InSPECtor: an end-to-end design framework for compressive pixelated hyperspectral instruments

Stockmans, T.A.; Snik, F.; Esposito, M.; Dijk, C. van; Keller, C.U.

Research Article Vol. 62, No. 27 / 20 September 2023 / Applied Optics 7185

applied optics

InSPECtor: an end-to-end design framework for compressive pixelated hyperspectral instruments

T. A. Stockmans,[1,*] F. Snik,[1] M. Esposito,[2] C. van Dijk,[2] and C. U. Keller[1,3]

[1]Leiden Observatory, Leiden University, P.O. Box 9513, 2300 RA Leiden, The Netherlands
[2]cosine Remote Sensing, Warmonderweg 14, 2171 AH Sassenheim, The Netherlands
[3]Lowell Observatory, 1400 W. Mars Hill Rd., Flagstaff, Arizona 86001, USA
*stockmans@strw.leidenuniv.nl

Received 14 June 2023; revised 28 August 2023; accepted 28 August 2023; posted 29 August 2023; published 13 September 2023

Classic designs of hyperspectral instrumentation densely sample the spatial and spectral information of the scene of interest. Data may be compressed after the acquisition. In this paper, we introduce a framework for the design of an optimized, micropatterned snapshot hyperspectral imager that acquires an optimized subset of the spatial and spectral information in the scene. The data is thereby already compressed at the sensor level but can be restored to the full hyperspectral data cube by the jointly optimized reconstructor. This framework is implemented with TensorFlow and makes use of its automatic differentiation for the joint optimization of the layout of the micropatterned filter array as well as the reconstructor. We explore the achievable compression ratio for different numbers of filter passbands, number of scanning frames, and filter layouts using data collected by the Hyperscout instrument. We show resulting instrument designs that take snapshot measurements without losing significant information while reducing the data volume, acquisition time, or detector space by a factor of 40 as compared to classic, dense sampling. The joint optimization of a compressive hyperspectral imager design and the accompanying reconstructor provides an avenue to substantially reduce the data volume from hyperspectral imagers.

https://doi.org/10.1364/AO.498021

1. INTRODUCTION

Hyperspectral imaging combines the acquisition of two-dimensional spatial and spectral information [1]; it is used in a broad range of research, including–but not limited to–remote sensing [2,3], food quality control [4,5], archaeology [6], astronomy [7], agriculture [8], medical imaging [9,10], and imaging on microscales for biological and chemical processes [11].

Hyperspectral imaging contains three dimensions of information (two spatial and one spectral dimension), but most detectors are two-dimensional. This requires a trade-off in the instrument design, which often results in using time as the third dimension. The three most common techniques for air- and spaceborne instruments are whisk broom, push broom (line scan), and staring [12]. In the whisk-broom imaging mode, the system measures the full spectrum of one geometrical pixel before stepping to the next in a track perpendicular to the flight direction. In the push-broom mode, the system simultaneously measures the spectrum of a line of geometrical pixels. Staring, as opposed to the other two modes, measures the whole image in one spectral band and then steps through the bands [12–14].

What the techniques described above all have in common is that the acquired hyperspectral data cube is densely sampled and therefore partially redundant [15]. This redundancy implies that there is a representation of the data cube in which most entries map to (approximately) zero and can be ignored. When only measuring the nonzero entries of this representation, all information could still be recovered while reducing detector space, data volume, and/or measurement time. Such reductions are particularly helpful for applications in space [16], where mass, volume, power, and data rates are limited. A simple example of the redundancy in hyperspectral data cubes is the success of hyperspectral band selection where the algorithms extract the spectral bands that contain the most information [17]. However, this is a postprocessing method that does not improve the detector size and/or the acquisition time.

The imaging techniques that go beyond dense sampling are typically referred to as *compressed sensing (CS)*. Several hyperspectral instruments based on CS have been designed [18]. Examples include the spaceborne concepts proposed by [19,20] and the *computed tomographic imaging spectrometry (CTIS)* system [7] based on [21]. The most common compressive sensor for hyperspectral imaging is the *coded aperture snapshot spectral imager (CASSI)* and its variations [22–26], which combine spectral dispersers and coded focal plane masks. Other designs combine coded focal plane masks with a dispersing lens [27], a diffuser in combination with a *color filter array (CFA)* [28], or a Fourier transform spectrometer and a single-pixel detector [29].

A compressive sensor can be described mathematically by a single matrix, **H**. This measurement matrix, when multiplied with the vector representation of the measured scene, $\vec{x}$, and adding the noise, $\vec{n}$, results in the vector representation of the detected signal, $\vec{y}$, i.e.,

$$\vec{y} = \mathbf{H}\vec{x} + \vec{n}. \quad (1)$$

The reconstructor estimates $\vec{x}$ from knowing $\vec{y}$ and **H**. This inversion is not trivial due to the non-uniqueness of the problem, which requires the addition of constraints. Examples of such reconstruction algorithms can be found in [30–32] and references therein.

Some authors have optimized the instrument and thereby the measurement matrix; they require the most accurate reconstructions, while operating the fastest or sparsest [33–36]. The numerical optimization of the imaging system, in light of the needed reconstruction, resembles the definition of *computational imaging (CI)*. This should not be surprising, since the field of CS has been closely intertwined with the field of CI for some time now [37]. CI has also produced efficient and relevant instruments, like the one described in [38] or in [39]. An overview of some instruments in this field that use deep learning mostly for the reconstruction of the hyperspectral data cube can be found in [40,41]. Finally, there is the work of Wang *et al*. [42], wherein the reconstruction was jointly optimized with the coded aperture mask of the CASSI system.

Another research field that is related to CS is called demosaicking [43]. It can be described within the CS framework as a specific type of reconstructor, but this is not normally done due to its origin and scope in *red–green–blue (RGB)* photography. In most common RGB instruments, the detector is covered by a Bayer pattern of filters [44], i.e., red, green, and blue pixels are alternating in a 2×2 superpixel, where the green filter occurs twice. When a scene is imaged with a single snapshot, the intensity of the blue and green light at the location of a red pixel is unknown, and vice versa. Demosaicking provides an approximation of the intensities of all the unknown colors of each pixel, creating a fully filled RGB image. Some examples of demosaicking algorithms for RGB imaging can be found in [43,45–57]. Since its beginning in RGB imaging, demosaicking has also found its way to detectors with more spectral bands than only the RGB broadband filters. Examples of demosaicking algorithms dealing with multispectral data cubes can be found in [58–66].

The demosaicking algorithms we referenced above make use of existing color filter array designs and instruments. The design of these *color filter arrays (CFAs)* themselves has also undergone development. The CFA design can be updated to enhance the quality of the resulting processed images. For instance, in the update of the Bayer layout for RGB imagers, we refer to [67–69] and other references in the latter publication. Some examples of multispectral CFA designs can be found in [70–73]. Finally, some of the commercially available instruments for snapshot hyperspectral imaging with a CFA include the SNAPCAM by [74], XIMEA's detector [75], and Silios' detector [76]. An older, but more general overview of snapshot spectral imagers is given in [77].

Some authors have also directly related RGB imaging and hyperspectral imaging, which is referred to as spectral recovery. In this field, RGB images are transformed into hyperspectral data cubes [78–80].

In the papers referenced above, the measurement design (instrument) and the reconstruction algorithm are treated as separate entities. By considering the system as a whole and jointly optimizing the instrument design and the reconstruction algorithm together, advances have been made in RGB imaging [81,82]. The joint optimization of instrument design and recovering algorithm is also done by [83]. The instrument they have optimized combines the CASSI system described above and a multispectral filter array. The broadband encoding stochastic (BEST) camera descibed in [84] and [85] is developed by a neural network that designs both the spectral filters of a hyperspectral camera and the dense neural network for the reconstruction afterwards. Finally, most closely related to this work is the conference paper by Li et al. [86]. They describe the use of a reinforcement-learning-based band selection algorithm in combination with a neural network to design the hyperspectral CFA and do the reconstruction afterwards. There are three key points that make the work described here different: (1) They do a band selection of common broadband filters. In our described framework, the spectral properties of the filters can be set as another optimizable parameter, which enables larger versatility. (2) Their reconstruction network first demosaics the images from the CFA and then uses a separate spectral recovery algorithm to reconstruct the hyperspectral data cube. We combine this step in a single mapping, which reduces the chance of error propagation between separate layers. (3) Finally, they solely focus on snapshot imaging. The methods we describe below also take the possibility for push-broom scanning into account, which can push the accuracy of the instrument for some applications far enough to make it a feasible alternative to classical hyperspectral instruments.

In this paper, we describe a new framework that can jointly design a spectral filter array and reconstruction function that combine into a compressive hyperspectral imager. The papers mentioned above either optimize only one part of these two or are focused on a different optical design altogether. The presented framework can optimize three aspects: (1) it optimizes the filters that contain the most spectral information; (2) it determines the layout of these filters to optimize the estimated spectra for all pixels; and (3) it determines a linear reconstructor that optimally demosaics the measurements into a filled hyperspectral data cube. In the following sections, we describe the framework in detail. Then we show the accuracy provided by designs that vary in terms of the number of filters and the number of push-broom scans. Finally, we present an outlook for future applications and improvements.

2. METHODS

To design the optimal compressive measurement setup and reconstructor, we developed the InSPECtor framework. Currently, InSPECtor consists of two components. The first component only takes the spectral dimension into account and disregards the spatial dimension of a hyperspectral data set. This component determines a given number of optimized spectral filter passbands to reconstruct the full spectra. The second component, however, takes both the spatial and spectral

dimensions into account. It can, for instance, decide on the optimal layout of the filters from the first component and return the matching reconstructor. The second component can also be used to decide the best passbands in a specific fixed layout and optimize a reconstructor for the resulting instrument. In the future, the optimization of both the filters and their layout will be combined in a single framework. Below we start with an explanation of the merit functions used in our paper. We continue with a mathematical formulation of the two components in a CS formulated manner. The end of this section entails the implementation of the two components in Python code using the TensorFlow package.

