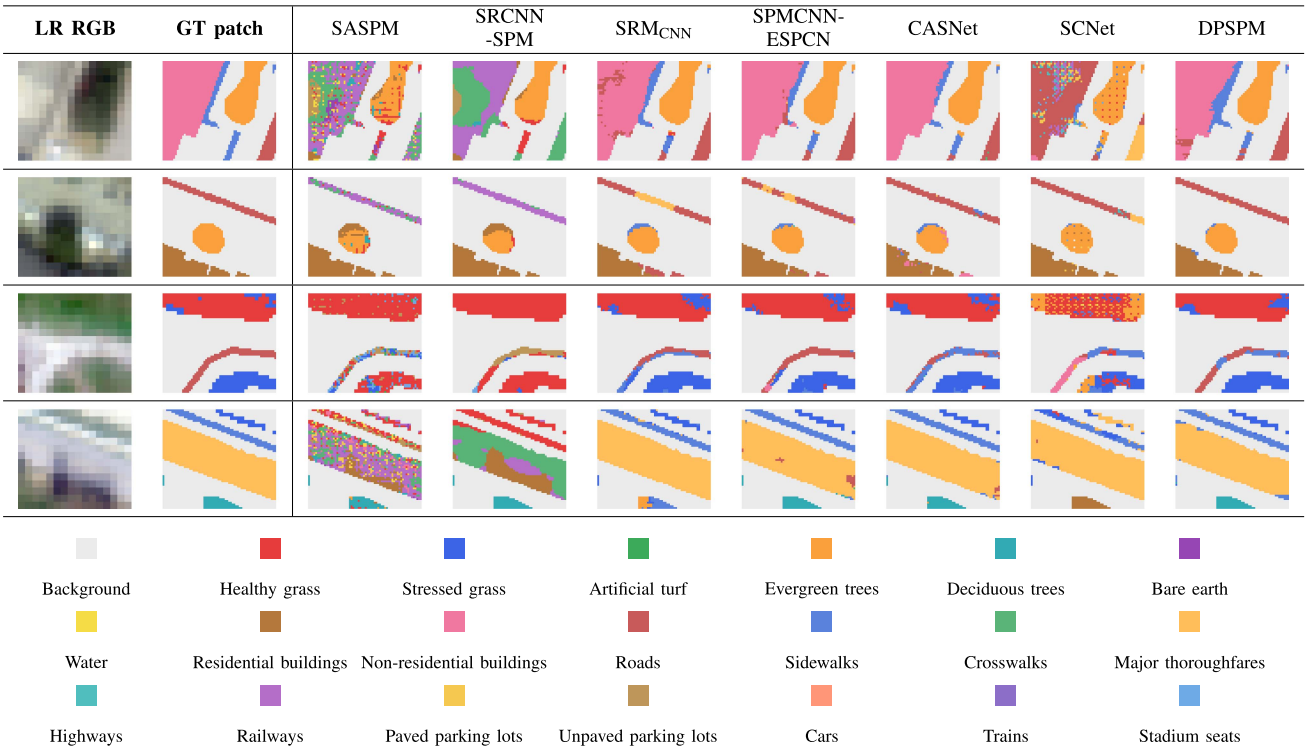

Fig. 5. Visual comparison of different SPM methods on Houston2018 University patches; upscale factor $\times 4$. The LR RGB patches were obtained by selecting the hyperspectral bands closest to 470 nm (B), 550 nm (G), and 650 nm (R).

TABLE III
QUANTITATIVE ASSESSMENT OF EACH ALGORITHM ON WHU-HI-LONGKOU DATASET (UPSCALE FACTORS $\times 2$ AND $\times 4$)

Metric	SASPM [27]	SRCNN-SPM [28]	SRM$_{CNN}$ [17]	SPMCNN-ESPCN [10]	CASNet [29]	SCNet [30]	DPSPM
Upscale Factor $\times 2$							
Corn	63.74	88.24	73.61	**98.21**	93.52	93.85	96.69
Cotton	50.25	62.93	66.02	65.92	2.91	**90.05**	0
Sesame	52.98	**90.53**	0	0	0	0	33.92
Broad-leaf soybean	69.13	76.91	92.97	95.13	96.89	86.44	**98.61**
Narrow-leaf soybean	24.55	28.65	0	41.82	30.39	0	**81.65**
Rice	64.10	85.57	30.84	93.85	42.51	45.71	**95.36**
Water	99.93	**100**	99.96	99.93	99.87	99.87	99.94
Roads and houses	66.63	84.89	**98.82**	95.22	94.23	92.90	95.17
Mixed weed	20.62	26.47	0	95.77	96.42	63.86	**99.26**
OA (%)	77.30	86.26	82.52	94.37	89.07	87.25	**95.03**
κ	0.7201	0.8163	0.7603	0.9239	0.8502	0.8272	**0.9322**
Upscale Factor $\times 4$							
Corn	61.79	90.99	70.46	94.13	88.78	74.92	**96.00**
Cotton	47.71	62.56	**95.96**	47.39	29.67	0	0
Sesame	51.11	**92.98**	0	0	0	0	22.34
Broad-leaf soybean	68.80	80.43	89.34	95.59	90.93	**96.20**	96.08
Narrow-leaf soybean	23.33	**25.04**	0	2.27	22.10	0	24.72
Rice	62.34	90.48	13.39	68.18	30.96	11.02	**91.57**
Water	99.72	**99.99**	98.83	99.74	99.67	98.78	99.85
Roads and houses	61.91	85.08	81.23	86.29	**92.21**	84.43	88.98
Mixed weed	14.26	20.16	0	94.52	82.54	65.54	**95.95**
OA (%)	76.25	87.61	79.75	90.22	86.09	81.31	**91.98**
κ	0.7087	0.8335	0.7209	0.8672	0.8096	0.7416	**0.8901**

Bold indicates best result.

