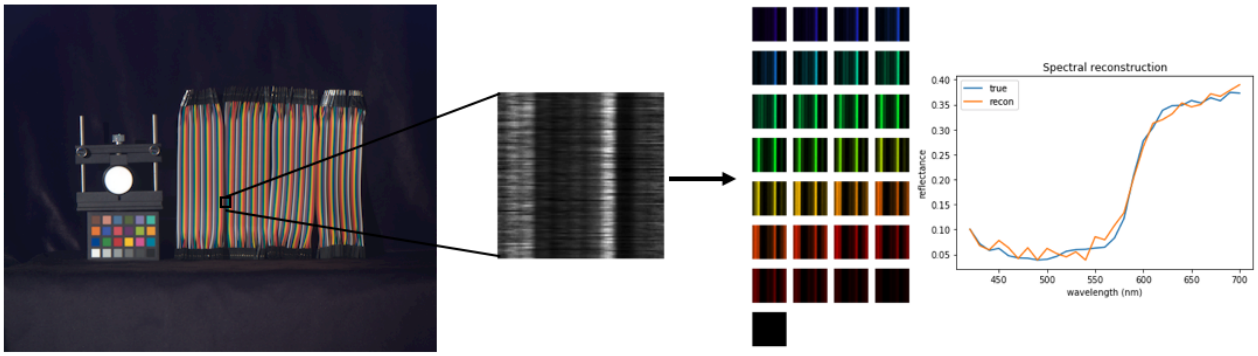
Compressive Hyperspectral Image Reconstruction with a Deep Generative Prior

Mark Nishimura
Stanford University
Stanford, CA
markn1@stanford.edu



Abstract

Hyperspectal images are 3D data cubes that contain spectral information of a scene in addition to its spectral content. Compressive hyperspectral imaging is a technique for taking a small number of linear measurements of this datacube and using techniques from compressed sensing to reconstruct the full data cube. In this work, based on the work of Choi et. al. 2017, we use a convolutional autoencoder to learn a deep generative model that can serve as a prior for the CASSI compressive hyperspectral imaging technique. We evaluate our technique on simulated data and show that it achieves essentially state-of-the-art performance on this task.

1. Introduction

Hyperspectral images are data cubes that capture the spectral content in addition to the spatial content, with height, width, and wavelength as their their dimensions. The information they contain is richer than that of RGB images, and the extra spectral information has applications in archaeology [8] and environmental monitoring [10]. Capturing hyperspectral images, however, poses a significant challenge, since each wavelength needs to be captured separately, which can be long and costly and limits the practicality of this technique. Compressive hyperspectral imaging solves this problem by capturing a small set of linear measurements of the data cube and using compressed sensing to recover the full data cube, exploiting the redundancy across spectral channels of natural images. In this paper, we present a compressive hyperspectral imaging reconstruction pipeline that uses ADMM and a prior constructed from a type of deep neural network called a convolutional autoencoder.

2. Related work

This work is based off of the work done by Choi et.al. in their 2017 paper "High-quality hyperspectral reconstruction using a spectral prior."[5] Other related work involving deep compressive sensing has been done by [2]. Other hyperspectral reconstruction techniques include Approximate Message Passing (AMP-3D-Wiener) [11], Two-step Iterative Shrinkage/Thresholding TwIST [1], and Gradient Projection for Sparse Reconstruction (a faster method for solving the basis pursuit problem) [6]. CASSI itself was developed by Wagadrikar et. al. [12]. Furthermore, an improved version of CASSI using a Digital Micromirror Device (DMD) was developed in [9]. The DMD allows greater control and flexibility over the coding of the different spectral channels (as opposed to CASSI, which codes each channel with a sheared version of a fixed coded aperture pattern), resulting in improved reconstruction.

