A Trainable Spectral-Spatial Sparse Coding Model for Hyperspectral Image Restoration
Théo Bodrito, Alexandre Zouaoui, Jocelyn Chanussot, Julien Mairal

To cite this version:

HAL Id: hal-03423559
https://hal.science/hal-03423559
Submitted on 17 Nov 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
A Trainable Spectral-Spatial Sparse Coding Model for Hyperspectral Image Restoration

Théo Bodrito, Alexandre Zouaoui, Jocelyn Chanussot, and Julien Mairal
Inria, Univ. Grenoble Alpes, CNRS, Grenoble INP, LJK, 38000 Grenoble, France
firstname.lastname@inria.fr

Abstract

Hyperspectral imaging offers new perspectives for diverse applications, ranging from the monitoring of the environment using airborne or satellite remote sensing, precision farming, food safety, planetary exploration, or astrophysics. Unfortunately, the spectral diversity of information comes at the expense of various sources of degradation, and the lack of accurate ground-truth “clean” hyperspectral signals acquired on the spot makes restoration tasks challenging. In particular, training deep neural networks for restoration is difficult, in contrast to traditional RGB imaging problems where deep models tend to shine. In this paper, we advocate instead for a hybrid approach based on sparse coding principles that retains the interpretability of classical techniques encoding domain knowledge with handcrafted image priors, while allowing to train model parameters end-to-end without massive amounts of data. We show on various denoising benchmarks that our method is computationally efficient and significantly outperforms the state of the art.

1 Introduction

Hyperspectral imaging (HSI) enables measurements of the electromagnetic spectrum of a scene on multiple bands (typically about a hundred or more), which offers many perspectives over traditional color RGB imaging. For instance, the high-dimensional information present in a single pixel is sometimes sufficient to identify the signature of a particular material, which is of course infeasible in the RGB domain. Not surprisingly, hyperspectral imaging is then of utmost importance and has a huge number of scientific and technological applications such as remote sensing [8, 22, 47], quality evaluation of food products [16, 19, 56], medical imaging [2, 18, 38], agriculture and forestry [1, 37, 41], microscopy imaging in biology [25, 57], or exoplanet detection in astronomy [24].

Information contained in hyperspectral signals is much richer than in RGB images, but the price to pay is the need to deal with complex degradations that may arise from multiple sources, including sparse noise with specific patterns (stripes), in addition to photon and thermal noise [30, 52]. As a consequence, HSI denoising is a crucial pre-processing step to enhance the image quality before using data in downstream tasks such as semantic segmentation or spectral unmixing [31]. A second issue is the lack of large-scale collection of ground-truth high-quality signals and the large diversity of sensor types, which makes it particularly challenging to train machine learning models for restoration such as convolutional neural networks. To deal with the scarcity of ground-truth data, most successful approaches typically encode strong prior knowledge about data within the model architecture, which may be low-rank representations of input patches [17, 23, 54, 61, 71], sparse coding [13, 21, 23], or image self-similarities [40, 50, 72], which have proven to be very powerful in the RGB domain [9].

1 Equal contribution

1 Code is available at https://github.com/inria-thoth/T3SC

In this paper, we propose a fully interpretable machine learning model for hyperspectral images that may be seen as a hybrid approach between deep learning techniques, where parameters can be learned end to end with supervised data, and classical methods that essentially rely on image priors. Since designing an appropriate image prior by hand is very hard, our goal is to benefit from deep learning principles (here, differentiable programming [6]) while encoding domain knowledge and physical rules about hyperspectral data directly into the model architecture, which we believe is a key to develop robust approaches that do not require massive amounts of training data.

More precisely, we introduce a novel trainable spectral-spatial sparse coding model with two layers, which performs the following operations: (i) The first layer decomposes the spectrum measured at each pixel as a sparse linear combination of a few elements from a learned dictionary, thus performing a form of linear spectral unmixing per pixel, where dictionary elements can be seen as basis elements for spectral responses of materials present in the scene. (ii) The second layer builds upon the output of the first one, which is represented as a two-dimensional feature map, and sparsely encodes patches on a dictionary in order to take into account spatial relationships between pixels within small receptive fields. To further reduce the number of parameters to learn and leverage classical prior knowledge about spectral signals [61], we also assume that the dictionary elements admit a low-rank structure—that is, dictionary elements are near separable in the space and spectrum domains, as detailed later. Even though dictionary learning has been originally introduced for unsupervised learning [43, 48], we adopt an unrolled optimization procedure inspired by the LISTA algorithm [26], which has been very successful in imaging problems for training sparse coding models from supervised data [34, 55, 56, 64].

Our motivation for adopting a two-layer model is to provide a shared architecture for different HSI sensors, which often involve a different number of bands with different spectral responses. Our solution consists of learning sensor-specific dictionaries for the first layer, while the dictionary of second layer is shared across modalities. This allows training simultaneously on several HSI signals, the first layer mapping input data to a common space, before processing data by the second layer.

We experimentally evaluate our HSI model on standard denoising benchmarks, showing a significant improvement over the state of the art (including deep learning models and more traditional baselines), while being computationally very efficient at test time. Perhaps more important than pure quantitative results, we believe that our work also draws interesting conclusions for machine learning. First, by encoding prior knowledge within the model architecture directly, we obtain models achieving excellent results with a relatively small number of parameters to learn, a conclusion also shared by [34, 35] for RGB imaging; nevertheless, the effect is stronger in our work due to the scarcity of training data for HSI denoising and the difficulty to train deep learning models for this task. Second, we also show that interpretable architectures are useful: our model architecture can adapt to different noise levels per band and modify the encoding function at test time in a principled manner, making it well suited for solving blind denoising problems that are crucial for processing hyperspectral signals.

2 Related Work on Hyperspectral Image Denoising

Learning-free and low-rank approaches. Classical image denoising methods such as BM3D [12] may be applied independently to each spectral band of HSI signals, but such an approach fails to capture relations between channels; Not surprisingly, multi-band techniques such as BM4D [40] have been shown to perform better for HSI, and other variants were subsequently proposed such as GLF [72]. Tensor-based methods such as LLRT [11] are able to exploit the underlying low-rank structure of HSI signals [17, 54, 68] and have shown particularly effective when combined with a non-local image prior as in NGMeet [28]. Finally, other approaches adapt traditional image processing priors such as total variation [66, 60], or wavelet sparsity [49, 53] but they tend to perform worse than GLF, LLRT, or NGMeet, see [32] for a survey on denoising techniques for HSI.