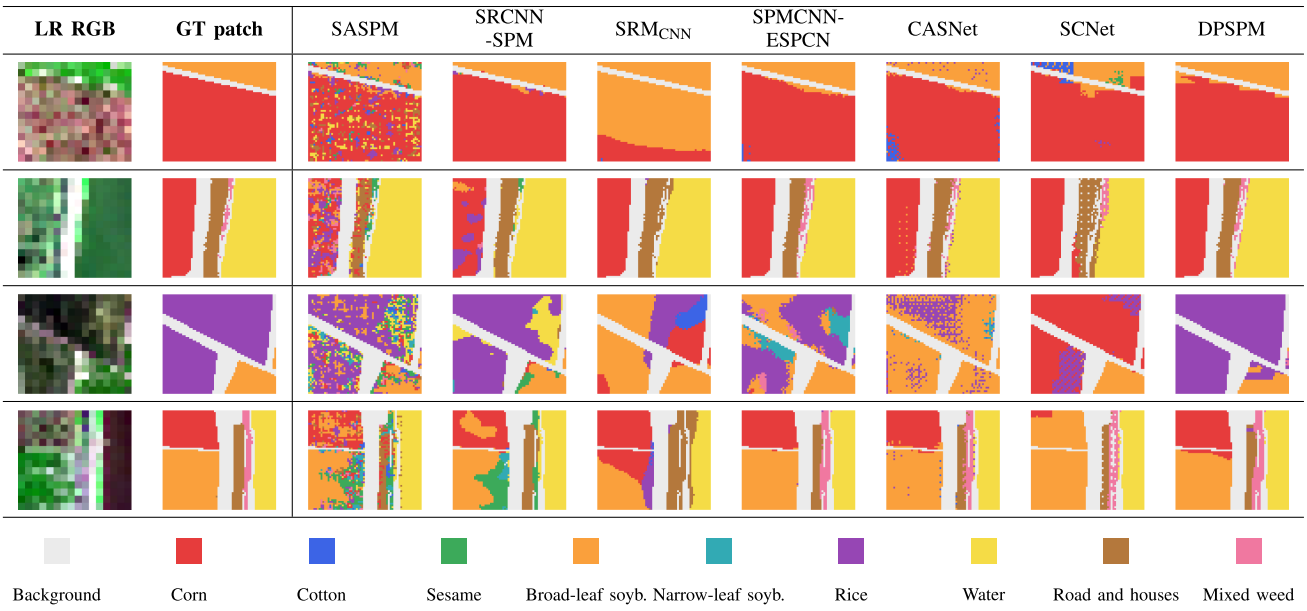

Fig. 6. Visual comparison of different SPM methods on WHU-Hi-LongKou University patches; upscale factor ×4. The LR RGB patches were obtained by selecting the hyperspectral bands closest to 475 nm (B), 561 nm (G), and 668 nm (R).

TABLE IV
INFERENCE SPEED

Metric	**SASPM** [27]	**SRCNN-SPM** [28]	**SRM$_{CNN}$** [17]	**SPMCNN-ESPCN** [10]	**CASNet** [29]	**SCNet** [30]	**DPSPM**
			Upscale Factor ×2				
Inference Time (ms)	4306.58	4174.92	0.79	0.4	2.42	4121.51	3.77
Real-Time Processing	×	×	✓	✓	✓	×	✓
			Upscale Factor ×4				
Inference Time (ms)	4354.31	4190.91	0.79	1.49	3.63	4111.57	4.17
Real-Time Processing	×	×	✓	✓	✓	×	✓

Runtime is normalized for an input of size 1 × 1000 × 66.

is computed on an input with 200 lines and then normalized to get the single-line time in order to avoid a possible speed overhead when processing a single input line. We can notice that DPSPM meets the real-time requirement by processing a line in 4.17 ms for the ×4 upsampling factor and 3.77 ms for the ×2 upsampling factor. We also notice that UTM-based methods SASPM and SRCNN-SPM are significantly slower due to complex end member extraction and FCLS optimization, falling short of this requirement by several orders of magnitude. SCNet also exhibits extremely high runtime due to FCLS preprocessing. On the other hand, CNN-based baselines SRM$_{CNN}$, SPMCNN-ESPCN, and to some extent CASNet are very fast, requiring, on average, a time to process a line shorter than its acquisition time.

However, all methods except DPSPM work on large 2-D tiles. This implies that a significant number of lines needs to be buffered for processing, increasing memory requirements. This can be seen in Figs. 7 and 8, which show the GPU memory requirements as a function of image size, for the ×4 and ×2 upsampling factors, respectively. The experiment confirms the design goal for DPSPM to have a low and constant memory requirement as function of the number of lines. We stress again that this behavior is a direct consequence of the line-wise design and the use of compact internal buffers: the model never holds a large 2-D spatial tile in memory and processes only three consecutive lines at a time, regardless of scene height. Moreover, the numbers of input columns and input bands have minimal impact on memory usage, with variations remaining within a narrow range, showing that the model keeps a small internal representation of the input. Moreover, we can see that DPSPM scales much slower than other methods as a function of the number of across-track columns or spectral bands, requiring orders of magnitudes less memory than other methods. This suggests that for realistic image sizes acquired by onboard sensors, methods other than DPSPM can quickly run out of the limited onboard memory. We note that SASPM was not included in the analysis because it cannot be accelerated by GPU, and it would not have been comparable with the other methods.