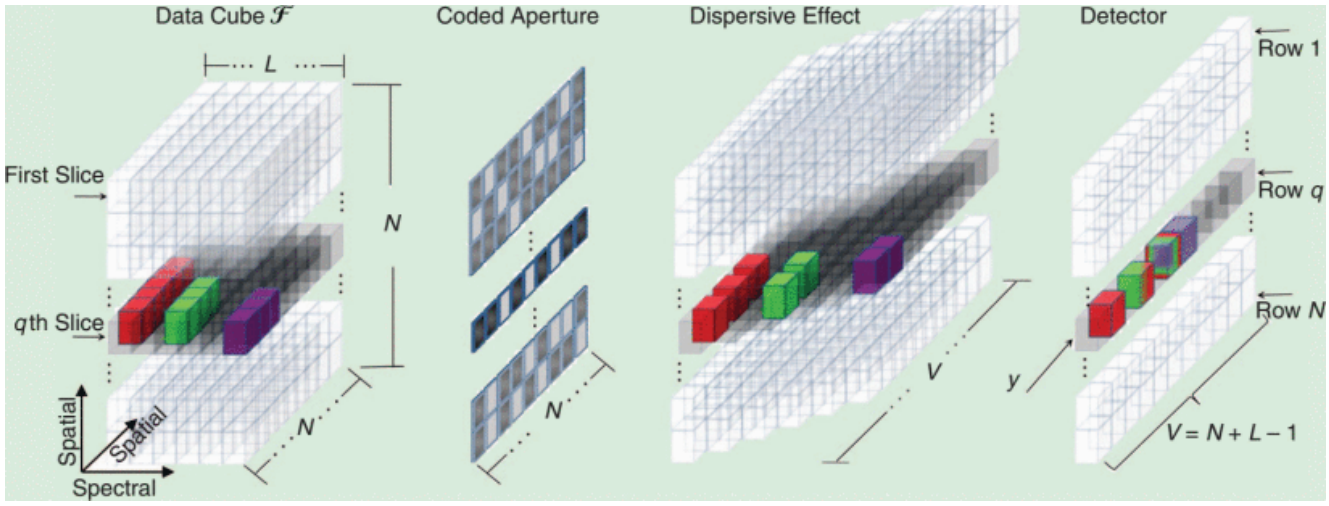


Figure 1: CASSI

3. Compressive Hyperspectral Imaging

There are various technique for compressive hyperspectral imaging, but here we will model our compressive image acquisition after a technique called spatial-spectral CASSI (Coded Aperture Snapshot Spectral Imaging). In this modality, light first passes through a dispersive element (i.e. a prism) which separates the wavelengths of light, essentially shearing the hyperspectral datacube in the x-direction. The light then passes through a coded aperture, which selectively blocks and admits the light according to its spatial location. Finally, the light is unsheared with another prism before contacting the sensor and producing a compressive hyperspectral image in which each pixel value is the integral over the wavelengths and spatial components of the datacube (as determined by the configuration of the prism and the coded aperture). Thus, given a datacube $F(x, y, \lambda)$, a the prism which acts in the x-axis bending light by a function $\phi(\lambda)$, and an aperture which modulates the light with the 2D function $T(x, y)$, the image formation process can be described with the following integral:

$$i(x, y) = \int_{\lambda} T(x - \phi(\lambda), y) F(x, y, \lambda) d\lambda$$

The image is then discretized on the sensor. Representing our process using pixels, we can therefore model our image formation process using the linear equation:

$$i = \Phi F$$

where Φ is the sensing matrix that captures the above linear measurement procedure. Thus, reconstructing F from i reduces to solving an underdetermined system of linear equations, setting the stage for our compressive sensing reconstruction technique.

4. Convolutional Autoencoders

Recent advances in computation and the widespread abundance of image data have enabled many new image-related applications of deep learning In particular, convolutional neural networks have achieved state-of-the-art performance on various image processing and classification tasks. The convolutional autoencoder is a particular neural network architecture which is designed to learn useful representations of data in an unsupervised fashion. The architecture is the concatenation of two subnetworks. The first network, the **Encoder**, takes as input a hyperspectral data cube, passes it through a series of convolutional layers and nonlinear activations, and outputs a data cube representing the encoded version of the input. The second network, the **Decoder**, takes a data cube of the same size as the output of the Encoder, passes it through another series of convolutional layers and nonlinear activations, and outputs a data cube of the same size as the original hyperspectral data cube.

As with many types of neural networks, arriving at clear and precise explanations for their behavior is a difficult task at best. However, as we will demonstrate, our model does quite well in practice.The basic idea behind this technique is that the Decoder and Encoder force each other to learn a transform from the image domain to some latent domain represented by the data block α. In some sense, α can be interpreted as a vector in a latent space that encodes the redundancy in hyperspectral images in some fashion. We then exploit this redundancy in our ADMM-based image reconstruction.

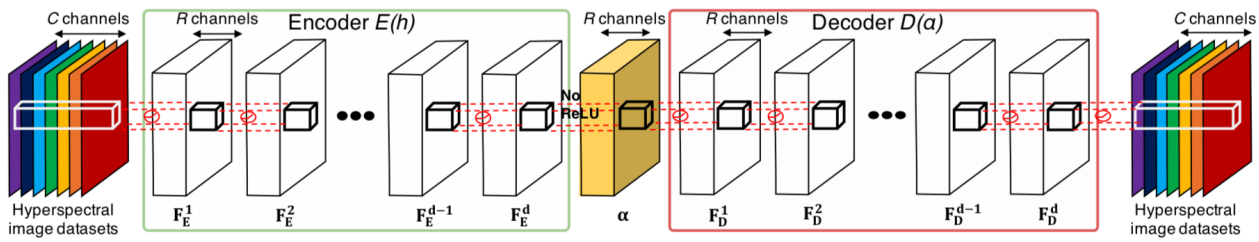


Figure 2: Convolutional Autoencoder (figure from [5])