A. Merit Functions

To determine the accuracy of the resulting data cubes as compared to the original ones, we calculate the mean square error (MSE) and the peak signal-to-noise ratio (PSNR). The PSNR is a widely used metric for spectral image comparisons [87]. We included the MSE to provide a nonlogarithmic scale of the error for direct comparison to the spectra. The MSE is defined as follows:

$$\mathrm{MSE} = \frac{1}{M}\sum_{k=0}^{M}(Y_k - P_k)^2, \tag{2}$$

in which Y is the true scene and P is the estimated scene and M is the total number of entries that the scene consists of. This scene can be either a single spectrum or the spectra of multiple spatial pixels. The PSNR is closely related to the MSE in a logarithmic inverse way. So note that a lower MSE means a better estimation and corresponds to a higher PSNR. The mathematical formulation of the PSNR is as follows:

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

where MAX is the maximum possible value of Y. MAX differs between applications and scenes under investigation. For example, for an 8-bit system, $\mathrm{MAX} = 255$. In this paper, we have used $\mathrm{MAX} = 2^{32} - 1$, since the images in the training and test data sets are 32-bit.

B. Optimal Filters Estimator

The first component of InSPECtor determines the optimal filter passbands independent of the spatial arrangement of the filters. The filters are implemented as a linear transformation from an input spectrum to filtered intensity measurements, from which a linear reconstructor estimates the input spectrum.

The linear transformation from an input spectrum to filtered intensity measurements is mathematically equivalent to Eq. (1). Here, $\vec{x}$ is the discretely sampled spectrum, $\vec{y}$ are the intensities through every filter, and $\mathbf{H}$ can be described as

$$\mathbf{H} = \begin{pmatrix} \vec{T}_1 \\ \vdots \\ \vec{T}_N \end{pmatrix}, \tag{4}$$

where $\vec{T}_n$ is the filter transmission at each wavelength. Here we used a normalized Gaussian spectral filter with a central wavelength, λ_n, and a full width at half-maximum, FWHM_n, i.e.,

$$\vec{T}_n = e^{\frac{\sqrt{2\ln 2}(\vec{\lambda}-\lambda_n)^2}{\mathrm{FWHM}_n^2}}. \tag{5}$$

The reconstructor is described by the following equation:

$$\mathbf{R}\vec{y} = \hat{\vec{x}}, \tag{6}$$

where $\mathbf{R}$ is the reconstruction matrix to obtain the approximation of the original spectrum: $\hat{\vec{x}}$.

This component is visualized in Fig. 1.

C. Optimal Layout Estimator

The optimal layout estimator is similar to the optimal filters estimator described above (see Fig. 2). It consists of a *layout of spectral filters* and a linear reconstructor that carries out the demosaicking and reconstructs the full hyperspectral data cube. The measurements, where individual pixels see the scene through different spectral filters, can be done either in a snapshot mode or in a push-broom fashion. In the former, only a single-intensity image is acquired. In the latter, the filters are shifted stepwise in one direction across the scene, and multiple images are taken; every step corresponds to one full image carrying different spectral information for all ground pixels.

The optimal layout estimator supports different configurations, which can be grouped into two main configurations: in the first configuration, the spectral filters are fixed and cannot be updated by the algorithm. In the second configuration, however, the filters can be updated as well. This will be further explained below in Section 2.D.

Mathematically, we can again describe this component in the compressed-sensing format, referring to Eqs. (1)–(4). However, $\vec{T}_{s,m}$ now describes the transmission of the filter focused on every geometrical pixel of the scene in every scanning frame, instead of every filter, as in Eq. (5). Assuming a detector with M pixels and taking S steps, $\mathbf{H}$ can be written as

$$\mathbf{H} = \begin{pmatrix} \vec{T}_{11} \\ \vdots \\ \vec{T}_{1M} \\ \vec{T}_{21} \\ \vdots \\ \vec{T}_{SM} \end{pmatrix}. \tag{7}$$

$\vec{x}$ is a serialized version of S times all spectra of all pixels. $\vec{y}$ contains the intensities on the detector of every pixel in every step. The linear reconstruction is the same as above in Eq. (6), but with this different $\vec{y}$.

D. TensorFlow Implementation

InSPECtor is implemented in TensorFlow, which provides rapid optimization of all free parameters of our analytical model that is fully differentiable [88]. Each component is implemented with a sequence of so-called Layers. A Layer consists of an input of

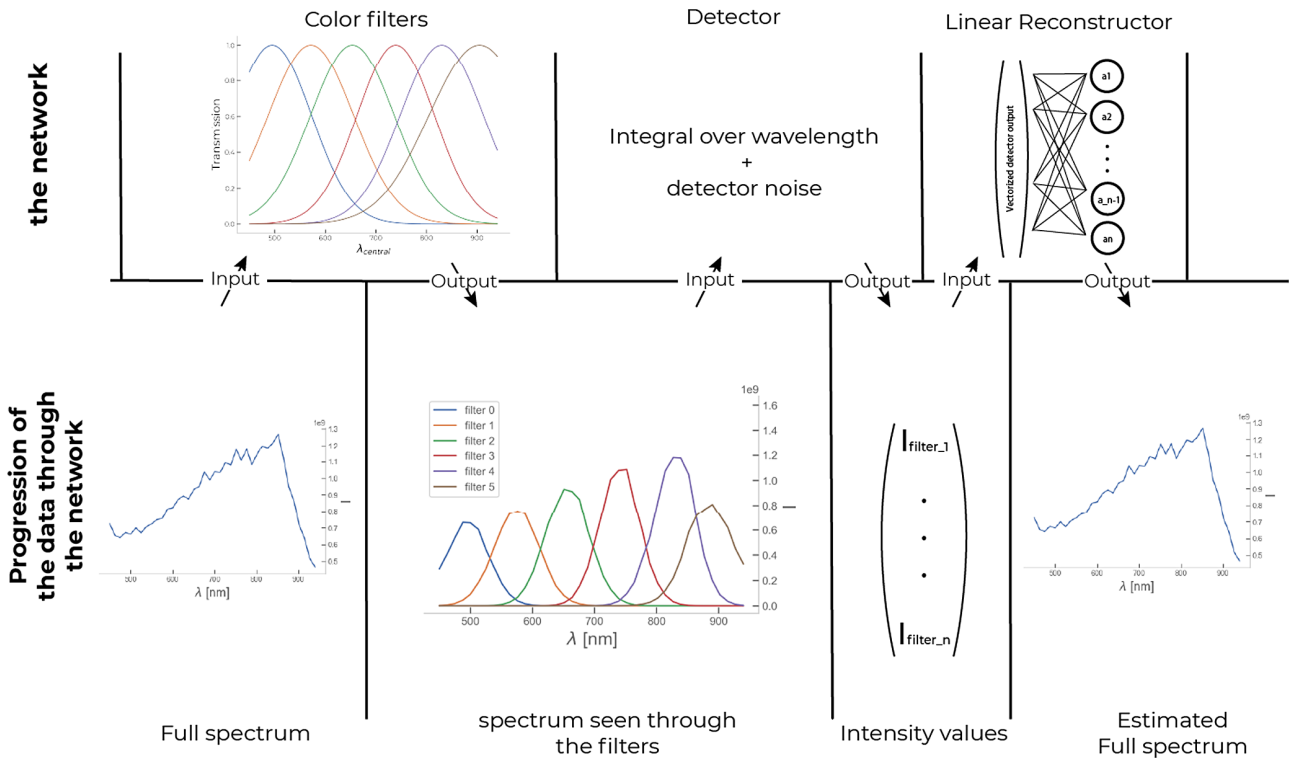

Fig. 1. Schematic overview of the optimal filter estimator and the propagation of the data through its linear transformations.

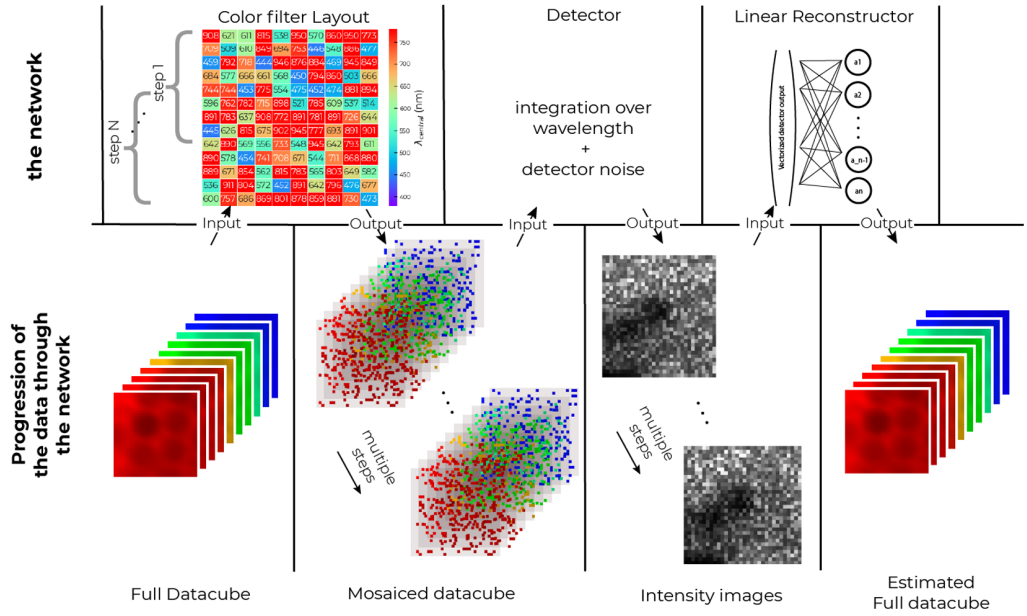

Fig. 2. Schematic overview of the network and the propagation of the data through it. It starts with a scene that passes through the spectral filter layout. Some information is being blocked by the filters, resulting in a mosaicked cube. This process is repeated for the total number of steps, which are taken in a push-broom fashion. The detector flattens the mosaicked hyperspectral cubes into 2D intensity measurements and adds noise. The multiplication of these 2D intensity images with the linear reconstructor results in an estimate of the original data cube.