Overall, DPSPM is the only method capable to meet the real-time inference speed and, at the same time, be memory-efficient, while providing state-of-the-art classification accuracy. All the other methods fall short of at least one of these three properties. In particular, each of them has a memory footprint

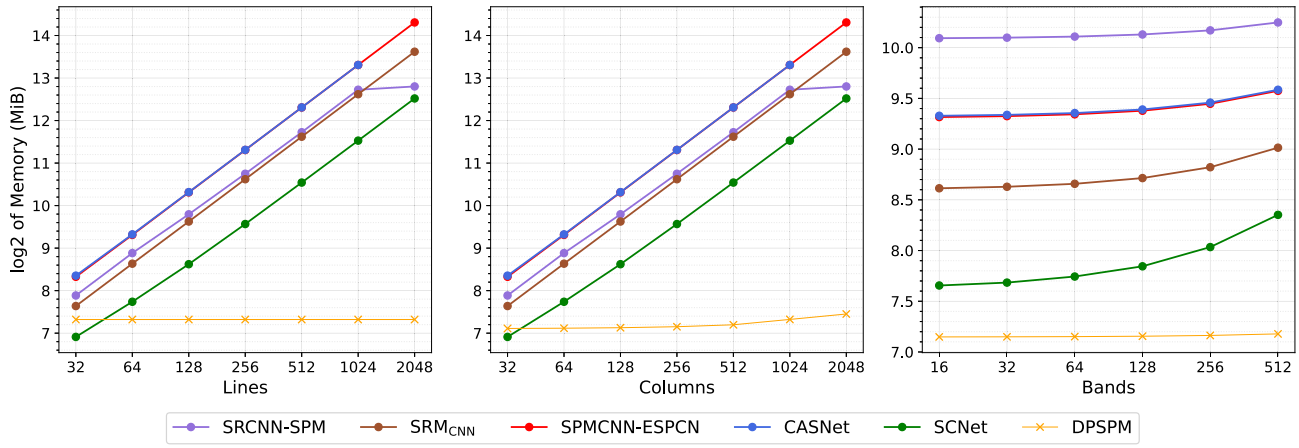

Fig. 7. Memory usage comparison for $\times 4$ SPM. Input image size is a) $256 \times 256 \times C$, b) $H \times 1000 \times 66$, and c) $1000 \times W \times 66$. CASNet runs out of memory when H and W exceed 1024 pixels.

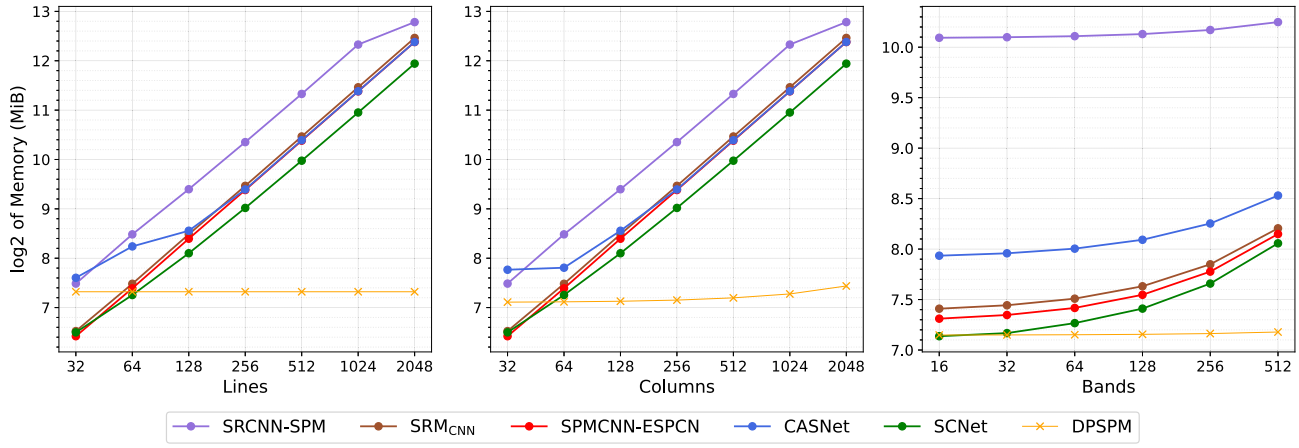

Fig. 8. Memory usage comparison for $\times 2$ SPM. Input image size is a) $256 \times 256 \times C$, b) $H \times 1000 \times 66$, and c) $1000 \times W \times 66$.

that is incompatible with constrained environments: their 2-D tile-based processing forces the memory to grow rapidly with the real input size and to exceed practical payload limits. We stress that this makes DPSPM the only method suitable for onboard deployment.

D. Robustness to Noise

Hyperspectral images acquired by remote sensing systems are inevitably degraded by noise due to a variety of factors such as thermal and shot noise or sensor nonuniformities. The resulting degradations are often band-dependent and spatially structured. Following the protocol adopted in the hyperspectral denoising literature [52], [53], we therefore train from scratch and evaluate the robustness of the proposed approach under synthetic, non-i.i.d. noise conditions that better approximate real HSI measurements. In order to generate noisy data, we considered two different noise settings: a non-i.i.d. Gaussian noise and a mixture noise, which reflects the fact that remote sensing HSIs are commonly contaminated by multiple noise sources simultaneously. In the non-i.i.d. Gaussian case, independent zero-mean Gaussian noise is added with a band-dependent standard deviation. For each spectral band, the noise level is sampled from a uniform interval with $\sigma \in [0, 95]$. As in [53], for the mixture noise, the data were constructed by combining: 1) non-i.i.d. band-dependent Gaussian noise with $\sigma \in [0, 95]$; 2) impulse noise applied to one third of the bands with corruption ratios uniformly sampled between 10% and 70%; 3) stripe noise affecting 5%–15% of columns on one third of the bands; and 4) deadline artifacts affecting 5%–15% of columns on one third of the bands.