4.1. Autoencoder Implementation Details

4.1.1 Autoencoder Data

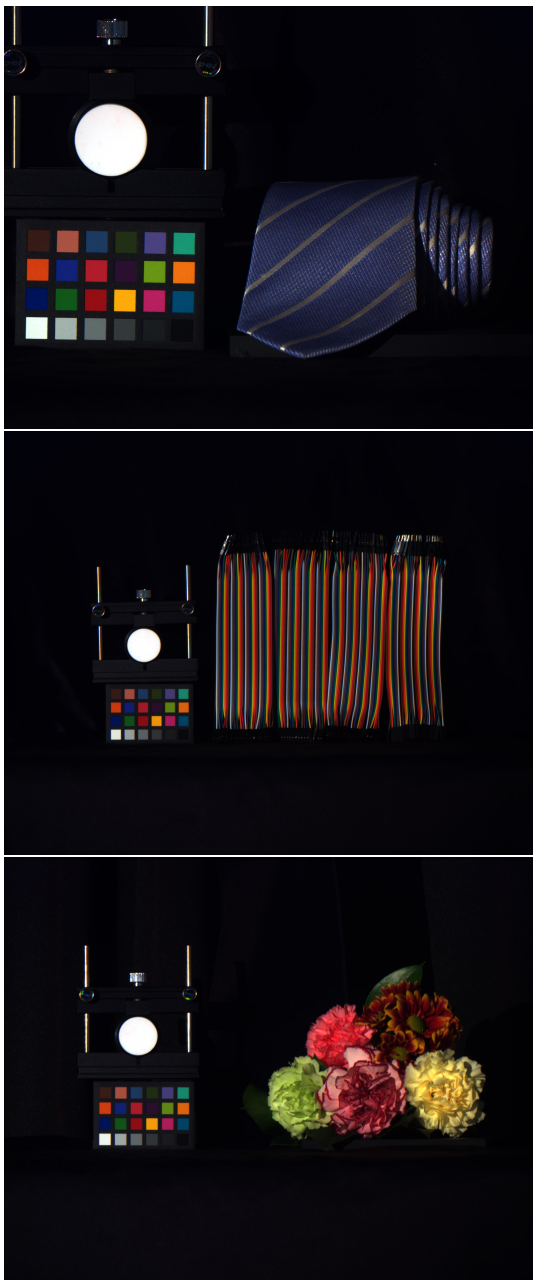


Figure 3: Sample photos from the KAIST dataset

We downloaded all 30 images from the KAIST Dataset of Hyperspectral Reflectance Images [5] containing channels from 400nm to 720nm in 10nm increments. We also downloaded the CAVE Multispectral Image Database [13], which contains 32 images with channels from 400nm to 700nm in 10nm increments. We extracted the data cubes from 420nm to 700nm (there was originally going to be a third dataset going from 420nm to 720nm). We performed data augmentation, adding to the image set versions of the original images that had been scaled up by a factor of 2 and other versions that had scaled down by a factor of 2. We also performed random flipping/mirroring across the vertical axis. We split the resulting images 8:1:1 into training, validation, and test sets, then subsampled data patches of size 96x96x29 from these images. Ultimately, our training set consisted of 22140 patches.

4.1.2 Autoencoder Hyperparameters

As advised by Choi et. al., our autoencoder uses 11 convolutional layers for the Encoder and 11 convolutional layers for the Decoder. All convolutional layers have 64 convolutional filters of size 3 and a zero padding of 1. We apply ReLU activation functions on the outputs of all the layers, except for the layer whose output is the output of the Encoder - this is so that we don't need to constrain α to be nonnegative later on in the reconstruction pipeline. We trained the autoencoder for 60 epochs with batch size 64 using the Adam optimizer [7] with learning rate `1e-5`, decaying the learning rate to `1e-6` at epoch 30. Our loss function was the norm-squared of the difference between the input and output, plus L2-regularization on the weights with $\lambda =$ `1e-8`:

$$\text{Loss} = \|D(E(i)) - i\|_2^2 + \lambda \sum_{i=1}^{11} \left(\|W_E^i\|_2^2 + \|W_D^i\|^2 \right)$$

The autoencoder was implemented in PyTorch and trained on a Pascal 1080 Ti GPU, taking roughly 14 hours to train completely.

5. ADMM Reconstruction

The Alternating-Direction Method of Multipliers [3] (ADMM) is an optimization technique for solving problems of the form:

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array} \tag{1}$$

with optimization variables x and z, functions f and g, and constants A, B, and c. After appropriate initialization of x, z, and a dual variable u, the problem can then be solved iteratively via the update rules:

$$x^{(k+1)} = \arg\min_x \left(f(x) + \frac{\rho}{2}\|Ax + Bz^{(k)} - c + u^{(k)}\|_2^2 \right)$$
$$z^{(k+1)} = \arg\min_z \left(g(z) + \frac{\rho}{2}\|Ax^{(k+1)} + Bz - c + u^{(k)}\|_2^2 \right)$$
$$u^{(k+1)} = u^{(k)} + Ax^{(k+1)} + Bz^{(k+1)} - c$$

We would like to use this optimization paradigm to reconstruct our image. Specifically, our goal will be to recover the latent variable α via ADMM

Our optimization problem is as follows:

$$\begin{aligned} \text{minimize}_\alpha \quad & \|i - \Phi D(\alpha)\|_2^2 + \tau_1\|\alpha - E(D(\alpha))\|_2^2 \\ & + \tau_2\|\nabla_{xy} D(\alpha)\|_1 \end{aligned}$$

This objective can be broken down into the sum of a data fidelity term $\|i - \Phi D(\alpha)\|_2^2$ and two prior terms. The first prior is based on the quality of of the autoencoder $\|\alpha - E(D(\alpha))\|_2^2$ and the second is an anisotropic total variation prior on the reconstructed hyperspectral data cube $D(\alpha)$. Now setting $z = \nabla_{xy} D(\alpha)$ and

$$f(\alpha) = \|i - \Phi D(\alpha)\|_2^2 + \tau_1\|\alpha - E(D(\alpha))\|_2^2$$
$$g(z) = \tau_2\|z\|_1$$

we can express our problem approximately in the ADMM framework as:

$$\begin{array}{ll} \text{minimize} & \|i - \Phi D(\alpha)\|_2^2 + \tau_1\|\alpha - E(D(\alpha))\|_2^2 + \tau_2\|z\|_1 \\ \text{subject to} & \nabla_{xy} D(\alpha) - z = 0 \end{array}$$