Tensors, an output of Tensors, a differentiable function, and the values of the variables (weights) used in this function along with the input Tensor. Tensors are the main multidimensional data containers of TensorFlow. The physical process represented by each Layer is described by the function and the weights, and the data propagate through each Layer as the inputs and outputs.

Each of these Layers can be made trainable where the values of the weights can be updated by the optimizer, in contrast to remaining fixed during the training phase. The optimizer uses

backpropagation to update the weights. Pieces of the training data set are fed to the network, and the resulting output is compared to the desired result using a loss function, in our case the MSE. The loss function is related to every parameter in the framework in the form of a partial derivative. Using that partial derivative, the value of each parameter is updated by the optimization algorithm, to minimize the loss function. In our case, we make use of the Adam optimizer [89] with different learning rates. The learning rate is a hyperparameter that is found by trial and error.

The TensorFlow model for the optimal filters estimator consists of three sequential Layers. First is a spectral filter and detector Layer, followed by a noise Layer and, finally, a reconstruction Layer. The second component, the layout estimator, is realized with three TensorFlow Layers. These are a spectral filter layout and detector Layer, again followed by the noise Layer and the reconstruction Layer. Each of these Layers is described in more detail below. In addition, we discuss further possible additions to the model, called regularizers.

1. Spectral Filters and Detector

We describe the filter of each pixel as in Eq. (5), a normalized Gaussian profile characterized by a central wavelength and bandwidth. These two numbers make up the weights of this custom spectral filter Layer, which can be optimized. The single-spectrum input is multiplied by all the filters. The detector part is simply an integration over wavelength, resulting in an intensity value for each filter.

The weights corresponding to the central wavelength and bandwidth, respectively, of the spectral filter Layer are scaled to the -1 to $+1$ range to accelerate training.

2. Spectral Filter Layout and Detector

This Layer is very close to the spectral filter and detector Layer described above. However, the input is a full hyperspectral data cube with the same spatial dimensions as the detector. Each detector pixel has its own filter associated with it, and the multiplication of the filter happens with the spectrum of geometrical pixel imaged on it. Again an integration over wavelength happens to result in one intensity image. For each additional push-broom step; this process is repeated with the filters shifted one pixel row with respect to the spatial sampling points, and the detector images are concatenated into a single, long vector.

Different configurations can be implemented with the optimal filter layout estimator. Most of these configurations have an influence on the weights of this Layer. If the configuration contains a fixed layout, the weights of this Layer will be the λ_c and FWHM of each filter of the fixed layout. However, if the configuration does not contain a fixed layout, the weights of this layer will be the λ_c and FWHM of each filter on each single detector pixel.

3. Noise

We have added a Gaussian noise Layer, which affects the intensity measurements coming from the preceding Layer. The amplitude of this added noise is comparable to the SNR of the input data, as noted in [90]. This Layer also ensures the robustness of the design to the physical detector noise and mitigates overfitting. Overfitting denotes fitting not only the underlying patterns in the training data but also the random noise patterns, which will be different and unpredictable when validating the design with different data.

4. Reconstruction

The intensity values from the noisy detector are reconstructed in the final Layer as either a full spectrum or as the full hyperspectral data cube. The reconstruction is implemented as a linear reconstructor. This corresponds to a single Dense Layer in TensorFlow, with as many weights as there are entries in the reconstructed spectrum or hyperspectral cube; all bias weights are fixed to 0, to ensure strict proportionality between the input measurement and output data cube. This Dense Layer is initialized with zeros instead of the more common random numbers to help the network converge. To determine the spectrum of one geometrical pixel in the optimal filter layout estimator, the reconstruction can, in theory, make use of all the measurements of all geometrical pixels. However, in practice, it will focus on the connections that contain the most information, e.g., the closest ones.

5. Regularizers

Next to the loss function described above, which compares the output of the network with the desired output, additional loss terms, called regularizers, can be added to the TensorFlow model. Each added loss term must be differentiable as well for it to be able to influence the optimization of the weights. The regularizers can be as important as the network structure itself. Here we apply different regularizers to limit the noise propagation in the linear reconstructor and to implement the discrete filter selection in a differentiable manner.

The reconstruction Layer can be made less prone to overfitting by adding an L2 regularizer. L2 regularization adds the L2-norm of the weights of the linear reconstructor as a loss function and leads to a preference for smaller weights.

The second regularizer is custom-made to specifically give preference to a fixed selection of filters described by their central wavelengths and widths. This regularizer adds a loss for each filter, but this loss is reduced when the filter resembles one of the selected filters. Since there are two parameters defining each potential passband, the central wavelength and FWHM, the loss function is two-dimensional (2D). For each specified filter, there is a negative 2D normalized Lorentzian function with a global minimum at the specified filter coordinates. A Lorentzian function is preferred over a Gaussian function due to its broader wings, which accelerate convergence. All these Lorentzian functions are summed to create the full loss landscape with local minima at all specified filter coordinates. This sum of Lorentzians is evaluated for each filter in the spectral filter Layer, and all these values are added as the additional loss term, which is expressed in the following equation:

$$L = \alpha_{\text{reg}} \sum_{\text{pixels}} \sum_{\text{filters}} 1 - \frac{A^2}{(\lambda - \lambda_{\text{filter}})^2 + (\text{FWHM} - \text{FWHM}_{\text{filter}})^2 + A^2}, \quad (8)$$

where A controls how fast a deviation from the desired filter results in a big loss term. α_{reg} determines the weight of this regularizer with respect to the other regularizers and the global loss function. λ_{filter} and $\text{FWHM}_{\text{filter}}$ are the central wavelength and FWHM of the specified filter, respectively. An example of the loss landscape for this regularizer can be seen in Fig. 3. Note that this loss is a unitless quantity with the sole purpose of optimization with respect to certain design constraints; it is not related to any physics in the system.

The regularizer described above is the key link between the two frameworks of InSPECtor: when the layout is not fixed, it ensures that the optimal filters from the optimal filters framework are highly preferred over all others in the layout design.

6. Configurations of the Framework

As mentioned in Section 2.C, the filter layout estimator can be configured in many ways. The weights in the color-filter Layer and the reconstruction Layer can be set to be trainable or not. In addition, the initialization of the weights of these layers and the choice of regularizers can be selected.

A configuration of the framework is determined by the following items:

- Determine the initialization of the filters.
- Either optimize the filters or fixate the filters to the initialized values.
- Determine the initialization of the layout.
- Either optimize the layout or fixate the layout to the initialized pattern.
- If both the filters and the layout need to be optimized, do they need to be optimized simultaneously?

Throughout this paper, we will initialize the filters in the same way. We will call this initialization "regular filters." The term "regular filters" represents filters with identical Gaussian passbands, spaced in wavelength by their FWHM and spanning the complete wavelength range of interest (450–940 nm). Two regular filters would both have a FWHM of 245 nm and be centered at 577.5 and 818.5 nm, respectively.

When the filters are optimized, there are two possibilities called "best filters" and "optimized filters." They are related to the last item that determines the configuration: the term best filters means that the filters are the result of the optimization done by the "Optimal filters estimator;" see Section 2.B. The final term, optimized filters, means that the filters are optimized together with the layout in the "optimal layout estimator," described above in Section 2.C.

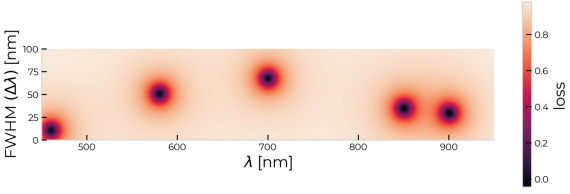

Fig. 3. Loss that each filter adds according to its wavelength and FWHM when the following combinations (λ, $\Delta\lambda$) are desired: (460 nm, 10 nm), (580 nm, 50 nm), (850 nm, 34 nm), (900 nm, 29 nm) and (700 nm, 67 nm). These combinations were randomly selected for display purposes.

The layout is initialized either randomly or with a fixed pattern. In the section below, the two fixed patterns that we use are described: an LVF-like pattern and the "squarish" pattern. In future work, more patterns could be implemented in this framework as long as they are generated with a specific function that is differentiable and not limited to certain sizes of the simulated detector.

7. Fixed Patterns

We describe two different fixed patterns in this section, an LVF-like pattern and our own squarish pattern.

The LVF-like pattern is a repreating arrangement of the different filters with central wavelengths increasing in the scanning direction, while being uniform in the other direction.

The squarish pattern can be best defined as a lattice in statistical physics, by a unit cell and two linearly independent primitive translation vectors. We keep to the definition 12.2.1 in [91] for a unit cell as follows: A unit cell is the repeated motif that is the elementary building block of the periodic structure.

In our case, a unit cell contains each spectral filter at least once in a fixed layout. For instance, if a 500-nm filter is directly to the right of a 650-nm filter in the unit cell, all 650-nm filters in the total layout will find a 500-nm filter to their right except for the edges of the sensor.