As shown in Table V, across both upscaling factors, DPSPM exhibits only a limited sensitivity to non-i.i.d. Gaussian noise, with a limited drop from the clean setting, amounting to about 5.11% in OA and 0.0683 in κ at $\times 2$ and about 10.87 % in OA and 1.461 in κ at $\times 4$. In contrast, the mixture-noise setting is markedly more challenging and induces a larger degradation, since it combines Gaussian perturbations with structured artifacts, such as impulse, stripe, and deadline noise. Nevertheless, DPSPM remains the top-performing method in both scenarios,

TABLE V
QUANTITATIVE ASSESSMENT OF EACH ALGORITHM ON PAVIA UNIVERSITY DATASET UNDER GAUSSIAN AND MIXTURE NOISE WITH $\sigma \in [0,95]$ (UPSCALE FACTORS $\times 2$ AND $\times 4$)

Metric	**SASPM** [27]	**SRCNN-SPM** [28]	**SRM$_{CNN}$** [17]	**SPMCNN-ESPCN** [10]	**CASNet** [29]	**SCNet** [30]	**DPSPM**
Upscale Factor $\times 2$							
Gaussian noise							
OA (%)	46.25	60.17	74.35	73.59	75.78	63.44	**83.22**
κ	0.3527	0.4970	0.6528	0.6523	0.6658	0.5162	**0.7727**
Mixture noise							
OA (%)	48.54	48.80	66.53	65.88	65.15	63.71	**70.06**
κ	0.3736	0.3876	0.5475	0.5406	0.5326	0.5253	**0.5979**
Upscale Factor $\times 4$							
Gaussian noise							
OA (%)	43.46	60.30	66.55	66.88	65.73	52.20	**73.37**
κ	0.3234	0.4978	0.5647	0.5644	0.5369	0.3917	**0.6437**
Mixture noise							
OA (%)	46.14	50.09	66.12	58.91	63.99	43.46	**67.24**
κ	0.3481	0.4016	0.5543	0.4601	0.5130	0.3037	**0.5625**

TABLE VI
QUANTITATIVE ASSESSMENT OF EACH ALGORITHM USING LARGE UPSCALE FACTORS $\times 8$ AND $\times 16$ ON WHU-OHS DATASET

Metric	**SASPM** [27]	**SRCNN-SPM** [28]	**SRM$_{CNN}$** [17]	**SPMCNN-ESPCN** [10]	**CASNet** [29]	**SCNet** [30]	**DPSPM**
Upscale Factor $\times 8$							
OA (%)	10.04	15.35	69.10	65.96	73.69	65.20	**77.34**
κ	0.0595	0.0863	0.6490	0.6123	0.7026	0.6029	**0.7442**
Upscale Factor $\times 16$							
OA (%)	10.24	15.59	63.84	65.47	70.42	64.55	**76.35**
κ	0.0612	0.0900	0.5874	0.6065	0.6643	0.5939	**0.7325**

supporting its robustness and practical validity in realistic noisy acquisition conditions.

E. Robustness to Larger Scale Factors

Since higher upsampling ratios make the SPM problem significantly more challenging and require the model to recover finer semantic details from increasingly limited LR information, it is also important to assess whether the proposed method generalizes well to larger scale factors. To this end, we introduce an additional evaluation on the WHU-OHS dataset, which allows us to consider more demanding scale factors, namely $\times 8$ and $\times 16$. As reported in Table VI, DPSPM continues to achieve the best performance among all compared methods at both scale factors, while classic nonneural methods are significantly challenged by such large scaling factors. These results indicate that the proposed approach preserves its effectiveness even in more challenging super-resolution conditions, therefore confirming its good generalization capability to larger scale factors.

F. Ablation Experiments

We conducted a series of ablation experiments to assess the influence of the main architectural components of DPSPM. Unless otherwise stated, all results are reported on the Pavia University dataset at upscaling factors of $\times 2$ and $\times 4$, and every memory usage throughout all the tables is measured on inputs that replicate the acquisition geometry of the PRISMA VNIR instrument, i.e., inputs of size $1 \times 1000 \times 66$, unless otherwise specified.

First, we study the effect of the number of input lines used by the model, i.e., the window size of the input, on the model effectiveness. Every model collapses the different number of input lines into one with the MSLE block, producing a super-resolved segmentation map corresponding to one input line. Indeed, what changes among the different models is the context to which each MSLE block has access. As shown in Table VII, the proposed 3-L configuration achieves the best performance at both scales, while requiring only a small memory overhead with respect to the most efficient 1-L one. Reducing the context to a single line results in a performance degradation of about 6.5% at $\times 2$ and 2.5% at $\times 4$, indicating that incorporating along-track information is beneficial. Conversely, with two lines, the model yields worse performance with respect to the 1-L one and we conjecture it is due to the possible over-focusing on the future context y of MSLE: having access only to the future and not to the past y - 2, the MSLE block produces features that cannot recover a smooth joint compression of past and future in a single line feature. Expanding the window beyond three lines leads again to

TABLE VII
COMPARISON WITH THE N-LINE MODELS ON PAVIA UNIVERSITY DATASET (UPSCALE FACTORS X2 AND X4); THE MEMORY IS COMPUTED ON INPUTS OF SIZE N-LINES X 1000 X 66 FOR RESPECTIVE MODELS

Model	OA (%)	κ	Memory
Upscale $\times 2$			
1-L	81.72	0.7514	**153.94 MiB**
2-L	81.42	0.7504	154.26 MiB
3-L	**88.33**	**0.8410**	159.85 MiB
5-L	77.59	0.6931	159.91 MiB
Upscale $\times 4$			
1-L	81.68	0.7521	**158.63 MiB**
2-L	76.13	0.6839	158.95 MiB
3-L	**84.24**	**0.7898**	159.85 MiB
5-L	70.95	0.6092	159.91 MiB

Bold indicates best result.