Sparse coding models. Dictionary learning [48] is an unsupervised learning technique consisting of representing a signal as a linear combination of a few elements from a learned dictionary, which has shown to be very effective for various image restoration tasks [15, 44]. Several approaches have then combined dictionary learning and low-rank regularization. For instance, 3D patches are represented as tensors in [50] and are encoded by using spatial-spectral dictionaries [58]. In [71], 2D patches are extracted from the band-vectorized representation of the 3D HSI data and sparsely encoded on a dictionary, while encouraging low-rank representations with a trace norm penalty on
the reconstructed image. The low-rank constraint can also be enforced by designing the dictionary as the result of the matrix multiplication between spatial and spectral dictionaries learned by principal component analysis as in [21]. However, these methods typically compute sparse representations with an iterative optimization procedure, which may be computationally demanding at test time.

Deep learning. Like BM3D above, convolutional neural networks for grayscale image denoising (e.g., DnCNN [69]) may also be applied to each spectral band, which is of course suboptimal. Because deep neural networks have been highly successful for RGB images with often low computational inference cost, there have been many attempts to design deep neural networks dedicated to HSI denoising. For instance, to account for the large number of hyperspectral bands, several approaches based on convolutional neural networks are operating on sliding windows in the spectral domain, [38] [55] [67], which allows training models on signals with different number of spectral bands, but the sliding window significantly increases the inference time. More precisely, attention layers are used in [55], while more traditional CNNs are used in [39], possibly with residual connections [67]. Recently, an approach based on recurrent architecture was proposed in [63] to process signals with an arbitrary number of bands, achieving impressive results for various denoising tasks.

Hybrid approaches. SMDS-Net [64] adopts a hybrid approach between sparse coding and deep learning models by adapting the RGB image restoration method of [35] to HSI images. The resulting pipeline however lacks interpretability: SMDS-Net first denoises the input image with non-local means [9], then performs subspace projection in the spectral domain using HySime [7], before sparsely encoding 3D patches (cubes) with a trainable version of Tensor-based ISTA [51]. Although this method reduces considerably the number of parameters in comparison to vanilla deep learning models, the spectral sliding windows approach lacks interpretability since the same denoising procedure is applied across different bands, which may not suffer from the same level of noise. In contrast, we propose a much simpler sparse coding model, which is physically consistent with the nature of hyperspectral signals, by introducing a novel differentiable low-rank sparse coding layer.

3 A Trainable Spectral-Spatial Sparse Coding Model (T3SC)

In this section, we introduce our trainable spectral-spatial sparse coding model dedicated to hyperspectral imaging, and start by presenting some preliminaries on sparse coding.

3.1 Background on Sparse Coding

Image denoising with dictionary learning. A classical approach introduced by Elad and Aharon [15] for image denoising consists in considering the set of small overlapping image patches (e.g., 8 × 8 pixels) from a noisy image, and compute a sparse approximation of these patches onto a learned dictionary. The clean estimates for each patch are then recombined to produce the full image. Formally, let us consider a noisy image y in \( \mathbb{R}^{c \times h \times w} \) with c channels and two spatial dimensions. We denote by \( y_1, y_2, \ldots, y_n \) the n overlapping patches from y of size \( c \times s \times s \), which we represent as vectors in \( \mathbb{R}^m \) with \( m = cs^2 \). Assuming that a dictionary \( D = [d_1, \ldots, d_p] \) in \( \mathbb{R}^{m \times p} \) is given—we will discuss later how to obtain a “good” dictionary—each patch \( y_i \) is processed by computing a sparse approximation:

\[
\min_{\alpha_i \in \mathbb{R}^p} \frac{1}{2} \|y_i - D\alpha_i\|^2 + \lambda \|\alpha_i\|_1,
\]

where \( \| \cdot \|_1 \) is the \( l_1 \)-norm, which is known to induce sparsity in the problem solution [43], and \( \alpha_i \) is the sparse code representing the patch \( y_i \), while \( \lambda \) controls the amount of regularization. Note that the \( l_0 \)-penalty, which counts the number of non-zero elements, could also be used, leading to a combinatorial problem whose solution is typically approximated by a greedy algorithm. After solving the n problems [1], each patch \( y_i \) admits a “clean” estimate \( D\alpha_i \). Because each pixel belongs to several patches, the full denoised image \( \hat{x} \) is obtained by averaging these estimates.

Finding a good dictionary can be achieved in various manners. In classical dictionary learning algorithms, \( D \) is optimized such that the sum of the loss functions [1] is as small as possible, see [43] for a review. Adapting the dictionary with supervision is also possible [42], as discussed next.
We are now in shape to introduce a trainable layer encoding both sparsity and low-rank principles. which we found to be a too stringent condition to achieve good reconstruction in practice.

where

we adopt a slightly different approach, where we consider a single dictionary $D$ where

$(2)$ with the presence of a matrix $R$ which differs from $(2)$ with the presence of a matrix $S$ with pointwise non-linearities $\eta > 0$.

This is the approach chosen in [35, 56] for image restoration, which considers the following iterations:

Interestingly, the LISTA algorithm can also be used to train dictionaries for supervised learning tasks. This is the approach chosen in [35, 55] for image restoration, which considers the following iterations:

Finally, the denoised image $\hat{x}$ is reconstructed by averaging the patch estimates:

where $R_i$ is the linear operator that places the patch $x_i$ at position $i$ in the image, and we assume—by neglecting border effects for simplicity—that each pixel admits the same number $m$ of estimates.

In contrast to classical restoration techniques based on dictionary learning, the LISTA point of view enables us to learn the model parameters $C, D, W$ in a supervised fashion. Given a training set of pairs of noisy/clean images, we remark that the estimate $\hat{x}$ is obtained from a noisy image $y$ by a sequence of operations that are differentiable almost everywhere, as typical neural networks with rectified linear unit activation functions. A typical loss, which we optimize by stochastic gradient descent, is then

where $(x, y)$ is a pair of clean/noisy images drawn from some training distribution from which we can sample, and $\hat{x}(y)$ is the clean estimate obtained from $[4]$, given the noisy image $y$.

### 3.2 A Trainable Low-Rank Sparse Coding Layer

We are now in shape to introduce a trainable layer encoding both sparsity and low-rank principles.