From which we can derive the ADMM updates:

$$\alpha^{(k+1)} = \arg\min_\alpha \left(f(\alpha) + \frac{\rho}{2}\|\nabla_{xy} D(\alpha) - z^{(k)} + u^{(k)}\|_2^2 \right)$$
$$\begin{aligned} z^{(k+1)} &= \arg\min_z \left(\tau_2\|z\|_1 + \frac{\rho}{2}\|\nabla_{xy} D(\alpha^{(k+1)}) - z + u^{(k)}\|_2^2 \right) \\ &= S_{\tau_2/\rho}\left(\nabla_{xy} D(\alpha^{(k+1)}) + u^{(k)} \right) \end{aligned}$$
$$u^{(k+1)} = u^{(k)} + \nabla_{xy} D(\alpha^{(k+1)}) - z^{(k+1)}$$

Where $S_{\tau/\rho}$ is the element-wise soft thresholding operator, defined as:

$$S_\beta(v) = \begin{cases} v - \beta & v - \beta > 0 \\ v + \beta & v + \beta < 0 \\ 0 & \text{otherwise} \end{cases}$$

5.1. ADMM Implementation Details

For each input ADMM was run for 10 iterations with parameter $\rho = 0.1$. The minimization for α was performed with 200 iterations of ADAM with learning rate 0.05. The regularization parameters were set with $\tau_1 = 0.1$ and $\tau_2 = 0.001$.

6. Results

We evaluated our autoencoder-aided ADMM reconstruction on simulated data. We started by simulating the spatial-spectral CASSI image acquisition by generating a 96x96 0/1/ binary mask, then expanding it and shearing it to be 96x96x29. For each input image, we multiplied elementwise by this binary mask block, then summed along the spectral dimension to produce our input image i.

We also attempted to reconstruct our image by using the following naive optimization reconstruction method:

$$\min_\alpha \quad \|i - \Phi(x)\|_2^2 + \tau\|\nabla_\lambda x\|_2^2 + \lambda\|\text{DCT}(x)\|_1$$

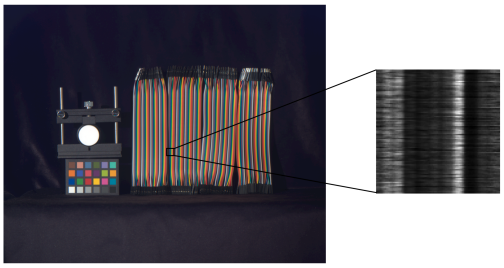
This method is a generic compressive sensing technique with two priors. The first, $\tau\|\nabla_\lambda x\|_2^2$, is a smoothness prior in the spectral dimension. This is the crucial prior - it allows us to take full advantage of both the Encoder and Decoder (and, by extension, all of the training data we fed in to them) during our reconstruction. The second, $\lambda\|\text{DCT}(x)\|_1$, encourages sparsity in the channel-wise 2-dimensional DCT domain (i.e. after taking the DCT independently on all channels of the hyperspectral image, we should get a sparse data cube). We solved this problem with ADMM with $\tau = 1$ and $\lambda = 1e-3$ and using the following updates:

$$\begin{aligned} x^{(k+1)} &= \arg\min_\alpha \|i - \Phi(x)\|_2^2 + \tau\|\nabla_\lambda x\|_2^2 \\ &\quad + \frac{\rho}{2}\|\text{DCT}(x) - z^{(k)} + u^{(k)}\|_2^2 \\ z^{(k+1)} &= \arg\min_z \left(\lambda\|z\|_1 + \frac{\rho}{2}\|\text{DCT}(x^{(k+1)}) - z + u^{(k)}\|_2^2 \right) \\ &= S_{\lambda/\rho}\left(\text{DCT}(x^{(k+1)}) + u^{(k)} \right) \\ u^{(k+1)} &= u^{(k)} + \text{DCT}(x^{(k+1)}) - z^{(k+1)} \end{aligned}$$

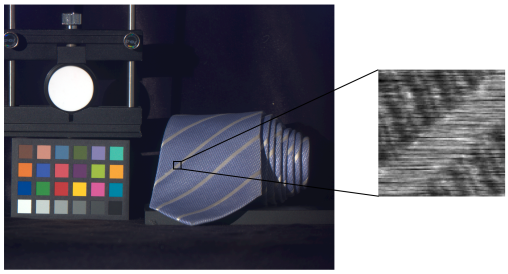
We solved the first update using Adam again. The number of ADMM iterations was 10, and the number of Adam iterations was 200.

We show selected results on the following three patches:

(a) Patch 1



(b) Patch 2



(c) Patch 3

Qualitatively, we observe that the naive method is able to make significant progress, but struggles to understand the underlying pattern in the spectral content of the scene. This is possibly because simple smoothness of gradients in the spectral domain is not a strong enough prior to enable the reconstruction of the spectrum. It seems that the autoencoder has indeed learned a stronger prior on the spectral content of the scene.

7. Conclusions and Future Work

We successfully implemented and validated a compressive hyperspectral image reconstruction pipeline using a convolutional autoencoder. Future avenues of inquiry might include

- Using a GAN (generative adversarial network) or a more advanced variant of autoencoder (e.g. the Importance Weighted Autoencoder [4])instead.
- Applying this technique to other settings where compressed sensing is useful. For example, MRI, compressive light field imaging, etc.
- Leveraging this new application to understand the structure and/or function of autoencoders. For example, investigating what is being learned at each layer of the autoencoder.

7.1. Code Availability

The code for this project is available (in the form of several jupyter notebooks) at `https://github.com/nishi951/hyperspectral-cs`. Setup instructions can be found there.

References

[1] J. M. Bioucas-Dias and M. A. T. Figueiredo. A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration. *IEEE Transactions on Image Processing*, 16(12):2992–3004, Dec 2007.

[2] A. Bora, A. Jalal, E. Price, and A. G. Dimakis. Compressed Sensing using Generative Models. *ArXiv e-prints*, Mar. 2017.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, Jan. 2011.

[4] Y. Burda, R. B. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. *CoRR*, abs/1509.00519, 2015.

[5] I. Choi, D. S. Jeon, G. Nam, D. Gutierrez, and M. H. Kim. High-quality hyperspectral reconstruction using a spectral prior. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2017)*, 36(6):218:1–13, 2017.

[6] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, Dec 2007.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[8] M. Kubik. Chapter 5 hyperspectral imaging: A new technique for the non-invasive study of artworks. In D. Creagh and D. Bradley, editors, *Physical Techniques in the study of Art, Archaeology and Cultural Heritage*, volume 2 of *Physical Techniques in the Study of Art, Archaeology and Cultural Heritage*, pages 199 – 259. Elsevier, 2007.

[9] X. Lin, G. Wetzstein, Y. Liu, and Q. Dai. Dual-coded compressive hyperspectral imaging. *Opt. Lett.*, 39(7):2044–2047, Apr 2014.

[10] M. Moroni, E. Lupo, E. Marra, and A. Cenedese. Hyperspectral image analysis in environmental monitoring: Setup of a new tunable filter platform. *Procedia Environmental Sciences*, 19:885 – 894, 2013. Four Decades of Progress in Monitoring and Modeling of Processes in the Soil-Plant-Atmosphere System: Applications and Challenges.

[11] J. Tan, Y. Ma, H. Rueda, D. Baron, and G. R. Arce. Compressive hyperspectral imaging via approximate message passing. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):389–401, March 2016.

[12] A. Wagadarikar, R. John, R. Willett, and D. Brady. Single disperser design for coded aperture snapshot spectral imaging. *Appl. Opt.*, 47(10):B44–B51, Apr 2008.

[13] F. Yasuma, T. Mitsunaga, D. Iso, and S. K. Nayar. Generalized assorted pixel camera: Postcapture control of resolution, dynamic range, and spectrum. *IEEE Transactions on Image Processing*, 19(9):2241–2253, Sept 2010.

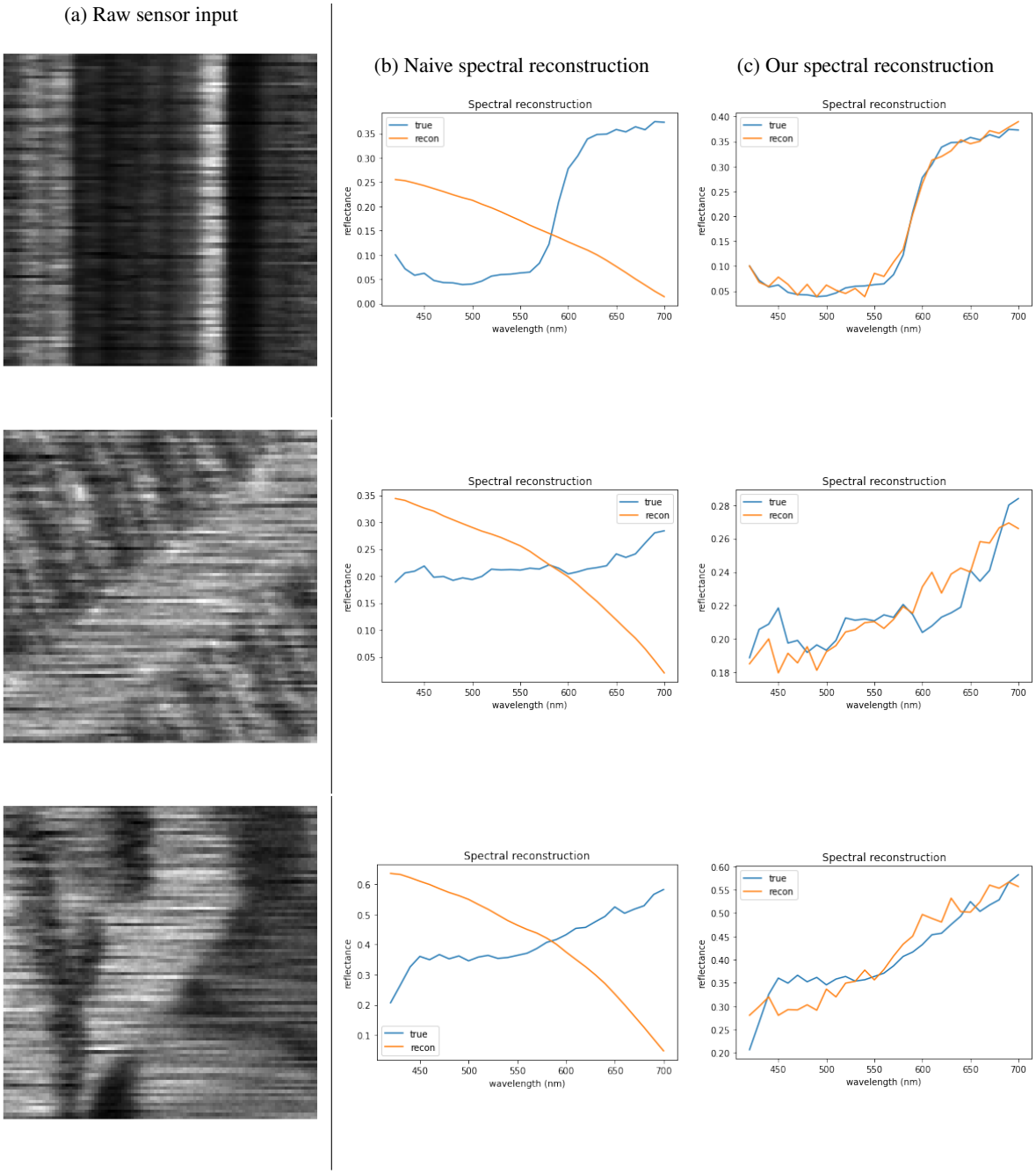
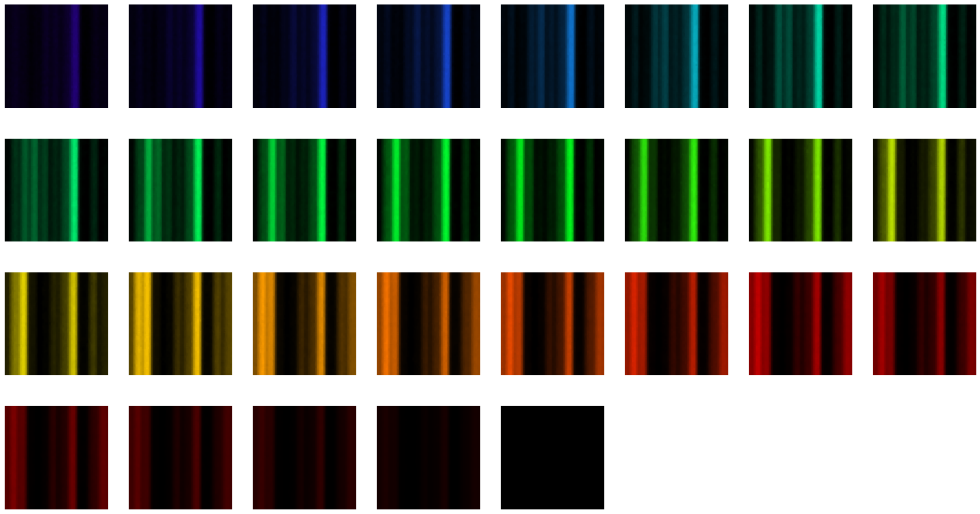
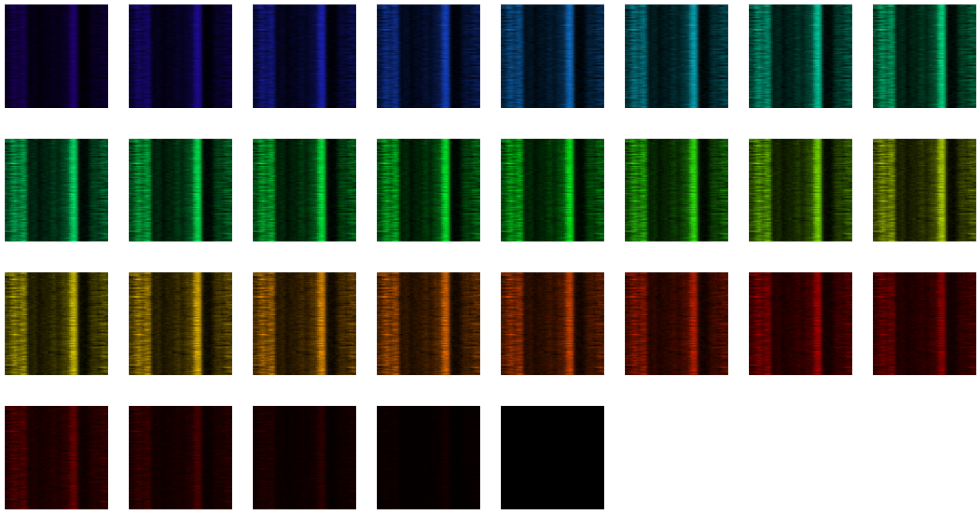


Figure 5: Input images and their reconstructed spectra (for a single pixel)

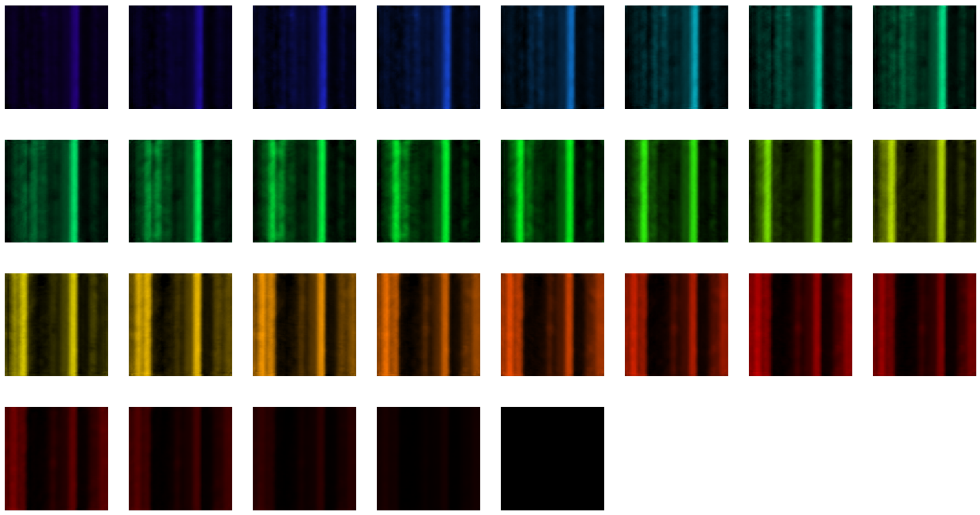
(a) Patch 1: Ground truth



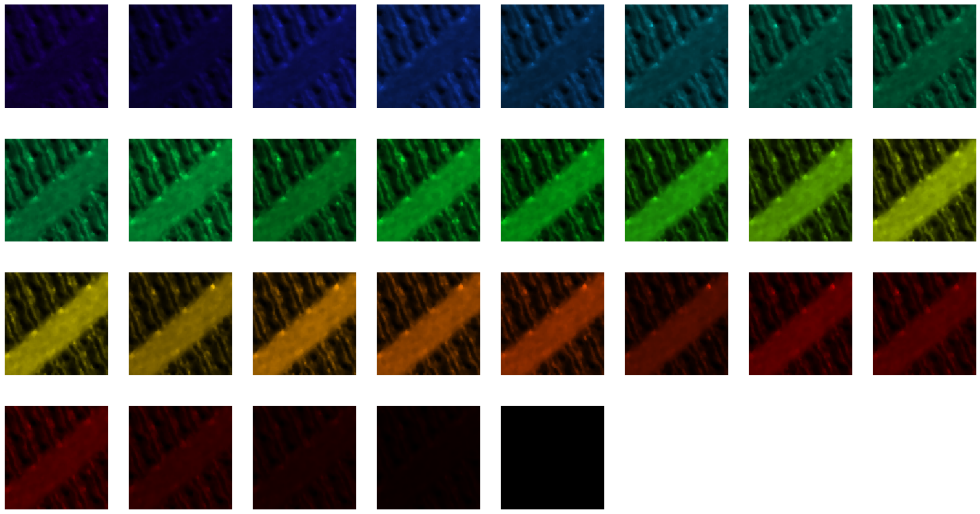
(b) Patch 1: Naive Reconstruction. PSNR = 17.09



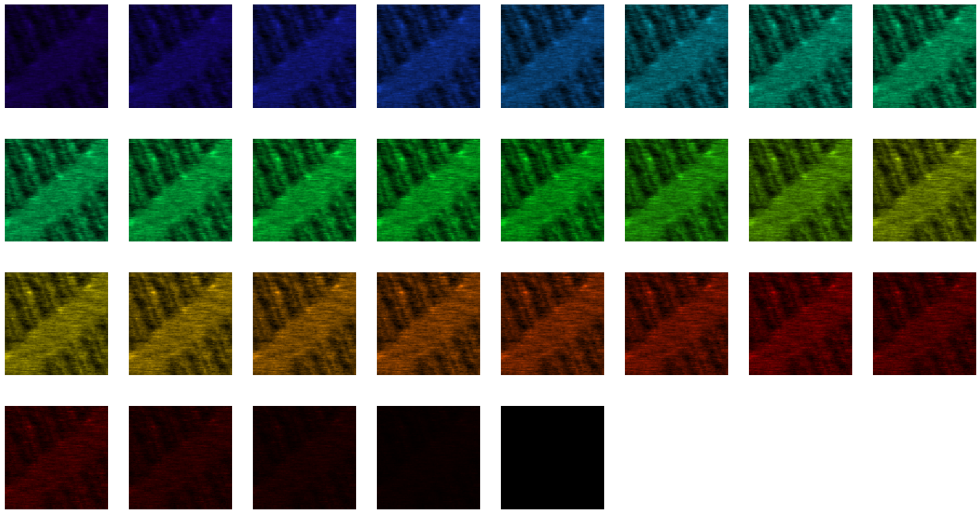
(c) Patch 1: Our reconstruction. PSNR = 34.18



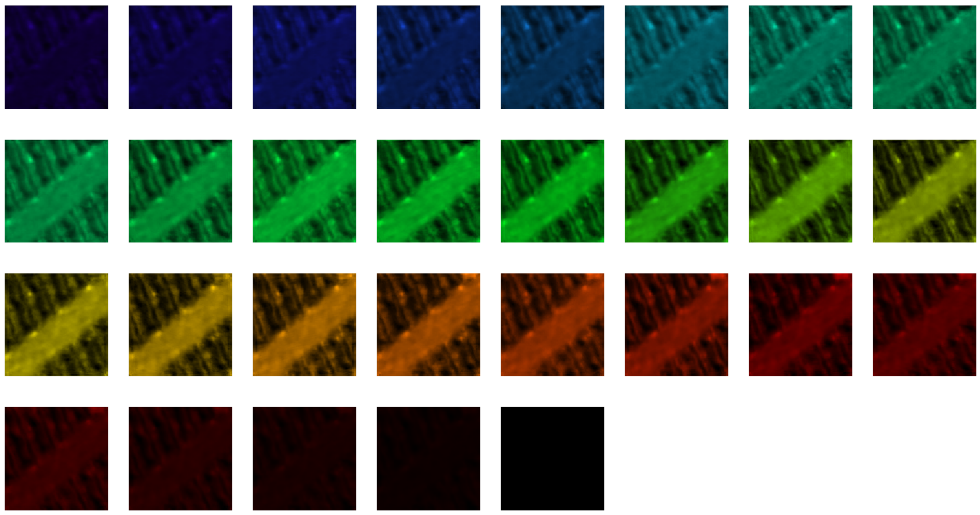
(a) Patch 2: Ground truth



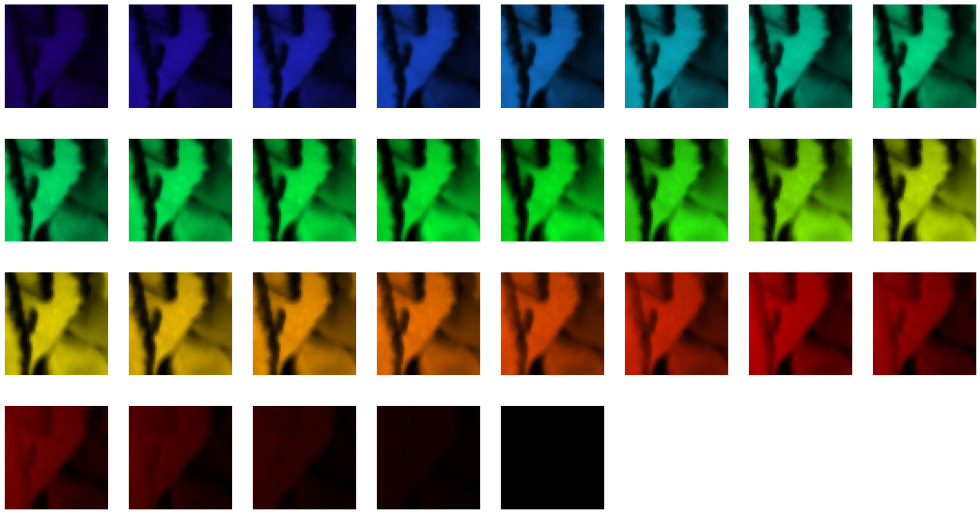
(b) Patch 2: Naive Reconstruction. PSNR = 19.89



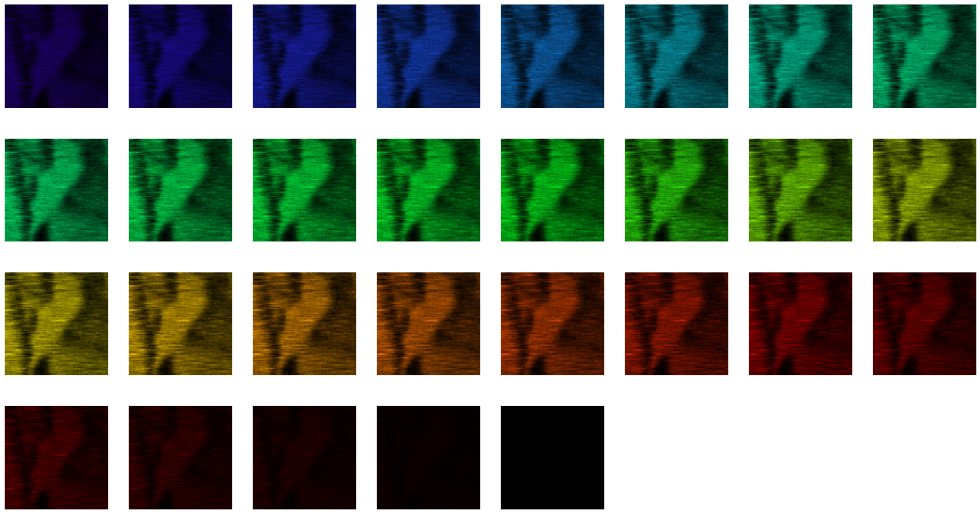
(c) Patch 2: Our reconstruction. PSNR = 32.18



(a) Patch 3: Ground truth



(b) Patch 3: Naive Reconstruction. PSNR = 13.06



(c) Patch 3: Our reconstruction. PSNR = 27.49