TABLE VIII
IMPACT OF THE ORDER OF ATR (A) AND CLSSM (B) BLOCKS ON THE PERFORMANCE MEASURED ON THE PAVIA UNIVERSITY AND WHU-HI-LONGKOU DATASETS (UPSCALE FACTORS $\times 2$ AND $\times 4$)

Model Config.	Pavia		WHU-Hi-LongKou	
	OA (%)	κ	OA (%)	κ
Upscale $\times 2$				
AA-BB	85.23	0.7970	93.51	0,9178
BB-AA	86.23	0.8127	93.81	0,9159
BA-BA	86.13	0.8115	94.76	0,9303
AB-AB	**88.33**	**0.8410**	**95.03**	**0.9322**
Upscale $\times 4$				
AA-BB	81.60	0.7567	90.77	0,8888
BB-AA	83.97	0.7878	91.30	0,8511
BA-BA	**85.39**	**0.8071**	91.93	0.8824
AB-AB	84.24	0.7898	**91.98**	**0.8901**

Bold indicates best result.

a substantial drop in accuracy: the 5-L model falls to 77.59% and 70.95% OA at $\times 2$ and $\times 4$, respectively. We hypothesize that this is due to the excessive information that the MLSE block has to compress in just one feature line, leading to representations that include too much of useless past information without being able to effectively and selectively filter the features. Memory usage remains almost constant across configurations; however, using larger windows would lead to an increased buffering and thus to a higher latency in streaming processing. These results validate the choice of a compact 3-line buffer for pushbroom-aligned processing.

We proceed to analyze the sequence in which ATR and CLSSM modules are applied, which has a measurable impact. As shown in Table VIII, the AB–AB configuration (A is ATR, B is CLSSM) achieves the highest accuracy for $\times 2$ upscaling on both datastes and for $\times 4$ upscaling on the Pavia dataset. Apart from the case of $\times 4$ upscaling on the Pavia dataset where the BA–BA configuration attains the best result, all the other permutations perform consistently below the AB–AB one. These findings indicate that ATR and CLSSM play complementary roles: ATR primarily strengthens spatial refinement by emphasizing informative structures and edges, while CLSSM promotes line-wise sequential modeling by propagating context along the pushbroom dimension. The ordering therefore determines when spatial cues are injected into the representation versus when long-range line context is consolidated. The consistently weaker performance of grouped layouts (AA–BB, BB–AA) supports the hypothesis that early over-specialization is detrimental. In contrast, interleaving (AB–AB or BA–BA) enables an iterative exchange where spatial refinement is repeatedly conditioned by line-wise context (and vice versa), yielding a better trade-off between local detail preservation and global consistency. Overall, AB–AB provides the strongest results so, considering the general performance and that $\times 2$ upscaling is also the most common and practically relevant setting, we opted for the AB–AB configuration.

TABLE IX
IMPACT OF THE PRESENCE OF α_i GATES IN CLSSM BLOCKS ON THE PERFORMANCE MEASURED ON THE PAVIA UNIVERSITY DATASET (UPSCALE FACTORS $\times 2$ AND $\times 4$)

Model	OA (%)	κ	Memory
Upscale $\times 2$			
w/o α_i	85.30	0.7962	159.85 MiB
α_i scal. init$=0$	87.26	0.8270	159.85 MiB
α_i scal. init$=1$	87.48	0.8289	159.85 MiB
α_i vec. init$=0$	85.01	0.8012	159.85 MiB
$\boldsymbol{\alpha_i}$ **vec. init$=1$**	**88.33**	**0.8410**	159.85 MiB
Upscale $\times 4$			
w/o α_i	82.16	0.7647	159.85 MiB
α_i scal. init$=0$	72.96	0.6569	159.85 MiB
α_i scal. init$=1$	78.19	0.7232	159.85 MiB
α_i vec. init$=0$	80.58	0.7401	159.85 MiB
$\boldsymbol{\alpha_i}$ **vec. init$=1$**	**84.24**	**0.7898**	159.85 MiB

Bold indicates best result.

Moreover, Table IX evaluates the gating mechanism α, a gating tensor operating on the feature dimension of activations in the CLSSM blocks. Removing the gate reduces performance to 85.30% for for $\times 2$ upscaling and to 82.16% OA for $\times 4$ upscaling. Scalar gates with initialization to 0 or 1 either under perform or lead to unstable optimization at $\times 4$, with OA ranging from 72.96% to 78.19%. The best results are obtained using vector-valued gates initialized to 1, achieving 88.33 % OA and 0.8410 κ at $\times 2$ and 84.24% OA and 0.7898 κ at $\times 4$. This confirms that fine-grained channel-wise modulation is essential for extracting discriminative temporal features.

Table X further examines the impact of the kernel size used in the causal convolution of CLSSM blocks. A kernel size of 3 yields the strongest performance, outperforming both smaller and larger alternatives. As expected, reducing the kernel to 2 decreases accuracy, suggesting insufficient receptive-field coverage. Increasing to a kernel of 4 slightly improves over the 2-element version, but still remains below the optimal configuration (81.46% OA, 0.7518 κ). We hypothesized that a large kernel would compress information coming from too far into the past into a single feature line, limiting performance. Memory usage remains essentially unchanged across settings, confirming that the observed differences are attributable to the modeling capacity rather than efficiency constraints.