#### Spatial-Spectral Representation.

As shown in [10, 21], HSI patches can be well reconstructed by using only a few basis elements obtained by principal component analysis. The authors further decompose these into a Cartesian product of separate spectral and spatial dictionaries. In this paper, we adopt a slightly different approach, where we consider a single dictionary $D = [d_1, \ldots, d_p]$ in $\mathbb{R}^{m \times p}$ as in the previous section with $m = cs^2$, but each element may be seen as a matrix of size $c \times s^2$ with low-rank structure. More precisely, we enforce the following representation

where $U_j$ is in $\mathbb{R}^{s^2 \times r}$, $V_j$ is in $\mathbb{R}^{r \times c}$, $r$ is the desired rank of the dictionary elements, and vec$(\cdot)$ is the operator than flattens a matrix to a vector. The hyperparameter $r$ is typically small with $r = 1, 2$ or $3$. When $r = 1$, the dictionary elements are said to be separable in the spectral and spatial domains, which we found to be a too stringent condition to achieve good reconstruction in practice.
The low-rank assumption allows us to build a model with a reduced number of parameters, while encoding natural assumptions about the data directly in the model architecture. Indeed, whereas a classical full-rank dictionary \( D \) admits \( cs^2p \) parameters, the decomposition (5) yields dictionaries with \((s^2 + c)rp\) parameters only. Matrices \( C \) and \( W \) are parametrized in a similar manner.

**Convolutional variant and implementation tricks.** Whereas traditional sparse coding reconstructs local signals (patches) independently according to the iterations (3), another variant called convolutional sparse coding (CSC) represents the whole image by a sparse linear combination of dictionary elements placed at every possible location in the image [56]. From a mathematical point of view, the reconstruction loss for computing the codes \( \alpha \) given an input image \( y \) becomes

\[
\min_{\{\alpha_i \in \mathbb{R}^p\}_{i=1,...,n}} \frac{1}{2} \left\| y - \frac{1}{m} \sum_{i=1}^n R_i D \alpha_i \right\|_2^2 + \lambda \sum_{i=1}^n \| \alpha_i \|_1. \tag{6}
\]

An iterative approach for computing these codes can be obtained by a simple modification of (5) consisting of replacing the quantity \( D \alpha_i^{(t)} \) by the \( i \)-th patch of the reconstructed image \( \frac{1}{m} \sum_{i=1}^n R_i D \alpha_i^{(t)} \). All of these operations can be efficiently implemented in standard deep learning frameworks, since the corresponding operations correspond to a transposed convolution with \( D \), followed by convolution with \( C \), see [56] for more details. In this paper, we experimented with the CSC variant (6) and SC one (1), both with low-rank dictionaries, which were previously described. We observed that CSC was providing slightly better results and was thus adopted in our experiments. Following [35], another implementation trick we use is to consider a different \( \lambda \) parameter per dictionary element, which slightly increases the number of parameters, while allowing to learn with a weighted \( \ell_1 \)-norm in (6).

### 3.3 The Two-Layer Sparse Coding Model with Sensor-Specific Layer

One of the main challenges in hyperspectral imaging is to train a model that can generalize to several types of sensors, which typically admit different number of spectral bands. Whereas learning a model that is tuned to a specific sensor is perfectly acceptable in many contexts, it is often useful to learn a model that is able to generalize across different types of HSI signals. To alleviate this issue, several strategies have been adopted such as (i) projecting signals onto a linear subspace of fixed dimension, with no guarantee that representations within this subspace can be comparable between different signals, or (ii) processing input data using a sliding window across the spectral domain.

In this paper, we address this issue by learning a two-layer model, presented in Figure 1 where the first layer is tuned to a specific sensor, whereas the second layer could be generic. Note that the
second layer carries most of the model parameters (about $20\times$ more than in the first layer in our experiments). Formally, let us denote by $\alpha$ in $\mathbb{R}^{p \times h \times w}$ the sparse encoding of an input tensor $y$ in $\mathbb{R}^{c \times h \times w}$ as previously described. A sparse coding layer $\Phi$ naturally yields an encoder and a decoder such that:

$$\Phi^{enc}: y \mapsto \alpha, \quad \text{and} \quad \Phi^{dec}: \alpha \mapsto \frac{1}{n} \sum_{i=1}^{n} R_i W \alpha_i.$$

(7)

Given a noisy image $y$, the denoising procedure described in the previous section with one layer can be written as

$$\hat{x}(y) = \Phi^{dec} \circ \Phi^{enc}(y).$$

Then, a straightforward multilayer extension of the procedure may consist of stacking several sparse coding layers $\Phi_1, \ldots, \Phi_L$ together to form a multilayer sparse coding denoising model:

$$\hat{x}(y) = \Phi^{dec}_L \circ \ldots \circ \Phi^{dec}_2 \circ \Phi^{enc}_L \circ \ldots \circ \Phi^{enc}_1(y).$$

The model we propose is composed of two layers, as shown in Figure 1. The first layer encodes spectrally the input HSI image, meaning that it operates on $1 \times 1$ patches, whereas the second layer encodes both spectrally and spatially the output of the first layer.

### 3.4 Noise Adaptive Sparse Coding

An advantage of using a model based on a sparse coding objective (1) is to give the ability to encode domain knowledge within the model architecture. For instance, the Lasso problem (1) seen from a maximization a posteriori estimator implicitly assumes that the noise is i.i.d. If the noise variance is different on each spectral band, a natural modification to the model is to introduce weights and use a weighted-$\ell_2$ data fitting term (which could be applied as well to the CSC model of (6)):

$$\min_{\alpha_i \in \mathbb{R}^p} \frac{1}{2} \sum_{j=1}^{c} \beta_j \| M_j (y_i - D \alpha_i) \|^2 + \lambda \| \alpha_i \|_1,$$

(8)

where $M_j$ is a linear operator that extracts band $j$ from a given HSI signal. From a probabilistic point of view, if $\sigma_j^2$ denotes the variance of the noise for band $j$, we may choose the corresponding weight $\beta_j$ to be proportional to $1/\sigma_j^2$. Yet, estimating accurately $\sigma_j^2$ is not always easy, and we have found it more effective to simply learn a parametric function $\beta_j = g(M_j y)$—here, a very simple CNN with three layers, see supplementary material for details—which is applied independently to each band. It is then easy to modify the LISTA iterations accordingly to take into account these weights, and learn the model parameters jointly with those of the parametric function $g$.

### 3.5 Self-Supervised Learning: Blind-Band Denoising with No Ground Truth Data

Even though acquiring limited ground truth data for a specific sensor is often feasible, it is also interesting to be able to train models with no ground truth at all, e.g., for processing images without physical access to the sensor. In such an unsupervised setting, deep neural networks are typically trained for RGB images by using blind-spot denoising techniques [33], consisting of predicting pixel values given their context. Here, we propose a much simpler approach exploiting the spectral redundancy between channels. More precisely, each time we draw an image for training, we randomly mask one band (or more), and train the model to reconstruct the missing band from the available ones. Formally, the training objective becomes

$$\min_{C,D,W,\lambda} \mathbb{E}_{x,y,S} \left[ \sum_{j \in S} \| M_j (\hat{x}_S(y) - y) \|^2 \right],$$

(9)

where $S$ is the set of bands that are visible for computing the sparse codes $\alpha_i$, leading to a reconstructed image that we denote by $\hat{x}_S(y)$. Formally, it would mean considering the objective (5), but replacing the sum $\sum_{j=1}^{c}$ by $\sum_{j \in S}$. This is in spirit similar to blind-spot denoising, except that bands are masked instead of pixels, making the resulting implementation much simpler.
### Table 1: Simplified comparison between learning-free and learning-based approaches.

<table>
<thead>
<tr>
<th></th>
<th>Data req.</th>
<th>training</th>
<th>inference</th>
<th>adapt. to new data</th>
<th>complex noise</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>learning-free</strong></td>
<td>no req.</td>
<td>no training</td>
<td>slow</td>
<td>easy</td>
<td>poor</td>
</tr>
<tr>
<td><strong>learning-based</strong></td>
<td>clean data</td>
<td>slow</td>
<td>fast</td>
<td>complicated</td>
<td>good perf.</td>
</tr>
</tbody>
</table>

### 4 Experiments

We now present various experiments to demonstrate the effectiveness of our approach for HSI denoising, but first, we discuss the difficulty of defining the state of the art in this field. We believe indeed that it is not always easy to compare learning-free approaches based on supervised learning. These two classes of approaches have very different requirements/characteristics, making one class more relevant than the other one in some scenarios, and less in others. Table 1 summarizes their characteristics, displaying advantages and drawbacks of both approaches.

#### Benchmarked models.

Keeping in mind the previous dichotomy, we choose to compare our method to traditional methods such as bandwise BM3D [12] (implementation based on [45, 46]), BM4D [40], GLF [72], LLRT [11], NGMeet [28]. We also included deep learning models in our benchmark such as HSID-CNN [67], HSI-SDeCNN [39], 3D-ADNet [55], SMDS-Net [64] and QRNN3D [63]. Results of HSID-CNN, HSI-SDeCNN and 3D-ADNet on Washington DC Mall (available in the Appendix) are taken directly from the corresponding papers, as the train/test split is the same. Otherwise, the results were obtained by running the code obtained directly from the authors, except for SMDS-Net, where our implementation turned out to be slightly more effective. Note that the same architecture for our model was used in all our experiments (see Appendix).

#### Datasets.

We evaluate our approach on two datasets with significantly different properties.

- **ICVL** [4] consists of 204 images of size $1392 \times 1300$ with 31 bands. We used 100 images for training and 50 for testing as in [63] but with a different train/test split ensuring that similar images—e.g., picture from the same scene—are not used twice.
- **Washington DC Mall** is perhaps the most widely used dataset for HSI denoising and consists of a high-quality image of size $1280 \times 307$ with 191 bands. Following [55], we split the image into two sub-images of size $600 \times 307$ and $480 \times 307$ for training and one sub-image of size $200 \times 200$ for testing. Even though the test image does not overlap with train images, they nevertheless share common characteristics. Interestingly, the amount of training data is very limited here.

Specific experiments were also conducted with the datasets APEX [29], Paviti [1] Urban [59] and CAVE [65], which appear in the supplementary material.

#### Normalization.

Before denoising, HSI images are normalized to $[0, 1]$. For remote sensing datasets, we pre-compute the 2nd and 98th percentiles for each band, on the whole the training set. Then, normalization is performed on train and test images by clipping each band between those percentiles before applying bandwise min-max normalization, similar to [5, 39]. For the close-range dataset ICVL, we simply apply global min-max normalization as in [64, 63].

#### Noise patterns.

We evaluate our model against different types of synthetic noise:

- **i.i.d Gaussian noise with known variance** $\sigma^2$, which is the same on all bands.
- **Gaussian noise with unknown band-dependent variance**: We consider Gaussian noise with different standard deviation $\sigma_j$ for each band, which is uniformly drawn in a fixed interval. These standard deviations change from an image to the other and are unknown at test time.
- **Noise with spectrally correlated variance**: We consider Gaussian noise with standard deviation $\sigma_j$ varying continuously across bands, following a Gaussian curve, see details in the appendix.
- **Stripes noise**: similar to [63], we applied additive stripes noise to 33% of bands. In those bands, 10-15% of columns are affected, meaning a value uniformly sampled in the interval $[-0.25, 0.25]$ is added to them. Moreover, all bands are disturbed by Gaussian noise with noise intensity $\sigma = 25$.  

---

https://engineering.purdue.edu/~biehl/MultiSpec/hyperspectral.html

Metrics. In order to assess the performances the previous methods, we used five different indexes widely used for HSI restoration, namely (i) Mean Peak-to-Noise Ratio (MPSNR), which is the classical PSNR metric averaged across bands; (ii) Mean Structural Similarity Index Measurement (MSSIM), which is based on the SSIM metric [62]; (iii) Mean Feature Similarity Index Measurement (MFSIM) introduced in [70]; (iv) Mean ERGAS [14], and (v) Mean Spectral Angle Map (MSAM) [3]. We use MPSNR and MSSIM in the main paper and report the other metrics in the appendix.

Implementation details. We trained our network by minimizing the MSE between the groundtruth and restored images. For ICVL, we follow the training procedure described in [63]: we first center crop training images to size \(1024 \times 1024\), then we extract patches of size \(64 \times 64\) at scales 1:1, 1:2, and 1:4, with stride 64, 32 and 32 respectively. The number of extracted patches for ICVL amounts to 52962. For Washington DC Mall, we do not crop training images and the patches are extracted with stride 16, 8 and 8, for a total of 1650 patches. One epoch in Washington DC Mall corresponds to 10 iterations on the training dataset. Basic data augmentation schemes such as \(90^\circ\) rotations and vertical/horizontal flipping are performed. Code and additional details about optimization, implementation, computational resources, are provided in the supplementary material. As reported in Table 4, augmenting the number unrolled iterations improves the denoising performances at the expense of inference time. Since the Spectral-Spatial SC layer is the most time-consuming, the number of unrolled iterations chosen for the first and second layers are 12 and 5 respectively.

Quantitative results on synthetic noise. We present in Table 2 the results obtained on the ICVL dataset (results on DCMall are presented in the appendix). Our method uses the vanilla model of Section 3.3 for the experiments with constant \(\sigma\) or correlated noise. For the blind denoising experiment with band-dependent \(\sigma\) or for the stripe noise experiment, we use the variant of Section 3.4, which is designed to deal with unknown noise level per channel. The method “T3SC-SSL” implements the self-supervised learning approach of Section 3.5 which does not rely on ground-truth data.

- Our supervised approach achieves state-of-the-art results (or is close to the best performing baseline) on all settings. GLF performs remarkably well given that this baseline is learning-free.
- Our self-supervised method achieves a relatively good performance under i.i.d. Gaussian noise, but does not perform as well under more complex noise. This is a limitation of the approach which is perhaps expected and overcoming this limitation would require designing a different self-supervised learning scheme; this is an interesting problem, which is beyond the scope of this paper.

A visual result on ICVL is shown in Figure 2 for stripes noise. Inference times are provided in Table 3 showing that our approach is computationally efficient.

Results on real noise. We also conducted a denoising experiment on the Urban dataset, reporting a visual result in Figure 3. Deep models were pre-trained on the APEX dataset, which has the same number of channels as Urban (even though the sensors are different), with band-dependent noise with \(\sigma \in [0 - 55]\). Please note that for this experiment we did not use Noise Adaptive Sparse Coding for T3SC, as it is highly dependent on the type of sensor used for training. We show that learning-based models trained on synthetic noise are able to transfer to real data.

Comments on the additional results presented in the appendix. The appendix also contains (i) results on the DCMall dataset including additional baselines mentioned above; (ii) error bars for parts of our experimental results in order to assess their statistical significance; (iii) an experiment when learning simultaneously on several datasets with different types of sensors showing that the second layer can be generic and effective at the same time; (iv) additional visual results; (v) various ablation studies to illustrate the importance of different components of our method.

Broader Impact

Our paper addresses the problem of denoising the signal, which is a key pre-processing step before using hyperspectral signals in concrete applications. As such, it is necessarily subject to dual use. For instance, HSI may be used for environmental monitoring, forestry, yield estimation in agriculture, natural disaster management planning, astronomy, archaeology, and medicine. Yet, HSI is also used by the petroleum industry for finding new oil fields, and has obvious military applications for surveillance. We believe the potential benefits of HSI for society are large enough to outweigh the
Table 2: Denoising performance on ICVL with various types of noise patterns. The first four rows correspond to i.i.d. Gaussian noise with fixed $\sigma$ per band. The next three rows corresponds to a noise level that depends on the band, taken uniformly on small interval. This is a blind-noise experiment since at test time, the noise level is unknown. The last two rows correspond to the scenarios with correlated $\sigma$ across bands, and with stripe noise, respectively. See main text for details.

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>Metrics</th>
<th>Noisy</th>
<th>BM3D</th>
<th>BM4D</th>
<th>GLF</th>
<th>LLRT</th>
<th>NGMeet</th>
<th>SMDS</th>
<th>QRNN3D</th>
<th>T3SC</th>
<th>T3SC-SSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>MPSNR</td>
<td>34.47</td>
<td>46.17</td>
<td>48.85</td>
<td>51.25</td>
<td>51.86</td>
<td><strong>52.74</strong></td>
<td>50.91</td>
<td>48.80</td>
<td><strong>52.62</strong></td>
<td>51.42</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.7618</td>
<td>0.984</td>
<td>0.9916</td>
<td>0.9949</td>
<td>0.995</td>
<td><strong>0.9960</strong></td>
<td>0.9944</td>
<td>0.9918</td>
<td><strong>0.9959</strong></td>
<td>0.9952</td>
</tr>
<tr>
<td>25</td>
<td>MPSNR</td>
<td>21.44</td>
<td>37.86</td>
<td>39.89</td>
<td>43.16</td>
<td>43.43</td>
<td><strong>44.74</strong></td>
<td>42.83</td>
<td>44.20</td>
<td><strong>45.38</strong></td>
<td>44.73</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.1548</td>
<td>0.9269</td>
<td>0.9510</td>
<td>0.9695</td>
<td>0.9746</td>
<td><strong>0.9796</strong></td>
<td>0.9700</td>
<td>0.9782</td>
<td><strong>0.9825</strong></td>
<td>0.9805</td>
</tr>
<tr>
<td>50</td>
<td>MPSNR</td>
<td>16.03</td>
<td>34.22</td>
<td>34.22</td>
<td>39.26</td>
<td>39.69</td>
<td><strong>41.08</strong></td>
<td>39.25</td>
<td>41.81</td>
<td><strong>42.16</strong></td>
<td>41.82</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.0502</td>
<td>0.8654</td>
<td>0.8654</td>
<td>0.9197</td>
<td>0.9504</td>
<td>0.9602</td>
<td>0.9382</td>
<td>0.9655</td>
<td><strong>0.9677</strong></td>
<td>0.9646</td>
</tr>
<tr>
<td>100</td>
<td>MPSNR</td>
<td>10.85</td>
<td>30.43</td>
<td>32.47</td>
<td>34.79</td>
<td>36.39</td>
<td>37.55</td>
<td>35.64</td>
<td>37.19</td>
<td><strong>38.99</strong></td>
<td>38.50</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.0144</td>
<td>0.7557</td>
<td>0.8155</td>
<td>0.7982</td>
<td>0.9182</td>
<td>0.9311</td>
<td>0.8815</td>
<td>0.9140</td>
<td><strong>0.9439</strong></td>
<td>0.9394</td>
</tr>
<tr>
<td>[0-15]</td>
<td>MPSNR</td>
<td>33.89</td>
<td>45.81</td>
<td>45.35</td>
<td>50.57</td>
<td>48.50</td>
<td>41.67</td>
<td>48.23</td>
<td>52.07</td>
<td><strong>53.31</strong></td>
<td>51.26</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.6386</td>
<td>0.9767</td>
<td>0.9735</td>
<td>0.9948</td>
<td>0.9899</td>
<td>0.9078</td>
<td>0.9900</td>
<td>0.9957</td>
<td><strong>0.9967</strong></td>
<td>0.9955</td>
</tr>
<tr>
<td>[0-55]</td>
<td>MPSNR</td>
<td>23.36</td>
<td>39.06</td>
<td>38.43</td>
<td>44.22</td>
<td>41.13</td>
<td>32.94</td>
<td>41.76</td>
<td>47.13</td>
<td><strong>48.64</strong></td>
<td>46.82</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.2601</td>
<td>0.9231</td>
<td>0.9074</td>
<td>0.9818</td>
<td>0.9580</td>
<td>0.7565</td>
<td>0.9620</td>
<td>0.9884</td>
<td><strong>0.9911</strong></td>
<td>0.9882</td>
</tr>
<tr>
<td>[0-95]</td>
<td>MPSNR</td>
<td>19.06</td>
<td>36.17</td>
<td>35.55</td>
<td>41.43</td>
<td>38.44</td>
<td>29.40</td>
<td>38.94</td>
<td>43.98</td>
<td><strong>46.30</strong></td>
<td>44.75</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.1614</td>
<td>0.8760</td>
<td>0.8540</td>
<td>0.9674</td>
<td>0.9354</td>
<td>0.6609</td>
<td>0.9357</td>
<td>0.9753</td>
<td><strong>0.9859</strong></td>
<td>0.9822</td>
</tr>
<tr>
<td>Corr.</td>
<td>MPSNR</td>
<td>28.85</td>
<td>42.73</td>
<td>42.13</td>
<td>47.05</td>
<td>45.76</td>
<td>38.06</td>
<td>45.98</td>
<td>48.90</td>
<td><strong>49.89</strong></td>
<td>48.78</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.4740</td>
<td>0.9599</td>
<td>0.9070</td>
<td>0.9881</td>
<td>0.9824</td>
<td>0.8536</td>
<td>0.9835</td>
<td>0.9911</td>
<td><strong>0.9923</strong></td>
<td>0.9911</td>
</tr>
<tr>
<td>Strip</td>
<td>MPSNR</td>
<td>21.20</td>
<td>34.88</td>
<td>37.70</td>
<td>42.06</td>
<td>39.38</td>
<td>39.78</td>
<td>41.98</td>
<td>44.60</td>
<td><strong>44.74</strong></td>
<td>43.80</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.1508</td>
<td>0.8641</td>
<td>0.9198</td>
<td>0.9628</td>
<td>0.9258</td>
<td>0.9333</td>
<td>0.9655</td>
<td>0.9806</td>
<td><strong>0.9805</strong></td>
<td>0.9773</td>
</tr>
</tbody>
</table>

Table 3: Inference time per image on ICVL with $\sigma = 50$; SMDS, QRNN3D and T3SC are using a V100 GPU; BM4D, GLF, LLRT and NGMeet are using an Intel(R) Xeon(R) CPU E5-1630 v4 @ 3.70GHz. Note that unlike GLF, NGMeet, and LLRT, learning-based approaches such as QRNN3D and our approach require a training procedure, which may be conducted offline. The cost of such a training step was about 13.5 hours for our method and 19 hours for QRNN3D on a V100 GPU.

<table>
<thead>
<tr>
<th></th>
<th>BM3D</th>
<th>BM4D</th>
<th>GLF</th>
<th>LLRT</th>
<th>NGMeet</th>
<th>SMDS</th>
<th>QRNN3D</th>
<th>T3SC</th>
<th>T3SC-SSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference time (s)</td>
<td>1677</td>
<td>2382</td>
<td>5565</td>
<td>24384</td>
<td>2686</td>
<td><strong>74.3</strong></td>
<td><strong>5.8</strong></td>
<td><strong>54.2</strong></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: Denoising results with Gaussian noise $\sigma = 25$ on ICVL with bands 9, 15, 28.
Table 4: Impact of the number of unrolled iterations per layer on denoising performances and inference time. This experiment was carried out on ICVL with $\sigma = 50$.

<table>
<thead>
<tr>
<th>Unrolled iterations per layer</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPSNR</td>
<td>40.16</td>
<td>41.48</td>
<td>42.15</td>
<td>42.45</td>
</tr>
<tr>
<td>Inference time (s)</td>
<td>0.38</td>
<td>1.44</td>
<td>5.27</td>
<td>14.91</td>
</tr>
</tbody>
</table>

Figure 3: Visual result on a real HSI denoising experiment on Urban dataset with bands 1, 108, 208.

Acknowledgments and Funding

This project was supported by the ERC grant number 714381 (SOLARIS project) and by ANR 3IA MIAI@Grenoble Alpes (ANR-19-P3IA-0003). This work was granted access to the HPC resources of IDRIS under the allocation 2021-[AD011012382] made by GENCI.

References


Supplementary Material

A Implementation details

In this section, we provide additional implementation details, which are useful to reproduce our experiments (note that the code is also provided).

Noise with spectrally correlated variance. For each band \(i \in [0, c - 1]\), the standard deviation of the Gaussian noise is defined as :

\[
\sigma_i = \beta \exp \left[ -\frac{1}{4\eta^2} \left( \frac{i}{c} - \frac{1}{2} \right)^2 \right]
\]

with \(\beta = 23.08\) and \(\eta = 0.157\).

Preprocessing. A basic centering step is used for each input patch of our model. More precisely, for the first layer, each band of the input hyperspectral image is centered independently prior to patches extraction, and means are added back after decoding. For the second layer, patches are centered independently for each band (and similarly, the means are added back after decoding).

Code and patch sizes The hyperparameters of our model are presented in Table 5.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Patches size</th>
<th>Code size</th>
<th>Unrolled iterations</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spectral SC</td>
<td>1 \times 1</td>
<td>64</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>Spectral-Spatial SC</td>
<td>5 \times 5</td>
<td>1024</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 5: Architecture of our model

Table 14 shows that the combination of both layers is more performant than each layer independently.

Initialization All parameters are initialized with He initialization.

Blocks inference In order to apply our model to large images, we split them into blocks of size 256 \(\times\) 256 with an overlap of 6 pixels. Each block is denoised independently. The output image is obtained by aggregating the denoised blocks. Pixels comprised in several blocks are averaged.

Weights estimator For complex noise such as Gaussian noise with band-dependent variance or stripes noise, our model uses a CNN to estimate the weights \(\beta_j\) associated with each band. The CNN operates on centered patches of size 56 \(\times\) 56 both during training (random crops) and inference (blocks inference), and its architecture is described in Table 6.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Kernel size</th>
<th>Stride</th>
<th>#filters</th>
<th>Output size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td></td>
<td></td>
<td></td>
<td>1 \times 56 \times 56</td>
</tr>
<tr>
<td>Conv2D + ReLU</td>
<td>5</td>
<td>2</td>
<td>64</td>
<td>64 \times 26 \times 26</td>
</tr>
<tr>
<td>MaxPooling2D</td>
<td></td>
<td></td>
<td></td>
<td>64 \times 13 \times 13</td>
</tr>
<tr>
<td>Conv2D + ReLU</td>
<td>3</td>
<td>2</td>
<td>128</td>
<td>128 \times 6 \times 6</td>
</tr>
<tr>
<td>MaxPooling2D</td>
<td></td>
<td></td>
<td></td>
<td>128 \times 3 \times 3</td>
</tr>
<tr>
<td>Conv2D + Sigmoid</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1 \times 1 \times 1</td>
</tr>
</tbody>
</table>

Table 6: CNN architecture for estimating \(\beta_j\)

The ablation study presented in Table 15 shows that this extension improves performances substantially for complex noise.

Optimization Our models are trained with batch size of 16 for 60 epochs. We use the Adam optimizer, the initial learning rate is \(3 \times 10^{-4}\), and is divided by two at epoch 30 and 45.
Self-Supervised Learning. During the training, \( n \) bands randomly selected are masked simultaneously, and reconstructed from the available ones. The MSE loss is applied on the masked bands only. For testing, the \( n \) masked bands are evenly distributed along the spectral dimension. All bands are reconstructed after \( \lceil c/n \rceil \) iterations, where \( c \) denotes the total number of bands and \( \lceil \cdot \rceil \) is the ceiling operator. We used \( n = 4 \) for ICVL and \( n = 16 \) for Washington DC Mall.

For the SSL setting to be realistic, the noise is added to the clean image before patches are extracted. Otherwise, the model would have indirect access to the groundtruth by seeing the same patch with different noise realizations. As a consequence, the denoising task is much harder on complex noise when data is limited, as shown in Table 7.

B Additional Quantitative Results.

Washington DC Mall dataset. Results for this dataset are presented in Table 7. Additional baselines are presented in Table 8. The conclusions are similar to those already drawn in the main paper.

<table>
<thead>
<tr>
<th>( \sigma )</th>
<th>Metrics</th>
<th>Noisy</th>
<th>BM3D</th>
<th>BM4D</th>
<th>GLF</th>
<th>LLRT</th>
<th>NGMeet</th>
<th>SMDS</th>
<th>QRNN3D</th>
<th>T3SC</th>
<th>T3SC-SSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>MPSNR</td>
<td>34.31</td>
<td>35.10</td>
<td>41.13</td>
<td>39.57</td>
<td>41.83</td>
<td>37.57</td>
<td>42.83</td>
<td>43.42</td>
<td>43.85</td>
<td>42.56</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.9821</td>
<td>0.9875</td>
<td>0.9962</td>
<td>0.9953</td>
<td>0.9968</td>
<td>0.9928</td>
<td>0.9971</td>
<td>0.9973</td>
<td>0.9978</td>
<td>0.9967</td>
</tr>
<tr>
<td>25</td>
<td>MPSNR</td>
<td>20.70</td>
<td>24.51</td>
<td>31.08</td>
<td>35.25</td>
<td>34.95</td>
<td>35.38</td>
<td>35.64</td>
<td>35.04</td>
<td>36.74</td>
<td>35.92</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.7688</td>
<td>0.8859</td>
<td>0.9630</td>
<td>0.9883</td>
<td>0.9683</td>
<td>0.9886</td>
<td>0.9889</td>
<td>0.9864</td>
<td>0.9912</td>
<td>0.9894</td>
</tr>
<tr>
<td>50</td>
<td>MPSNR</td>
<td>15.25</td>
<td>20.80</td>
<td>26.69</td>
<td>31.77</td>
<td>30.94</td>
<td>31.88</td>
<td>31.76</td>
<td>31.72</td>
<td>33.12</td>
<td>31.96</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.5314</td>
<td>0.7508</td>
<td>0.9220</td>
<td>0.9761</td>
<td>0.9704</td>
<td>0.9759</td>
<td>0.9765</td>
<td>0.9761</td>
<td>0.9819</td>
<td>0.9762</td>
</tr>
<tr>
<td>100</td>
<td>MPSNR</td>
<td>10.48</td>
<td>17.65</td>
<td>22.51</td>
<td>27.81</td>
<td>26.82</td>
<td>27.86</td>
<td>28.02</td>
<td>27.41</td>
<td>29.48</td>
<td>28.04</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.2888</td>
<td>0.5427</td>
<td>0.8141</td>
<td>0.9475</td>
<td>0.9322</td>
<td>0.9460</td>
<td>0.9491</td>
<td>0.9375</td>
<td>0.9618</td>
<td>0.9460</td>
</tr>
</tbody>
</table>

Corr. MPSNR: 28.48 30.50 33.69 37.96 37.77 36.56 38.54 39.84 40.79 39.61 39.16
MSSIM: 0.9085 0.9515 0.9637 0.9928 0.9911 0.9934 0.9944 0.9960 0.9944
Strip. MPSNR: 20.47 24.08 29.07 35.27 34.13 34.94 35.24 35.25 36.34 34.50
MSSIM: 0.7621 0.8672 0.9433 0.9877 0.9811 0.9780 0.9859 0.9877 0.9874 0.9906 0.9853

Study of statistical significance for the ICVL dataset. In order to evaluate the statistical significance of our results, we present some results in Table 7 for some of our models and baselines, by running models with five different random seeds. Note that we did not conduct such a study for all results in this paper in order to keep the computational cost of the project reasonable. The conclusions of the paper remain unchanged.
### Table 9: Denoising performances on ICVL with multiple seeds

<table>
<thead>
<tr>
<th>σ</th>
<th>Metrics</th>
<th>Noisy</th>
<th>GLF</th>
<th>NGMeet</th>
<th>SMDS</th>
<th>QRNN3D</th>
<th>T3SC</th>
<th>T3SC-SSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>MPSNR</td>
<td>34.47 ± 0.01</td>
<td>51.25 ± 0.01</td>
<td>62.74 ± 0.01</td>
<td>50.78 ± 0.09</td>
<td>49.54 ± 0.28</td>
<td>52.62 ± 0.01</td>
<td>51.37 ± 0.03</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.7619 ± 0.0001</td>
<td>0.9951 ± 0.0001</td>
<td>0.9961 ± 0.0001</td>
<td>0.9941 ± 0.0001</td>
<td>0.9924 ± 0.0021</td>
<td>0.9960 ± 0.0001</td>
<td>0.9952 ± 0.0001</td>
</tr>
<tr>
<td>25</td>
<td>MPSNR</td>
<td>21.43 ± 0.04</td>
<td>44.16 ± 0.01</td>
<td>47.74 ± 0.01</td>
<td>42.64 ± 0.14</td>
<td>44.20 ± 0.16</td>
<td>45.37 ± 0.02</td>
<td>44.70 ± 0.02</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.1548 ± 0.0002</td>
<td>0.9906 ± 0.0001</td>
<td>0.9975 ± 0.0001</td>
<td>0.9867 ± 0.0009</td>
<td>0.9780 ± 0.0009</td>
<td>0.9882 ± 0.0001</td>
<td>0.9863 ± 0.0001</td>
</tr>
<tr>
<td>50</td>
<td>MPSNR</td>
<td>19.56 ± 0.01</td>
<td>39.36 ± 0.01</td>
<td>41.69 ± 0.01</td>
<td>39.09 ± 0.08</td>
<td>41.17 ± 0.14</td>
<td>42.16 ± 0.01</td>
<td>41.62 ± 0.01</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.0503 ± 0.0001</td>
<td>0.9190 ± 0.0002</td>
<td>0.9961 ± 0.0001</td>
<td>0.9939 ± 0.0012</td>
<td>0.9639 ± 0.0010</td>
<td>0.9677 ± 0.0001</td>
<td>0.9648 ± 0.0001</td>
</tr>
<tr>
<td>100</td>
<td>MPSNR</td>
<td>10.85 ± 0.04</td>
<td>34.78 ± 0.01</td>
<td>37.55 ± 0.01</td>
<td>35.59 ± 0.04</td>
<td>38.38 ± 0.60</td>
<td>38.99 ± 0.04</td>
<td>35.31 ± 0.04</td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.0144 ± 0.0001</td>
<td>0.9930 ± 0.0001</td>
<td>0.9993 ± 0.0001</td>
<td>0.9974 ± 0.0017</td>
<td>0.9916 ± 0.0011</td>
<td>0.9949 ± 0.0002</td>
<td>0.9950 ± 0.0001</td>
</tr>
</tbody>
</table>

#### CAVE dataset.

We report denoising performances of T3SC on the CAVE Dataset in Table 10. To evaluate T3SC, the dataset was divided in four splits: three were used for training and one for testing. The values reported for T3SC are averaged across all rotations of the test split.

### Table 10: Denoising performances on CAVE dataset with Gaussian noise.

<table>
<thead>
<tr>
<th>σ</th>
<th>Metrics</th>
<th>Noisy</th>
<th>NGMeet</th>
<th>T3SC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>MPSNR</td>
<td>35.05</td>
<td>47.96</td>
<td>49.16</td>
</tr>
<tr>
<td>25</td>
<td>MPSNR</td>
<td>21.99</td>
<td>42.44</td>
<td>42.77</td>
</tr>
<tr>
<td>50</td>
<td>MPSNR</td>
<td>16.37</td>
<td>38.89</td>
<td>39.7</td>
</tr>
<tr>
<td>100</td>
<td>MPSNR</td>
<td>10.96</td>
<td>34.99</td>
<td>36.48</td>
</tr>
</tbody>
</table>

#### Joint training across heterogeneous datasets.

In Table 11 we study the problem of training a single model on three different datasets, APEX, DC Mall, and Pavia, involving a different number of channels. As mentioned in the paper, this model involves a common second layer and a spectral dictionary per dataset. These results show that most of the model parameters (which are present in the second layer) can in fact be shared across datasets without significant loss of accuracy when compared to the training of three different models (thus involving three times more parameters).

### Table 11: Results for joint training experiment

<table>
<thead>
<tr>
<th>Training procedure</th>
<th>Model</th>
<th>Metrics</th>
<th>APEX</th>
<th>DC Mall</th>
<th>Pavia Center</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MPSNR</td>
<td>33.19</td>
<td>31.72</td>
<td>30.56</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSSIM</td>
<td>0.9619</td>
<td>0.9741</td>
<td>0.9569</td>
</tr>
<tr>
<td>Independant training</td>
<td>QRNN3D</td>
<td>34.91</td>
<td>33.12</td>
<td>31.32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T3SC</td>
<td>0.9730</td>
<td>0.9819</td>
<td>0.9617</td>
<td></td>
</tr>
<tr>
<td>Joint training</td>
<td>QRNN3D</td>
<td>31.95</td>
<td>30.97</td>
<td>29.12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T3SC</td>
<td>34.74</td>
<td>33.08</td>
<td>31.30</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSSIM</td>
<td>0.9711</td>
<td>0.9819</td>
<td>0.9616</td>
</tr>
</tbody>
</table>

#### Additional metrics.

Additional metrics are provided for the ICVL and DC Mall datasets, respectively in Tables 12 and 13. The conclusions of the paper are unchanged.
Ablation studies. In this paragraph, we present different ablation studies, demonstrating in Table 14 that our two-layer model outperforms single-layer models. We also demonstrate the usefulness of our variant with weights $\beta_j$ in Table 15 when the noise variance varies a lot between different bands.
Table 14: Combination of sparse coding layers: we denote by Spec the Spectral Sparse Coding layer and by SpecSpat the Spectral-Spatial Sparse Coding layer. This experiment was run on ICVL with $\sigma = 50$.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Noisy</th>
<th>Spec</th>
<th>SpecSpat</th>
<th>Spec + SpecSpat</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPSNR</td>
<td>16.03</td>
<td>30.96</td>
<td>40.13</td>
<td><strong>42.17</strong></td>
</tr>
<tr>
<td>MSSIM</td>
<td>0.0502</td>
<td>0.6884</td>
<td>0.9533</td>
<td><strong>0.9677</strong></td>
</tr>
<tr>
<td>MFSIM</td>
<td>0.8100</td>
<td>0.9708</td>
<td>0.9849</td>
<td><strong>0.9925</strong></td>
</tr>
<tr>
<td>MERGAS</td>
<td>51.48</td>
<td>8.84</td>
<td>3.00</td>
<td><strong>2.39</strong></td>
</tr>
<tr>
<td>MSAM</td>
<td>0.7546</td>
<td>0.1300</td>
<td>0.1021</td>
<td><strong>0.0547</strong></td>
</tr>
</tbody>
</table>

Table 15: Our model without/with band-wise noise estimator (NE) on ICVL with band-dependent Gaussian noise and stripes noise

<table>
<thead>
<tr>
<th>Metrics</th>
<th>T3SC</th>
<th>T3SC + NE</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0-15] MPSNR</td>
<td>52.85</td>
<td><strong>53.31</strong></td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.9963</td>
</tr>
<tr>
<td>[0-55] MPSNR</td>
<td>47.39</td>
<td><strong>48.64</strong></td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.9890</td>
</tr>
<tr>
<td>[0-95] MPSNR</td>
<td>44.92</td>
<td><strong>46.30</strong></td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.9821</td>
</tr>
<tr>
<td>Strip. MPSNR</td>
<td>44.68</td>
<td><strong>44.74</strong></td>
</tr>
<tr>
<td></td>
<td>MSSIM</td>
<td>0.9801</td>
</tr>
</tbody>
</table>

C Visual Examples

Finally, we show additional visual examples in Figure 4 and 5.

Figure 4: Simulated Gaussian noise ($\sigma = 100$) on DCMall
Figure 5: Visual results for the denoising experiment with stripes noise on ICVL with bands 9, 15, 28.

D GPU resources

The total number of GPU hours involved in this project is around 19k hours on NVIDIA Tesla V100 16Go, including preliminary experiments, model design, final experiments and running baseline methods.