The unit cell for our squarish pattern is defined to have a unit cell that is as close as possible to a square within the square grid graph [92]. When a perfect square is not possible, the direction perpendicular to the scanning direction is filled first.

The primitive translation vector is the vector between the same filter in two different unit cells. One of the primitive translation vectors can always be defined to be perfectly aligned perpendicular to the scanning direction. The second is then usually fixed due to the requirement that the pattern be uninterrupted. There is one exception when the unit cell is a perfect rectangle. In that case, the translation vector is chosen to never be perfectly aligned with the scanning direction, but always skewed by one pixel. This skew is introduced to ensure that a full cycle of all filters is repeated in the scanning direction, instead of a repetition of a subset, as in a Bayer pattern. (One example is shown in Figs. 8(c) and 8(d).)

E. Training, Validation and Test Data

1. Hyperscout Data

To train and validate the framework, we have made use of satellite data obtained by the Hyperscout instrument [93]. We used one of the 440 by 440-pixel images with a ground sampling distance of 70 m. There are 40 wavelength bands spanning 450–940 nm with a spectral resolution of about 15 nm each. The scene is shown in Fig. 4 as an RGB picture and contains agricultural fields as well as sand, rivers, and clouds. The RGB

Fig. 4. RGB representation of the sampled scene. The red line shows the divide between the test set on the right and the training and validation set on the left.

picture was generated by using the Python package "colour science" and scaling the colors to Google Maps satellite imagery.

The satellite data was separated into two different parts, as shown in Fig. 4. Three hundred-fifty columns, 154,000 spectra, were reserved for training (training set) and the hyperparameter selection (validation set). The separate test set consisted of the remaining 90 columns, or 39,600 spectra. This test set was kept separated during training and the selection of the best performing hyperparameters and was only used to present the final results of the network noted in the sections below.

The optimal filters were determined by training and validating on 100,000 randomly selected spectra from the training and validation part of the Hyperscout data set. The resulting filters are subsequently tested on 10,000 randomly selected spectra from the test set.

For the optimal layout estimator, we transformed the Hyperscout data set into patches of 10 pixels × 10 pixels. This size corresponds to the used detector size of 10 pixels × 10 physical pixels. The training data and validation data consisted of 10,000 patches from the corresponding part of the Hyperscout data set that were randomly augmented by mirroring and/or rotation by 90°, 180°, or 270°. Although there is a high likelihood that there is a partial overlap between some different patches, exact duplicates were avoided. The test set consisted of 1000 patches without any augmentation from the test part of the Hyperscout data set.

To separate between the training set and validation set, we used the inbuilt separation function of TensorFlow. This randomly selected 10% for the calculation of the validation loss and 90% for the actual training of the network.

2. *Information Content of the Data*

The power of CS lies in using the correlations in space and wavelength of the scene. In this section, we determine the information content of the data used to estimate the amount of compression that will be viable. To this end, we (1) assess the spatial correlations at a given wavelength with a Fourier analysis and the spatial and spectral correlations by calculating the PSNR between pixels as a function of their distance and (2) determine the information contained in the spectra with a principle component analysis (PCA).

The simplest method to analyze the spatial correlations is the power spectrum of the image at a given wavelength (see Fig. 5). Figure 5(a) shows that the power spectrum is approximately azimuthally symmetric, which allows us to limit ourselves to the azimuthal average in different wavelengths [see Fig. 5(b)]. We observe a gradual decline with increasing spatial frequency; the data set does not show any flattening at the higher frequencies, which indicates that the data are not dominated by white noise, even at the highest spatial frequencies.

A flattening of the power spectrum would indicate that the pixel binning is too high for the resolution of the optical system, and neighboring pixels would sample the same resolution element. In the absence of this flattening, we conclude that the system is not spatially oversampled.

To analyze the spectral correlations in space, we determined the average PSNR of two pixels as a function of their distance, which is shown in Fig. 6. When pixels are close by, their spectra are very similar (high PSNR). However, when two pixels are far from each other, their spectra differ greatly. When the distance

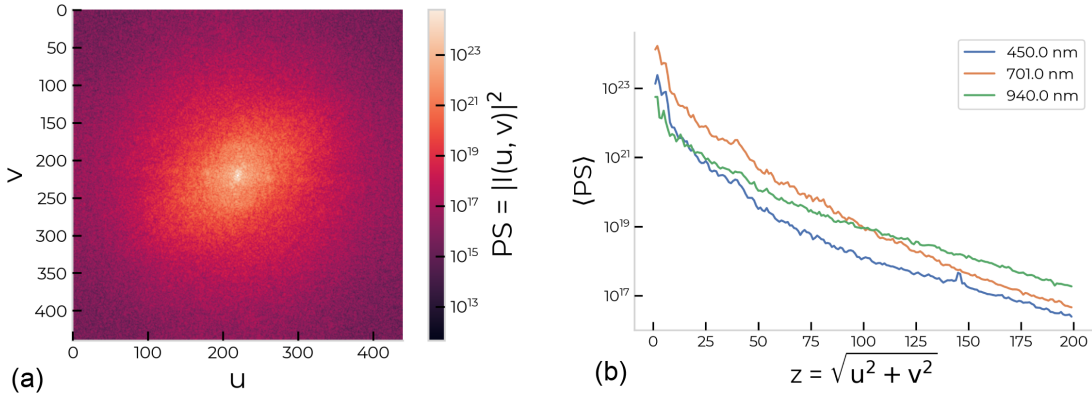

Fig. 5. Fourier analysis. (a) Power spectrum of the image at a wavelength of 701 nm, tapered with a Bartlett–Hann window toward the average; (b) azimuthal average of the power spectrum at different wavelengths.

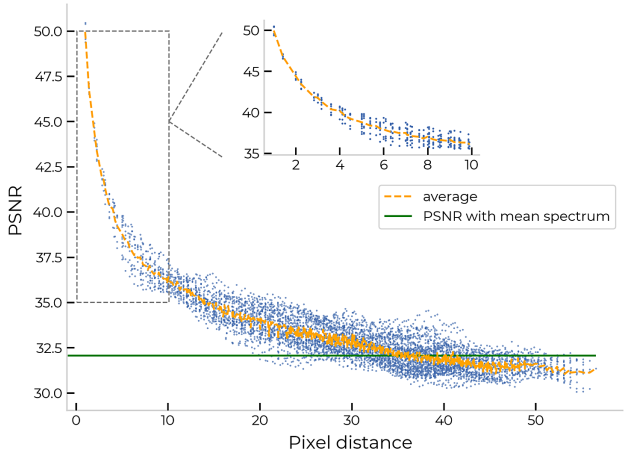

Fig. 6. Blue points show the PSNR between the spectra of two pixels as a function of the distance between them. The orange line is the average of all pairs with the same distance. The green line is the PSNR of a comparison of a pixel's spectrum to the mean spectrum of the whole image, averaged over all pixels. A close-up of the PSNR for distances of 10 pixels and fewer is shown in the upper right.

exceeds 34 pixels, the spectrum of a pixel is better approximated by the mean spectrum of the whole image than the spectrum of a random, far, far, away pixel. This indicates the distance at which all but the most basic correlation between pixels vanishes.

To assess the information content of the spectra, we carried out a PCA; see Fig. 7. The drop-off in the variance of successive PCA components is very sharp, indicating that much of the spectra can be approximated with a small number of PCA components [see Fig. 7(a)]. The individual PCA components shown in Fig. 7(b) seem to pick up the Vegetation Red Edge (component 2) and water vapor absorption in the NIR (component 4).

Finally, we have calculated the average PSNR between a spectrum and its approximation per number of PCA components used for this approximation. This calculation shows how much each PCA component adds to reproducing the original data. The number of components that results in an acceptable approximation is related to the number of filters needed for an acceptable reconstruction. However, since there are no negative filters, and interference filter transmission profiles have limitations, the number of PCA components cannot be converted directly to the number of required optical filters.

From four PCA components onwards, the improvements in approximation are minor. Our expectation is that the first PCA component could be approximated by a filter directly. However, for the second PCA component two filters would be necessary: one until 700 nm and one starting at 700 nm. For the third PCA component, we expect a filter from 500 to 600 nm and one from 600 to 700 nm. Finally, the fourth PCA component could be done with a single filter at 900 nm. Adding all these filters, we expect that seven filters would be necessary to adequately approximate the full spectrum at 40 wavelength bands.

3. RESULTS

In this section, we show the results produced by InSPECtor as a function of number of filters and number of steps. In the TensorFlow environment, these two variables are implemented as hyperparameters:

- the number of push-broom steps to take, going from a snapshot image (one step) to two or more frames
- the number of filters present in the layout, going from two different filters up to 19

The PSNR and MSE are evaluated at all permutations of these two hyperparameters.

We investigated five different configurations. The three best-performing configurations out of these five will be discussed in more detail. In order of complexity, the five different configurations are: regular filters in a fixed LVF-like layout, best filters in a fixed LVF-like layout, best filters in an optimized random layout, regular filters in a fixed squarish layout, and finally optimized filters in a fixed squarish layout. The resulting five layouts for the case of six filters and four steps are shown in Fig. 8.

Additional hyperparameters encode the learning rate and weight of the l2 regularizations on the linear reconstructor. We run the framework with the different values of these two additional hyperparameters, noted in Table 1. The resulting validation losses are then compared to determine the optimum design and reconstructor for each configuration. Furthermore, the "joint" configuration is run multiple times, starting with different random initializations of the spectral filter Layer, which is not necessary for the other configurations that feature a static spectral filter layout.