Last, we evaluate the influence of the expansion factor d_{state}, which controls the dimensionality of the latent state in the

TABLE X
ABLATION OF THE KERNEL OF THE CAUSAL CONVOLUTION INSIDE OF THE MAMBA BLOCKS ON PAVIA UNIVERSITY DATASET (UPSCALE FACTORS $\times 2$ AND $\times 4$); THE MEMORY IS COMPUTED ON INPUTS OF SIZE 3 X 1000 X 66, REFLECTING THE ACQUISITION DIMENSIONS OF PRISMA VNIR INSTRUMENT

Kernel dim.	OA (%)	κ	Memory
Upscale $\times 2$			
2	75.94	0,667	**158.60 MiB**
3	**88.33**	**0.8410**	159.85 MiB
4	87.52	0,8314	159.94 MiB
Upscale $\times 4$			
2	79.84	0.7274	**158.60 MiB**
3	**84.24**	**0.7898**	159.85 MiB
4	81.46	75.18	159.94 MiB

Bold indicates best result.

TABLE XI
ABLATION OF THE EXPANSION FACTOR d_{STATE} OF LATENT STATE $\mathbf{h}$ ON PAVIA UNIVERSITY DATASET (UPSCALE FACTORS $\times 2$ AND $\times 4$); THE MEMORY IS COMPUTED ON INPUTS OF SIZE 3 X 1000 X 66, REFLECTING THE ACQUISITION DIMENSIONS OF PRISMA VNIR INSTRUMENT

d_{state}	OA (%)	κ	Memory
Upscale $\times 2$			
1	75.96	0,6838	**149.17 MiB**
4	85.90	0.8094	151.19 MiB
8	88.29	0.8401	154.47 MiB
16	**88.33**	**0.8410**	159.85 MiB
32	83.51	0,7753	170.17 MiB
Upscale $\times 4$			
1	73.73	0.6386	**149.17 MiB**
4	82.45	0.7696	151.19 MiB
8	82.72	0.7683	154.47 MiB
16	**84.24**	**0.7898**	159.85 MiB
32	79.69	0.7226	170.17 MiB

Bold indicates best result.

CLSSM blocks. As reported in Table XI, performance increases consistently from $d_{\text{state}} = 1$ up to $d_{\text{state}} = 16$, which achieves the best results. After that, $d_{\text{state}} = 32$ leads to a degradation of performance, while memory consumption rises sharply.

Across all ablations, the results consistently support the architectural choices adopted in DPSPM: interleaved ATR–CLSSM blocks, a channel-wise α_i gating, a causal convolution of size 3 and a latent state of size ($d_{\text{state}} = 16$) collectively provide the best trade-off between accuracy, stability, and memory efficiency under onboard constraints.

V. CONCLUSION

This work introduced a deep SPM framework explicitly designed for real-time onboard processing with pushbroom hyperspectral sensors. Departing from conventional 2-D DLSPM pipelines—which require large spatial tiles, and present high computational and memory costs, we proposed a line-based architecture that processes only three consecutive LR lines and, together with a memory mechanism of previous lines based on SSMs, produces a super-resolved semantic segmentation map of the middle one. This design aligns naturally with the sequential acquisition geometry of pushbroom instruments while enabling streaming inference with minimal buffering and a constant memory footprint with respect to the scene height.

Extensive experiments on three benchmark datasets demonstrated that the proposed DPSPM consistently achieves state-of-the-art performance among methods that operate using only realistic onboard inputs. Moreover, tests conducted on the low-power Nvidia Jetson Orin Nano platform confirm that DPSPM satisfies the stringent real-time constraints of onboard processing and requires orders of magnitude less memory than existing methods.

We remark that the DPSPM line-based processing strategy naturally fits pushbroom sensors but could also be directly applied to whiskbroom instruments by line buffering. However, frame-based matrix sensors that acquire full 2-D images are less suitable for the proposed design. Nevertheless, satellite-based hyperspectral imaging is largely dominated by pushbroom sensors.

Finally, DPSPM is designed for a fixed upscaling factor r, selected a priori and kept constant during training and inference, in order to keep the design as simple and efficient as possible. This could be considered a limitation of the proposed method if multiple or variable upscaling factors are desired. Future work will explore extensions to multiscale mapping.

REFERENCES

[1] P. Fisher, "The pixel: A snare and a delusion," *Int. J. Remote Sens.*, vol. 18, no. 3, pp. 679–685, 1997, doi: 10.1080/014311697219015.

[2] N. Keshava and J. Mustard, "Spectral unmixing," *IEEE Signal Process. Mag.*, vol. 19, no. 1, pp. 44–57, Jan. 2002.

[3] C. Quintano, A. Fernández-Manso, Y. E. Shimabukuro, and G. Pereira, "Spectral unmixing," *Int. J. Remote Sens.*, vol. 33, no. 17, pp. 5307–5340, 2012, doi: 10.1080/01431161.2012.661095.

[4] P. Atkinson, M. Cutler, and H. Lewis, "Mapping sub-pixel proportional land cover with AVHRR imagery," *Int. J. Remote Sens.*, vol. 18, no. 4, pp. 917–935, 1997.

[5] P. M. Atkinson, "Mapping sub-pixel boundaries from remotely sensed images," in *Innovations in GIS*. Boca Raton, FL, USA: CRC Press, 1997, pp. 184–202.

[6] G. M. Foody, "Estimation of sub-pixel land cover composition in the presence of untrained classes," *Comput. Geosci.*, vol. 26, no. 4, pp. 469–478, 2000.

[7] K. C. Mertens, L. P. C. Verbeke, E. I. Ducheyne, and R. R. D. Wulf, "Using genetic algorithms in sub-pixel mapping," *Int. J. Remote Sens.*, vol. 24, no. 21, pp. 4241–4247, 2003, doi: 10.1080/0143116031000159507 3.

[8] L. Zhang, K. Wu, Y. Zhong, and P. Li, "A new sub-pixel mapping algorithm based on a BP neural network with an observation model," *Neurocomputing*, vol. 71, no. 10, pp. 2046–2054, 2008, Neurocomputing for Vision Research Advances in Blind Signal Processing. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231208001392

[9] P. V. Arun, K. M. Buddhiraju, and A. Porwal, "CNN based sub-pixel mapping for hyperspectral images," *Neurocomputing*, vol. 311, pp. 51–64, 2018.

[10] D. He, Y. Zhong, X. Wang, and L. Zhang, "Deep convolutional neural network framework for subpixel mapping," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 11, pp. 9518–9539, Nov. 2021. .

[11] D. He, Q. Shi, X. Liu, Y. Zhong, and X. Zhang, "Deep subpixel mapping based on semantic information modulated network for urban land use mapping," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 12, pp. 10628–10646, Dec. 2021.

[12] D. He, Q. Shi, X. Liu, Y. Zhong, and X. Liu, "Spectral–spatial fusion sub-pixel mapping based on deep neural network," *IEEE Geosci. Remote Sens. Lett.*, vol. 19, 2022, Art. no. 6004105.

[13] Y. Shi and Z. Wang, "Onboard generation of optimal trajectories for hypersonic vehicles using deep learning," *J. Spacecraft Rockets*, vol. 58, no. 2, pp. 400–414, 2021.