Figure 9 shows the configuration that reaches the best PSNR for a given pair of steps and filters. We can see that the squarish pattern with optimized filters performs best in almost all cases. When the number of filters is high, the difference between optimized filters and just regular filters begins to diminish.

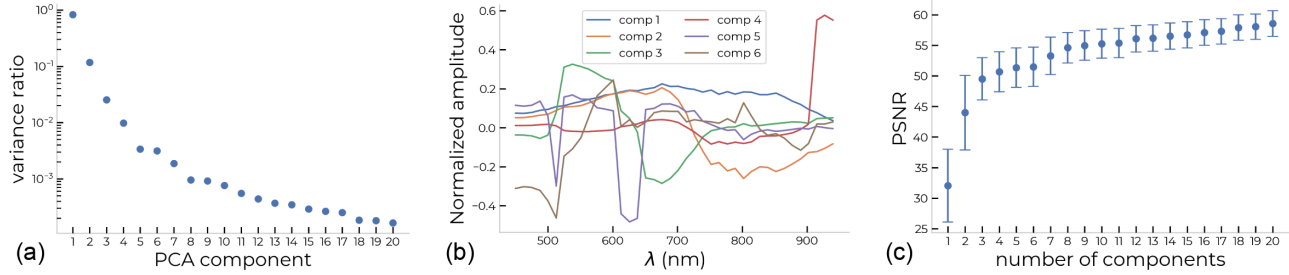

Fig. 7. PCA analysis. (a) Variances associated with each PCA component of the spectra in the image. The variances are normalized to add up to 1. (b) PCA components of the spectra in the Hyperscout data set; (c) average PSNR between a spectrum and its approximation as a function of the number of PCA components used for this approximation.

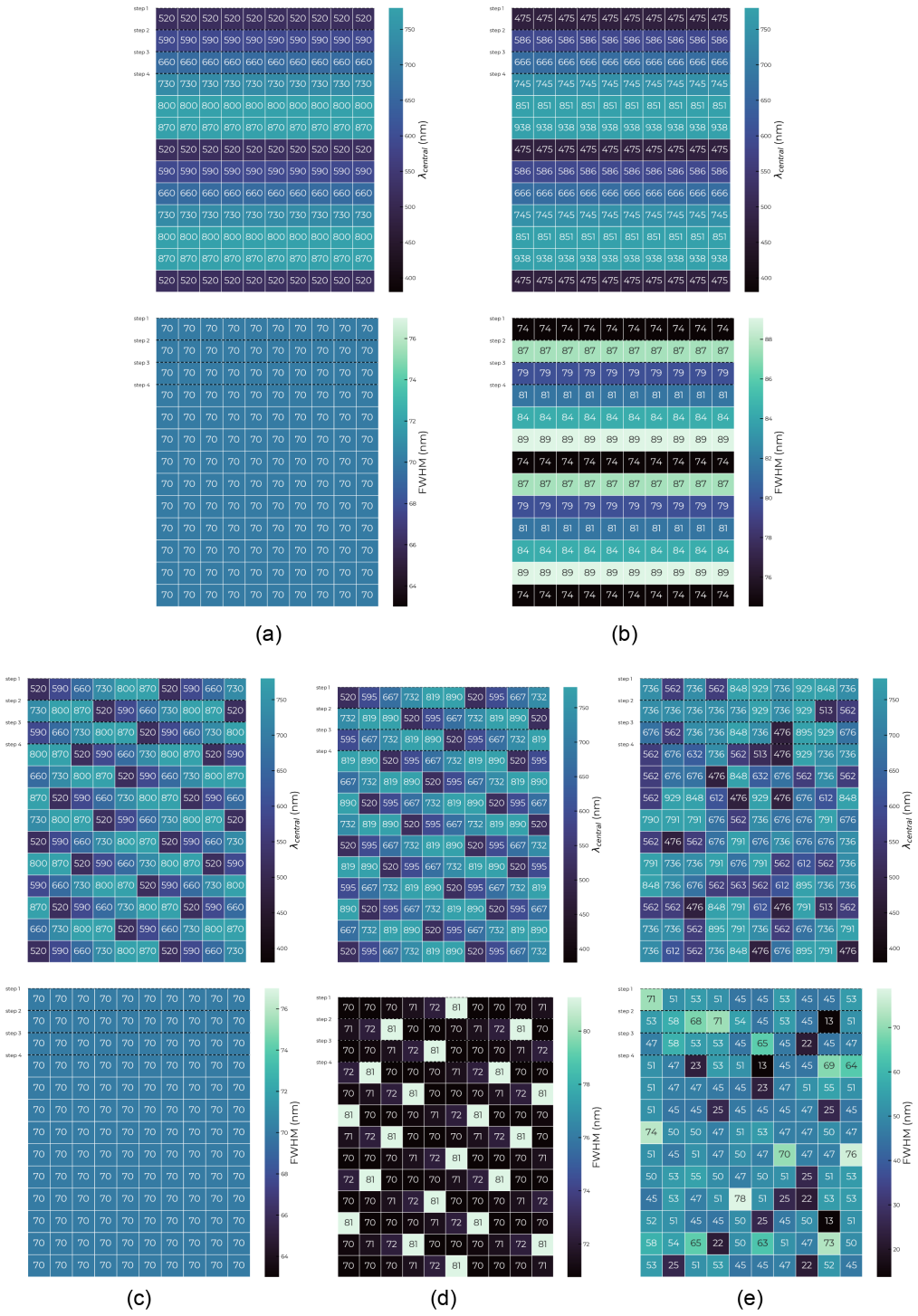

Fig. 8. Detector layout in different configurations after convergence by the network. Note that the layout in (a)–(c) are unchanged from their initialization. (a) LVF layout, with fixed regular filters; (b) LVF layout with fixed filters separately optimized by the first component; (c) squarish layout with fixed regular filters; (d) squarish layout with optimized filters; (e) optimized layout with fixed filters separately optimized by the first component.

Table 1. Different Possible Values of the Hyperparameters

L2 weight	0	0.0001	0.001
Learning rate	0.0001	0.0003	0.001

Every additional push-broom step implies an additional image that has to be acquired and transmitted by the instrument. The original LVF design of the Hyperscout instrument needs 40 push-broom steps, which requires 40 images to construct the full data cube. We define compression as the number of images that need to be taken compared to the original 40

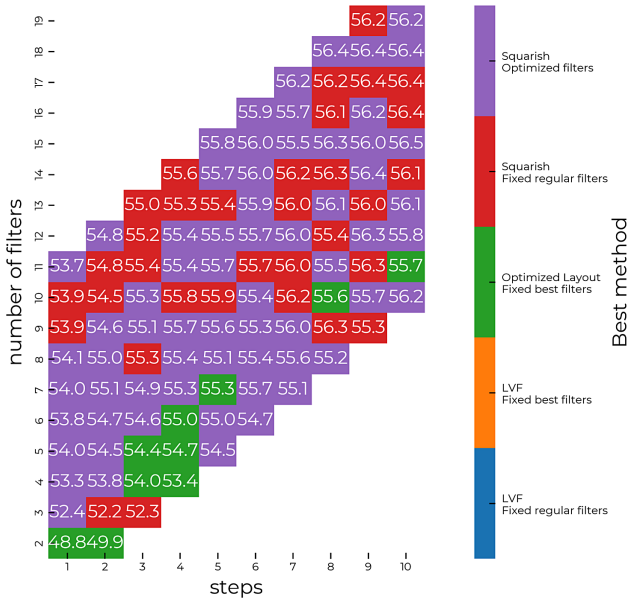

Fig. 9. Color-coded map to indicate what setup produces the most accurate results. In each block the best achievable PSNR is noted, with a color corresponding to which setup has achieved this.

images. This is directly related to compression in data rate and acquisition time, since data rate is the amount of data that has to be downlinked, or the sum of images, and acquisition time is the time it takes to make all the images.

Figure 10 shows the accuracy that can be achieved for a given fraction of the data rate of the original LVF setup. The reduction of the data rate is only related to the number of push-broom steps, not to the number of filters. Therefore, the y axis of this figure is the highest PSNR at each number of steps; a data rate reduction of 95% then corresponds to two steps, i.e., taking only two images. In Fig. 9, we can see that the highest possible PSNR for two push-broom steps (denoted on the x axis) occurs at seven filters (denoted on the y axis) by the squarish optimized filters (denoted by the color purple), which corresponds to the quoted PSNR in Fig. 10. As expected, higher compression leads to lower accuracy. With a compression by a factor of 40 (snapshot), the achievable PSNR is still 54.1.

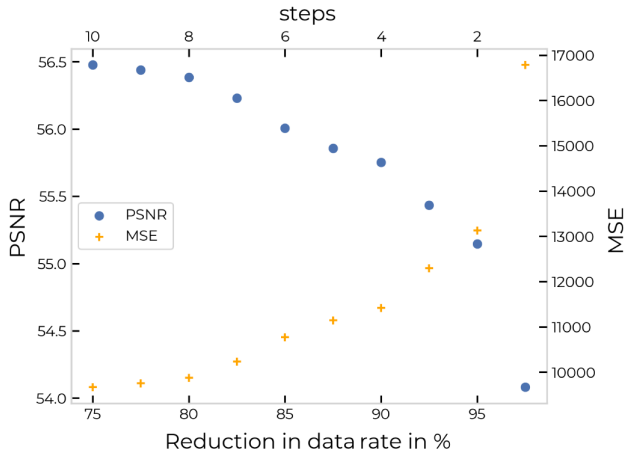

Fig. 10. Reduction in data rate that can be achieved compared to the classical LVF setup.

We could estimate the expected compression rate in Section 2.E.2, where we showed the power spectrum and the PCA results. The power spectrum showed spatial correlations. We could see that most of the power is concentrated in lower frequencies. At $z > 100$, the power has dropped by 4 orders of magnitude. Removing all the information at the $z > 100$ frequencies would therefore only reduce the accuracy by 1%. This corresponds with a compression of a factor 2 in both spatial directions.

From the PCA analysis, we expected seven filters to be enough to recover about 99% of the spectral information. Compared to the 40 original filters, we expected a possible compression rate of around 6 times in the spectral dimension.

Multiplying these expected compression rates, we expect to be able to compress the data by a factor of about 24. Figure 10 indeed indicates the largest drop-off in quality occurs between compression factors of 20 (95%) to a factor of 40 (97.5%).

In order to give a better understanding of the difference between a PSNR of 54.1 and one of 56.5, we have included Figs. 11 and 12. In these figures, we show the difference in spectra and spatial images. At this level, the difference between how well the reconstructions are done becomes hard to discern by eye and the use of the figures of merit over visual inspection becomes apparent.

4. DISCUSSION

The inSPECtor framework designs pixelated spectral filter layouts in the focal plane along with a linear reconstructor. The resulting instruments are expected to achieve high accuracy while substantially reducing the data rate and/or acquisition time.

The results noted in this paper are a proof of concept of this framework. In the actual use of this framework, it is highly advisable to make use of a more diverse training set than the single hyperspectral data cube used throughout this paper. The best results are expected to come from a training set that contains all the expected scenes in a balanced manner.

In some cases, the optimized configuration was outperformed by the equivalent nonoptimized filter arrangement. This would not be expected, since the static configurations are within the solution space of the optimizer. If the performance of the static cases is better than the performance of the optimized design, the latter has not converged to its optimal solution.

However, the data-driven optimization using gradient descent is not always able to converge to the global minimum of the problem, as can be seen by the comparisons between different optimization algorithms in [89]. With respect to our results, this means that the optimized design generally converges to a local minimum and that this minimum can be slightly worse than either other local minima or the global minimum. Which local minimum the network converges to depends on both the type of gradient descent algorithm and the initialization of the weights. When a static configuration outperforms the optimizable counterpart, the layout will be at or close to one of those better-performing local minima. However, these differences are no more than a PSNR difference of 0.9, or an MSE difference of a factor of 1.2.

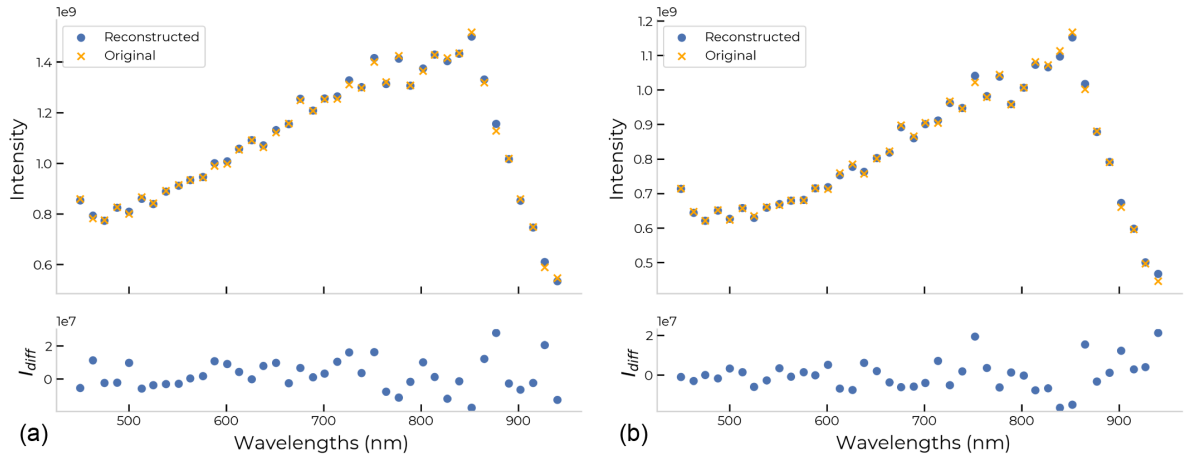

Fig. 11. Two comparisons of the original spectrum (orange crosses) with a retrieved spectrum (blue dots) are shown for two different PSNR levels. Both spectra come from the median performing data cubes from the test set after having been thrown into the design of the leftmost and rightmost point of Fig. 10. (a) PSNR of 54.1; (b) PSNR of 56.5.

When the amount of filters becomes large, the difference between the filters optimized by the optimal filters estimator (Section 2.B) and the regularly spaced filters also becomes small, and their performance becomes similar (at a maximum difference of 0.4 in PSNR). The optimal filters from the first component were estimated without regard to the spatial information, which could influence the best choice of filters.

What we show above is a comparison of the different results of our framework. Comparing with the results of other papers, we note a higher PSNR than previous joint design algorithms by Henz *et al*. [82] or Chakrabarti [81]. However, they focus on a different data product, an RGB image instead of a hyperspectral data cube. Jacome *et al*. [83] look at the hyperspectral retrieval as we do, but make use of an additional CASSI instrument. We could compare our result of taking a snapshot image with three filters to results from spectral recovery [79], where they start with a snapshot image made with three filters (RGB). However, this would only constitute a comparison of the best methods for spectral recovery on an actual camera with a basic linear reconstructor on a simulated detector. The linear reconstructor is something that can be interchanged, as will be mentioned in the future outlook section of the conclusion (Section 5), so this would not be an informative comparison.

It is important to note that the trained reconstructor for a given filter layout design is not the optimal reconstructor for the actual hardware. The reconstructor that comes with the design assumes a perfect Gaussian filter profile, a perfect match of the filters to the detector pixels, and an ideal performance of the detector at all wavelengths; all these assumptions will not hold in reality. The transmission profile for the filters will tend more towards a top-hat function, especially in the case of broader bandwidths. The match of the filters to the detector pixels is plagued by slight optical misalignment or other manufacturing errors. Finally, the detector has a wavelength-dependent efficiency and a nonzero background, which is not uniform over the whole array.

However, as long as all the optical and electrical components are still within a linear regime, where their response is linearly related to the number of photons entering the sensor, a linear reconstructor can still be used. When the response of the analog-to-digital converter, for instance, does not scale linearly with the infalling intensity of the light anymore, a linear reconstructor should not be expected to return accurate approximations of

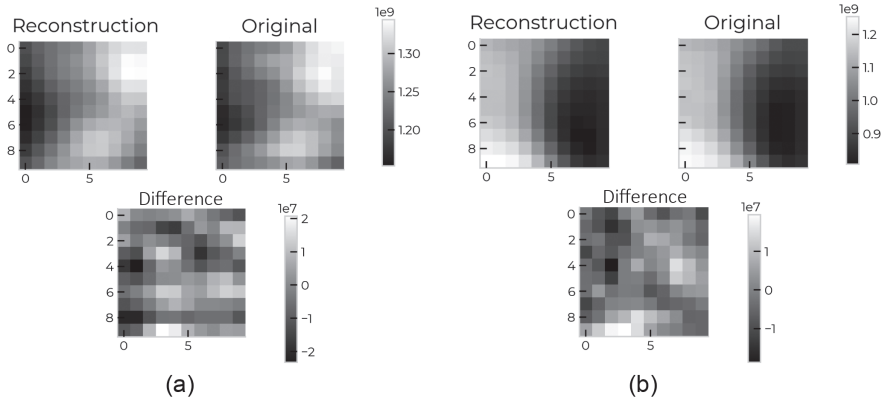

Fig. 12. Comparison of the 701 nm intensity images, where the reconstruction is to the left and the original to the right. The difference between the two is in the figure underneath. Both images come from the median performing data cubes from the test set after having been thrown into the design of the leftmost and rightmost point of Fig. 10. (a) PSNR of 54.1; (b) PSNR of 56.5.

the hyperspectral data cube. In the former case, the weights of the linear reconstructor, however, should still be relearned and cannot be copied from the reconstructor that came with the design. The relearning is done by feeding intensity images of calibrated/known sources created by the prototype into the linear reconstructor part of the optimal filter layout estimator and optimizing the weights for the reconstruction.

Finally, the goal of acquiring hyperspectral data goes beyond the acquisition of the data cube itself; instead, the final data product often requires further postprocessing like segmentation and classification [94–96]. One of the strengths of our design tool is that these postprocessing steps can be implemented right before the calculation of the loss function if they can be described in a differentiable manner. The loss function should then be modified to reflect the quality of the results after postprocessing.

5. CONCLUSION

In this paper, we describe a new tool for the design of spectral filter layouts on pixelated detectors as used for compressed, hyperspectral imaging. The resulting filter layout makes a partial measurement of the spectrum of every pixel. However, due to the simultaneously optimized linear reconstructor, this partial measurement can be used to reconstruct the full hyperspectral data cube with high accuracy. We show that the network can converge toward a filter layout that can recover known scenes with a snapshot image and high accuracy. This opens possibilities for extremely compact hyperspectral imagers with low data rates and short acquisition times.

As of now, InSPECtor does not yet do a full joint optimization, since the calculation of the optimal filters is still separate from the calculation of the optimal layout. This could be changed in future adaptations. Other possible additions include making the functional form of the spectral filters variable, so it can diverge from the Gaussian form it has now. A major change, carrying potentially big benefits, would be to replace the linear reconstructor with a nonlinear algorithm, such as a neural network or one of the algorithms mentioned in the introduction.

InSPECtor can be used to design a multitude of filter layouts for pixelated, hyperspectral imagers. The algorithm presented in this paper is easily adaptable to different scenes and applications, since it is a matter of optimizing InSPECtor with data comparable to the scene of interest. These scenes can range from remote sensing, to astronomy or defect inspection in factories. The framework could also be adapted to go further than hyperspectral imagers. Additional optical components, e.g., an array of linear polarizers [97], can be added before or after the existing Layers if they can be mathematically described in a differentiable and continuous manner. Finally, given known design constraints such as the desired data rate, minimally desired accuracy, and filter manufacturing constraints, the framework can return the optimal design of the filter layout.

Funding. Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO-TTW SYNOPTICS program).

Acknowledgment. This work was performed using the compute resources from the Academic Leiden Interdisciplinary Cluster Environment (ALICE) provided by Leiden University.

Disclosures. The authors declare that there are no conflicts of interest related to this paper.

Data availability. Data and code underlying the results presented in this paper are not publicly available at this time but may be obtained from the authors upon reasonable request.

REFERENCES

1. J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, *et al.*, "Hyperspectral unmixing overview: geometrical, statistical, and sparse regression-based approaches," IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. **5**, 354–379 (2012).
2. G. Vane, R. O. Green, T. G. Chrien, *et al.*, "The airborne visible/infrared imaging spectrometer (AVIRIS)," Remote Sens. Environ. **44**, 127–143 (1993).
3. W. L. Barnes, X. Xiong, and V. V. Salomonson, "Status of terra MODIS and aqua modis," Adv. Space Res. **32**, 2099–2106 (2003).
4. G. ElMasry and D.-W. Sun, "Chapter 1-Principles of hyperspectral imaging technology," in *Hyperspectral Imaging for Food Quality Analysis and Control*, D.-W. Sun, ed. (Academic, 2010), pp. 3–43.
5. A. A. Gowen, C. P. O'Donnell, P. J. Cullen, *et al.*, "Hyperspectral imaging–an emerging process analytical tool for food quality and safety control," Trends Food Sci. Technol. **18**, 590–598 (2007).
6. H. Liang, "Advances in multispectral and hyperspectral imaging for archaeology and art conservation," Appl. Phys. A **106**, 309–323 (2012).
7. E. K. Hege, D. O'Connell, W. Johnson, *et al.*, "Hyperspectral imaging for astronomy and space surveillance," Proc. SPIE **5159**, 380–391 (2004).
8. T. Adão, J. Hruška, L. Pádua, *et al.*, "Hyperspectral imaging: a review on UAV-based sensors, data processing and applications for agriculture and forestry," Remote Sens. **9**, 1110 (2017).
9. B. Fei, "Chapter 3.6-Hyperspectral imaging in medical applications," in *Data Handling in Science and Technology*, J. M. Amigo, ed., Vol. **32** of Hyperspectral Imaging (Elsevier, 2020), pp. 523–565.
10. S. Ortega, M. Halicek, H. Fabelo, *et al.*, "Hyperspectral and multispectral imaging in digital and computational pathology: a systematic review [Invited]," Biomed. Opt. Express **11**, 3195–3233 (2020).
11. X. Dong, G. Tong, X. Song, *et al.*, "DMD-based hyperspectral microscopy with flexible multiline parallel scanning," Microsyst. Nanoeng. **7**, 68 (2021).
12. M. T. Eismann, *Hyperspectral Remote Sensing* (SPIE, 2012).
13. Q. Li, X. He, Y. Wang, *et al.*, "Review of spectral imaging technology in biomedical engineering: achievements and challenges," J. Biomed. Opt. **18**, 100901 (2013).
14. S. L. Ustin and E. M. Middleton, "Current and near-term advances in earth observation for ecological applications," Ecol. Process. **10**, 1 (2021).
15. R. M. Willett, M. F. Duarte, M. A. Davenport, *et al.*, "Sparsity and structure in hyperspectral imaging: sensing, reconstruction, and target detection," IEEE Signal Process. Mag. **31**(1), 116–126 (2014).
16. D. Guzzi, G. Coluccia, D. Labate, *et al.*, "Optical compressive sensing technologies for space applications: instrumental concepts and performance analysis," in *International Conference on Space Optics—ICSO 2018* (2019).
17. W. Sun and Q. Du, "Hyperspectral band selection: a review," IEEE Geosci. Remote Sens. Mag. **7**, 118–139 (2019).
18. X. Cao, T. Yue, X. Lin, *et al.*, "Computational snapshot multispectral cameras: toward dynamic capture of the spectral world," IEEE Signal Process. Mag. **33**(5), 95–108 (2016).
19. G. Coluccia, C. Lastri, D. Guzzi, *et al.*, "Optical compressive imaging technologies for space big data," IEEE Trans. Big Data **6**, 430–442 (2020).
20. A. Barducci, D. Guzzi, C. Lastri, *et al.*, "Compressive sensing for hyperspectral earth observation from space," Proc. SPIE **10563**, 1056353 (2014).
21. T. Okamoto and I. Yamaguchi, "Simultaneous acquisition of spectral image information," Opt. Lett. **16**, 1277–1279 (1991).

22. A. Wagadarikar, R. John, R. Willett, *et al.*, "Single disperser design for coded aperture snapshot spectral imaging," Appl. Opt. **47**, B44–B51 (2008).
23. G. R. Arce, D. J. Brady, L. Carin, *et al.*, "Compressive coded aperture spectral imaging: an introduction," IEEE Signal Process. Mag. **31**(1), 105–115 (2014).
24. M. E. Gehm, R. John, D. J. Brady, *et al.*, "Single-shot compressive spectral imaging with a dual-disperser architecture," Opt. Express **15**, 14013 (2007).
25. Y. Wu, I. O. Mirza, G. R. Arce, *et al.*, "Development of a digital-micromirror-device-based multishot snapshot spectral imaging system," Opt. Lett. **36**, 2692–2694 (2011).
26. Y. August, C. Vachman, Y. Rivenson, *et al.*, "Compressive hyperspectral imaging by random separable projections in both the spatial and the spectral domains," Appl. Opt. **52**, D46–D54 (2013).
27. O. F. Kar and F. S. Oktem, "Compressive spectral imaging with diffractive lenses," Opt. Lett. **44**, 4582–4585 (2019).
28. K. Monakhova, K. Yanny, N. Aggarwal, *et al.*, "Spectral DiffuserCam: lensless snapshot hyperspectral imaging with a spectral filter array," Optica **7**, 1298–1307 (2020).
29. S. Jin, W. Hui, Y. Wang, *et al.*, "Hyperspectral imaging using the single-pixel Fourier transform technique," Sci. Rep. **7**, 45209 (2017).
30. J. A. Tropp and S. J. Wright, "Computational methods for sparse solution of linear inverse problems," Proc. IEEE **98**, 948–958 (2010).
31. L. Wang, K. Lu, and P. Liu, "Compressed sensing of a remote sensing image based on the priors of the reference image," IEEE Geosci. Remote Sens. Lett. **12**, 736–740 (2015).
32. Y. Yang, Y. Xie, X. Chen, *et al.*, "Hyperspectral snapshot compressive imaging with non-local spatial-spectral residual network," Remote Sens. **13**, 1812 (2021).
33. B. Gözcü, R. K. Mahabadi, Y.-H. Li, *et al.*, "Learning-based compressive MRI," IEEE Trans. Med. Imaging **37**, 1394–1406 (2018).
34. S. Wu, A. Dimakis, S. Sanghavi, *et al.*, "Learning a compressed sensing measurement matrix via gradient unrolling," in *Proceedings of the 36th International Conference on Machine Learning (PMLR)* (2019), pp. 6828–6839.
35. Y.-H. Li and V. Cevher, "Learning data triage: linear decoding works for compressive MRI," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2016), pp. 4034–4038.
36. L. Baldassarre, Y.-H. Li, J. Scarlett, *et al.*, "Learning-based compressive subsampling," IEEE J. Sel. Top. Signal Process. **10**, 809–822 (2016).
37. J. N. Mait, G. W. Euliss, and R. A. Athale, "Computational imaging," Adv. Opt. Photon. **10**, 409 (2018).
38. L. Gao, Y. Qu, L. Wang, *et al.*, "Computational spectrometers enabled by nanophotonics and deep learning," Nanophotonics **11**, 2507–2529 (2022).
39. H. Arguello, J. Bacca, H. Kariyawasam, *et al.*, "Deep optical coding design in computational imaging," arXiv, arXiv:2207.00164 (2022).
40. L. Huang, R. Luo, X. Liu, *et al.*, "Spectral imaging with deep learning," Light Sci. Appl. **11**, 61 (2022).
41. J. Bacca, E. Martinez, and H. Arguello, "Computational spectral imaging: a contemporary overview," J. Opt. Soc. Am. A **40**, C115–C125 (2023).
42. L. Wang, T. Zhang, Y. Fu, *et al.*, "HyperReconNet: joint coded aperture optimization and image reconstruction for compressive hyperspectral imaging," IEEE Trans. Image Process. **28**, 2257–2270 (2019).
43. E. S. Kaur and V. K. Banga, "A survey of demosaicing: issues and challenges," Int. J. Sci. Eng. Technol. **2**, 9–17 (2015).
44. B. E. Bayer, "Color imaging array," U.S. patent 3,971,065 (20 July 1976).
45. M. Gharbi, G. Chaurasia, S. Paris, *et al.*, "Deep joint demosaicking and denoising," ACM Trans. Graph. **35**, 1–12 (2016).
46. K. Cui, Z. Jin, and E. Steinbach, "Color image demosaicking using a 3-stage convolutional neural network structure," in *25th IEEE International Conference on Image Processing (ICIP)* (IEEE, 2018), pp. 2177–2181.
47. S. Guo, Z. Liang, and L. Zhang, "Joint denoising and demosaicking with green channel prior for real-world burst images," IEEE Trans. Image Process. **30**, 6930–6942 (2021).
48. F.-L. He, Y.-C. F. Wang, and K.-L. Hua, "Self-learning approach to color demosaicking via support vector regression," in *19th IEEE International Conference on Image Processing* (2012), pp. 2765–2768.
49. T. Heinze, M. von Löwis, and A. Polze, "Joint multi-frame demosaicing and super-resolution with artificial neural networks," in *19th International Conference on Systems, Signals and Image Processing (IWSSIP)* (2012), pp. 540–543.
50. T. Iriyama, M. Sato, H. Aomori, *et al.*, "Deep demosaicking considering inter-channel correlation and self-similarity," Nonlinear Theory Appl. IEICE **12**, 453–463 (2021).
51. Q. Jin, G. Facciolo, and J. M. Morel, "A review of an old dilemma: demosaicking first, or denoising first?" *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2020, pp. 2169–2179.
52. D. Kiku, Y. Monno, M. Tanaka, *et al.*, "Beyond color difference: residual interpolation for color image demosaicking," IEEE Trans. Image Process. **25**, 1288–1300 (2016).
53. D. Menon and G. Calvagno, "Color image demosaicking: an overview," Signal Process. Image Commun. **26**, 518–533 (2011).
54. S. M. A. Sharif, R. Ali Naqvi, and M. Biswas, "Beyond joint demosaicking and denoising: an image processing pipeline for a pixel-bin image sensor," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (IEEE, 2021), pp. 233–242.
55. Y.-Q. Wang, "A multilayer neural network for image demosaicking," in *IEEE International Conference on Image Processing (ICIP)* (2014), pp. 1852–1856.
56. X. Wu, "Color demosaicking by local directional interpolation and nonlocal adaptive thresholding," J. Electron. Imaging **20**, 023016 (2011).
57. J. Zhang, J. Shao, H. Luo, *et al.*, "Learning a convolutional demosaicing network for microgrid polarimeter imagery," Opt. Lett. **43**, 4534–4537 (2018).
58. T. A. Habtegebrial, G. Reis, and D. Stricker, "Deep convolutional networks for snapshot hypercpectral demosaicking," in *10th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)* (2019), pp. 1–5.
59. K. Dijkstra, J. van de Loosdrecht, L. R. Schomaker, *et al.*, "Hyperspectral demosaicking and crosstalk correction using deep learning," Mach. Vis. Appl. **30**, 1–21 (2019).
60. P. Li, M. Ebner, P. Noonan, *et al.*, "Deep learning approach for hyperspectral image demosaicking, spectral correction and high-resolution RGB reconstruction," in *MICCAI Workshop on Augmented Environments for Computer-Assisted Interventions, Computer Assisted and Robotic Endoscopy, and Context Aware Operating Theaters* (2022), p. 12.
61. X. Wang, J.-B. Thomas, J. Y. Hardeberg, *et al.*, "Discrete wavelet transform based multispectral filter array demosaicking," in *Colour and Visual Computing Symposium (CVCS)* (2013), pp. 1–6.
62. L. Zhuang, M. K. Ng, X. Fu, *et al.*, "Hy-demosaicing: hyperspectral blind reconstruction from spectral subsampling," IEEE Trans. Geosci. Remote Sens. **60**, 5515815 (2022).
63. G. Tsagkatakis, M. Bloemen, B. Geelen, *et al.*, "Graph and rank regularized matrix recovery for snapshot spectral image demosaicing," IEEE Trans. Comput. Imaging **5**, 301–316 (2019).
64. S. Mihoubi, O. Losson, B. Mathon, *et al.*, "Multispectral demosaicing using pseudo-panchromatic image," IEEE Trans. Comput. Imaging **3**, 982–995 (2017).
65. P. Amba, J. B. Thomas, and D. Alleysson, "N-LMMSE demosaicing for spectral filter arrays," J. Imaging Sci. Technol. **61**, 40407-1–40407-11 (2017).
66. B. Arad, R. Timofte, R. Yahel, *et al.*, "NTIRE 2022 spectral demosaicing challenge and data set," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 882–896.
67. R. Lukac and K. Plataniotis, "Color filter arrays: design and performance analysis," IEEE Trans. Consum. Electron. **51**, 1260–1267 (2005).
68. K. Hirakawa and P. Wolfe, "Spatio-spectral color filter array design for optimal image recovery," IEEE Trans. Image Process. **17**, 1876–1890 (2008).

69. J. Li, C. Bai, Z. Lin, *et al.*, "Optimized color filter arrays for sparse representation-based demosaicking," IEEE Trans. Image Process. **26**, 2381–2393 (2017).
70. L. Miao and H. Qi, "The design and evaluation of a generic method for generating mosaicked multispectral filter arrays," IEEE Trans. Image Process. **15**, 2780–2791 (2006).
71. Y. Li, A. Majumder, H. Zhang, *et al.*, "Optimized multi-spectral filter array based imaging of natural scenes," Sensors **18**, 1172 (2018).
72. P. J. Lapray, X. Wang, J. B. Thomas, *et al.*, "Multispectral filter arrays: recent advances and practical implementation," Sensors-Switzerland **14**, 21626–21659 (2014).
73. S. Saxe, L. Sun, V. Smith, *et al.*, "Advances in miniaturized spectral sensors," Proc. SPIE **10657**, 106570B (2018).
74. J. Pichette, W. Charle, and A. Lambrechts, "Fast and compact internal scanning CMOS-based hyperspectral camera: the snapscan," Proc. SPIE **10110**, 1011014 (2017).
75. S. Lemmens, T. Van Craenendonck, J. Van Eijgen, *et al.*, "Combination of snapshot hyperspectral retinal imaging and optical coherence tomography to identify alzheimer's disease patients," Alz. Res. Ther. **12**, 144 (2020).
76. H. Cheng, C. Deng, Y. Li, *et al.*, "An on-line color defect detection method for printed matter based on snapshot multispectral camera," Proc. SPIE **10816**, 1081612 (2018).
77. N. Hagen and M. W. Kudenov, "Review of snapshot spectral imaging technologies," Opt. Eng. **52**, 090901 (2013).
78. H. Fu, L. Bian, X. Cao, *et al.*, "Hyperspectral imaging from a raw mosaic image with end-to-end learning," Opt. Express **28**, 314–324 (2020).
79. B. Arad, R. Timofte, R. Yahel, *et al.*, "NTIRE 2022 spectral recovery challenge and data set," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 863–881.
80. C. Tao, H. Zhu, X. Wang, *et al.*, "Compressive single-pixel hyperspectral imaging using RGB sensors," Opt. Express **29**, 11207–11220 (2021).
81. A. Chakrabarti, "Learning sensor multiplexing design through back-propagation," in *Advances in Neural Information Processing Systems* (2016), pp. 3089–3097.
82. B. Henz, E. S. Gastal, and M. M. Oliveira, "Deep joint design of color filter arrays and demosaicing," Comput. Graph. Forum **37**, 389–399 (2018).
83. R. Jacome, J. Bacca, and H. Arguello, "D2uf: Deep coded aperture design and unrolling algorithm for compressive spectral image fusion," IEEE J. Sel. Top. Signal Process. **17**, 502–512 (2022).
84. W. Zhang, H. Song, X. He, *et al.*, "Deeply learned broadband encoding stochastic hyperspectral imaging," Light: Sci. Appl. **10**, 108 (2021).
85. H. Song, Y. Ma, Y. Han, *et al.*, "Deep-learned broadband encoding stochastic filters for computational spectroscopic instruments," Adv. Theory Simul. **4**, 2000299 (2021).
86. K. Li, D. Dai, and L. Van Gool, "Jointly learning band selection and filter array design for hyperspectral imaging," in *Winter Conference on Applications of Computer Vision*, (2023), pp. 6373–6383.
87. A. Horé and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *20th International Conference on Pattern Recognition* (2010), pp. 2366–2369.
88. M. Abadi, P. Barham, J. Chen, *et al.*, "TensorFlow: a system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation* (2016), p. 21.
89. D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization* (2017).
90. M. Esposito, S. S. Conticello, M. Pastena, *et al.*, "In-orbit demonstration of artificial intelligence applied to hyperspectral and thermal sensing from space," Proc. SPIE **11131**, 111310C (2019).
91. S. H. Simon, *The Oxford Solid State Basics*, 1st ed. (Oxford University, 2013).
92. B. Acharya and M. Gill, "On the index of gracefulness of a graph and the gracefulness of two-dimensional square lattice graphs," Indian J. Math **23**, 14 (1981).
93. M. Esposito and A. Z. Marchi, "In-orbit demonstration of the first hyperspectral imager for nanosatellites," Proc. SPIE **11180**, 1118020 (2019).
94. N. Audebert, B. Le Saux, and S. Lefevre, "Deep learning for classification of hyperspectral data: a comparative review," IEEE Geosci. Remote Sens. Mag. **7**(2), 159–173 (2019).
95. M. Imani and H. Ghassemian, "An overview on spectral and spatial information fusion for hyperspectral image classification: current trends and challenges," Inf. Fusion **59**, 59–83 (2020).
96. B. Rasti, D. Hong, R. Hang, *et al.*, "Feature extraction for hyperspectral imagery: the evolution from shallow to deep: overview and toolbox," IEEE Geosci. Remote Sens. Mag. **8**(4), 60–88 (2020).
97. T. Stockmans, F. Snik, M. Smit, *et al.*, "End-to-end design framework for compressed on-chip pixel-wise spectro-polarimeters," Proc. SPIE **12236**, 122360E (2022).