[14] C. Serief, Y. Ghelamallah, and Y. Bentoutou, "Deep-learning-based system for change detection onboard earth observation small satellites," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 16, pp. 8115–8124, 2023.
[15] D. Piccinini, D. Valsesia, and E. Magli, "Towards deep line-based architectures for onboard hyperspectral image super-resolution," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2025, pp. 6241–6245.
[16] D. Valsesia, T. Bianchi, and E. Magli, "Onboard deep lossless and near-lossless predictive coding of hyperspectral images with line-based attention," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 5532714.
[17] Y. Jia, Y. Ge, Y. Chen, S. Li, G. B. Heuvelink, and F. Ling, "Super-resolution land cover mapping based on the convolutional neural network," *Remote Sens.*, vol. 11, no. 15, 2019, Art. no. 1815, [Online]. Available: https://www.mdpi.com/2072-4292/11/15/1815
[18] J. Zhong, K. Wu, and Y. Xu, "Deep spatial feedback refined network with multilevel feature fusion for hyperspectral image subpixel mapping," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 4409620.
[19] M. Xu, X. Zou, S. Liu, H. Sheng, and Y. Dong, "Multiscale semantically modulated mixed convolutional networks for subpixel mapping," *IEEE Trans. Geosci. Remote Sens.*, vol. 63, 2025, Art. no. 5521816.
[20] D. Piccinini, D. Valsesia, and E. Magli, "Onboard hyperspectral super-resolution with deep pushbroom neural network," *Remote Sens.*, vol. 17, no. 21, 2025, Art. no. 3634.
[21] A. Gu and T. Dao, "MAMBA: Linear-time sequence modeling with selective state spaces," 2023, *arXiv:2312.00752*..
[22] A. Vaswani et al., "Attention is all you need," in *31st Int. Conf. on Neural Info. Process. Syst. (NeurIPS 2017)*, vol. 30. I. Guyon, Eds., Long Beach, CA, USA: Curran Associates, Inc., Dec. 2017, pp. 5998–6008, [Online]. Available: https://arxiv.org/abs/1706.03762
[23] P. M. Atkinson, E. Pardo-Iguzquiza, and M. Chica-Olmo, "Downscaling cokriging for super-resolution mapping of continua in remotely sensed images," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 2, pp. 573–580, Feb. 2008.
[24] G. Y. Ge Yong, "Sub-pixel land-cover mapping with improved fraction images upon multiple-point simulation," *Int. J. Appl. Earth Observ. Geoinformation*, vol. 22, pp. 115–126, 2013.
[25] Y. Zhang, Y. Du, F. Ling, and X. Li, "Improvement of the example-regression-based super-resolution land cover mapping algorithm," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 8, pp. 1740–1744, Aug. 2015.
[26] K. Mertens, B. De Baets, L. Verbeke, and R. De Wulf, "Direct sub-pixel mapping exploiting spatial dependence," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2004, vol. 5, pp. 3046–3049.
[27] K. C. Mertens, B. de Baets, L. P. C. Verbeke, and R. R. de Wulf, "A sub-pixel mapping algorithm based on sub-pixel/pixel spatial attraction models," *Int. J. Remote Sens.*, vol. 27, no. 15, pp. 3293–3310, 2006, doi: 10.1080/01431160500497127.
[28] X. Ma, Y. Hong, Y. Song, and Y. Chen, "A super-resolution convolutional-neural-network-based approach for subpixel mapping of hyperspectral images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 12, pp. 4930–4939, Dec. 2019.
[29] D. He, Q. Shi, X. Liu, Y. Zhong, and L. Zhang, "Generating 2m fine-scale urban tree cover product over 34 metropolises in China based on deep context-aware sub-pixel mapping network," *Int. J. Appl. Earth Observ. Geoinformation*, vol. 106, 2022, Art. no. 102667, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0303243421003743
[30] X. Zhang et al., "High-quality super-resolution mapping using spatial deep learning," *Iscience*, vol. 26, no. 6, 2023, Art. no. 106875.
[31] L. Diana and P. Dini, "Review on hardware devices and software techniques enabling neural network inference onboard satellites," *Remote Sens.*, vol. 16, no. 21, 2024, Art. no. 3957, [Online]. Available: https://www.mdpi.com/2072-4292/16/21/3957
[32] Y. Yao, Z. Jiang, H. Zhang, and Y. Zhou, "On-board ship detection in micro-nano satellite based on deep learning and COTS component," *Remote Sens.*, vol. 11, no. 7, 2019, Art. no. 762, [Online]. Available: https://www.mdpi.com/2072-4292/11/7/762
[33] G. Giuffrida et al., "CloudScout: A deep neural network for on-board cloud detection on hyperspectral images," *Remote Sens.*, vol. 12, no. 14, 2020, Art. no. 2205, [Online]. Available: https://www.mdpi.com/2072-4292/12/14/2205
[34] G. Giuffrida et al., "The ϕ-sat-1 mission: The first on-board deep neural network demonstrator for satellite earth observation," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5517414.
[35] G. Guerrisi, F. Del Frate, and G. Schiavon, "Artificial intelligence based on-board image compression for the ϕ-sat-2 mission," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 16, pp. 8063–8075, 2023.
[36] N. Melega et al., "Implementation of the ϕsat-2 on board image processing chain," in *Sensors, Systems, and Next-Generation Satellites XXVII*, vol. 12729. S. R. Babu, A. Hélière, and T. Kimura, Eds., Bellingham, WA, USA: SPIE, 2023, Art. no. 127290Z, doi: 10.1117/12.2679044.
[37] M. Ziaja et al., "Benchmarking deep learning for on-board space applications," *Remote Sens.*, vol. 13, no. 19, 2021, Art. no. 3981, [Online]. Available: https://www.mdpi.com/2072-4292/13/19/3981
[38] V. Ružička et al., "RaVÆn: Unsupervised change detection of extreme events using ML on-board satellites," *Sci. Rep.*, vol. 12, no. 1, 2022, Art. no. 16939.
[39] E. Kervennic et al., "Embedded cloud segmentation using AI : Back on years of experiments in orbit on OPS-SAT," in *Proc. Eur. Data Handling Data Process. Conf.*, Juan-Les-Pins, France, Oct. 2023, pp. 1–8.
[40] J. A. Justo et al., "Hyperspectral image segmentation for optimal satellite operations: In-orbit deployment of 1D-CNN," *Remote Sens.*, vol. 17, no. 4, 2025, Art. no. 642, [Online]. Available: https://www.mdpi.com/2072-4292/17/4/642
[41] C. Ieracitano et al., "An explainable embedded neural system for on-board ship detection from optical satellite imagery," *Eng. Appl. Artif. Intell.*, vol. 133, 2024, Art. no. 108517, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197624006754
[42] A. S. Italiana, "Prisma algorithm theoretical basis document (ATBD)," 2021, Accessed: Jul. 05, 2025. [Online]. Available: https://prisma.asi.it/missionselect/docs/PRISMA%20ATBD_v1.pdf
[43] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convNet for the 2020s," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 11966–11976.
[44] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis.*, Munich, Germany, Sep. 2018, pp. 3–19.
[45] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," 2021, *arXiv:2111.00396*..
[46] W. Shi et al., "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, Jun./Jul. 2016, pp. 1874–1883.
[47] P. P. G. from the telecommunications and p. u. remote sensing laboratory, "Pavia university hyperspectral dataset," rOSIS airborne hyperspectral image over Pavia, Italy, 2020. [Online]. Available: https://www.kaggle.com/datasets/syamkakarla/pavia-university-hsi
[48] songyz2019, "rs_fusion_datasets," 2025. [Online]. Available: https://github.com/songyz2019/rs-fusion-datasets
[49] Y. Zhong et al., "WHU-Hi: UAV-borne hyperspectral with high spatial resolution (H2) benchmark datasets and classifier," *Remote Sens. Environ.*, vol. 250, 2020, Art. no. 112012.
[50] J. Li, X. Huang, and L. Tu, "WHU-OHS: A benchmark dataset for large-scale hersepctral image classification," *Int. J. Appl. Earth Observ. Geoinformation*, vol. 113, 2022, Art. no. 103022.
[51] D. C. Heinz et al., "Chein-I-Chang Fully constrained least squares linear spectral mixture analysis method for material quantification in hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 39, no. 3, pp. 529–545, Mar. 2001.
[52] T. Bodrito, A. Zouaoui, J. Chanussot, and J. Mairal, "A trainable spectral-spatial sparse coding model for hyperspectral image restoration," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 5430–5442.
[53] G. Fu, F. Xiong, J. Lu, J. Zhou, J. Zhou, and Y. Qian, "Hyperspectral image denoising via spatial–spectral recurrent transformer," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 5511214.

Matteo Impieri received the M.Sc. degree in ICT for Smart Societies from the Politecnico di Torino, Torino, Italy, in 2023.

He is currently a Research Fellow with Politecnico di Torino, working in the area of deep learning for anomaly detection and image processing. His research interests include artificial intelligence, remote sensing and probabilistic modeling techniques.

Davide Piccinini (Student Member, IEEE) received the master's degree in mathematics in 2023 from Università di Torino, Dipartimento di Matematica, where he is currently working toward the Ph.D. degree in deep learning.

He later completed a fellowship in the image processing and learning group of the Politecnico di Torino, Turin, Italy.

Diego Valsesia (Member, IEEE) received the Ph.D. degree in electronic and communication engineering from the Politecnico di Torino, Torino, Italy, in 2016.

He is currently an Associate 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 a Senior Area Editor for 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.

Enrico Magli (Fellow, IEEE) received the M.Sc. and Ph.D. degrees from the Politecnico di Torino, 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 applications. He is a Senior Associate Editor of IEEE JOURNAL ON SELECTED TOPICS IN SIGNAL PROCESSING, and a former Associate Editor of the *EURASIP Journal on Image and Video Processing*, the IEEE TRANSACTIONS ON MULTIMEDIA and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. 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 IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY.