Blind Hyperspectral Unmixing Using Autoencoders: A Critical Comparison

Burkni Palsson, *Student Member, IEEE*, Johannes R. Sveinsson, *Senior Member, IEEE*, and Magnus O. Ulfarsson, *Senior Member, IEEE*

***Abstract*—Deep learning (DL) has heavily impacted the data-intensive field of remote sensing. Autoencoders are a type of DL methods that have been found to be powerful for blind hyperspectral unmixing (HU). HU is the process of resolving the measured spectrum of a pixel into a combination of a set of spectral signatures called endmembers and simultaneously determining their fractional abundances in the pixel. This article details the various autoencoder architectures used in HU and provides a critical comparison of some of the existing published blind unmixing methods based on autoencoders. Eleven different autoencoder methods and one traditional method will be compared in blind unmixing experiments using four real datasets and four synthetic datasets with different spectral variability. Additionally, extensive ablation experiments with a simple spectral unmixing autoencoder will be performed. The results are interpreted in terms of the various implementation details, and the question of why autoencoder methods are so powerful compared to traditional methods is unraveled. The source codes for all methods implemented in this article can be found at https://github.com/burknipalsson/hu_autoencoders.**

***Index Terms*—Autoencoder, deep learning, hyperspectral data unmixing, image processing, multitask learning (MTL), neural network, spectral–spatial model.**

I. Introduction

OVER the last decade, deep learning (DL) has opened new possibilities in processing data in data-intensive fields, such as hyperspectral imaging. Hyperspectral imaging belongs to imaging spectrometry, where an entire spectrum is acquired at every pixel. The technique has been defined by Goetz *et al.* [1] as "the acquisition of images in hundreds of contiguous, registered, spectral bands such that for each pixel a radiance spectrum can be derived."

Hyperspectral remote sensing is a passive remote sensing technique that combines spectroscopy and digital photography. As a remote sensing application, it involves extracting information from scenes on the surface of the earth. Because of the high spectral resolution, hyperspectral imaging data are very high dimensional and large in size compared to data from other imaging techniques. This high dimensionality, along with both linear and nonlinear spectral mixing, requires sophisticated data analysis methods. These methods are often in the form of nonconvex optimization and modeling [2].

The main topics within hyperspectral remote sensing are classification, data fusion, spectral unmixing, target detection, and physical parameter retrieval [3]. Hyperspectral imaging is not confined to remote sensing. It has applications in agriculture and the food industry, biotechnology, medical sciences, the pharmaceutical industry, manufacturing, and forensic science [4]–[9].

Due to the high spectral resolution of hyperspectral images (HSIs), it is possible to determine which pure materials (endmembers) are present in a scene. However, due to the low spatial resolution of HSIs, a single pixel often contains multiple endmembers. Therefore, determining the spectra of the endmembers in an HSI and their proportions in each pixel, i.e., their abundances, is a challenging inverse problem, which is the central problem of hyperspectral unmixing (HU). As a result, HU methods often determine only the endmembers, and their abundances are then subsequently determined by another method using the extracted endmembers. Methods that determine both the endmembers and their abundances simultaneously are known as blind unmixing methods.

The problem of blind HU can be formulated as a nonnegative matrix factorization (NMF) problem, a type of blind source separation problem. The idea of autoencoders has been around since the 1990s [10], and they are neural networks [11] well suited for solving aforementioned problems.

An autoencoder consists of an encoder that generates a compressed representation of its input and a decoder that reconstructs the input from the representation. By imposing a bottleneck in the network, the network is forced to discover and learn to leverage any structure present in the data, resulting in compressed knowledge representations in the bottleneck itself. Applications of autoencoders include dimensionality reduction, feature extraction, image generation, data compression, and many more [12], [13].

The first autoencoder-based method for HSI classification was published in 2014 [14], where an autoencoder was used for feature extraction. Autoencoders are commonly used for feature extraction and dimensional reduction in hyperspectral classification [15]. Works based on autoencoders for pan-sharpening and image fusion start to appear in 2015 with the method in [16] among the first to be published. Use of autoencoders for change and anomaly detection appears much later or in 2018, with [17] being among the first publications; similar for target

Manuscript received October 5, 2021; revised November 26, 2021 and December 15, 2021; accepted December 29, 2021. Date of publication January 6, 2022; date of current version February 2, 2022. This work was supported in part by the Icelandic Research Fund under Grant 174075-05 and 207233-052. *(Corresponding author: Johannes R. Sveinsson.)*

The authors are with the Faculty of Electrical and Computer Engineering, University of Iceland, 107 Reykjavík, Iceland (e-mail: burkni74@gmail.com; sveinsso@hi.is; mou@hi.is).

Digital Object Identifier 10.1109/JSTARS.2021.3140154

detection, with [17] from 2018 being among the first papers to utilize autoencoders. The first published method using an autoencoder for HU dates back to 2015 [18], and now, six years later, more than two dozen papers have been published on HU using autoencoders.

In recent years, excellent reviews have been published on traditional HU methods; examples include [19]–[26]. The work [19] reviews nonlinear HU methods. In [20], efforts to incorporate spatial information are reviewed. The paper [21] gives a good review of HU in general. The work in [22] gives a comprehensive exploration of all of the major unmixing approaches and their applications. The papers [23] and [24] focus on endmember variability in HU and review how it can be addressed, and the work [25] gives a signal processing perspective on HU. Finally, [26] gives a comprehensive review on spectral variability.

There have also been reviews published on DL in remote sensing in general and for hyperspectral imaging, such as [15] and [27]. The only review paper focusing on DL for spectral unmixing is [28]. The work [2] is a good review of interpretable hyperspectral artificial intelligence. This article attempts to give a comprehensive overview of autoencoder architectures and provide a critical comparison of autoencoder-based methods for HU, with a particular focus on blind methods. Recent autoencoder-based methods will be catalogued, discussed, and compared in experiments with both synthetic and real datasets. Nonblind methods using DL techniques are discussed, but the experiments will only involve blind methods.

Notation: The notation shown below will be used in this article.

P	Number of pixels in an HSI
B	Number of bands in an HSI
R	Number of endmembers to estimate
$\boldsymbol{A}$	The matrix of endmembers
$\boldsymbol{S}$	The matrix of abundances
$\boldsymbol{a}_i$	Endmember i
$\boldsymbol{s}_p$	Abundance vector of pixel p
$\boldsymbol{x}_p$	Spectrum of pixel p
g	Activation function
$\boldsymbol{z}_j^{(l)}$	Activation of unit j in layer l
$\boldsymbol{W}^{(l)}$	Weights of layer l

II. Hyperspectral Unmixing

A. Hyperspectral Imaging

Airborne or spaceborne hyperspectral sensors simultaneously acquire images in up to several hundred contiguous spectral bands. The sensors capture both the light emitted and the light reflected by objects as a spectrum consisting of several hundreds of channels. This results in a spectral response curve for each pixel in the image. Because the measured signal from the surface is affected by atmospheric effects, such as clouds, water vapor, and aerosols, it is converted to reflectance. Reflectance is defined as the ratio between the flux coming from the surface and the incidental flux. This makes it an intrinsic property of the materials being imaged. This conversion minimizes the effects of the imaging conditions.

Since every pixel in an HSI corresponds to a spectral response curve, the image itself is a 3-D cube of either radiance or reflectance values. The reflectance spectra of pure materials are known as *endmembers*. Because of the low spatial resolution of hyperspectral sensors, a pixel usually contains multiple endmembers. This makes the spectrum of the pixel some combination of the endmembers, and the pixel is said to be mixed. The proportional area of the pixel that each endmember covers is known as the *abundance fraction* of the endmember.

B. Mixing Models

The problem HU aims to solve arises from the fact that limited spatial resolution leads to mixed pixels. In what manner this spectral mixing happens and what assumptions can be made regarding it is central to unmixing and is captured through the so-called mixing models. There are a linear mixing model (LMM) and nonlinear mixing models, with the LMM being the most widely used because of its simplicity and effectiveness.

1) Linear Models: Linear mixing is valid when the mixing scale is macroscopic, and the incident light only interacts with one pure material [19]. The spectral mixing occurs within the sensor because the resolution is not fine enough to separate the materials. Under the LMM, the spectrum of a pixel $\boldsymbol{x}_p$ of an HSI $\boldsymbol{Y} \in \mathbb{R}^{w \times h \times B}$ having B bands is modeled as a convex combination of R endmembers $\boldsymbol{a}_m \in \mathbb{R}^{B \times 1}$ as

$$\boldsymbol{x}_p = \sum_{m=1}^{R} s_{mp} \boldsymbol{a}_m + \boldsymbol{\epsilon}_p \tag{1}$$

where the abundance fractions, s_{mp}, of pixel, $\boldsymbol{x}_p$, must satisfy the following two physically inspired constraints: $s_{mp} \geq 0$ or abundance nonnegativity constraint (ANC) and $\sum_{m=1}^{R} s_{mp} = 1$ or abundance sum-to-one constraint (ASC). $\boldsymbol{\epsilon}_p$ is noise. This can also be written as a matrix–vector multiplication as

$$\boldsymbol{x}_p = \boldsymbol{A}\boldsymbol{s}_p + \boldsymbol{\epsilon}_p \tag{2}$$

where $\boldsymbol{A} \in \mathbb{R}^{B \times R}$ is the endmember matrix having the endmembers in its columns and $\boldsymbol{s}_p$ is the abundance vector. The main drawback of the LMM is its inability to represent spectral variability, such as variations due to varying illumination conditions. There are other extended and augmented LMMs that can model spectral variability, such as the perturbed LMM in [29], the extended linear model in [30], the augmented model in [31], and the data-dependent multiscale model in [32].

2) Nonlinear Models: When the mixing scale is not macroscopic as for intimate mixtures of materials, or the incident light scatters off multiple materials, nonlinear models are needed [19], [33]. Usually, only bilinear interactions are modeled, i.e., when light reflected off one material reflects off another, a case of secondary illumination. Models that model bilinear interactions are called bilinear models [34]–[36].

Another case is intimate mixtures of grains or particles in close contact with each other, e.g., mineral particles in sand and soil. Light in such mixtures will typically interact multiple times with the particles making up the mixture before reaching the observer.

The modeling of the optical characteristics of intimate mixtures is highly nontrivial. The Hapke model [37] is one example.

An example of a general nonlinear model is

$$\boldsymbol{x}_p = \Theta(\boldsymbol{A}, \boldsymbol{s}_p) + \boldsymbol{\epsilon}_p \tag{3}$$

where nonlinear interactions between the endmembers in $\boldsymbol{A}$ are given implicitly by the function Θ and parameterized by the abundance vector $\boldsymbol{s}_p$. This article is only concerned with a restriction of (3) to the post-nonlinear structure given by

$$\boldsymbol{x}_p = \Psi(\boldsymbol{A}\boldsymbol{s}_p) + \boldsymbol{\epsilon}_p \tag{4}$$

where Ψ is a nonlinear function acting on the linear transform $\boldsymbol{A}\boldsymbol{s}_p$. The Hapke model can be considered a special case of (4) [38]. Later, it will be seen that (4) allows for easier extraction of the endmember matrix $\boldsymbol{A}$ and the abundances $\boldsymbol{s}_p$ than the general model (3).

C. Spectral Unmixing Methods

Often the first step in HU is to determine how many endmembers are in a given HSI. The most common methods for this are the method of virtual dimensionality [39], hyperspectral signal identification with minimum error method [40], the eigenvalue likelihood maximization method [41], and hyperspectral subspace identification using SURE [42].

Traditional methods for spectral unmixing are often grouped into three main categories, with the categories defined by how the problem is interpreted [21]. Among the earliest methods are the sparse regression methods that seek to express the observed spectra as linear combinations of known spectral signatures from spectral libraries [43]–[46]. Methods based on compressed sensing, such as [47]–[49], also belong to this category of sparse regression methods.

Another category of methods is geometrical methods. These methods are centered around the observation that the spectral vectors generated according to the LMM lie in an $R-1$ simplex in $\mathbb{R}^{B\times 1}$ with the endmembers at the vertices. Geometrical methods can be further categorized into pure pixel methods or minimum volume simplex methods based on whether they rely on the presence of pure pixels, i.e., the spectra of pure materials in the data or not. A well-known and widely used pure pixel method is the vertex component analysis (VCA) [50], while minimum volume simplex analysis [51] is an excellent example of a minimum volume technique.

Further examples are $\ell_{1/2}$ and ℓ_q sparsity constrained minimum volume methods in [52] and [53], respectively. The last category of traditional methods is statistical methods, which reformulate the unmixing problem as an inference problem [54]–[56]. Because of the nonnegativity constraint in mixing models, NMF has been widely used by blind unmixing methods [52], [53], [57]. Most statistical approaches to unmixing are either variants or some extensions of NMF.

III. Autoencoders

Fig. 1 shows a schematic of an autoencoder. An autoencoder consists of two parts: an encoder and a decoder. The encoder, $\mathcal{G}_E$, encodes the input, $\boldsymbol{x}_p$, into a latent code or a hidden representation, $\boldsymbol{h}_p = \mathcal{G}_E(\boldsymbol{x}_p)$, in a latent space of typically much lower dimension. The decoder, $\mathcal{G}_D$, must then reconstruct the original input from the latent code, $\hat{\boldsymbol{x}}_p = \mathcal{G}_D(\boldsymbol{h}_p)$, to minimize the loss

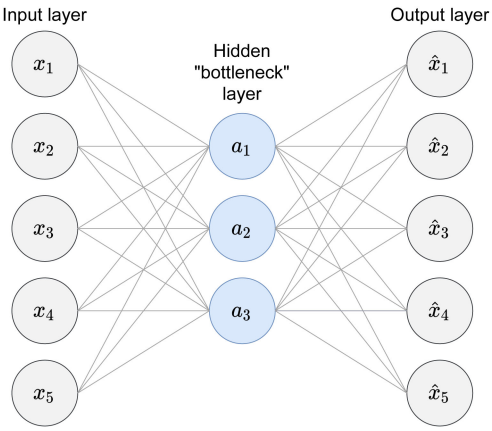


Fig. 1. Schematic of a simple autoencoder with a single hidden layer bottleneck.

$$\mathcal{L}(\boldsymbol{x}_p, \mathcal{G}_D(\mathcal{G}_E(\boldsymbol{x}_p))) \tag{5}$$

where $\mathcal{L}(\boldsymbol{x}_p, \hat{\boldsymbol{x}}_p)$ is some measure of the discrepancy between the original input $\boldsymbol{x}_p$ and the reconstruction by the autoencoder, $\hat{\boldsymbol{x}}_p$. If $\boldsymbol{W}_E$ denotes the weight matrix of the encoder and $\boldsymbol{W}_D$ denotes the weight matrix of the decoder, the forward pass of the simple autoencoder in Fig. 1 can be written as

$$\hat{\boldsymbol{x}}_p = g_D(\boldsymbol{W}_D(g_E(\boldsymbol{W}_E\boldsymbol{x}_p))) \tag{6}$$

where g_E and g_D are the activation functions of the hidden layer and the output layer, respectively. The reconstruction loss is indifferent to latent space characteristics; therefore, generally, we need additional constraints to ensure that the autoencoder learns meaningful representations, i.e., that it learns the data manifold.

Thus, using appropriate regularization can force the autoencoder to learn the necessary variations to reconstruct training examples. We want a balance between accurate reconstruction and staying on the data manifold. The regularized loss has the form

$$\mathcal{L}_{\text{regularized}}(\boldsymbol{x}_p, \hat{\boldsymbol{x}}_p) = \mathcal{L}(\boldsymbol{x}_p, \hat{\boldsymbol{x}}_p) + \lambda J(\boldsymbol{h}_p, \boldsymbol{W}_E, \boldsymbol{W}_D) \tag{7}$$

where $J(\boldsymbol{h}_p, \boldsymbol{W}_E, \boldsymbol{W}_D)$ is some penalty that can be a function of the activities of the units in the bottleneck layer, the weights of the decoder, or the encoder, and λ is the tuning parameter, which controls how strong the regularization is.

Now, if we wish to perform spectral unmixing using an autoencoder, such as in Fig. 1, it is easy to derive the basic architecture of the autoencoder by comparing the forward pass given by (6) to the LMM given by (2). The activation function of the decoder, g_E, is required to be linear; therefore, $\boldsymbol{A} = \boldsymbol{W}_D$ and $\boldsymbol{s}_p = g_E(\boldsymbol{W}_E\boldsymbol{x}_p)$.

The encoder encodes the input spectrum to a latent code that can be interpreted as the abundance fractions. The linear decoder layer then reconstructs the input as a convex combination of

the columns of its weights matrix (the endmembers) with the abundance fractions as the coefficients. The simple autoencoder, however, fails to satisfy the ANC and ASC constraints. Hence, a more sophisticated architecture is needed to fully implement the LMM.

The next few sections will give a brief overview of different types of autoencoder that have been used for HU.

A. Sparse Nonnegative Autoencoders

It is possible to make autoencoders discover and utilize structures in the data without having a bottleneck layer with fewer units than the input layer. This is done by forcing the activations in the hidden layers to be sparse. Fig 2 shows a schematic of such an autoencoder. This again results in an information bottleneck that prevents the autoencoder from just learning the identity function.

However, since the number of bands in an HSI is much greater than the number of endmembers to estimate, sparse autoencoders for unmixing use a *sparse bottleneck* layer, i.e., a layer that has much fewer units than the input layer, and which is required to be sparse via a regularizing term in the loss function such as ℓ_1-norm penalty on the activities of the units in the bottleneck layer.

In order to satisfy the ANC constraint, the abundance fractions, i.e., the output of the encoder part, must be nonnegative. This can be done in more than one way. One way is to select an activation function for the last encoder layer that outputs only nonnegative numbers such as the rectified linear unit (ReLU) activation [58] given by

$$\text{ReLU}(z_i) = \max(0, z_i) \tag{8}$$

or the sigmoid activation function given by

$$\text{sigmoid}(z_i) = \frac{e^{z_i}}{1 + e^{z_i}}. \tag{9}$$

Another way is to let the implementation of the ASC constraint also take care of the ANC constraint. Using, e.g., the softmax activation function

$$\sigma(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{10}$$

to implement the ASC constraint ensures that the abundance fractions sum-to-one and that they are nonnegative. Later in this article, different ways to implement the ASC constraint will be discussed.

It is also necessary to make sure that the weights of the decoder are all nonnegative as they correspond to the endmembers, which represent reflectance values and must, therefore, be nonnegative. This can be done in two ways: employing a nonnegativity kernel constraint or weights clamping in the DL framework used to implement the method and using a regularization that penalizes negative weights. The sparse autoencoder that satisfies all of these nonnegativity constraints is known as a sparse nonnegative autoencoder.

Such an autoencoder is effectively implementing NMF. It factors the observed HSI into the product of two nonnegative matrices, namely, the nonnegative matrix of abundance fractions and the mixing matrix having the endmembers as its columns. Such autoencoders are widely used in problems that can be solved through NMF, such as various blind separation problems [59], [60].

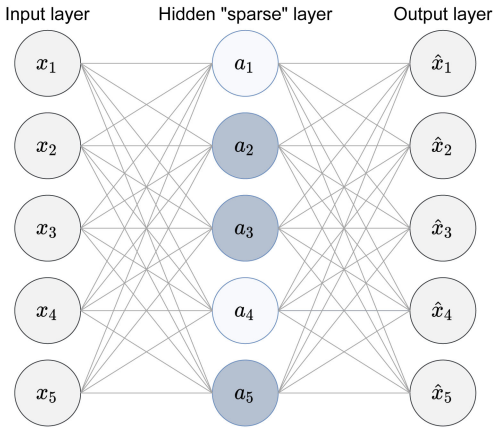


Fig. 2. Schematic of a sparse autoencoder with a single hidden layer. Dark hidden units have higher activation strength than the lighter ones.

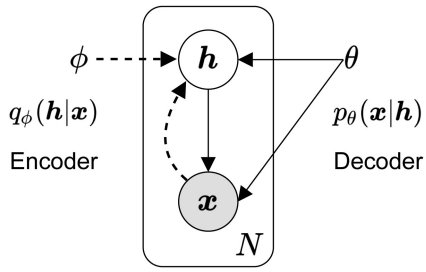


Fig. 3. Probabilistic graphical model of a VAE.

B. Variational Autoencoders

The latent space of traditional autoencoders is irregular, making them unsuitable as generative models [61]. Here, irregular means that two points close in latent space are not guaranteed to be similar once decoded. This means that two similar inputs are not necessarily encoded to points close together in the latent space. Variational autoencoders (VAEs) [62] solve this problem by utilizing regularized training that ensures that the latent space has good properties.

Fig. 3 shows the probabilistic graphical model for VAEs. The VAE contains a probability model of data $\boldsymbol{x}$ and latent variables $\boldsymbol{h}$. The joint probability distribution of $\boldsymbol{x}$ and $\boldsymbol{h}$, $p(\boldsymbol{x}, \boldsymbol{h})$, factorizes over the graph as $p(\boldsymbol{x}, \boldsymbol{h}) = p(\boldsymbol{x}|\boldsymbol{h})p(\boldsymbol{h})$. Only $\boldsymbol{x}$ is observed as indicated by the shaded circle, and we wish to infer the characteristics of $\boldsymbol{h}$, i.e., the conditional probability distribution $p(\boldsymbol{h}|\boldsymbol{x})$

$$p(\boldsymbol{h}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{h})p(\boldsymbol{h})}{p(\boldsymbol{x})} \tag{11}$$

where the evidence $p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{h})p(\boldsymbol{h})d\boldsymbol{h}$ is an intractable distribution. However, $p(\boldsymbol{h}|\boldsymbol{x})$ can be approximated with a

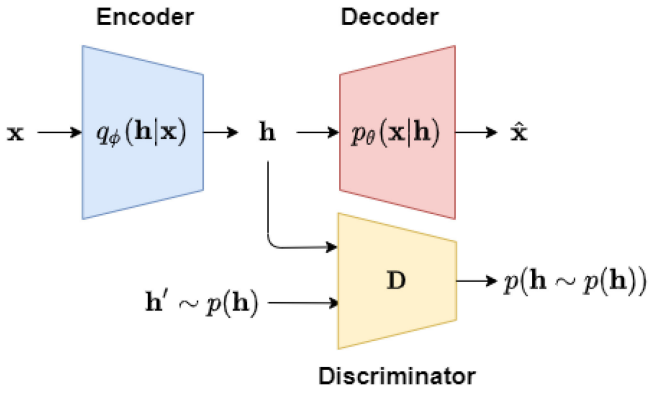


Fig. 4. Schematic of an adversarial autoencoder.

tractable distributions $q(\boldsymbol{h}|\boldsymbol{x})$ by minimizing the Kullback–Leibler (KL) divergence between $p(\boldsymbol{h}|\boldsymbol{x})$ and $q(\boldsymbol{h}|\boldsymbol{x})$ as

$$\min \mathrm{KL}(q(\boldsymbol{h}|\boldsymbol{x})\|p(\boldsymbol{h}|\boldsymbol{x})). \tag{12}$$

It is easy to show that minimizing the KL divergence above is equal to maximizing the following expression known as the variational lower bound:

$$\mathbb{E}_{q(\boldsymbol{h}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{h})] - \mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{h})). \tag{13}$$

We can use neural networks to parameterize $q(\boldsymbol{h}|\boldsymbol{x})$ as $q_\phi(\boldsymbol{h}|\boldsymbol{x})$ (encoder) and $p(\boldsymbol{x}|\boldsymbol{h})$ as $p_\theta(\boldsymbol{x}|\boldsymbol{h})$ (decoder). The reconstruction of $\boldsymbol{x}$ is then $\hat{\boldsymbol{x}} \sim p_\theta(\boldsymbol{x}|\boldsymbol{h})$. The first term in (13) is the reconstruction loss $\mathcal{L}(\boldsymbol{x}, \hat{\boldsymbol{x}})$, and the second term can be viewed as a regularization term that ensures that $q(\boldsymbol{h}|\boldsymbol{x})$ is similar to $p(\boldsymbol{h})$. The loss of the networks is then given by the negative of (13) as minimizing its negative is equivalent to maximizing it or

$$\mathcal{L}_{\text{total}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \mathcal{L}(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \sum_j \mathrm{KL}(q_{\phi_j}(\boldsymbol{h}|\boldsymbol{x})\|p(\boldsymbol{h})) \tag{14}$$

where the sum is over the dimensions of the latent space and $p(\boldsymbol{h})$ is usually chosen as an unit Gaussian distribution as it gives a closed form of the KL term and the reconstruction loss becomes the mean squared error (MSE) loss.

C. Adversarial Autoencoders

An adversarial autoencoder is an approach that borrows ideas from generative adversarial networks (GANs) to turn a standard autoencoder into a useful generative model [63]. A schematic of an adversarial autoencoder is shown in Fig. 4. Here, $p(\boldsymbol{h})$ is some prior distribution that we want to impose on the codes $\boldsymbol{h}$. The upper part of the figure is a standard autoencoder that tries to reconstruct its input $\boldsymbol{x}$. It has an encoder model, $q_\phi(\boldsymbol{h}|\boldsymbol{x})$, which produces a latent code $\boldsymbol{h}$, and a decoder model, $p_\theta(\boldsymbol{x}|\boldsymbol{h})$, which reconstructs the input from the latent code. The encoder model defines an aggregated posterior $q(\boldsymbol{x})$ as

$$q(\boldsymbol{x}) = \int_x q_\phi(\boldsymbol{h}|\boldsymbol{x}) p_d(\boldsymbol{x}) d\boldsymbol{x} \tag{15}$$

where $p_d(\boldsymbol{h})$ is the probability density function of the data.

An adversarial autoencoder is obtained by regularizing the autoencoder described above by matching the aggregated posterior $q(\boldsymbol{x})$ to some arbitrary prior distribution $p(\boldsymbol{h})$ [63]. This is done by borrowing the concept of an adversarial network from GANs. The network labeled D in Fig. 4 is trained to tell if the latent code comes from the aggregate posterior $q(\boldsymbol{x})$ or from the prior $p(\boldsymbol{h})$. The autoencoder's encoder acts as a generator of an adversarial network, while the network D acts as a discriminator.

During training, the adversarial network and the autoencoder are trained alternately. First, the autoencoder updates both the encoder and the decoder to minimize the reconstruction error. Then, the adversarial network updates its discriminator network D to tell apart the true samples ($\boldsymbol{h}'$ sampled from the prior $p(\boldsymbol{h})$) from the ones generated by the encoder part and updates its generator (the encoder) to confuse the discriminator D. The general loss function of an adversarial autoencoder has the form

$$\mathcal{L}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \mathcal{L}_{\text{reconstruction}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \mathcal{L}_{\text{adversarial}}(D(\boldsymbol{h}), D(\boldsymbol{h}')) \tag{16}$$

where the adversarial loss can be binary cross entropy [64] loss or some other loss such as a feature loss as in [65].

D. Denoising Autoencoders

Autoencoders having only a single hidden layer and which take a partially corrupted input and are trained to recover the original uncorrupted input are known as denoising autoencoders (DAs) [66]. At training time, the input $\boldsymbol{x}$ is partially corrupted using a stochastic mapping, resulting in $\tilde{\boldsymbol{x}}$. The partially corrupted input is then reconstructed in the standard way, $\hat{\boldsymbol{x}} = \mathcal{G}_D(\mathcal{G}_E(\tilde{\boldsymbol{x}}))$, but the loss is calculated between the reconstructed input $\hat{\boldsymbol{x}}$ and the original uncorrupted input, $\boldsymbol{x}$ [66]. It is important to notice that the input is only degraded during training. To perform the denoising well, the model needs to extract features that capture useful structure in the input distribution.

1) Marginalized DAs: Many simple DAs are stacked to form a deep network by feeding the latent code of the previous DA as input to the current DA in the stack [66]. The DAs are pretrained one layer at a time, with each layer trained as a DA by minimizing the error in reconstructing its input, which is the output latent code of the previous layer. When stacked DAs are used for feature learning, linear autoencoders are sufficient, i.e., the activation function can be set to be linear, and the effect of each autoencoder can be given by a single linear transformation $\boldsymbol{W}$. When considering a deep enough stack of such single-layer linear denoisers, the corruption can be marginalized out, and a closed-form solution can be obtained for the effective reconstructing matrix $\boldsymbol{W}$ [67]. The resulting single-layer autoencoder, which applies the reconstructing matrix, is known as a marginalized denoising autoencoder (mDAE) and has found use in spectral unmixing [67], [68].

E. Convolutional Autoencoders

Convolutional autoencoders [69] are based on convolutional neural networks (CNNs). These networks use convolutional layers instead of fully connected layers. Convolutional layers convolve learnable filters with input patches to produce feature

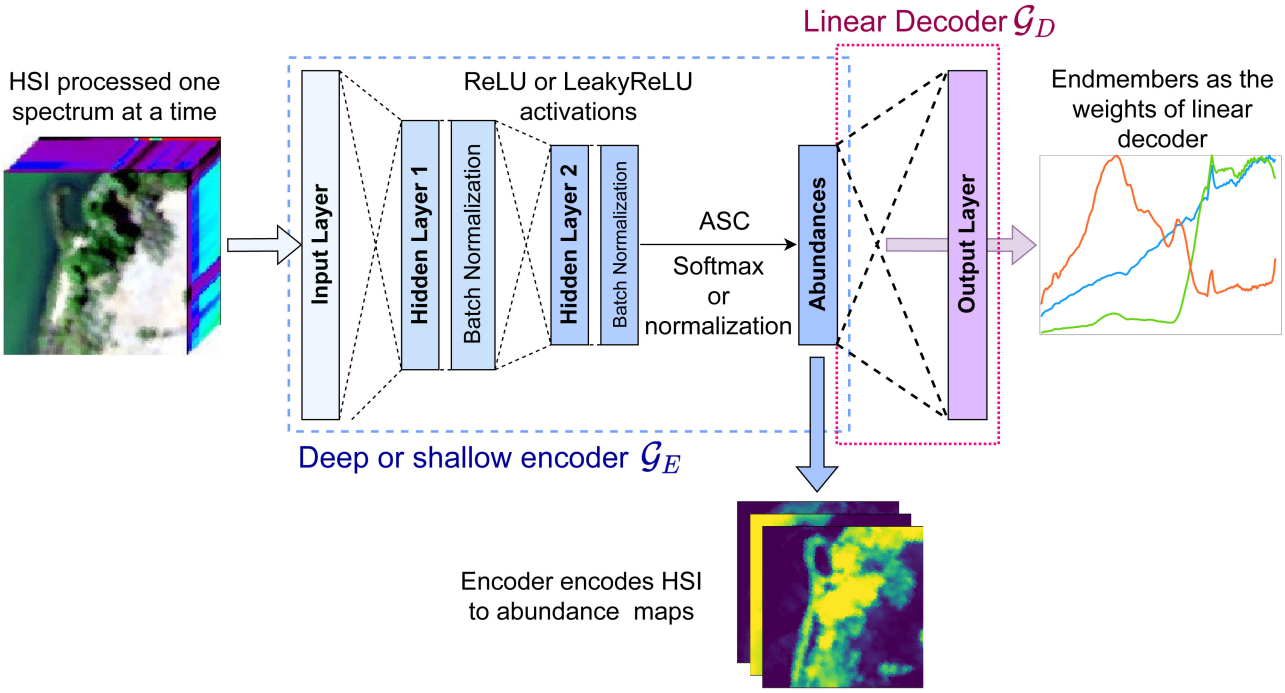


Fig. 5. Schematic of a general unmixing autoencoder. The figure shows an example having two hidden layers, but the actual number could vary from one to several. The decoder has a nonnegative weight constraint and linear activation. The ASC constraint is enforced with a normalizing utility layer or softmax activation.

maps. In the context of spectral unmixing, they differ from conventional autoencoders in one crucial aspect, which is that they are inherently spatial and make use of the spatial correlations existing in HSIs. In contrast, conventional autoencoders look at one spectrum at a time and need special regularization to consider spatial information.

Architecturally, they are similar to conventional autoencoders used in spectral unmixing. For example, the abundance maps arise naturally as the feature maps of a "spectral bottleneck" layer, i.e., a convolutional layer having R feature maps, one for each endmember. Similarly, the weights of the linear decoder convolutional layer can be interpreted in terms of endmembers, but to do so for the nontrivial case of filter size larger than 1×1 requires a new spectral–spatial LMM as the reconstruction of a pixel's spectrum involves the abundances of neighboring pixels [70].

IV. Spectral Unmixing Autoencoders

The majority of autoencoder-based methods for blind unmixing are based on the fully connected nonnegative autoencoder. Here, we will look at various implementation details and choices taken by methods in blind unmixing that can influence the performance of spectral unmixing, e.g., choice of loss functions, choice of activation functions, and so forth. Fig. 5 shows a generic autoencoder that performs spectral unmixing.

It has a deep encoder with two hidden layers, but the hidden layers can vary from one to several. The decoder has a nonnegative constraint for its weights and a linear activation. It is also possible to make the decoder's weights nonnegative using a regularization that penalizes negative weights. The activation function of the encoder is the ReLU or Leaky ReLU (LReLU) [71]. The ASC constraint is enforced using a normalizing utility layer or softmax activation. In practice, it is convenient to create a custom layer that enforces the ASC because it makes it easier to implement abundance regularizations. The forward pass of this general autoencoder can be written out, using the notation introduced in Section I-A, as

$$\hat{\boldsymbol{x}}_p = \boldsymbol{W}_D(\boldsymbol{\sigma}(\,\mathrm{BN}\, g(\boldsymbol{W}^{(2)}\mathrm{BN}\, g(\boldsymbol{W}^{(1)}\boldsymbol{z}^{(0)})))) \tag{17}$$

where BN denotes batch normalization using operator notation, $\boldsymbol{\sigma}$ is the ASC enforcing layer, i.e., $\boldsymbol{\sigma}(\boldsymbol{z})$ sums-to-one, $\boldsymbol{z}^{(0)} = \boldsymbol{x}_p$, i.e., the input layer activations, and $\boldsymbol{W}_D$ is the weight matrix of the linear decoder.

A. Deep Versus Shallow Encoder

The power of depth in feedforward neural networks is still not fully understood in the field of DL theory [72]. Deep neural networks can extract better features than shallow networks for the same amount of computation. In [72], it is proven that deep networks can approximate the class of compositional functions with the same accuracy as shallow networks but with an exponentially lower number of training parameters. However, deep networks are more prone to overfitting and can be harder to train than shallow networks and require more regularizations. Whether a deep encoder will extract better abundance fractions than a shallow encoder most likely will depend on the mixing model being assumed. It could be argued that a deep encoder can extract better abundances under a nonlinear mixing model. We will come back to this question when discussing the experimental results. The methods in [38], [70], and [73]–[77] employ

deep encoders. The method in [78] employs a deep 1-D CNN encoder.

B. Choice of Activation Function for Hidden Layers

The choice of activation function for the hidden layers and especially the last hidden layer of a deep encoder can significantly influence the behavior and performance of unmixing autoencoders. Like in DL in general, the trend has been away from using saturating activation functions, which squeeze their input, e.g., like the sigmoid function, to using nonsaturating activation functions such as the LReLU variant given by

$$\text{LReLU}(z_i) = \max(\beta z_i, z_i) \tag{18}$$

where β is a real number between 0 and 1. The problem with saturating activation functions is that the value of hidden units often becomes stuck at their asymptotic values, e.g., 0 or 1 for the sigmoid activation, indicating that the preactivation sum of products is relatively large. This leads to vanishing gradients during training. The methods in [18] and [79] use a parameterized sigmoid activation function. The ReLU activation largely mitigates the problem of vanishing gradients, but the function is still saturated to the left, and hidden units can become stuck at zero. In unmixing, this can cause the abundance maps of certain endmembers to become zero-valued. The works [68], [76], [77], [80], and [81] all use the ReLU activation. The LReLU activation is a good choice of an activation function as it is nonsaturating in both directions. Other nonsaturating activation function include exponential linear unit (ELU) [82], scaled ELU [83], and Swish [84]. The methods in [38], [70], and [74] all use LReLU as the activation function of hidden units.

C. Implementation of the ASC Constraint

The last step the encoder needs is making sure the latent code resulting from previous layers sums-to-one in accordance with the ASC constraint of the LMM, or

$$\boldsymbol{s} = \boldsymbol{\sigma}(\boldsymbol{h})$$

where $\boldsymbol{\sigma}$ is the action of the layer used for the ASC constraint enforcement. One way to do this is to normalize the latent code vector, $\boldsymbol{h}$, resulting from the last layer of the encoder as

$$s_i = \frac{h_i}{\sum_{j=1}^{R} h_j} \tag{19}$$

where $\boldsymbol{s}$ is the abundance vector. If this method is to be used, it must be ensured that all the values h_i, $i = 1, \ldots, R$, are nonnegative. In practice, this means that the choice of activation function for the last layer of the encoder is restricted to strictly nonnegative functions such as ReLU or sigmoid activation functions. It might be argued that thresholding the values using ReLU or dynamic ReLU

$$\hat{\boldsymbol{s}} = \max(0, \boldsymbol{s} - \boldsymbol{\alpha}) \tag{20}$$

where α is a nonnegative $R \times 1$-vector learned by the network, as some works do, might deactivate many units in the network and not utilize its full capacity. The methods in [18] and [73] use this implementation of the ASC, while [73] additionally uses it with a dynamically thresholding ReLU activation. Another ASC implementation that does not require nonnegative activation of the last hidden layer is

$$s_i = \frac{|h_i|}{\sum_{j=1}^{R} |h_j|}. \tag{21}$$

This implementation also takes care of the ANC constraint. The method in [38] uses this implementation. The method described in [80] uses a variation of (21), where taking the absolute value in the nominator is omitted. Yet another implementation of the ASC that many works use is to apply the softmax function directly to $\boldsymbol{h}$ or scale it first as in

$$\boldsymbol{s} = \text{softmax}(\gamma \boldsymbol{h}) \tag{22}$$

where γ is a constant larger than 1. This implementation also takes care of the ANC constraint and, by choosing large γ, achieves something similar to ℓ_1-norm regularization on the abundance maps. The papers [76], [77], and [80] use the softmax function without scaling to implement the ASC, while the methods in [70] and [74] use it with a scaling.

The ASC constraint can also be implemented through regularization by adding a penalty term to the loss function that penalizes if the abundances do not sum-to-one. This can, however, increase the instability of the training process and the time needed for convergence. The techniques described in [76] and [77] enforce the ASC through such regularization. A related implementation is to augment the data matrix and the decoder weight's matrix with constant vectors. The methods in [68], [75], [81], and [85] use this to weakly enforce the ASC constraint.

D. Batch Normalization

Batch normalization is used to speed up the training of neural networks and make them more stable through normalization of the inputs to a layer by recentering and rescaling [86]. Initially, it was believed that batch normalization did work by mitigating internal covariance shift, which reduces the learning rate of the network [86], but recent research shows that it works by smoothing the objective function [87]. Batch normalization layers are usually used after or before a nonlinear activation function such as ReLU, and they are only active during training.

Regarding the choice of batch size, it is the authors' experience that small batch sizes yield better unmixing performance than large batch sizes. In [88], it is argued that large batch sizes can lead to degradation of the quality of DL models as measured by their generalization abilities as large-batch methods tend to converge to sharp minimizers of the training and testing functions, and sharp minima lead to inferior generalization.

If we let $\boldsymbol{z}_i$, $i = 1, \ldots, m$, be the values of the inputs from the previous layer for a batch $\mathcal{B}$, we can write the effect of the batch normalization layer as

$$\boldsymbol{z}_{\text{BN}_i} = \text{BN}_{\gamma,\beta}(\boldsymbol{z}_i) = \gamma \hat{\boldsymbol{z}}_i + \beta \tag{23}$$

where

$$\hat{\boldsymbol{z}}_i = \frac{\boldsymbol{z}_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{z}_i$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m} (\boldsymbol{z}_i - \mu_{\mathcal{B}})^2$$

γ and β are learnable parameters, and ϵ is a very small number.

E. Dropout

Dropout is a powerful regularization technique that helps with preventing overfitting and improving the generalization of deep networks. It works by randomly omitting or dropping units of a layer during the training of a neural network [89]. This prevents complex coadaptations of units on the training data. It can be thought of as a way of performing model averaging with neural networks. In convolutional networks, feature maps, instead of units, are set randomly to zero or dropped, a case of dropout known as spatial dropout [90]. When properly used, dropout can give a better unmixing performance, primarily when used with deep encoders as they need more regularization. Dropout is not commonly used in blind unmixing methods. The methods in [73], [74], and [80] use dropout, while the method in [70], being a convolutional network, uses spatial dropout.

F. Choices of Loss Fidelity Function and Spectral Variability

The choice of the similarity measure used for the fidelity term of the loss function is crucial and can significantly influence the performance of the unmixing autoencoder. The main reason for this is that some similarity measures are scale invariant, while others are not. Furthermore, due to variations in illumination or topography, real hyperspectral data often have considerable spectral variability, and the LMM cannot model them. This means that the same material or mixture of materials can have spectra that differ in scale even though they have the same abundances.

A similarity measure that is not scale invariant will penalize the LMM reconstructions that differ in scale from the original spectra and, thus, cannot handle spectral (scale) variability. In contrast, a scale-invariant similarity measure will only penalize based on the difference in shape. Thus, a scale-sensitive similarity measure should not be used for an HSI having substantial spectral variability. The most common scale-invariant similarity measures used in unmixing are the spectral angle distance (SAD) given by

$$J_{\text{SAD}}(\boldsymbol{x}_p, \hat{\boldsymbol{x}}_p) = \arccos\left(\frac{\boldsymbol{x}_p^T \hat{\boldsymbol{x}}_p}{\|\boldsymbol{x}_p\|_2 \, \|\hat{\boldsymbol{x}}_p\|_2}\right) \tag{24}$$

and the spectral information divergence (SID) measure given by

$$J_{\text{SID}}(\boldsymbol{x}_p, \hat{\boldsymbol{x}}_p) = \sum_{n=1}^{B} p_n \log\left(\frac{p_n}{q_n}\right) + \sum_{n=1}^{B} q_n \log\left(\frac{q_n}{p_n}\right) \tag{25}$$

where

$$p_n = \frac{\boldsymbol{x}_{p,n}}{\sum_{n=1}^{B} \boldsymbol{x}_{p,n}}, \quad q_n = \frac{\hat{\boldsymbol{x}}_{p,n}}{\sum_{n=1}^{B} \hat{\boldsymbol{x}}_{p,n}}$$

are estimates of the probability mass functions of the target and estimated spectra, respectively. The methods in [38], [70], [73], and [74] all use the SAD similarity measure as the fidelity term of the objective function, while the method in [91] uses the SID measure. The most common non-scale-invariant similarity measure is the MSE given by

$$J_{\text{MSE}}(\boldsymbol{x}_p, \hat{\boldsymbol{x}}_p) = \|\boldsymbol{x}_p - \hat{\boldsymbol{x}}_p\|_2^2. \tag{26}$$

All the methods in [18], [68], [75], [79], [81], [85], [92], and [93] use the MSE measure as the fidelity term of the network's loss function. It is also possible to use a combination of both scale-invariant and non-scale-invariant loss terms. The work in [80] uses such a combination as the fidelity term. In [76], it is argued that the histogram of the reconstruction error of an unmixing autoencoder follows a hyper-Laplacian distribution, and hence, an optimal fidelity term should have the form

$$J_{\text{HL}}(\boldsymbol{x}_p, \hat{\boldsymbol{x}}_p) = \|\boldsymbol{x}_p - \hat{\boldsymbol{x}}_p\|_q^q \tag{27}$$

where q is a positive real number, the optimal value determined as 0.7 in [76].

G. Abundance Regularizations

In order to further constrain the unmixing problem, reasonable assumptions or priors regarding the abundance fractions are used to construct regularization terms. These additional terms to the loss force the latent codes or the learned representations to live in certain latent space subspaces, reducing its effective dimensionality. The most common assumption made is that the abundance maps are sparse. Increased sparsity can be achieved by ℓ_1-norm regularization or, more generally, by ℓ_q-norm regularization, i.e., by an objective function of the form

$$\mathcal{L}_{\ell_q \text{ regularized}}(\boldsymbol{x}_p, \hat{\boldsymbol{x}}_p) = \mathcal{L}(\boldsymbol{x}_p, \hat{\boldsymbol{x}}_p) + \lambda \|\boldsymbol{s}_p\|_q^q \tag{28}$$

where $0 \leq q \leq 1$, and $\boldsymbol{s}_p$ is the abundance vector for spectrum $\boldsymbol{x}_p$. The methods in [76], [77], [80], and [93] all use ℓ_1-norm regularization to promote sparsity of the abundances.

In [76], it is argued that different abundance maps are close to orthogonal since no more than two endmembers are usually mixed within one pixel. This work introduces a new prior based on this, named orthogonal sparse prior (OSP), given by

$$\mathcal{L}_{\text{OSP}}(\mathcal{B}) = \sum_{i<j} \frac{\mathcal{B}_{\cdot i} \cdot \mathcal{B}_{\cdot j}}{\|\mathcal{B}_{\cdot i}\|_2 \|\mathcal{B}_{\cdot j}\|_2} \tag{29}$$

where $\mathcal{B} \in \mathbb{R}^{b \times R}$ is a batch of size b of generated abundance maps. An ablation study shows that the OSP regularization leads to more sparse abundance maps than the conventional norm based prior. The work in [94] uses inhomogeneous Gaussian Markov random field (IGMRF) as a prior for spatial abundances regularization. In [92], a VAE is used as an abundance regularization to enforce the ASC constraint.

H. Weights and Endmember Regularizations

Weight decay, ℓ_2-norm regularization on the weights of the encoder and decoder, is commonly employed to reduce overfitting and for better generalization. It can be applied by adding a

term in the loss function as

$$\mathcal{L}_{\ell_2 \text{ regularized}}(\boldsymbol{x}_p, \hat{\boldsymbol{x}}_p) = \mathcal{L}(\boldsymbol{x}_p, \hat{\boldsymbol{x}}_p) + \lambda \|\boldsymbol{W}\|_{\mathbf{F}} \quad (30)$$

where

$$\|\boldsymbol{W}\|_{\mathrm{F}} = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij}^2 \right)^{1/2}$$

and $\boldsymbol{W}$ is the weight matrix of the layer we wish to regularize. ℓ_2 regularization penalizes the sum of squares of the weights and, thus, encourages smaller weights. Unlike ℓ_1 regularization, it does not lead to sparse solutions. In [80] and [93], ℓ_2 is used both for the weights of the encoder and the decoder. The method in [81] applies ℓ_{21} regularization to the encoder's weights with ℓ_{21}-norm being a rotational invariant ℓ_1-norm (R1-norm) defined as [95]

$$\|\boldsymbol{W}\|_{21} = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} W_{ij}^2 \right)^{1/2}$$

for an $m \times n$ matrix $\boldsymbol{W}$.

Another well-known regularization of the endmembers, i.e., the weights of the decoder, is minimum volume regularization. This regularization aims to minimize the volume of the simplex defined by the origin and the endmembers, i.e., the columns of $\boldsymbol{W}_D$. All reconstructed spectra are guaranteed to lie inside this simplex, being convex combinations of the endmembers. The volume of this simplex is often denoted by

$$\text{MinVol}(\boldsymbol{W}_D) = |\det \boldsymbol{W}_D|.$$

Therefore, the loss function with minimum volume regularization has the form

$$\mathcal{L}_{\text{MV regularized}}(\boldsymbol{x}_p, \hat{\boldsymbol{x}}_p) = \mathcal{L}(\boldsymbol{x}_p, \hat{\boldsymbol{x}}_p) + \lambda \text{MinVol}(\boldsymbol{W}_D). \quad (31)$$

The methods in [85] and [92] employ this regularization. In [94], an IGMRF prior is used for spectral regularization of the endmembers. The reconstructed HSI is also regularized using a spectral–spatial IGMRF prior.

I. Multitask Learning

Recently, there have been two techniques for HU [74], [75] that have utilized multitask learning (MTL). In the context of neural networks, MTL means that the network learns multiple related tasks in parallel, and parameters are shared between the tasks, usually via shared layers [96]. Thus, the different tasks are implemented as different branches of one network, which can have multiple inputs and outputs, depending on the problem being solved [97]. The benefits of MTL in spectral unmixing autoencoders include the following.

1) *Faster learning:* When tasks are correlated, they will contribute to the aggregate gradient during backpropagation and increase the effective learning rate on the input to hidden layer weights [96]. This means that useful features will form faster in the shared hidden layer of the network.
2) *Reduced risk of overfitting:* It has been shown that when layers are shared between tasks, the risk of overfitting the shared parameters can be up to an order N smaller, where N is the number of tasks [98].
3) *Improved stability:* It has been shown that MTL tasks prefer similar hidden layer representations [96]. For MTL autoencoders, where each task is unmixing a different pixel of a patch, this can increase the method's stability and consistency.
4) *Incorporation of spatial information:* MTL allows for the construction of autoencoders that can unmix a whole patch at a time by having one task for unmixing each pixel from the patch. This allows for fully connected networks that can directly exploit the spatial correlation in the HSI with all the above-listed benefits of MTL.

In [74], MTL is utilized for spectral–spatial unmixing, and in [75], MTL is utilized to perform bilinear mixing model spectral unmixing. One task performs the linear unmixing, and another task updates the bilinear components.

J. Approaches Utilizing Adversarial and VAEs

Recently, several HU methods have utilized VAEs. In [92], a VAE is used to ensure the nonnegativity and sum-to-one constraints of the abundances. The paper [99] employs a VAE to learn a spectral variability model and generate endmembers. The generated endmembers are then used to solve a spectral unmixing problem, cast as an alternating nonlinear least-squares problem that is solved iteratively, alternately adjusting the abundances and the low-dimensional representations of the endmembers in the generative model.

Recently, several papers where adversarial autoencoders and generative models are used for unmixing have been published. A joint metric neural network in the form of an adversarial autoencoder is suggested in [65]. There, the Wasserstein distance [100] between features in the discriminator is used to regularize the autoencoder, which is using the SAD as the loss function. The Wasserstein distance between features of real and reconstructed spectra provides useful gradient information that promotes the autoencoder to reach a solution with better unmixing performance.

In [101], an abundance estimation method is suggested using 1-D convolution kernels and spectral uncertainty. High-level representations are computed and are further modeled with the multinomial mixture model to estimate abundance fractions under high spectral uncertainty. A new trainable uncertainty term based on a nonlinear neural network is used in the reconstruction step. The uncertainty models are optimized by the Wasserstein GAN [102] to improve stability and capture uncertainty.

In [103], an adversarial loss is used to guide an autoencoder network by encouraging the encoder to match an abundance prior derived from superpixels using VCA and fully constrained least squares. A conceptually similar paper is [104]. Here, two autoencoders share decoder weights. One network reconstructs endmember bundles obtained by VCA from superpixels of an HSI, while the other is a standard spectral unmixing network reconstructing the pixels of the HSI. This is an example of a two stream network, where one stream guides the other.

The method in [105] borrows the concept of cycle consistency, similar to the concept behind CycleGan [106]. Here, two 2-D CNN unmixing autoencoders are learned in cascade. The first autoencoder performs blind unmixing of the original HSI, and the second performs blind unmixing of the reconstruction by the first one. The network loss is the self-perception loss defined as the sum of the reconstruction terms w.r.t. the original HSI and a term that penalizes the network if the abundance maps of the autoencoders are dissimilar. A conceptually similar cascade approach, which additionally utilizes self-supervised learning, is [107]. Here, a two-stage network is utilized; one stage performs unmixing (inverse model), while the other (forward model) learns the physics of the hyperspectral data acquisition to handle noise and perturbations better.

K. Utilization of Spatial Information

Like other natural images, HSIs have highly correlated pixels, and it is desirable to use this spatial information. However, most unmixing methods are strictly spectral and do not directly exploit the spatial structure of HSIs. This means that the methods work with one pixel at a time, even though it may use regularizations based on assumptions about the spatial structure, which will not be considered here to count as directly exploiting the spatial structure. At the time of writing this, there are three blind unmixing methods based on autoencoders that directly exploit the spatial structure of HSIs. The method in [74] directly unmixes a patch at a time using a multiple branch architecture that is inspired by MTL [96], [97].

The other techniques directly exploiting the spatial structure of HSIs are proposed in [70] and [105] and are fully convolutional autoencoders that are inherently spatial in nature. They exploit the spatial and spectral structure of HSIs both for endmember and abundance map estimation. Working directly with patches of HSIs and not using pooling or upsampling layers preserves the spatial structure throughout the network. The abundance maps arise then naturally as feature maps of a hidden convolutional layer. This makes it very easy to apply complex spatial regularization on the abundances such as total variation.

L. Working in Transformed Domains

Working in transformed domains, e.g., the wavelet or curvelet domains, has been shown to be beneficial for blind NMF-based HU [108], [109]. It is reasonable that this should also apply to autoencoder-based methods. A literature search shows that there are many applications of autoencoders working in transformed domains but only one in HU, the work in [93].

M. Nonlinear Approaches

In the last two years, a number of autoencoder-based methods for nonlinear unmixing have been published. Here, some of the latest approaches will be briefly discussed. Many of the methods are based on the post-nonlinear structure (3), but some extend it and are more general.

The method in [110] is the first blind HU method to utilize a long short-term memory [111] network. It additionally utilizes an attention mechanism to selectively focus on parts of the input during learning.

A method that addresses a general model that consists of a linear component and a nonlinear component was proposed in [112]. The encoder is in the form of a 3-D CNN network to better capture the spectral–spatial priors from the data. The decoder has a model-based structure so that nonlinear interactions are imposed on the endmembers weighted by the abundances.

In [113], a method for nonlinear unmixing is given that is a combination of a kernelization layer using radial basis functions and autoencoder structures. The method proposed in [114] is a model-based autoencoder method that considers the mixing model to be a nonlinear fluctuation over a linear mixture.

N. Hyperparameter Selection

Tuning hyperparameters is important in DL-based methods, and much effort has been devoted to the problem. In DL, algebraic models are usually unavailable, and the model usually has to be treated as a blackbox when tuning hyperparameters. To evaluate the performance of a model for a given set of parameters, k-fold cross-validation [115] is often used. Generally, any gradient-free optimization method can be applied to hyperparameter tuning of DL models. Two standard but primitive methods are grid search and random search.

In grid search, a finite set of values of each hyperparameter is specified, and the model is evaluated on the Cartesian product of these sets. Grid search suffers from the curse of dimensionality as the required number of evaluations grows exponentially with the dimension of the configuration space [116]. Random search, which samples configurations randomly, usually works better than grid search when some hyperparameters are much more important than others and allows for better parallelization [117].

In recent years, Bayesian optimization [118] has emerged as a state-of-the-art optimization framework for the optimization of blackbox functions, especially expensive ones. Bayesian optimization is an iterative method to optimize such blackbox objective functions. The objective function is treated as a random function, and a prior is placed over it. Then, the posterior distribution, known as a surrogate model of the objective function, is used to determine what set parameters should be queried in the next iteration. After each iteration, the posterior gets updated, and the objective function is mapped out better. The Bayesian approach uses a Gaussian process to model the objective function and defines an acquisition or selection function used to determine the next point in the parameter space to evaluate. The most common choice of the acquisition function is the expected improvement, which estimates how much the objective function is expected to improve.

Applying loss-based optimization methods to spectral unmixing autoencoders is very difficult. The main reason for this is that the network's actual loss is not a measure of its unmixing performance. Instead, the network's loss is a similarity measure between the input and the reconstructed input. The unmixing performance measures the quality of the weights of the linear

decoder (the endmembers) and the abundances that are the encoder part's latent codes.

Thus, it cannot be said that autoencoder methods converge to a solution regarding the endmembers or abundances, in contrast with many traditional methods. Only if ground truth is available is it possible to construct a function to measure the unmixing performance and score it. In [119], Bayesian optimization is used with such an objective function that scores the extracted endmembers with the average SAD from given reference endmembers. In addition, in [53], the Bayesian information criterion is used to select optimal regularization parameters.

It is, therefore, challenging to use traditional hyperparameter tuning methods such as Bayesian hyperparameter optimization or maximum *a posteriori* [120] to select hyperparameters for autoencoder unmixing methods. This is partly because the number of epochs used for training is one of the hyperparameters, and we do not have strict convergence concerning the quality of the endmembers. In addition, because of random weight initialization, many runs are required per hyperparameter configuration to get a reliable measure for any automatic or manual hyperparameter selection. This makes hyperparameter tuning for spectral unmixing autoencoders very costly. To make matters worse, different hyperparameters are usually needed for different HSIs.

One approach that has been published on automatic regularization parameter tuning on HSIs without any ground truth is described in [121]. There, a technique for the automatic tuning of a minimum volume regularization parameter for a geometrical HU method is suggested. The parameter controls the tradeoff between the reconstruction error and the simplex volume of the endmember matrix. The technique is based on the geometrical insight that a good parameter value should result in an estimated endmember simplex whose boundary is close to the boundary data points. A similar approach could be adopted to autoencoder-based methods regarding the tuning of regularization parameters in the absence of ground truth.

For the experiments in this article, the hyperparameter values used for the methods were the ones recommended by their authors.

O. Recommended Implementation Choices

Here, some pros and cons of the main architectures and implementation details will be discussed, and recommendations based on the authors' experiences will be given. For a method that will use spatial regularizations of the abundance maps, the best architecture is a fully convolutional autoencoder. This architecture makes it very easy to work with complex spatial regularizations of the abundances as they arise as feature maps of hidden layers in the network and can be easily worked with. In addition, the CNN approach is inherently spectral–spatial. For mostly spectral methods, a fully connected autoencoder is the best choice.

For most purposes, a shallow encoder with at most one hidden layer besides the bottleneck abundance layer is the best choice. Deeper encoders do not give better performance for simple mixing models such as the LMM. Extra hidden layers also result in extra hyperparameters, i.e., the number of hidden units. If the decoder is nonlinear with many layers, a deeper encoder could be beneficial. The recommended activation function for the hidden layers of the encoder is the LReLU function. It is nonsaturating in both directions and prevents units from getting stuck with zero activations.

The decoder should be made nonnegative using a weights constraint or a regularization that penalizes nonnegative weights. In the authors' experience, regularizations on the endmembers or the weights of the network, such as weight decay, are not as important as appropriate regularizations on the abundances. Initializing the weights of the decoder, e.g., using VCA, does not give better performance when using scale-invariant loss functions and randomly initialized encoder.

For enforcing the ASC, the softmax function with a scaling factor as in (10) is a good choice and also the normalization with absolute values in (21). Using Batch normalization is essential when training spectral unmixing autoencoders, and the batch size should be kept low for best performance.

The most important implementation choice is the choice of the fidelity term. For datasets with a lot of spectral variability, scale-invariant losses, such as SAD, will give better performance for mixing models such as the LMM. Not unless the method is specifically designed to work with spectral variability should a non-scale-invariant loss such as MSE be used.

Adversarial regularization, multistream architectures, and VAEs are becoming popular, and such architectures are the way forward in the authors' opinion. What has been said here regarding implementation choices also applies to these architectures.

V. Nonblind Methods Based on Autoencoders

Until now, only blind unmixing methods based on autoencoders have been discussed. Nonblind approaches are methods that estimate the abundances of endmembers but not the endmembers themselves. Many methods use neural networks for abundance estimation, and in most cases, such networks are essentially autoencoders or utilize autoencoders. This section will discuss various nonblind approaches based on autoencoders, but it is not meant to be a comprehensive review of such methods.

We have seen a direct correspondence between the weights of a linear decoder in a spectral unmixing autoencoder and the endmembers matrix, according to the LMM. If a set of endmembers is given, they can be used as the weights of a nontrainable linear decoder. It is then prevented from changing during training, which then only trains the encoder part. Such a network is still an autoencoder but works like a nonlinear regression method that determines the abundances maps corresponding to the given endmembers.

This approach is used in [122], where the encoder part of a spectral unmixing network is improved to utilize convolution layers to enable deeper architectures of the encoder, and custom spectral normalization layers are used instead of batch normalization. The decoder has fixed weights with endmembers coming from another method. The work [123] also uses a deep convolutional encoder and a fully connected decoder with fixed weights to perform supervised or nonblind unmixing. Another method based on this concept is in [124], where the encoder is a deep

convolutional network using a deep prior. The fixed decoder approach can be generalized to use a whole spectral library of endmembers, resulting in a combination of autoencoder unmixing and sparse-regression-based methods using a spectral library.

The technique in [125] is an example of such a method. The encoder has a deep convolutional architecture, but the last two layers are fully connected and use the softmax activation, essentially a classifier that picks the most suitable endmembers out of many. The input to the decoder layer has a much higher number of units than the number of endmembers to be estimated, but most of these activations will be zero since the output of the encoder is required to be sparse.

Another approach for abundance estimation utilizes MTL (multistream networks) and involves two different networks or tasks that share layers. A reconstructing autoencoder network is guided by another network that shares layers with the autoencoder in this approach. This guiding network is typically trained to learn the relationship between endmember candidates obtained by another endmember extraction method and their abundances as determined by the LMM. This network will guide the reconstructing autoencoder and make its encoder part encode spectra to abundances. The papers [126] and [127] are examples of this type of approach.

A variation of this is the method described in [128], where the guiding network is itself a spectral unmixing autoencoder that shares decoder weights with another spectral unmixing autoencoder. The guiding network is trained on endmember bundles obtained by VCA from superpixel segmentation of an HSI. Both networks are encouraged to produce latent codes that follow the Dirichlet distribution, i.e., sum-to-one.

This short overview of nonblind methods illustrates further how powerful and widespread the autoencoder architecture is becoming in HU.

VI. Experimental Results

This section provides an experimental foundation for the previous section and determines experimentally what makes unmixing autoencoders perform well. The performance of 11 blind unmixing methods will be evaluated and compared using four real datasets with accompanying reference endmembers and reference abundance maps.

Furthermore, all methods will be evaluated on four synthetic datasets having the same absolute ground truths but with different spectral variability. In addition, ablation experiments will be performed on a very general and simple spectral unmixing autoencoder. First, the ablation experiments aim to demonstrate what makes autoencoder-based methods powerful compared to traditional methods. Second, the effects of three different ways to enforce the ASC will be studied for two different activation functions for the encoder. Finally, the effects of nonlinear versus linear decoders will be investigated. The endmember extracted by methods will be evaluated using the mean spectral angle distance (mSAD), given by

$$\text{mSAD} = \frac{1}{R}\sum_{j=1}^{R} \arccos\left(\frac{\langle \hat{\boldsymbol{a}}_j, \boldsymbol{a}_j\rangle}{\|\hat{\boldsymbol{a}}_j\|_2 \|\boldsymbol{a}_j\|_2}\right) \tag{32}$$

where $\hat{\boldsymbol{a}}_i$ are the endmembers extracted by the method and $\boldsymbol{a}_i$ are the reference endmembers. The lower the mSAD, the higher the similarity. In the result tables, the SAD for individual endmembers will also be given. Abundance maps generated by the methods will be compared to reference abundance maps using the RMSE measure given by

$$\text{RMSE} = \sqrt{\frac{1}{R}\sum_{j=1}^{R} \|\mathbf{S}_j - \hat{\mathbf{S}}_j\|^2} \tag{33}$$

where $\hat{\mathbf{S}}_i$ are the abundance fractions of all pixels for endmember i and $\mathbf{S}_i$ are the reference abundance fractions.

The results of both endmember extraction and abundance maps generation for all the methods will be discussed, especially how they can be interpreted in light of the discussions in previous sections. Finally, the computation costs of the methods will be compared and discussed.

A. Datasets

All experiments were performed using four real HSIs described below and four synthetic datasets with increasing spectral variability. The methodology for determining the reference endmembers and abundance maps is described in [129]. The datasets are as follows.

1) *Samson:* The SAMSON sensor obtained this widely used dataset, and it is a cropped image from a larger image. The size is 95 × 95 pixels, and the number of bands is 145 covering the 401–889 nm wavelength range. The dataset has the following endmembers: Water, Soil, and Tree.
2) *Urban:* Obtained by the Hyperspectral Digital Image Collection Experiment [131] sensor, this image is 307 × 307 pixels and has 210 bands covering the 400–2500 nm wavelength range. After removing corrupted and noisy bands, 162 bands remain. Here, four, five, and six reference endmembers are used in experiments. Grass, Tree, Asphalt, and Roof were selected as the references for four endmembers. The five endmembers' reference additionally includes the Soil endmember, and the six endmembers' reference additionally includes the Soil and Metal endmembers.
3) *Houston:* This dataset is a cropped image from a larger one acquired over the University of Houston campus, Houston, TX, USA, in June 2012, and has 144 bands covering the wavelengths of 380–1050 nm with a spatial resolution of 2.5 m. The cropped image is 170 × 170 and is centered on the Robertson Stadium on the Houston campus and surrounding area. For this dataset, four endmembers are estimated, and the reference endmembers are selected from the averages of the 15 classification ground truth categories that come with the dataset.
4) *Apex:* This hyperspectral dataset is a cropped image from a much larger one acquired by the APEX [132] sensor during a clear day in June 2011 at an altitude of 4600 m above sea level with a heading of 56° in the vicinity of Baden, Switzerland. Two hundred eighty-five bands cover the wavelength range between 413 and 2412 nm, all

(a) (b) (c) (d)

Fig. 6. RGB images of the datasets used in the experiments. The Samson and Urban images were colored using the technique in [130]. The Apex and Houston images are actual RGB images. (a) Samson—simulated. (b) Urban—simulated. (c) Houston—actual. (d) Apex—actual.

usable. The cropped image is 300 × 300 pixels subimage cropped from the larger 1500×1000 image at location (70,650). Here, four endmembers are estimated, and the reference endmembers are Asphalt, Vegetation, Water, and Roof.

5) *Synthetic datasets:* These datasets use a 100 × 100 pixel cropped image from the reference abundance maps for the Urban dataset as ground truth for the abundance maps. A VAE is used for every pixel to generate samples of the reference endmembers with controllable spectral variability. Here, four endmembers are estimated, and the reference endmembers are Asphalt, Vegetation, Water, and Roof.

Simulated and actual RGB images of the real datasets used are shown in Fig. 6.

B. Methods Compared in Experiments

The methods compared in the experiments are listed in Table I. The source code for methods number 4, 6, and 7 could not be obtained, so the authors implemented them. Method 12, sparsity-regularized NMF, is a benchmark traditional method. The Endnet method was not published as a usable blind method as the generated abundance maps were of low quality due to particular implementation of a batch normalization layer and high ℓ_1-norm sparsity regularization. In our implementation, the batch normalization does not ruin the abundance maps, and neither does the chosen optimal strength of the ℓ_1 regularization. Therefore, the method produces usable abundance maps in our implementation. The method mDAE was initially implemented in MATLAB, but we used the Tensorflow DL framework for our implementation. Methods 1–4 are randomly initialized. All other methods are initialized using VCA.

C. Ablation Experiments

These experiments use a simple spectral unmixing autoencoder. Its architecture is shown in Fig. 7.

The encoder only has a single layer, and the activation can be either the ReLU or the LReLU activation with slope parameter 0.1 for the negative preactivations. A batch normalization layer is used after the activation to speed up learning. The ASC is enforced using one of the following: the softmax function, simple normalization according to (19), or normalization of absolute values according to (21). The loss function is either the SAD or MSE function.

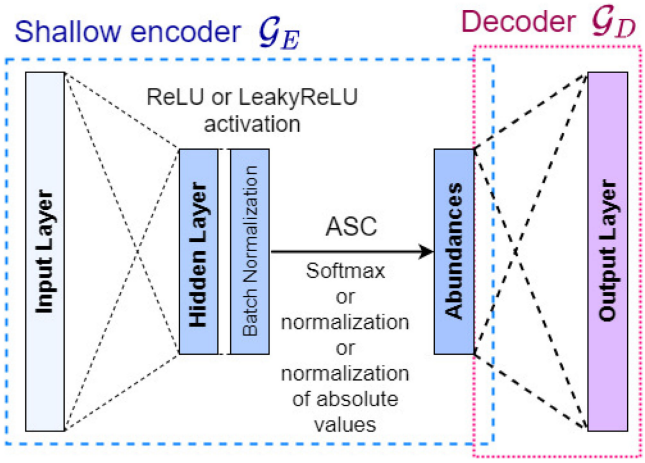


Fig. 7. Architecture of the simple autoencoder used in the ablation experiments.

1) Comparison With NMF-$\ell_{1/2}$: In this experiment, the simple autoencoder is compared with the NMF-$\ell_{1/2}$, a widely used NMF method that is sparsity regularized. The ASC will be enforced using the softmax function, and the ReLU activation function is used for the encoder. The loss function is either MSE or SAD. Fig. 8 shows the results of the experiment for the Urban and the Samson datasets.

Fig. 8 shows that a simple spectral unmixing autoencoder using the MSE function for the fidelity term and LReLU activation is only slightly better than the NMF-$\ell_{1/2}$ method for the Urban dataset and slightly worse for the Samson dataset. When the autoencoder uses the SAD loss function and the LReLU activation, the performance becomes significantly better than the performance of the NMF method. However, when the activation function is the ReLU, we see a much worse performance with the autoencoder.

This is because ReLU units can get stuck when using the ReLU activation and batch normalization. The ReLU activation has a zero gradient for negative preactivations, while the LReLU

TABLE I
METHODS USED IN THE EXPERIMENTS

#	Name	Architecture	Loss	ASC	Regularization	Spatial	Implementation
1	DAEU [73]	Deep encoder	SAD	Uses (19)	None	No	Developed by the authors
2	MTLAEU [74]	Branched deep encoder utilizing multi-task learning with shared decoder	SAD	Scaled softmax	None	Yes	Developed by the authors
3	CNNAEU [70]	Fully CNN encoder and decoder	SAD	Scaled softmax	None	Yes	Developed by the authors
4	SIDAEU [91]	Shallow encoder	SID	Uses (19)	dynamical thresholding	No	Implemented by the authors
5	OSPAEU [76]	Deep encoder	Hyper-Laplacian	Uses (21)	Orthogonal sparse prior	No	Implemented by the authors
6	Endnet [80]	Single layer encoder with a custom matrix-vector product	SAD + small MSE term	Softmax	l_1 on abund. and l_2 on endm.	No	Implemented by the authors
7	mDAU [18]	Marginalized denoising autoencoder in cascade with an unmixing autoencoder with tied weights	MSE	Uses augmented matrices	None	No	Implemented by the authors
8	NLAEU [38]	Deep autoencoder nonlinear unmixing with a decoder having additionally two extra layers that model the interactions between the endmembers according to (4)	MSE	Uses (21) but no ANC	None	No	Code made available to the authors
9	SNSA [85]	Stacked nonnegative sparse autoencoder with outlier detection prior to unmixing	MSE	Uses augmented matrices	minimum volume	No	Code made available to the authors
10	uDAS [68]	An untied denoising autoencoder with sparsity	MSE	Uses augmented matrices	ℓ_{21} on endmembers	No	Code for the method available on GitHub
11	DAEN [92]	The method uses an variational autoencoder for spectral unmixing which uses the VAE indirectly to enforce the ASC constraint. The method optionally uses stacked autoencoders to prepare a outlier free training data set for the unmixing autoencoder	MSE	Uses VAE for ASC and ANC	Minimum volume	No	Code made available to the authors
12	NMF-$\ell_{1/2}$ [52]	Sparsity constrained nonnegative matrix factorization ($\ell_{1/2}$-NMF) [55]	MSE	Uses augmented matrices	$\ell_{1/2}$ on abundances	No	Code is available on the web

always has a nonzero gradient. This shows that care should be taken when using the ReLU activation.

This experiment largely answers why autoencoder methods using the SAD loss can achieve much better unmixing performance than autoencoder methods using the MSE loss function. The scale invariance of the SAD loss function makes the autoencoder able to handle the spectral variability inherent in most real HSIs.

2) Comparison Between Different ASC and Activation Combinations: In this experiment, the effect of different ways to enforce the ASC will be compared for both MSE and SAD losses and using the ReLU and LReLU activations. Three different ways of enforcing the ASC will be tested:

a) using the softmax function;
b) normalizing the activations according to (19);
c) normalizing the absolute value of the activations (absnorm) according to (21).

The results of this experiment are shown in Fig. 9 . The results are interesting but not surprising. For LReLU and the Urban dataset, the difference between the different ways of enforcing the ASC is negligible. However, for the Samson dataset with LReLU, softmax is the best option for both losses.

This difference is because LReLU allows negative activations, and for the Samson dataset, there seems to be a much more preference for negative activations than for the Urban dataset. Negative activations in the layer just before the ASC is enforced will negatively affect the simple normalization as it will break the ANC constraint. It might be argued that LReLU activation will lead to less sparse abundance maps than if ReLU activation combined with batch normalization is used.

A ReLU activation for the last layer might work well in deep encoders and possibly give more sparse maps. The absnorm method is not as affected, but softmax does not care about the signs of the activations and will always return a vector that sums-to-one. It seems that the neural network can best adapt to the softmax method of enforcing the ASC.

The ReLU part of this experiment is plagued by the ReLU units getting stuck, certain endmembers stopping converging,

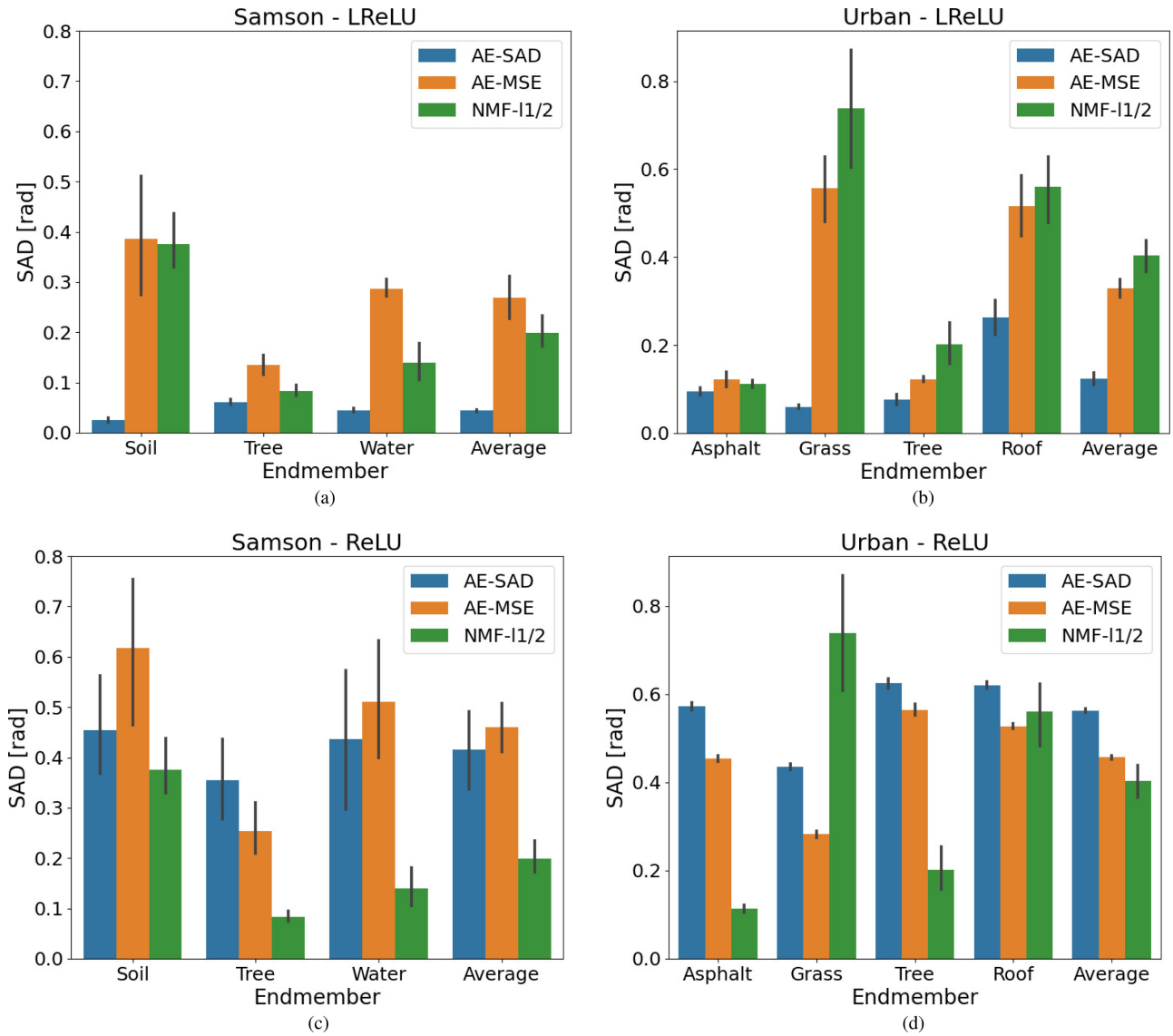


Fig. 8. Comparison between a simple autoencoder, using either the MSE or SAD function for the fidelity term, and the NMF-$\ell_{1/2}$ method. The autoencoder uses either the LReLU or ReLU activation. The black bars show the standard deviation. All experiments consisted of 25 runs. (a) and (c) Samson dataset. (b) and (d) Urban dataset.

and is hard to interpret. This experiment shows that for some datasets, it can matter how the ASC is enforced. Using the softmax method gives good performance in both datasets and can be considered the safest option to enforce the ASC. By using a scaling factor for the activations entering the softmax function, the function can be soft thresholding and act very similarly to ℓ_1-sparsity regularization.

3) Linear and Nonlinear Decoders: Until now, we have mainly discussed autoencoders having linear decoders, where the endmembers are the weights of the decoder at the end of training. It is possible to have multilayered and nonlinear decoders in spectral unmixing autoencoders. If the decoder has multiple layers, the endmembers cannot be extracted as the weights of any layers. However, the endmembers can be obtained from the decoder by decoding one-hot vectors, i.e., the abundances of pure pixels.

In this final ablation experiment, we will compare two autoencoders with identical encoders, but one has a linear decoder and the other a nonlinear decoder. In the case of the nonlinear decoder, the endmembers are obtained by a prediction by the decoder part on the identity matrix (abundances of pure pixels). The nonlinear decoder has three layers: one with $4\times R$ units and two layers having B units.

Three datasets are used, the Urban, Samson, and Houston datasets, and the LReLU activation was used for the encoder. Four different batch sizes are used: 6, 10, 15, and 25. Fig. 10 shows the results of the experiment. The figures in the top row are the mSAD of extracted endmembers, and the figures in the bottom row show the RMSE for the abundance maps.

What first stands out when looking at Fig. 10 is that the batch size is a hyperparameter that can affect the performance, especially for the endmembers. The best SAD performance on

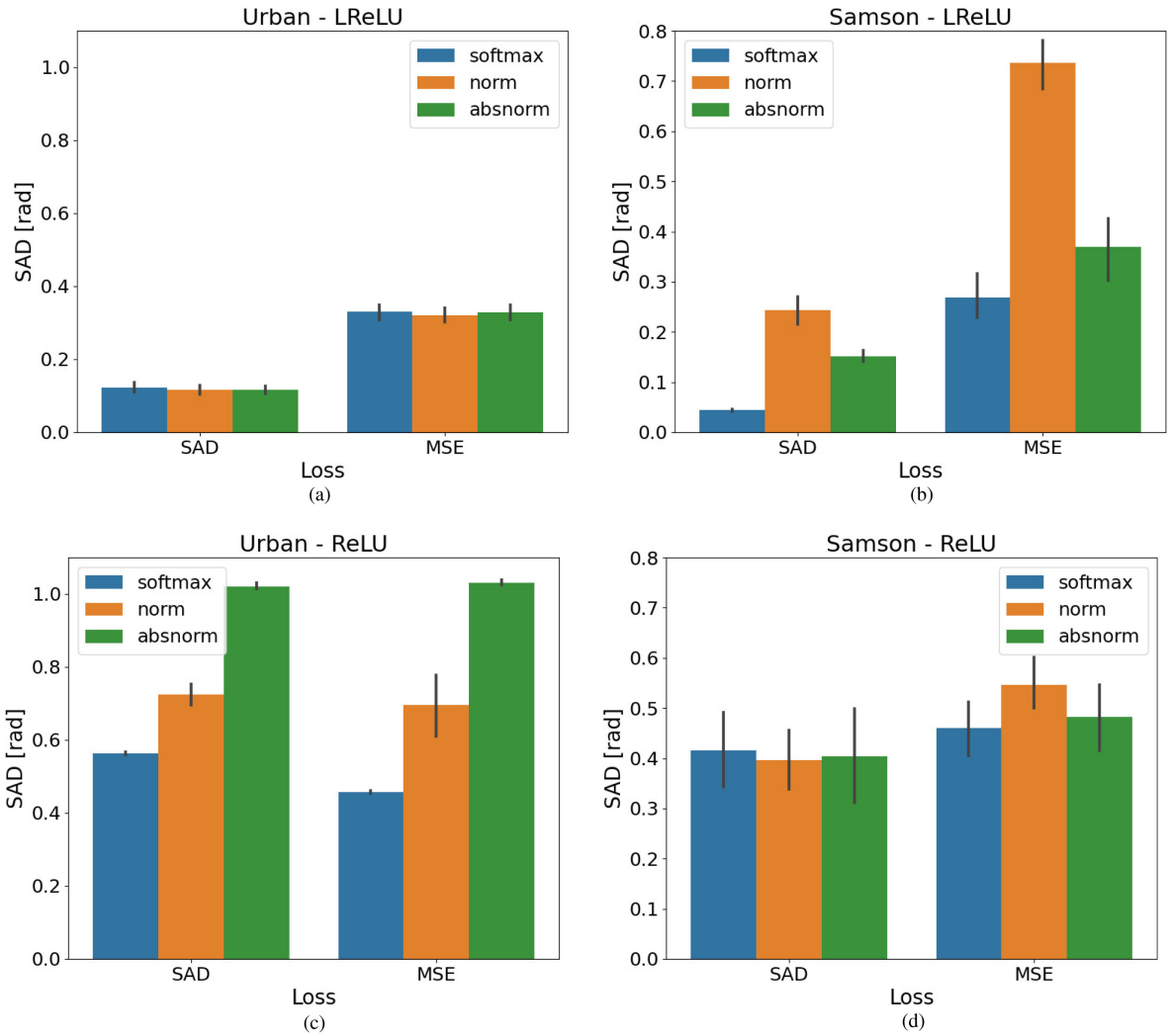


Fig. 9. Comparison between three ways of enforcing the ASC constraint for both the MSE and SAD losses and using the LReLU or ReLU activations. The black bars show the standard deviation. All experiments consisted of 25 runs. (a) Urban dataset with LReLU. (b) Samson dataset with LReLU. (c) Urban dataset with ReLU. (d) Samson dataset with ReLU.

the Urban and Houston images is for small batch sizes, while for Samson, it is best to use moderately large batch sizes. It can be seen that for the Urban image, having more parameters in the decoder is beneficial for the mSAD scores of extracted endmembers.

The case of the Samson image is not as clear and depends on the batch size. The results for the Houston image show that a nonlinear decoder gives better mSAD scores, indicating that this image has more nonlinear mixing than the other images. Regarding the quality of the abundance maps, the experiment is not conclusive. However, it is interesting to see that a nonlinear decoder that does not correspond to the LMM can benefit endmember extraction, while the quality of abundance maps does not change much. A deeper encoder in combination with the nonlinear decoder might give better abundance maps.

4) Summary: The results of the ablation experiments can now be summarized as follows.

a) Autoencoder methods should use a scale-invariant loss such as SAD, and the use of MSE should be avoided unless the methods specifically handle spectral variability through an extended LMM.
b) Using the LReLU activation in combination with softmax to enforce the ASC works well, but abundance maps might be less sparse than if ReLU is used. Care should be taken when using the ReLU activation to prevent units from getting stuck.
c) Batch size affects the performance of unmixing autoencoders and is dataset dependent.
d) Using a nonlinear decoder with more parameters can benefit the quality of extracted endmembers, especially for images with nonlinear mixing. The effect on the quality of the abundance maps is not very significant.

D. Synthetic Datasets With Varying Spectral Variability

For this experiment, a VAE was used to learn the Urban image and the reference endmembers (four endmembers) were encoded into codes in the 2-D latent space of the encoder.

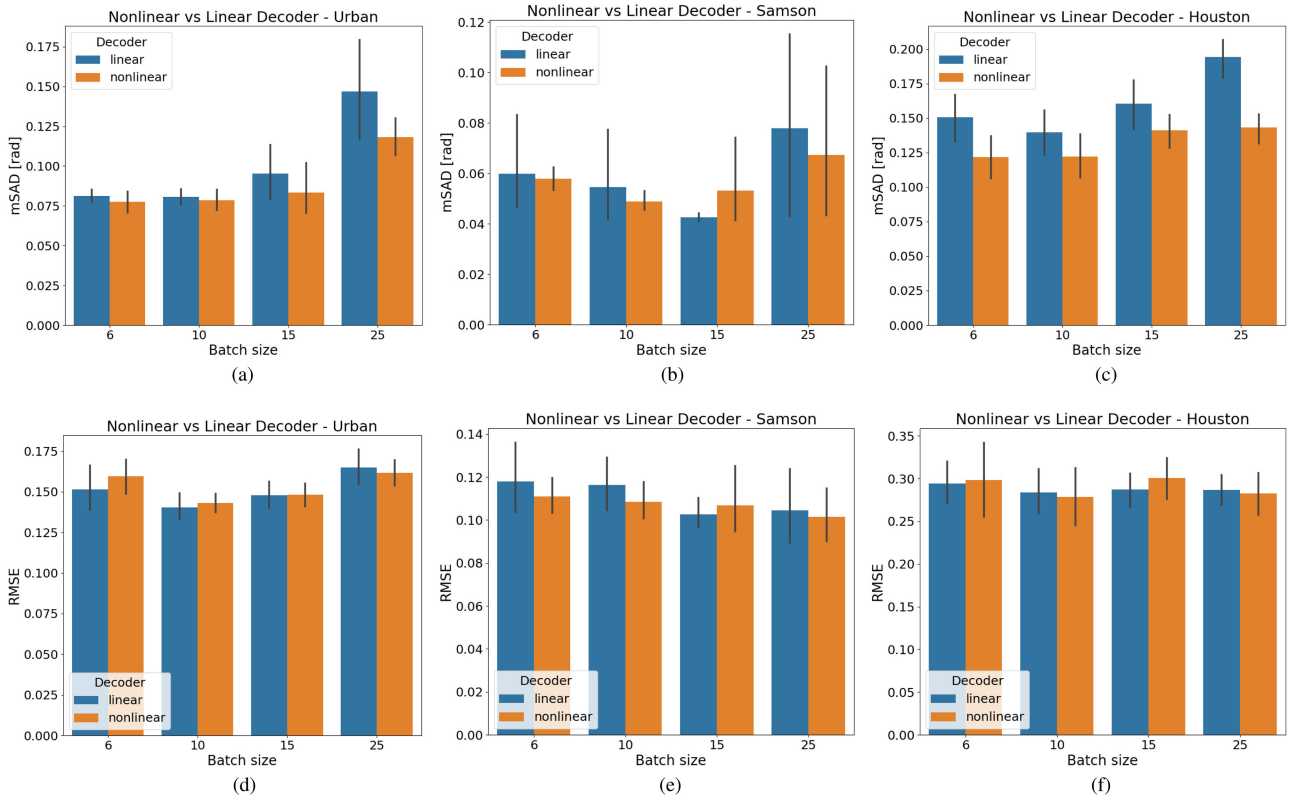


Fig. 10. Comparison between having a linear or nonlinear decoder. The encoder uses the LReLU activation. Four different batch sizes are tested. The graphs in the top row show the average SAD for extracted endmembers, while the bottom row shows the RMSE of the abundances. The black lines show the standard deviation. All experiments consisted of 25 runs. (a) Urban dataset—SAD. (b) Samson dataset—SAD. (c) Houston dataset—SAD. (d) Urban dataset—RMSE. (e) Samson dataset—RMSE. (f) Houston dataset—RMSE.

By sampling within a circle in the latent space centered on the codes of the reference endmembers, new variations of the endmembers can be generated using the generator part. The radius of the sampling circle controls the spectral variability of the sampled endmembers. A 100×100 pixel crop of the Urban dataset reference abundance maps was then used to provide the ground truth abundance fractions for every pixel and new spectra generated by using sampled endmembers for every pixel in the abundance maps.

Four such datasets were generated with sampling radii of 0, 0.1, 0.2, and 0.4. Fig. 11 shows examples of sampled endmembers for two different radii. Every method was run 25 times for each of these synthetic datasets, and a bar plot of the average SAD for each dataset is shown in Fig. 12.

Fig. 12 is very informative. It shows well the effect of spectral variability on the performance of methods using the MSE objective function. The six rightmost methods all use the MSE fidelity term, and they all achieve very good mSAD for zero spectral variability. Even the mDAE method that performs very poorly on real datasets achieves good mSAD for zero spectral variability. It is striking to see how similar the performance of the NAE-VCA, SNSA, uDAS, and DAEN is to NMF-$\ell_{1/2}$ on these datasets. As the spectral variability increases, the non-scale-invariant methods perform increasingly worse.

The methods using scale-invariant fidelity terms exhibit strong robustness against spectral variability. Only DAEU performs increasingly worse with increasing spectral variability. The methods CNNAEU and MTAEU seem to benefit from being spectral–spatial as demonstrated by the low mSAD score. The Endnet method also has good performance and is not affected by the increasing spectral variability. OSPAEU method using the hyper-Laplacian loss shows some sensitivity to the spectral variability, which is to be expected, but less than the methods using the MSE loss. The results of this experiment indicate that experiments with synthetic datasets having low or zero spectral variability are not a good benchmark for autoencoder methods in general. Low spectral variability results in a bias toward non-scale-invariant fidelity terms on such datasets that would not reflect real datasets.

E. Experiments With Real HSIs

1) Endmember Extraction: When evaluating the endmember extraction performance of the methods, the following three qualities will be in focus:

a) The average SAD from the reference endmembers;
b) The variance of the average SAD score. A good method should have a low variance;
c) The accuracy of the method is studied.

Accuracy is not quite the same thing as consistency. Here, we are looking at if the method finds the same correct endmembers every time or most of the time. A consistent method is a method

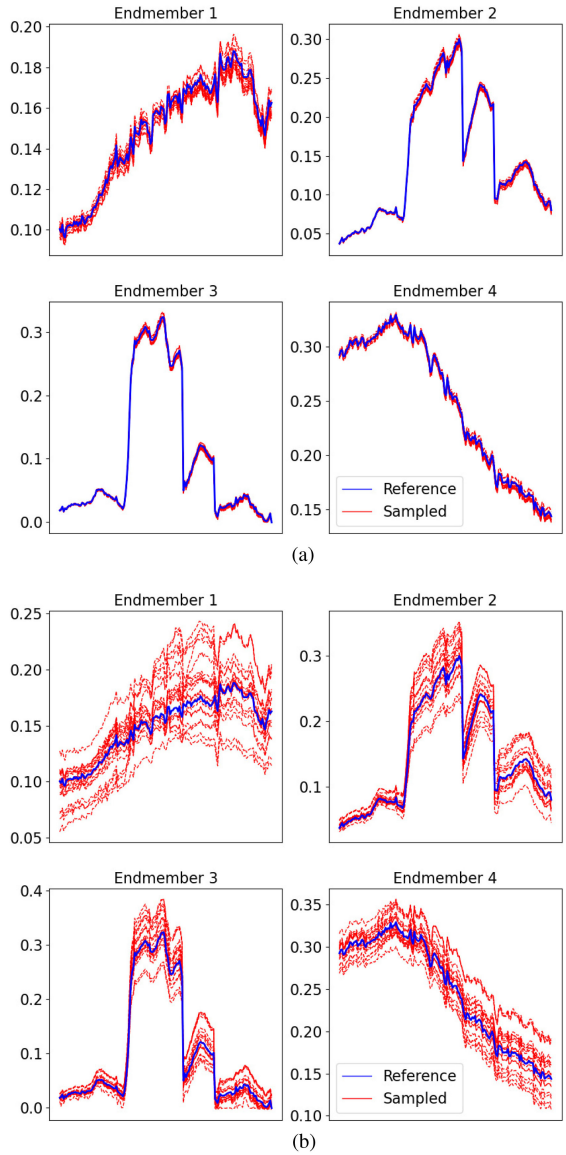


Fig. 11. Sampled endmembers for the synthetic experiment based on the Urban image for two different sampling radii. (a) Sampled endmembers using radius = 0.05. (b) Sampled endmembers using radius = 0.4.

that finds similar endmembers in each run, i.e., has low variance. Plotting all extracted endmembers for every run and showing these together in a single figure is an excellent way to evaluate the methods. The Urban, Houston, and Apex datasets were used to evaluate the endmember extraction performance of the methods.

a) Urban dataset: In the first experiment, we use the Urban dataset, extract the first four, then five, and finally six endmembers, and compare the extracted endmembers to the reference endmembers. As the number of estimated endmembers increases, it becomes harder to get good solutions. It is interesting to see how the solutions for the initial endmembers behave as new additional endmembers are estimated. Figs. 13–15 show extracted endmembers for all the methods for the Urban dataset with four, five, and six reference endmembers, respectively. The reference endmembers are shown with red color. Tables II– IV tabulate the average SAD from reference endmembers in radians along with the standard deviation for the Urban dataset with four, five, and six reference endmembers, respectively.

When estimating four endmembers, CNNAEU, MTAEU, and Endnet show good performance, low variance, and good accuracy. The DAEN method has good consistency but bad accuracy. The mDAE method, which is the only method with tied weights, is very unstable, and this instability can most likely be attributed to tying the weights and spectral variability in the image. The method NLAEU has highly negative endmembers as it lacks a nonnegativity constraint. This makes the method essentially unusable when not initialized and trained carefully.

The methods uDAS, SNSA, and DAEN interestingly all yield similar endmembers. The DAEU and SIDAEU methods have medium consistency and not very good accuracy, i.e., mSAD score. The OSPAEU technique has bad accuracy because the "Roof" endmember oscillates between three different solutions.

As the number of estimated endmembers is increased to five, the average SAD scores of all methods increase and many methods' accuracy decreases significantly. Endmembers that previously had good accuracy become unstable. Again, CNNAEU, MTAEU, and Endnet show the best performance and accuracy, with CNNAEU having the best accuracy and consistency and Endnet having the second-best consistency. MTAEU has one inconsistent run, where it estimates the Asphalt endmember as a "Tree" endmember variant, resulting in worse mSAD and lower consistency.

The OSPAEU technique estimates the new "Soil" endmember quite well, but the "Asphalt" endmember, which previously, when estimating four endmembers was good, becomes unstable. Interestingly, the NLAEU method, which previously yielded highly negative endmembers, estimates them consistently mostly nonnegative. Still, the consistency is not good. DAEN, uDAS, and SNSA seem to have trouble estimating the new "Soil" endmembers.

When estimating six endmembers, the trend of increasing average SAD and decreasing consistency continues. The best-performing methods are again CNNAEU, MTAEU, and Endnet. CNNAEU and MTAEU have greater difficulty estimating the "Metal" endmember than the Endnet method, but they retain excellent accuracy and good SAD scores for the five other endmembers. Both methods are spectral–spatial methods. For Endnet, the cost of a good "Metal" endmember is a poorer solution for the "Soil" endmember. All the other methods show bad accuracy for most endmembers other than "Tree," which is the best-represented endmember in the Urban scene.

Interestingly, the only method to consistently estimate the "Metal" endmember well is the Endnet method. This method uses a highly customized hidden layer in the encoder, where the standard dot product used to calculate the layer's activations from its weight matrix and inputs is replaced with a custom and more "spectrally discriminating" dot product based on the SAD similarity measure. For some reason, the DAEN method sometimes took too long to extract six endmembers for this dataset, so it had to be omitted.

b) Houston dataset: The extracted endmembers by all methods for the Houston dataset are shown in Fig. 16, and the SAD scores for individual endmembers along with the average SAD

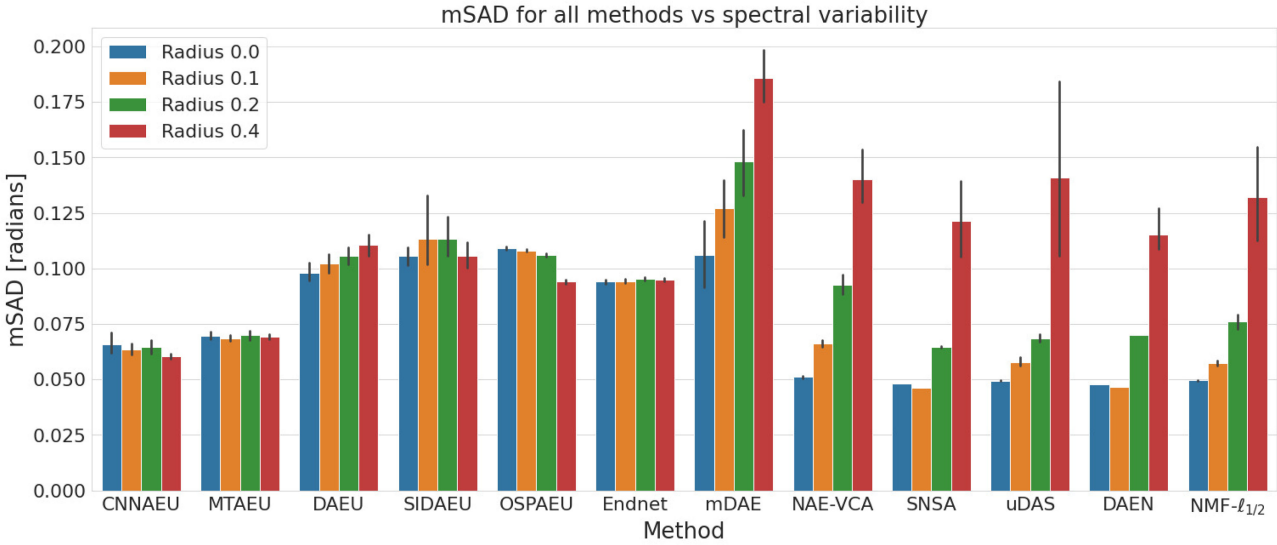


Fig. 12. Bar plot showing the average SAD for the four synthetic datasets. The spectral variability increases with increasing sampling radius. The six rightmost methods all use the MSE objective function. The black vertical lines show the standard deviation. All experiments consisted of 25 runs.

TABLE II
MEAN SAD FROM REFERENCE ENDMEMBERS IN RADIANS ALONG WITH THE STANDARD DEVIATION FOR ALL METHODS FOR THE URBAN DATASET WITH FOUR REFERENCE ENDMEMBERS

Endmember Method	Asphalt	Grass	Tree	Roof	Average
CNNAEU	0.0575±0.0059	0.0366±0.0048	0.0321±0.0040	0.0332±0.0067	0.0398±0.0031
MTAEU	0.0843±0.0047	0.0421±0.0037	0.0539±0.0040	0.0415±0.0046	0.0555±0.0020
DAEU	0.0720±0.0240	0.0769±0.0291	0.0707±0.0297	0.2155±0.0862	0.1088±0.0239
DAEN	0.2996±0.0023	1.2468±0.0076	0.0800±0.0003	0.1057±0.0024	0.4330±0.0017
SIDAEU	0.1039±0.0386	0.0995±0.0663	0.0640±0.0234	0.2150±0.1206	0.1206±0.0323
OSPAEU	0.1290±0.0126	0.0901±0.0193	0.1350±0.0600	0.2005±0.2722	0.1386±0.0635
Endnet	0.0618±0.0118	0.0538±0.0169	0.0363±0.0045	0.0508±0.0160	0.0507±0.0050
mDAE	0.1626±0.0378	0.8122±0.3883	0.4852±0.4101	0.4945±0.1394	0.4886±0.0750
NAE-VCA	0.2023±0.1126	0.4304±0.3105	0.934±0.3909	0.2274±0.1049	0.4485±0.0388
SNSA	0.2858±0.0214	1.2022±0.1571	0.0788±0.0014	0.1202±0.0577	0.4218±0.0279
uDAS	0.2104±0.0198	1.0720±0.1988	0.1401±0.0251	0.2650±0.0428	0.4219±0.0629
NMF-$\ell_{1/2}$	0.1161±0.0238	0.5730±0.3293	0.1571±0.0940	0.4891±0.2107	0.3338±0.0657

The number of runs is 25. Best results are in red and the second best in blue.

TABLE III
MEAN SAD FROM REFERENCE ENDMEMBERS IN RADIANS ALONG WITH THE STANDARD DEVIATION FOR ALL METHODS FOR THE URBAN DATASET WITH FIVE REFERENCE ENDMEMBERS

Endmember Method	Asphalt	Grass	Tree	Roof	Soil	Average
CNNAEU	0.0379±0.0031	0.0611±0.0097	0.1320±0.0085	0.0404±0.0069	0.1075±0.0098	0.0758±0.0034
MTAEU	0.1211±0.2363	0.0565±0.0354	0.1132±0.0168	0.0661±0.0096	0.1652±0.0467	0.1044±0.0482
DAEU	0.1164±0.0603	0.2005±0.0709	0.1404±0.0343	0.4171±0.0821	0.3157±0.1796	0.2380±0.0327
DAEN	0.2267±0.0642	0.5417±0.0894	0.1635±0.0026	0.1244±0.0178	0.8648±0.0516	0.3842±0.0164
SIDAEU	0.2383±0.2266	0.2001±0.0634	0.0986±0.0210	0.1511±0.1116	0.4342±0.2715	0.2245±0.0302
OSPAEU	0.3597±0.3319	0.0772±0.0138	0.0894±0.0306	0.3011±0.2869	0.0507±0.0123	0.1756±0.0614
Endnet	0.0368±0.0056	0.0406±0.0117	0.1171±0.0112	0.1313±0.0463	0.1507±0.0330	0.0953±0.0087
mDAE	0.2218±0.0623	0.9421±0.3245	0.7784±0.4005	0.5338±0.1578	0.6751±0.2270	0.6303±0.0777
NAE-VCA	0.2147±0.1472	0.8616±0.5259	0.3360±0.2192	0.1212±0.0513	0.4691±0.3534	0.4005±0.0474
SNSA	0.7912±0.3134	0.7764±0.1663	0.1512±0.0084	0.2163±0.1163	0.4402±0.1480	0.4750±0.0514
uDAS	0.1666±0.0403	0.5917±0.4973	0.1701±0.0472	0.2133±0.0476	0.7817±0.4235	0.3847±0.0409
NMF-$\ell_{1/2}$	0.2423±0.2066	0.7433±0.5346	0.3612±0.4552	0.2350±0.2113	0.2568±0.2319	0.3677±0.0512

The number of runs is 25. Best results are in red and the second best in blue.

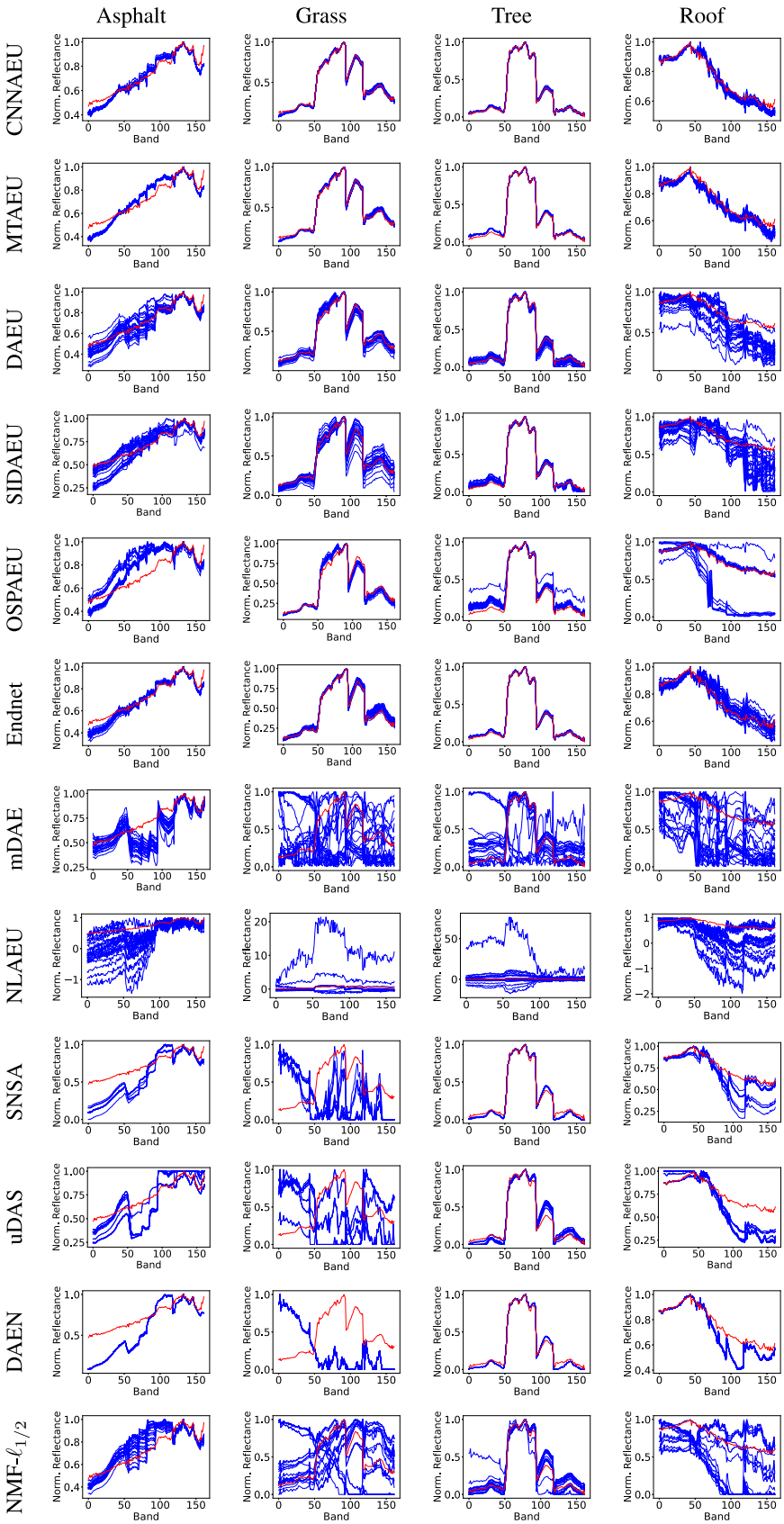


Fig. 13. Plots of all extracted endmembers by all methods (blue curves) and the reference endmembers (red) for the Urban dataset with four reference endmembers. The number of runs is 25.

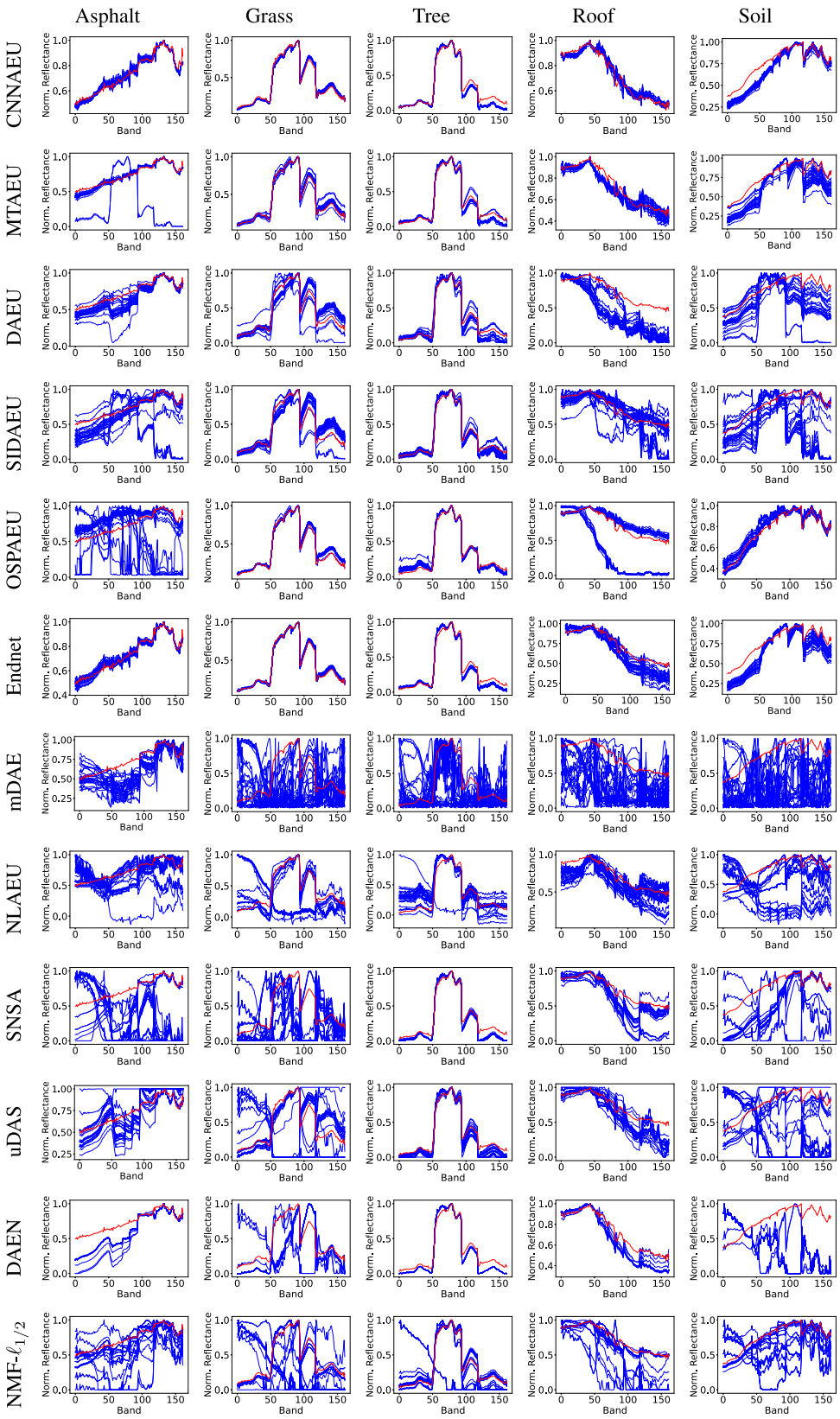


Fig. 14. Plots of all extracted endmembers by all methods (blue curves) and the reference endmembers (red) for the Urban dataset with five reference endmembers. The number of runs is 25.

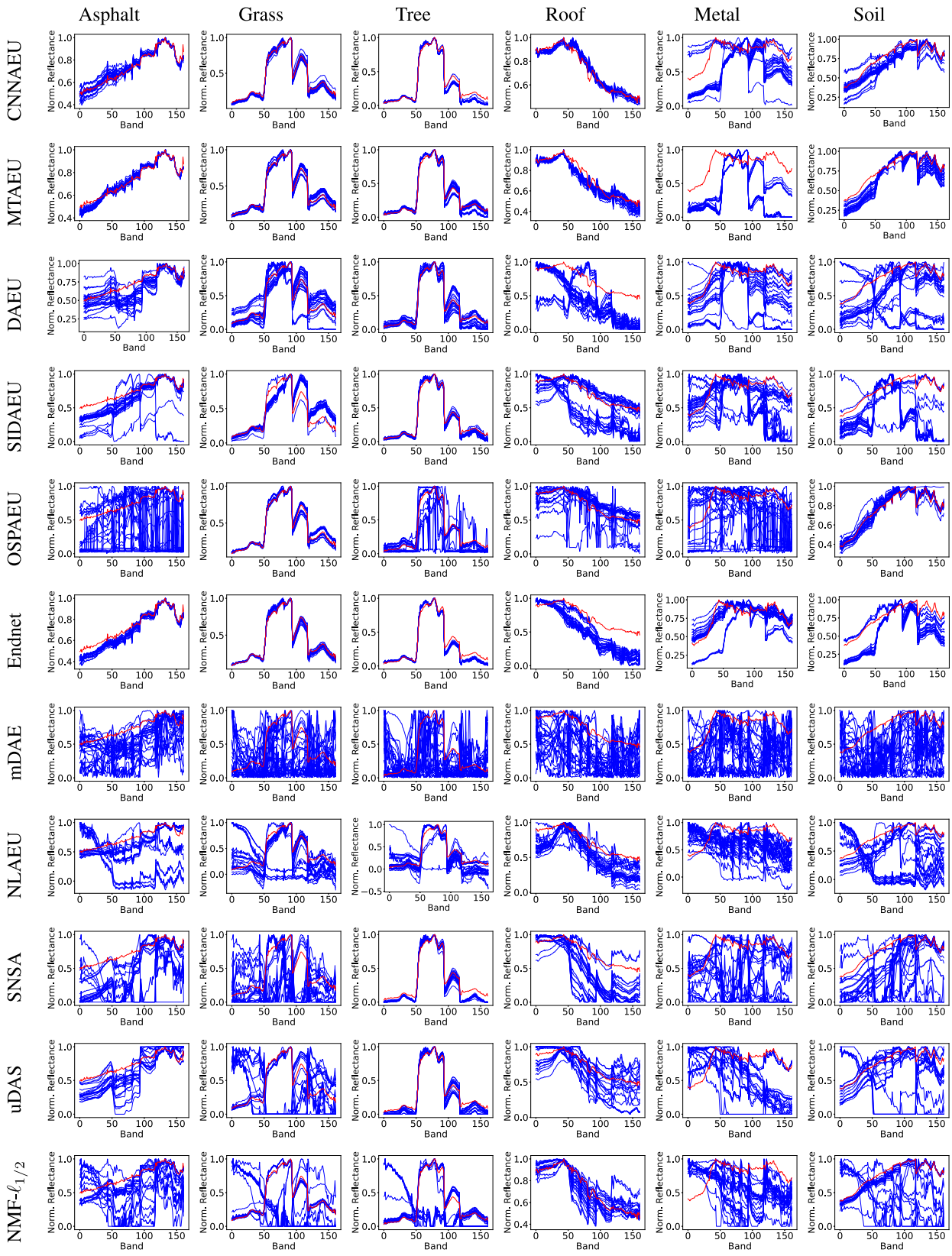


Fig. 15. Plots of all extracted endmembers by all methods (blue curves) and the reference endmembers (red) for the Urban dataset with six reference endmembers. The number of runs is 25.

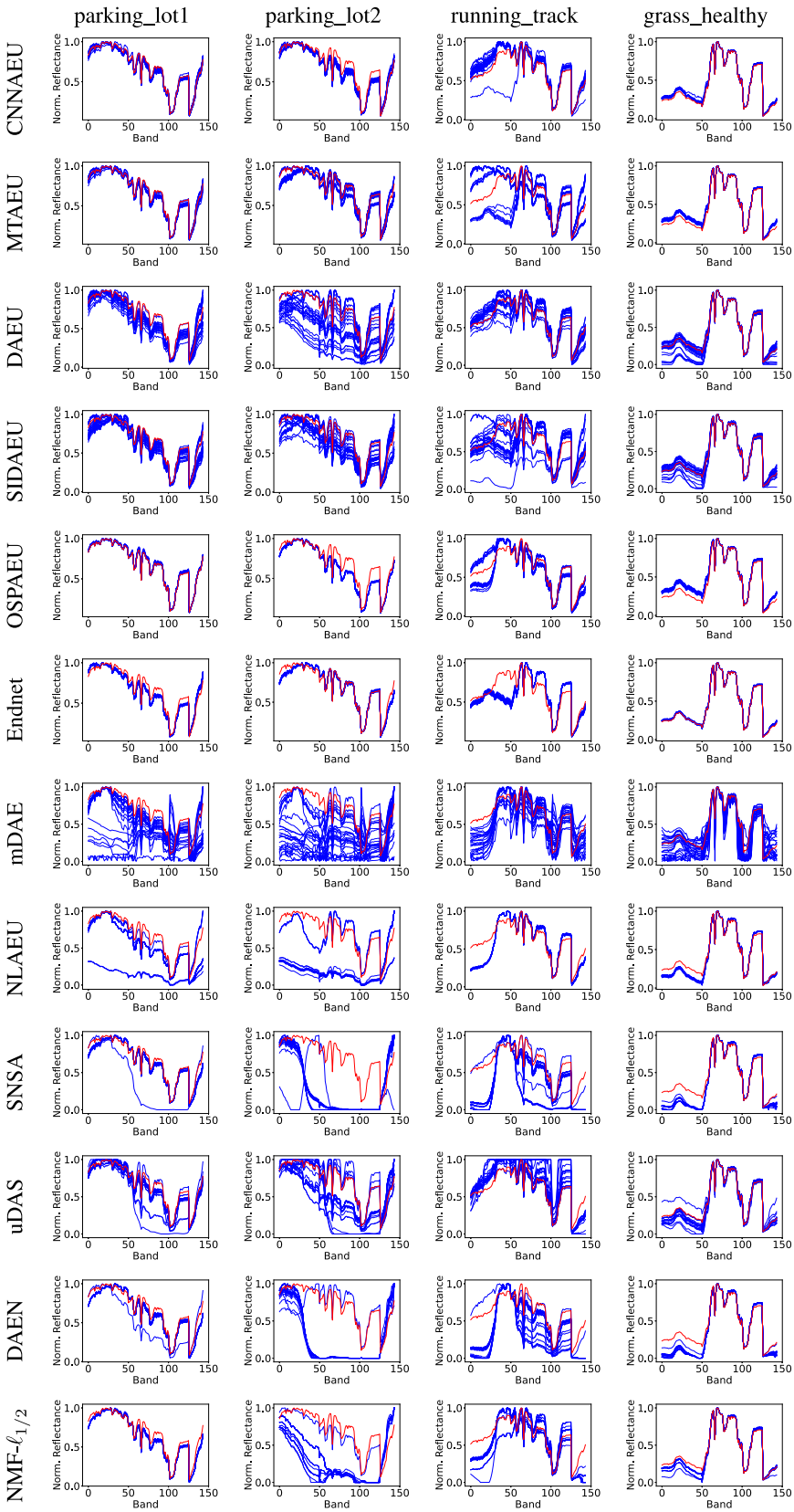


Fig. 16. Plots of all extracted endmembers by all methods (blue curves) and the reference endmembers (red) for the Houston dataset with four reference endmembers. The number of runs is 25.

TABLE IV
MEAN SAD FROM REFERENCE ENDMEMBERS IN RADIANS ALONG WITH THE STANDARD DEVIATION FOR ALL METHODS (EXCEPT DAEN) FOR THE URBAN DATASET WITH SIX REFERENCE ENDMEMBERS

Endmember Method	Asphalt	Grass	Tree	Roof	Metal	Soil	Average
CNNAEU	0.0435±0.0099	0.0491±0.0248	0.1339±0.0166	0.0430±0.0081	0.3628±0.1133	0.1030±0.0304	0.1226±0.0173
MTAEU	0.0449±0.0070	0.0751±0.0213	0.0768±0.0398	0.0610±0.0078	0.7070±0.1808	0.1211±0.0408	0.1810±0.0302
DAEU	0.1625±0.0694	0.2164±0.1207	0.1298±0.0464	0.4398±0.0690	0.3801±0.3045	0.4534±0.3186	0.2970±0.0419
SIDAEU	0.2189±0.2070	0.2371±0.0324	0.0874±0.0148	0.2797±0.1756	0.3750±0.2469	0.5802±0.2498	0.2964±0.0286
OSPAEU	0.6643±0.2984	0.0676±0.0147	0.2678±0.2685	0.1437±0.1375	0.5003±0.2567	0.0450±0.0147	0.2815±0.0711
Endnet	0.0583±0.0103	0.0549±0.0215	0.1245±0.0110	0.3107±0.0605	0.1505±0.0929	0.2434±0.0681	0.1571±0.0128
mDAE	0.3135±0.1483	0.7965±0.3205	0.9032±0.3802	0.4987±0.1753	0.5266±0.1990	0.5577±0.2287	0.5994±0.0606
NAE-VCA	0.3144±0.2901	0.5095±0.4841	0.2878±0.2299	0.1475±0.0457	0.3806±0.1412	0.5158±0.3964	0.3593±0.0440
SNSA	0.5239±0.2209	0.7632±0.2685	0.1626±0.0218	0.4614±0.1543	0.6706±0.3076	0.5311±0.2976	0.5188±0.0377
uDAS	0.2429±0.0895	0.8975±0.5233	0.1408±0.0579	0.2292±0.1052	0.5613±0.2055	0.3699±0.4091	0.4069±0.0584
NMF-$\ell_{1/2}$	0.3220±0.2514	0.7347±0.5753	0.4559±0.5594	0.0858±0.0489	0.5260±0.1694	0.3352±0.3704	0.4100±0.0309

The number of runs is 25. Best results are in red and the second best in blue.

TABLE V
SAD FROM REFERENCE ENDMEMBERS IN RADIANS ALONG WITH THE STANDARD DEVIATION FOR ALL METHODS FOR THE HOUSTON DATASET WITH FOUR REFERENCE ENDMEMBERS

Endmember Method	parking_lot1	parking_lot2	running_track	grass_healthy	Average
CNNAEU	0.0313±0.0051	0.0750±0.0096	0.0790±0.0721	0.0410±0.0092	0.0566±0.0214
MTAEU	0.0427±0.0092	0.0786±0.0130	0.2894±0.0905	0.0642±0.0106	0.1187±0.0231
DAEU	0.1263±0.0359	0.2798±0.1620	0.1140±0.0397	0.0937±0.0842	0.1534±0.0361
DAEN	0.0910±0.0401	0.8354±0.1277	0.4425±0.1521	0.2458±0.0272	0.4037±0.0580
SIDAEU	0.0965±0.0231	0.1561±0.0465	0.2534±0.1375	0.0691±0.0559	0.1438±0.0355
OSPAEU	0.0127±0.0031	0.1058±0.0026	0.1340±0.0255	0.1028±0.0114	0.0888±0.0063
Endnet	0.0800±0.0075	0.0635±0.0050	0.2339±0.0119	0.0165±0.0032	0.0985±0.0036
mDAE	0.3353±0.2477	0.5537±0.2683	0.2768±0.0826	0.2435±0.0821	0.3523±0.0606
NAE-VCA	0.2423±0.1817	0.6680±0.1632	0.2159±0.0053	0.1156±0.0098	0.3105±0.0083
SNSA	0.0971±0.0972	0.8269±0.0374	0.4958±0.2028	0.2485±0.0177	0.4171±0.0580
uDAS	0.1130±0.1123	0.3039±0.1523	0.1550±0.0541	0.1080±0.0572	0.1700±0.0396
NMF-$\ell_{1/2}$	0.0671±0.0054	0.6981±0.1519	0.2460±0.0568	0.0842±0.0311	0.2738±0.0317

The number of runs is 25. Best results are in red and the second best in blue.

are tabulated in Table V. The three best-performing methods for this dataset are CNNAEU, OSPAEU, and Endnet. Endnet has excellent consistency, but its solution for the "running_track" endmember is different from the reference. CNNAEU also has good accuracy, with only one run out of 25 coming up with a different solution for the "running_track" endmember, and it also achieves the lowest average SAD score. OSPAEU has some difficulty with the "running_track" endmember and oscillates between two solutions, one of them being the reference endmember. The OSPAEU method achieves the second-lowest mSAD score for the dataset.

MTAEU also has trouble with the "running_track" endmember, showing similar behavior as OSPAEU. The DAEU and SIDAEU methods have a reasonably good consistency. mDAE does better on this dataset than it did on Urban but is still unstable. The methods SNSA, uDAS, and DAEN all have trouble with the parking_lot2 endmember, which is highly correlated with the parking_lot1 endmember. In addition, only uDAS is extracting the "running_track" endmember similar to the reference. NLAEU has some trouble with the parking_lot endmembers, and the accuracy is not good with these endmembers. Its solution for the "running_track" endmember has a good consistency, but it is not the reference endmember.

Overall, the methods are doing better on the average on this dataset than on the Urban dataset. The difference in performance between methods that use scale-invariant fidelity terms and those that do not is less than for the Urban dataset, indicating that the Houston dataset has less spectral variability concerning scaling of spectra. The common ambiguity in the solutions for the "running_track" endmember is probably because it is the most under-represented endmember.

c) Apex dataset: The Apex dataset is the final dataset for the evaluation of blind unmixing performance. Fig. 17 shows all extracted endmembers for all the methods, while Table VI tabulates the average SAD scores for individual endmembers along with the mSAD score. This dataset has very distinct and easily identifiable endmembers. The water endmember in this dataset can be challenging because the water spectra have a tiny scale compared to the other endmembers. Methods using scale-sensitive similarity measures such as MSE can have difficulties with extracting such endmembers. A scale-sensitive objective function can be lowered more by extracting a variant of a well-represented endmember instead of the tiny scale endmember. An endmember having a tiny scale can also manifest as a poorly learned endmember having irregularities and generally being badly formed.

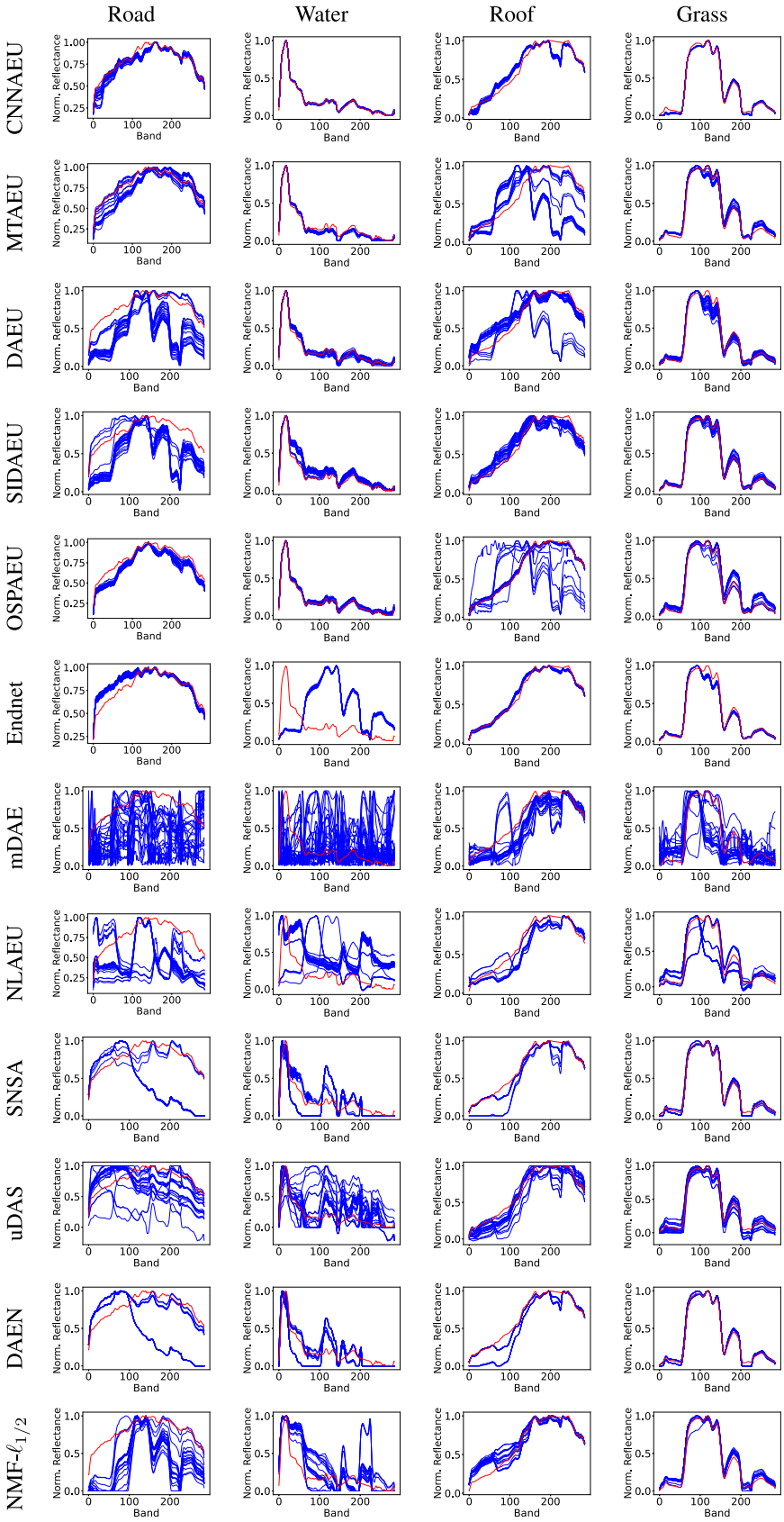


Fig. 17. Plots of all extracted endmembers by all methods (blue curves) and the reference endmembers (red) for the Apex dataset with four reference endmembers. The number of runs is 25.

TABLE VI
SAD FROM REFERENCE ENDMEMBERS IN RADIANS ALONG WITH THE STANDARD DEVIATION FOR ALL METHODS FOR THE APEX DATASET WITH FOUR REFERENCE ENDMEMBERS

Endmember Method	Road	Water	Roof	Grass	Average
CNNAEU	0.0587±0.0200	0.0417±0.0037	0.1233±0.0061	0.0621±0.0038	0.0714±0.0046
MTAEU	0.1126±0.0411	0.1365±0.0166	0.5460±0.2512	0.0992±0.0077	0.2236±0.0726
DAEU	0.3637±0.1136	0.1061±0.0250	0.2320±0.1767	0.1193±0.0377	0.2053±0.0350
DAEN	0.5575±0.2397	0.5120±0.1233	0.1219±0.0747	0.0585±0.0047	0.3125±0.0732
SIDAEU	0.3444±0.0450	0.1695±0.0444	0.1284±0.0229	0.0561±0.0167	0.1746±0.0232
OSPAEU	0.0749±0.0085	0.0662±0.0144	0.1652±0.2523	0.1318±0.0245	0.1095±0.0607
Endnet	0.0902±0.0099	1.0985±0.0044	0.0515±0.0068	0.0935±0.0079	0.3334±0.0039
mDAE	0.4818±0.1768	1.0567±0.1836	0.2302±0.0877	0.4956±0.0765	0.5661±0.0625
NAE-VCA	0.3967±0.0904	0.7177±0.1627	0.1404±0.0206	0.3821±0.0669	0.4092±0.0565
SNSA	0.6324±0.1827	0.5669±0.0928	0.0946±0.0616	0.0607±0.0024	0.3387±0.0536
uDAS	0.3082±0.1042	0.5828±0.1590	0.1257±0.0499	0.1004±0.0479	0.2793±0.0589
NMF-$\ell_{1/2}$	0.5166±0.0363	0.4635±0.0742	0.1304±0.0200	0.1008±0.0162	0.3028±0.0237

The number of runs is 25. Best results are in red and the second best in blue.

TABLE VII
RMSE BETWEEN GENERATED ABUNDANCE MAPS AND REFERENCE MAPS FOR THE URBAN DATASET WITH FOUR REFERENCE ENDMEMBERS

Endmember Method	Asphalt	Grass	Tree	Roof	Average
CNNAEU	0.2567±0.0082	0.2944±0.0124	0.2084±0.0171	0.1675±0.0188	0.2369±0.0104
MTAEU	0.1517±0.0033	0.1498±0.0054	0.0822±0.0052	0.0889±0.0043	0.1226±0.0032
DAEU	0.1689±0.0475	0.2292±0.0563	0.1856±0.0693	0.1314±0.0407	0.1862±0.0380
DAEN	0.3585±0.0001	0.4007±0.0005	0.4048±0.0002	0.2370±0.0002	0.3568±0.0002
SIDAEU	0.1721±0.0554	0.1973±0.0520	0.1625±0.0376	0.1620±0.0621	0.1767±0.0424
OSPAEU	0.2657±0.0456	0.3464±0.0302	0.3090±0.0447	0.1938±0.0467	0.2868±0.0203
Endnet	0.1424±0.0263	0.1738±0.0153	0.0896±0.0109	0.1102±0.0238	0.1338±0.0126
mDAE	0.3427±0.0393	0.3273±0.0413	0.3617±0.0332	0.2948±0.0735	0.3351±0.0262
NAE-VCA	0.2744±0.0106	0.3449±0.0342	0.3741±0.0747	0.1749±0.0311	0.3048±0.0126
SNSA	0.3356±0.0095	0.4112±0.0022	0.3535±0.0037	0.2074±0.0020	0.3353±0.0037
uDAS	0.3265±0.0104	0.4340±0.0177	0.3140±0.0200	0.2040±0.0090	0.3302±0.0032
NMF-$\ell_{1/2}$	0.4163±0.0016	0.4511±0.0044	0.4343±0.0038	0.2391±0.0012	0.3945±0.0006

The number of runs is 25. Best results are in red and the second best in blue.

In Fig. 17, we see that all methods utilizing SAD as the similarity measure in the loss function have no trouble extracting the "Water" endmember, and it is close to the reference. The water spectra in the dataset have minimal spectral variability, which explains the excellent match. CNNAEU has good accuracy and consistency for this dataset. OSPAEU, which uses a scale-sensitive fidelity term, the hyper-Laplacian loss, manages to extract the "Water" endmember very well. The loss is a p-norm of the difference between input and reconstruction raised to the power p and having $p = 0.7$, which can explain why the lower scale of the endmember is not as problematic as it is for the MSE loss.

The Endnet method fails consistently to extract the "Water" endmember and extracts some vegetation endmember instead. This can be explained by the MSE term in the loss of this method. Despite this, the method has a good consistency. The MTAEU and DAEU methods fail to consistently extract the reference form of the "Roof" endmember and extract some vegetation endmember instead. It is hard to attribute this to some implementation details without doing some experimentation. However, the methods that have bad accuracy for the "Roof" endmember all have deep encoders, i.e., MTAEU, DAEU, and OSPAEU. NLAEU, SNSA, uDAS, and DAEN all have trouble with the "Road" and the "Water" endmember.

d) Average mSAD versus scale invariance: Fig. 19 shows the average mSAD for all real datasets of method groups, where their loss function is scale invariant or non-scale invariant. The SAD and SID similarity measure are scale invariant. All methods using the MSE and the hyper-Laplacian loss function belong to the last group of scale-sensitive methods. The graph clearly shows that having a non-scale-invariant loss function leads to substantially lower performance than if a scale-invariant loss had been used.

2) Abundance Maps: The quality of generated abundance maps by blind unmixing methods depends on the quality of the extracted endmembers. The discussion of abundance maps will be limited to the Urban dataset. Fig. 18 shows the generated abundance maps by all methods for the best mSAD score. Table VII tabulates the RMSE scores for individual endmembers and the average of all maps.

The method that achieves the lowest RMSE score is the MTAEU method. CNNAEU, despite having a lower mSAD score, does not achieve a good RMSE score. Fig. 18 shows that the abundance maps for CNNAEU are intense and sparse. This is a consequence of using the softmax function to enforce the ASC and using a sizeable spatial filter for the decoder convolutional layer. This causes high values of the feature maps entering the softmax function, which then acts like ℓ_1-sparsity regularization.

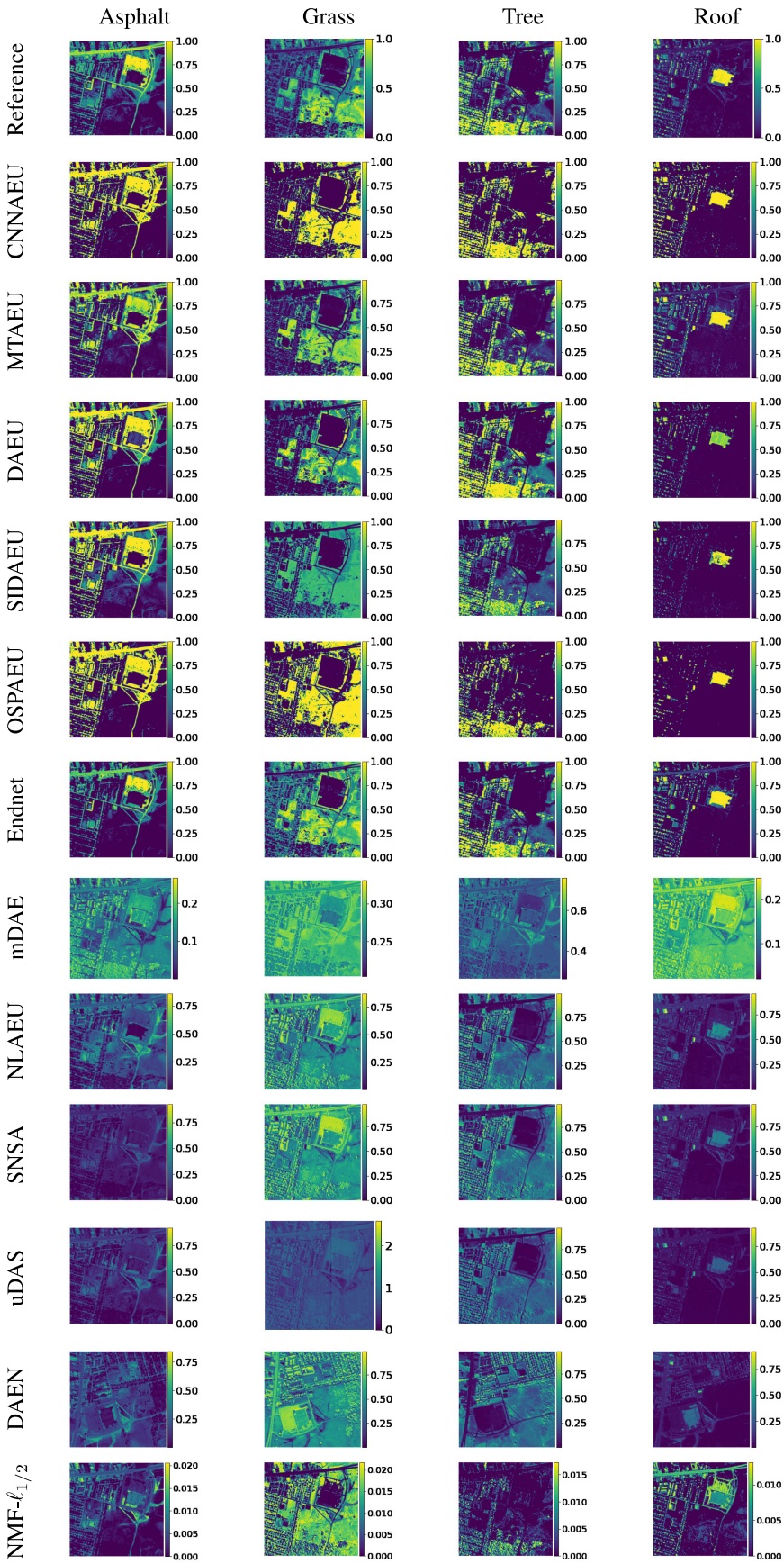


Fig. 18. Abundance maps for the run with the best mSAD score for all methods for the Urban dataset. The reference abundance maps are in the top row.

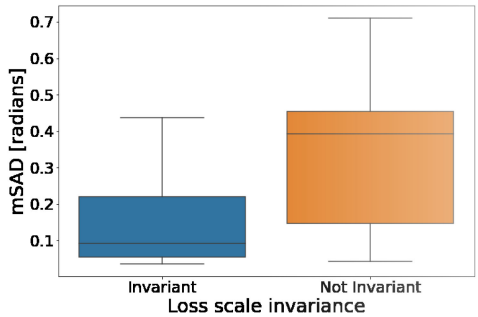


Fig. 19. Box plot of the average mSAD of methods grouped by whether their loss function is scale invariant, semi-invariant, or not at all.

The OSPAEU method also produces intense and binary-looking abundance maps. This results likely from the orthogonality prior of the abundance maps. The Endnet method achieves the second-lowest RMSE score and produces good abundance maps. This method employs forced sparsity by applying a "top_k" function to the abundances, selecting the k highest abundances and setting all the others to zero. A value of $k = 2$ was used in the experiments.

F. Robustness to Noise

A blind unmixing experiment was performed using the Samson dataset to investigate the methods' robustness to noise. The dataset was corrupted with noise to obtain four versions having signal-to-noise ratio (SNR) of 10, 20, 30, and 40 dB. Fig. 20 shows a bar chart of the mSAD score of all the methods for the Samson dataset for the four different SNR levels and the original uncorrupted dataset. Fig. 20 shows that most methods are relatively robust to noise. MTAEU and OSPAEU seem largely unaffected by the noise for this dataset. Endnet shows good performance and low variance for all SNR levels but is relatively largely affected by SNR $=$ 10 dB.

Paradoxically, SNSA and DAEN perform best at SNR $=$ 10 dB. NLAEU is heavily affected by noise, and SIDAEU does not handle SNR $=$ 10 dB well. CNNAEU performs best under moderate noise (SNR $=$ 30) corruption and also has a low variance for this SNR. The NMF method performs best at SNR $=$ 20 and 30 dB. We are not able to explain why some methods perform better at the highest noise level than with moderate noise.

Endnet has the best performance and the best consistency on the original dataset, and MTAEU has the second-best performance. CNNAEU comes third. Interestingly, the NMF method performs worse on the original dataset than on the datasets having SNR $=$ 20 and 30 dB.

Two synthetic datasets from the spectral variability experiment, having no and moderate spectral variability, were used for robustness experiments. They were corrupted with noise also such that each image has SNR of 20, 30, and 40 dB. This is useful because it decouples the effects of spectral variability and noise. The results are shown in Fig. 21(a) for no spectral variability and in Fig. 21(b) when there is moderate spectral variability.

In Fig. 21(a), we see that the MSE-based methods perform best, with the exception of NMF-$\ell_{1/2}$ and mDAE. The methods SNSA, uDAS, and DAEN perform overall best. This is not very surprising as the HSI does not have any spectral variability, and all the methods, except NMF-$\ell_{1/2}$, are designed to deal with noisy data. Comparing these methods to NMF-$\ell_{1/2}$ confirms this.

CNNAEU and MTAEU that use SAD loss are the best-performing methods. Somewhat surprisingly, the method Endnet is the worst-performing method. This is very different from the Samson experiment where it was the best-performing method. The SAD-based methods are practically unaffected by the different noise levels with the exception of the Endnet and SIDAEU methods.

From Fig. 21(b), we can see a different situation with the addition of some spectral variability. The MSE-based methods perform worse compared to SAD-based methods, as expected. It is interesting to see that the NMF-$\ell_{1/2}$ method performs better for SNR $=$ 30 and 40 dB than the other MSE methods and some SAD methods. Again, Endnet performs relatively poorly on this dataset.

The NAE-VCA performs best overall of the MSE methods. In addition, the addition of spectral variability increases the variance of the uDAS method substantially. CNNAEU also shows increased variance for two noise levels, while SIDAEU shows decreased variance. It can be seen by comparing the figures that the SAD-based methods, especially MTAEU, CNNAEU, and OSPAEU, show good robustness against both spectral variability and noise. It is puzzling that both NAE-VCA and mDAE perform best at the highest SNR.

This experiment indicates that using the SAD loss for unmixing autoencoders can provide good robustness to both spectral variability and noise, although they are not designed for that. The consistency of CNNAEU and MTAEU regarding noise and spectral variability demonstrates this.

G. Computation Cost

It is challenging to meaningfully compare the running time of different methods since it depends on hyperparameters such as batch size, the number of training data samples, and implementation details such as what programming language and DL framework are being used. In addition, CNN methods run on a GPU, while dense neural network methods run on a CPU if implemented using Tensorflow or MATLAB. Table VIII shows running times in seconds for a single run for the methods for three different datasets. The main conclusion that can be drawn from Table VIII is that methods implemented in the Python programming language using the Tensorflow or Pytorch DL frameworks are significantly faster than the methods implemented in MATLAB. The experiments were performed on a computer with an eight-core CPU and 64 GB of memory and a GPU with 11 GB of memory.

VII. Conclusion

This article critically compares numerous blind autoencoder-based unmixing methods and gives an overview of the architectures and implementation details utilized up until now. Eleven

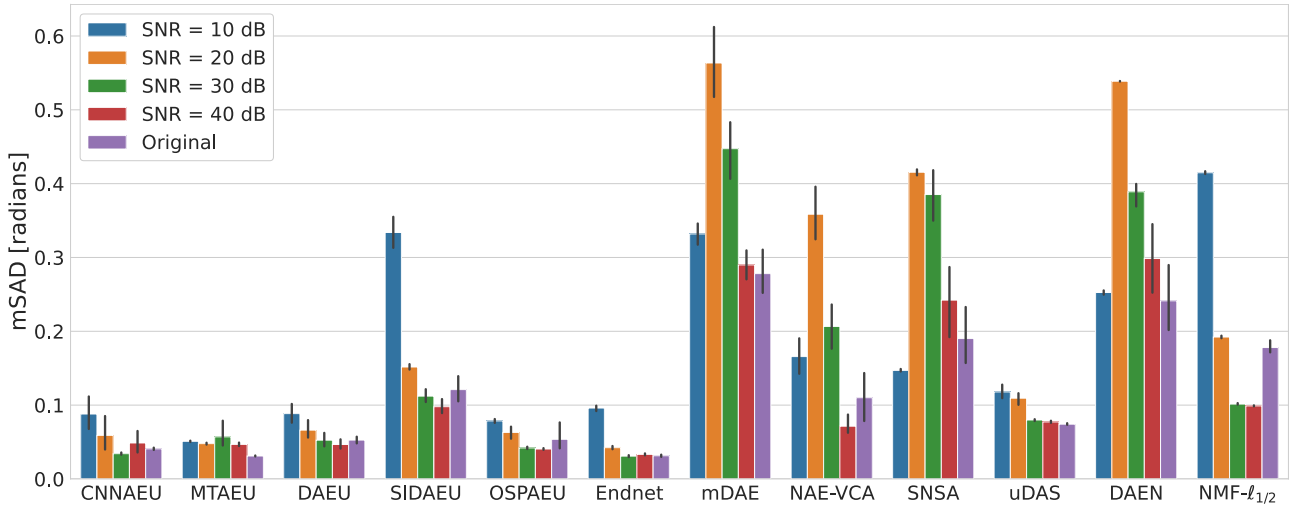
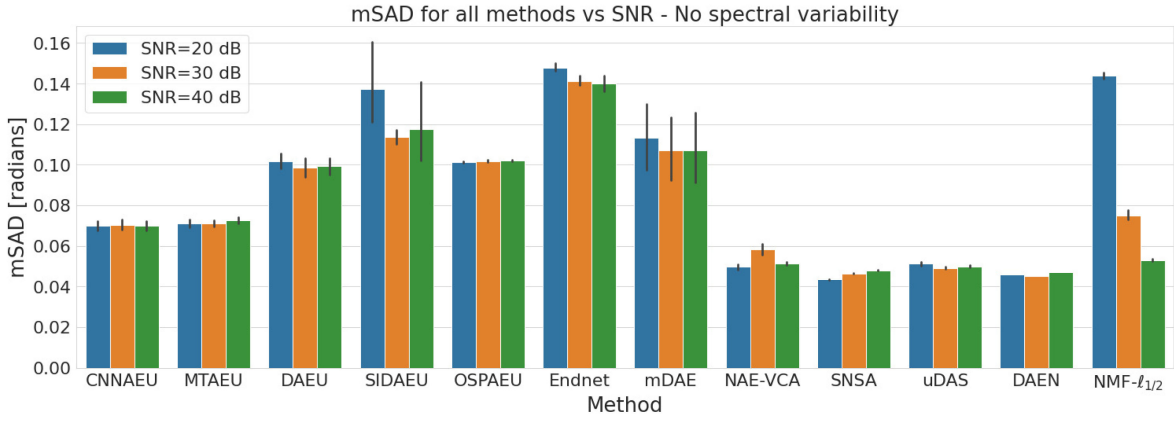
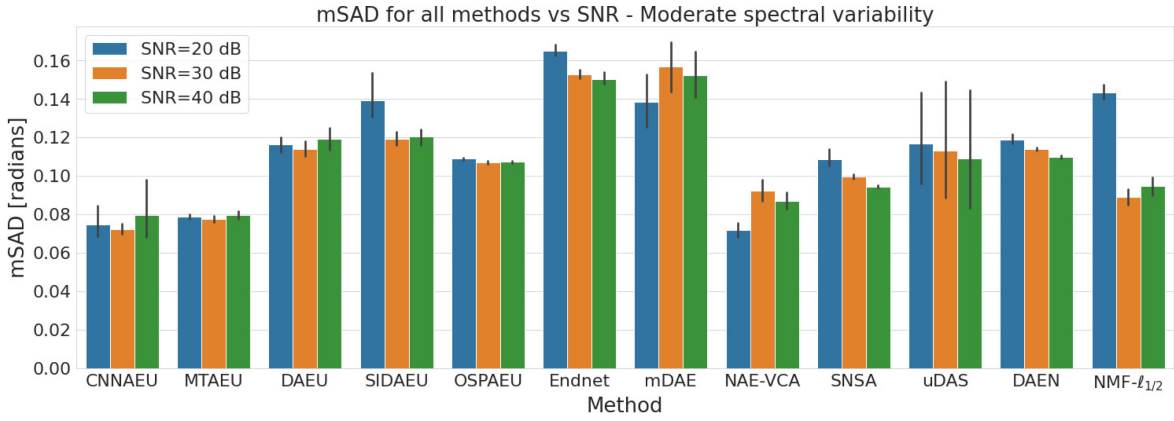


Fig. 20. Bar plot of the mSAD score for all methods for the Samson dataset with four different levels of SNR and the original uncorrupted dataset. The black vertical lines show the standard deviation. All experiments consisted of 25 runs.



(a)



(b)

Fig. 21. Bar plot of the mSAD score for all methods for the synthetic datasets having (a) no spectral variability and (b) moderate spectral variability ($r = 0.2$) and with three different levels of SNR (20, 30, and 40 dB). (a) Synthetic dataset having no spectral variability with three different levels of SNR. (b) Synthetic dataset having moderate spectral variability with three different levels of SNR.

TABLE VIII
COMPUTATION TIME IN SECONDS FOR A SINGLE RUN FOR THE METHODS FOR THREE DATASETS

Dataset	CNNAEU	MTAEU	DAEU	SIDAEU	OSPAEU	Endnet	mDAE	NLAEU	DAEN	SNSA	uDAS	NMF-$\ell_{1/2}$
Urban	95	58	79	47	34	112	27	220	3400	809	940	84
Houston	24	55	77	44	18	98	36	22	1102	708	158	32
Apex	347	60	83	52	34	105	69	215	3602	1813	1654	140

The values for the Urban image are for estimating four endmembers.

different autoencoder methods were compared by performing blind unmixing on four different real hyperspectral datasets and four synthetic datasets with a varying degree of spectral variability.

It is generally tough to interpret the results of experiments in terms of concrete implementation details of neural network methods when they differ in so many ways. Because of this, comprehensive ablation experiments with a simple spectral unmixing autoencoder were carried out. These ablation experiments provide further experimental evidence for many of the details discussed in this article.

Why do autoencoders work well for spectral unmixing compared to traditional methods? What exactly is it that makes them perform so well? A linear nonnegative autoencoder using the MSE loss function essentially performs NMF and should perform similarly to traditional NMF methods in unmixing. However, autoencoders can use any loss functions that allow for backpropagation. The option to use scale-invariant loss functions gives autoencoders increased flexibility over traditional methods for linear unmixing. In addition, the encoder and the decoder can be arbitrarily nonlinear, which can further improve performance.

Because of inherent spectral variability in real HSIs and the inability of the LMM to model it, the scale-invariance of the loss function used by the methods matters extensively. The results of the experiments with real datasets confirm this. The results of ablation experiments where a simple autoencoder was compared to the NMF-$\ell_{1/2}$ method and experiments with the synthetic datasets with varying spectral variability further strengthen this conclusion.

Scale-sensitive fidelity terms should be avoided if the method performs unmixing according to the LMM model. Unless the architecture of the method is designed to handle spectral variability, such as the method in [77], a scale-invariant loss or as close to it as possible should be chosen. Even if spectral variability is handled by the method, another problem remains if a strongly scale-sensitive similarity measure such as MSE is used. It is the difficulty with extracting endmembers that have a tiny scale compared to all the other endmembers in a scene, such as the water endmember in the Apex dataset.

The ablation experiments show that an LReLU activation that allows negative activations, coupled with the softmax function to enforce the ASC constraint, works well and is robust. In addition, the batch size is an important parameter for spectral unmixing autoencoders, and it is dataset dependent. Furthermore, using nonlinear decoders with a higher number of parameters can benefit the quality of extracted endmembers, especially if the images have substantial nonlinear mixing.

Another observation concerns the use of spatial information, i.e., whether methods process a single spectrum at a time or if they operate on a patch at a time and make use of the spatial correlations existing within real HSIs. The spectral–spatial methods seem to benefit from the additional spatial information by operating on whole patches of HSIs at a time. This is especially evident on the synthetic dataset. Using as much of the available information within an HSI as possible can lead to better unmixing performance if done correctly. Better performance can come, e.g., in the form of better consistency.

At this moment, the greatest challenge for HU using autoencoders is creating methods that perform well and have a good consistency for many diverse datasets. Methods can show very strong performance on one dataset and mediocre performance on a different dataset. Increasing the robustness of autoencoder methods will continue to be a crucial issue going forward. Handling endmember variability is an aspect of this.

Recently, multistream autoencoders utilizing adversarial regularizations or VAEs have become popular. VAEs allow for treating endmembers as distributions instead of having them fixed for every pixel. Doing so enables methods to address spectral variability with scale-sensitive object functions. This is a promising approach that abandons unmixing models such as the LMM, which can be too restrictive and simple. Using generative models in this way is a trend that the authors expect to continue. In addition, borrowing concepts from image-to-image translation such as cycle consistency and perceptual loss shows promise. The performance of autoencoder methods adhering strictly to restrictive and simple models such as the LMM is unlikely to see much more improvement and more complex, e.g., multistream, combined with adversarial or variational architectures, will be needed for better performance and robustness in the future.

The intersection of HU and DL is currently a very vibrant field of research. More and more autoencoder-based methods are being published every year, utilizing the latest results from the study of autoencoders in the context of DL.

ACKNOWLEDGMENT

The authors would like to thank Jun Li for the code in [85] and [92], Jie Chen for the code in [38], and Ying Qu for making the code in [68] available on GitHub.

REFERENCES

[1] A. F. Goetz, G. Vane, J. E. Solomon, and B. N. Rock, "Imaging spectrometry for earth remote sensing," *Science*, vol. 228, no. 4704, pp. 1147–1153, 1985.

[2] D. Hong *et al.*, "Interpretable hyperspectral artificial intelligence: When nonconvex modeling meets hyperspectral remote sensing," *IEEE Geosci. Remote Sens. Mag.*, vol. 9, no. 2, pp. 52–87, Jun. 2021.

[3] J. M. Bioucas-Dias, A. Plaza, G. Camps-Valls, P. Scheunders, N. Nasrabadi, and J. Chanussot, "Hyperspectral remote sensing data analysis and future challenges," *IEEE Geosci. Remote Sens. Mag.*, vol. 1, no. 2, pp. 6–36, Jun. 2013.

[4] C. Yang, J. H. Everitt, and J. M. Bradford, "Airborne hyperspectral imagery and linear spectral unmixing for mapping variation in crop yield," *Precis. Agriculture*, vol. 8, no. 6, pp. 279–296, 2007.

[5] T. D. McRae, D. Oleksyn, J. Miller, and Y.-R. Gao, "Robust blind spectral unmixing for fluorescence microscopy using unsupervised learning," *PLoS One*, vol. 14, no. 12, 2019, Art. no. e0225410.

[6] G. Edelman, E. Gaston, T. Van Leeuwen, P. Cullen, and M. Aalders, "Hyperspectral imaging for non-contact analysis of forensic traces," *Forensic Sci. Int.*, vol. 223, no. 1–3, pp. 28–39, 2012.

[7] A. T. Badaró *et al.*, "Near infrared hyperspectral imaging and spectral unmixing methods for evaluation of fiber distribution in enriched pasta," *Food Chem.*, vol. 343, 2021, Art. no. 128517.

[8] G. P. Asner and K. B. Heidebrecht, "Spectral unmixing of vegetation, soil and dry carbon cover in arid regions: Comparing multispectral and hyperspectral observations," *Int. J. Remote Sens.*, vol. 23, no. 19, pp. 3939–3958, 2002.

[9] M. B. Lopes, J. M. Bioucas-Dias, M. A. Figueiredo, J.-C. Wolff, N. Mistry, and J. Warrack, "Comparison of near infrared and raman hyperspectral unmixing performances for chemical identification of pharmaceutical tablets," in *Proc. 3rd Workshop Hyperspectral Image Signal Process., Evol. Remote Sens.*, 2011, pp. 1–4.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Internal Representations by Error Propagation*. Cambridge, MA, USA: MIT Press, 1986, pp. 318–362.

[11] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[12] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," 2020, *arXiv:2003.05991*.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[14] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2094–2107, Jun. 2014.

[15] X. X. Zhu *et al.*, "Deep learning in remote sensing: A comprehensive review and list of resources," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 4, pp. 8–36, Dec. 2017.

[16] W. Huang, L. Xiao, Z. Wei, H. Liu, and S. Tang, "A new pan-sharpening method with deep neural networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 5, pp. 1037–1041, May 2015.

[17] N. Ma, Y. Peng, S. Wang, and D. Liu, "Hyperspectral image anomaly targets detection with online deep learning," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf.*, 2018, pp. 1–6.

[18] R. Guo, W. Wang, and H. Qi, "Hyperspectral image unmixing using autoencoder cascade," in *Proc. 7th Workshop Hyperspectral Image Signal Process.: Evol. Remote Sens.*, 2015, pp. 1–4.

[19] R. Heylen, M. Parente, and P. Gader, "A review of nonlinear hyperspectral unmixing methods," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 1844–1868, Jun. 2014.

[20] C. Shi and L. Wang, "Incorporating spatial information in spectral unmixing: A review," *Remote Sens. Environ.*, vol. 149, pp. 70–87, 2014.

[21] J. M. Bioucas-Dias *et al.*, "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 5, no. 2, pp. 354–379, Apr. 2012.

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

[23] A. Zare and K. Ho, "Endmember variability in hyperspectral analysis: Addressing spectral variability during spectral unmixing," *IEEE Signal Process. Mag.*, vol. 31, no. 1, pp. 95–104, Jan. 2014.

[24] B. Somers, G. Asner, L. Tits, and P. Coppin, "Endmember variability in spectral mixture analysis: A review," *Remote Sens. Environ.*, vol. 115, no. 7, pp. 1603–1616, Jul. 2011.

[25] W.-K. Ma *et al.*, "A signal processing perspective on hyperspectral unmixing: Insights from remote sensing," *IEEE Signal Process. Mag.*, vol. 31, no. 1, pp. 67–81, Jan. 2014.

[26] R. Borsoi *et al.*, "Spectral variability in hyperspectral data unmixing: A comprehensive review," *IEEE Geosci.Remote Sens. Mag.*, to be published, doi: 10.1109/MGRS.2021.3071158.

[27] A. Signoroni, M. Savardi, A. Baronio, and S. Benini, "Deep learning meets hyperspectral image analysis: A multidisciplinary review," *J. Imag.*, vol. 5, no. 5, 2019, Art. no. 52.

[28] J. S. Bhatt and M. V. Joshi, "Deep learning in hyperspectral unmixing: A review," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2020, pp. 2189–2192.

[29] P.-A. Thouvenin, N. Dobigeon, and J.-Y. Tourneret, "Hyperspectral unmixing with spectral variability using a perturbed linear mixing model," *IEEE Trans. Signal Process.*, vol. 64, no. 2, pp. 525–538, Jan. 2016.

[30] L. Drumetz, M. Veganzones, S. Henrot, R. Phlypo, J. Chanussot, and C. Jutten, "Blind hyperspectral unmixing using an extended linear mixing model to address spectral variability," *IEEE Trans. Image Process.*, vol. 25, no. 8, pp. 3890–3905, Aug. 2016.

[31] D. Hong, N. Yokoya, J. Chanussot, and X. X. Zhu, "An augmented linear mixing model to address spectral variability for hyperspectral unmixing," *IEEE Trans. Image Process.*, vol. 28, no. 4, pp. 1923–1938, Apr. 2019.

[32] R. A. Borsoi, T. Imbiriba, and J. C. M. Bermudez, "A data dependent multiscale model for hyperspectral unmixing with spectral variability," *IEEE Trans. Image Process.*, vol. 29, pp. 3638–3651, 2020.

[33] N. Dobigeon, J. Tourneret, C. Richard, J. C. M. Bermudez, S. McLaughlin, and A. O. Hero, "Nonlinear unmixing of hyperspectral images: Models and algorithms," *IEEE Signal Process. Mag.*, vol. 31, no. 1, pp. 82–94, Jan. 2014.

[34] Y. Altmann, N. Dobigeon, and J. Tourneret, "Bilinear models for nonlinear unmixing of hyperspectral images," in *Proc. 3rd Workshop Hyperspectral Image Signal Process., Evol. Remote Sens.*, 2011, pp. 1–4.

[35] A. Halimi, Y. Altmann, N. Dobigeon, and J. Tourneret, "Nonlinear unmixing of hyperspectral images using a generalized bilinear model," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 11, pp. 4153–4162, Nov. 2011.

[36] J. Sigurdsson, M. O. Ulfarsson, and J. R. Sveinsson, "Blind sparse nonlinear hyperspectral unmixing using an ℓ_q penalty," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 12, pp. 1907–1911, Dec. 2018.

[37] B. Hapke, "Bidirectional reflectance spectroscopy: 1. Theory," *J. Geophys. Res.: Solid Earth*, vol. 86, no. B4, pp. 3039–3054, 1981.

[38] M. Wang, M. Zhao, J. Chen, and S. Rahardja, "Nonlinear unmixing of hyperspectral data via deep autoencoder networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 9, pp. 1467–1471, Sep. 2019.

[39] C.-I. Chang and Q. Du, "Estimation of number of spectrally distinct signal sources in hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 3, pp. 608–619, Mar. 2004.

[40] J. Bioucas-Dias and J. Nascimento, "Hyperspectral subspace identification," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 8, pp. 2435–2445, Aug. 2008.

[41] B. Luo, J. Chanussot, S. Douté, and L. Zhang, "Empirical automatic estimation of the number of endmembers in hyperspectral images," *IEEE Geosci. Remote Sens. Lett.*, vol. 10, no. 1, pp. 24–28, Jan. 2012.

[42] B. Rasti, M. O. Ulfarsson, and J. R. Sveinsson, "Hyperspectral subspace identification using SURE," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 12, pp. 2481–2485, Dec. 2015.

[43] M. D. Iordache, J. M. Bioucas-Dias, and A. Plaza, "Sparse unmixing of hyperspectral data," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 6, pp. 2014–2039, Jun. 2011.

[44] M. D. Iordache, J. M. Bioucas-Dias, and A. Plaza, "Total variation spatial regularization for sparse hyperspectral unmixing," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 11, pp. 4484–4502, Nov. 2012.

[45] M. D. Iordache, J. M. Bioucas-Dias, and A. Plaza, "Collaborative sparse regression for hyperspectral unmixing," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 1, pp. 341–354, Jan. 2014.

[46] M. D. Iordache, J. M. Bioucas-Dias, A. Plaza, and B. Somers, "MUSIC-CSR: Hyperspectral unmixing via multiple signal classification and collaborative sparse regression," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 7, pp. 4364–4382, Jul. 2014.

[47] L. Zhang, W. Wei, Y. Zhang, F. Li, and H. Yan, "Structured sparse Bayesian hyperspectral compressive sensing using spectral unmixing," in *Proc. 6th Workshop Hyperspectral Image Signal Process., Evol. Remote Sens.*, Jun. 2014, pp. 1–4.

[48] R. T. Albayrak, A. C. Gurbuz, and B. Gunyel, "Compressed sensing based hyperspectral unmixing," in *Proc. 22nd Signal Process. Commun. Appl. Conf.*, Apr. 2014, pp. 1438–1441.

[49] Xusu, "Compressive sensing for endmember extraction," in *Proc. 2nd IEEE Int. Conf. Comput. Commun.*, Oct. 2016, pp. 1345–1348.

[50] J. M. Nascimento and J. M. Dias, "Vertex component analysis: A fast algorithm to unmix hyperspectral data," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 4, pp. 898–910, Apr. 2005.

[51] J. Li, A. Agathos, D. Zaharie, J. M. Bioucas-Dias, A. Plaza, and X. Li, "Minimum volume simplex analysis: A fast algorithm for linear hyperspectral unmixing," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 9, pp. 5067–5082, Sep. 2015.

[52] Y. Qian, S. Jia, J. Zhou, and A. Robles-Kelly, "Hyperspectral unmixing via $l_{1/2}$ sparsity-constrained nonnegative matrix factorization," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 11, pp. 4282–4297, Nov. 2011.

[53] J. Sigurdsson, M. O. Ulfarsson, and J. R. Sveinsson, "Hyperspectral unmixing with l_q regularization," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 11, pp. 6793–6806, Nov. 2014.

[54] J. M. P. Nascimento and J. M. Bioucas-Dias, "Hyperspectral unmixing based on mixtures of Dirichlet components," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 3, pp. 863–878, Mar. 2012.

[55] N. Dobigeon, S. Moussaoui, M. Coulon, J. Y. Tourneret, and A. O. Hero, "Joint Bayesian endmember extraction and linear unmixing for hyperspectral imagery," *IEEE Trans. Signal Process.*, vol. 57, no. 11, pp. 4355–4368, Nov. 2009.

[56] O. Eches, N. Dobigeon, C. Mailhes, and J. Y. Tourneret, "Bayesian estimation of linear mixtures using the normal compositional model. Application to hyperspectral imagery," *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1403–1413, Jun. 2010.

[57] J. Sigurdsson, M. O. Ulfarsson, and J. R. Sveinsson, "Blind hyperspectral unmixing using total variation and ℓ_q sparse regularization," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 11, pp. 6371–6384, Nov. 2016.

[58] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," May 2015, *arXiv:1505.00853*.

[59] M. Benyamin *et al.*, "Autoencoder based blind source separation for photoacoustic resolution enhancement," *Sci. Rep.*, vol. 10, 2020, Art. no. 21414.

[60] K. H. Tsai *et al.*, "Blind monaural source separation on heart and lung sounds based on periodic-coded deep autoencoder," *IEEE J. Biomed. Health Informat.*, vol. 24, no. 11, pp. 3203–3214, Nov. 2020.

[61] M. Mikulski and J. Duda, "Toroidal autoencoder," 2019, *arXiv:1903.12286*.

[62] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *Statistics*, vol. 1050, p. 10, 2013, *arXiv:1312.6114*.

[63] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow, "Adversarial autoencoders," 2016. [Online]. Available: http://arxiv.org/abs/1511.0564

[64] S. Mannor, D. Peleg, and R. Rubinstein, "The cross entropy method for classification," in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 561–568.

[65] A. Min, Z. Guo, H. Li, and J. Peng, "JMnet: Joint metric neural network for hyperspectral unmixing," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5505412.

[66] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010.

[67] M. Chen, Z. Xu, K. Q. Weinberger, and F. Sha, "Marginalized stacked denoising autoencoders," in *j. Mach. Learn. Res.*, vol. 16, no. 1, pp. 3849–3875, 2015.

[68] Y. Qu and H. Qi, "uDAS: An untied denoising autoencoder with sparsity for spectral unmixing," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 3, pp. 1698–1712, Mar. 2019.

[69] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Proc. Int. Conf. Artif. Neural Netw.*, 2011, pp. 52–59.

[70] B. Palsson, M. O. Ulfarsson, and J. R. Sveinsson, "Convolutional autoencoder for spectral–spatial hyperspectral unmixing," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 1, pp. 535–549, Jan. 2021.

[71] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. Int. Conf. Mach. Learn.*, 2013, vol. 30, no. 1, p. 3.

[72] H. Mhaskar, Q. Liao, and T. Poggio, "When and Why Are Deep Networks Better than Shallow Ones?," in *Proc Thirty-First AAAI Conf. Artif. Intell.*, 2017, pp. 2343.2349.

[73] B. Palsson, J. Sigurdsson, J. R. Sveinsson, and M. O. Ulfarsson, "Hyperspectral unmixing using a neural network autoencoder," *IEEE Access*, vol. 6, pp. 25646–25656, 2018.

[74] B. Palsson, J. R. Sveinsson, and M. O. Ulfarsson, "Spectral-spatial hyperspectral unmixing using multitask learning," *IEEE Access*, vol. 7, pp. 148861–148872, 2019.

[75] Y. Su, X. Xu, J. Li, H. Qi, P. Gamba, and A. Plaza, "Deep autoencoders with multitask learning for bilinear hyperspectral unmixing," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 10, pp. 8615–8629, Oct. 2021.

[76] Z. Dou, K. Gao, X. Zhang, H. Wang, and J. Wang, "Hyperspectral unmixing using orthogonal sparse prior-based autoencoder with hyper-Laplacian loss and data-driven outlier detection," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 9, pp. 6550–6564, Sep. 2020.

[77] Z. Dou, K. Gao, X. Zhang, H. Wang, and J. Wang, "Blind hyperspectral unmixing using dual branch deep autoencoder with orthogonal sparse prior," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process*, 2020, pp. 2428–2432.

[78] M. M. Elkholy, M. Mostafa, H. M. Ebied, and M. F. Tolba, "Hyperspectral unmixing using deep convolutional autoencoder," *Int. J. Remote Sens.*, vol. 41, no. 12, pp. 4799–4819, 2020.

[79] Y. Su, A. Mariononi, J. Li, A. Plaza, and P. Gamba, "Nonnegative sparse autoencoder for robust endmember extraction from remotely sensed hyperspectral images," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2017, pp. 205–208.

[80] S. Ozkan, B. Kaya, and G. B. Akar, "EndNet: Sparse autoencoder network for endmember extraction and hyperspectral unmixing," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 1, pp. 482–496, Jan. 2019.

[81] Y. Qu, R. Guo, and H. Qi, "Spectral unmixing through part-based nonnegative constraint denoising autoencoder," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2017, pp. 209–212.

[82] D.-A. Clevert, T. Unterthiner and S. Hochreiter *et al.*, "Fast and accurate deep network learning by exponential linear units (elus)," *NiN*, vol. 8, pp. 35–68, 2015, *arXiv:1511.07289*.

[83] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 972–981.

[84] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: A self-gated activation function," 2017, *arXiv:1710.05941*.

[85] Y. Su, A. Marinoni, J. Li, J. Plaza, and P. Gamba, "Stacked nonnegative sparse autoencoders for robust hyperspectral unmixing," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 9, pp. 1427–1431, Sep. 2018.

[86] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[87] S. Santurkar, D. Tsipras, A. Ilyas, and A. Mądry, "How does batch normalization help optimization?," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 2488–2498.

[88] N. Keskar, J. Nocedal, P. Tang, D. Mudigere, and M. Smelyanskiy, "On large-batch training for deep learning: Generalization gap and sharp minima," in *Proc. 5th Int. Conf. Learn. Represent.*, 2016, *arXiv:1609.04836*.

[89] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[90] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 648–656.

[91] F. Palsson, J. Sigurdsson, J. R. Sveinsson, and M. O. Ulfarsson, "Neural network hyperspectral unmixing with spectral information divergence objective," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2017, pp. 755–758.

[92] Y. Su, J. Li, A. Plaza, A. Marinoni, P. Gamba, and S. Chakravortty, "DAEN: Deep autoencoder networks for hyperspectral unmixing," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 7, pp. 4309–4321, Jul. 2019.

[93] B. Yan, Z. Wu, H. Liu, Y. Xu, and Z. Wei, "Hyperspectral unmixing via wavelet based autoencoder network," in *Proc. 10th Workshop Hyperspectral Imag. Signal Process., Evol. Remote Sens.*, 2019, pp. 1–5.

[94] J. R. Patel, M. V. Joshi, and J. S. Bhatt, "Spectral unmixing using autoencoder with spatial and spectral regularizations," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2021, pp. 3321–3324.

[95] C. Ding, D. Zhou, X. He, and H. Zha, "R1-PCA: Rotational invariant l1-norm principal component analysis for robust subspace factorization," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 281–288.

[96] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[97] S. Ruder, "An overview of multi-task learning in deep neural networks," 2017, *arXiv:1706.05098*.

[98] J. Baxter, "A Bayesian/information theoretic model of learning to learn via multiple task sampling," *Mach. Learn.*, vol. 28, no. 1, pp. 7–39, 1997.

[99] R. A. Borsoi, T. Imbiriba, and J. C. M. Bermudez, "Deep generative endmember modeling: An application to unsupervised spectral unmixing," *IEEE Trans. Comput. Imag.*, vol. 6, pp. 374–384, 2020.

[100] C. Villani, *The Wasserstein Distances*. Heidelberg, Germany: Springer, 2009, pp. 93–111.

[101] S. Ozkan and G. B. Akar, "Spectral unmixing with multinomial mixture Kernel and Wasserstein generative adversarial loss," 2020, *arXiv:2012.06859*.

[102] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. 34th Int. Conf. Mach. Learn.*, Aug. 2017, vol. 70, pp. 214–223.

[103] Q. Jin, Y. Ma, F. Fan, J. Huang, X. Mei, and J. Ma, "Adversarial autoencoder network for hyperspectral unmixing," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2021.31142032021.
[104] Q. Jin, Y. Ma, X. Mei, and J. Ma, "TANet: An unsupervised two-stream autoencoder network for hyperspectral unmixing," *IEEE Trans. Geosci. Remote Sens.*, to be published, doi: 10.1109/TGRS.2021.3094884.
[105] L. Gao, Z. Han, D. Hong, B. Zhang, and J. Chanussot, "CyCU-Net: Cycle-consistency unmixing network by learning cascaded autoencoders," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5503914.
[106] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2242–2251.
[107] S. S. Vijayashekhar, V. S. Deshpande, and J. S. Bhatt, "A practical approach for hyperspectral unmixing using deep learning," *IEEE Geosci. Remote Sens. Lett.*, 2022, vol. 19, pp. 1–5, doi: 10.1109/LGRS.2021.3127075.
[108] V. S. S. and J. S. Bhatt, "A blind spectral unmixing in wavelet domain," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 10287–10302, 2021.
[109] X. Xu, J. Li, S. Li, and A. Plaza, "Curvelet transform domain-based sparse nonnegative matrix factorization for hyperspectral unmixing," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 4908–4924, 2020.
[110] M. Zhao, L. Yan, and J. Chen, "LSTM-DNN based autoencoder network for nonlinear hyperspectral image unmixing," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 2, pp. 295–309, Feb. 2021.
[111] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[112] M. Zhao, M. Wang, J. Chen, and S. Rahardja, "Hyperspectral unmixing for additive nonlinear models with a 3-D-CNN autoencoder network," *IEEE Trans. Geosci. Remote Sens.*, to be published, doi: 10.1109/TGRS.2021.3098745.
[113] K. T. Shahid and I. D. Schizas, "Unsupervised hyperspectral unmixing via nonlinear autoencoders," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5506513.
[114] H. Li, R. A. Borsoi, T. Imbiriba, P. Closas, J. C. M. Bermudez, and D. Erdoğmuş, "Model-based deep autoencoder networks for nonlinear hyperspectral unmixing," *IEEE Geosci. Remote Sens. Lett.*, vol. 19, 2022, Art. no. 5506105.
[115] P. Refaeilzadeh, L. Tang, and H. Liu, *Cross-Validation*. Boston, MA, USA: Springer, 2009, pp. 532–538.
[116] M. Feurer and F. Hutter, "Hyperparameter optimization," in *Automated Machine Learning*. Berlin, Germany: Springer, 2019, pp. 3–33.
[117] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. 10, pp. 281–305, 2012.
[118] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, vol. 2, pp. 2951–2959.
[119] J. Sigurdsson, M. O. Ulfarsson, and J. R. Sveinsson, "Parameter estimation for blind LQ hyperspectral unmixing using Bayesian optimization," in *Proc. 9th Workshop Hyperspectral Image Signal Process.: Evol. Remote Sens.*, 2018, pp. 1–5.
[120] M. Pereyra, J. M. Bioucas-Dias, and M. A. Figueiredo, "Maximum-a-posteriori estimation with unknown regularisation parameters," in *Proc. 23rd Eur. Signal Process. Conf.*, 2015, pp. 230–234.
[121] L. Zhuang, C.-H. Lin, M. A. T. Figueiredo, and J. M. Bioucas-Dias, "Regularization parameter selection in minimum volume hyperspectral unmixing," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 12, pp. 9858–9877, Dec. 2019.
[122] S. Ozkan and G. B. Akar, "Deep spectral convolution network for hyperspectral unmixing," in *Proc. 25th IEEE Int. Conf. Image Process.* 2018, pp. 3313–3317.
[123] F. Khajehrayeni and H. Ghassemian, "Hyperspectral unmixing using deep convolutional autoencoders in a supervised scenario," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 567–576, 2020.
[124] B. Rasti, B. Koirala, P. Scheunders, and P. Ghamisi, "UnDIP: Hyperspectral unmixing using deep image prior," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5504615.
[125] L. Qi, J. Li, Y. Wang, M. Lei, and X. Gao, "Deep spectral convolution network for hyperspectral image unmixing with spectral library," *Signal Process.*, vol. 176, 2020, Art. no. 107672.
[126] Z. Han, D. Hong, L. Gao, B. Zhang, and J. Chanussot, "Deep half-siamese networks for hyperspectral unmixing," *IEEE Geosci. Remote Sens. Lett.*, vol. 18, no. 11, pp. 1996–2000, Nov. 2021.
[127] D. Hong *et al.*, "Endmember-guided unmixing network (EGU-Net): A general deep learning framework for self-supervised hyperspectral unmixing," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2021.3082289.
[128] Q. Jin, Y. Ma, X. Mei, H. Li, and J. Ma, "UTDN: An unsupervised two-stream Dirichlet-Net for hyperspectral unmixing," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2021, pp. 1885–1889.
[129] F. Zhu, "Structured sparse method for hyperspectral unmixing," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 88, pp. 101–118, 2014.
[130] M. Magnusson, J. Sigurdsson, S. E. Armansson, M. O. Ulfarsson, H. Deborah, and J. R. Sveinsson, "Creating RGB images from hyperspectral images using a color matching function," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2020, pp. 2045–2048.
[131] L. J. Rickard, R. W. Basedow, E. F. Zalewski, P. R. Silverglate, and M. Landers, "HYDICE: An airborne system for hyperspectral imaging," in *Imaging Spectrometry of the Terrestrial Environment*, vol. 1937, G. Vane, Ed. Bellingham, WA, USA: SPIE, 1993, pp. 173–179.
[132] M. E. Schaepman *et al.*, "Advanced radiometry measurements and earth science applications with the airborne prism experiment (APEX)," *Remote Sens. Environ.*, vol. 158, pp. 207–219, 2015.

Burkni Palsson (Student Member, IEEE) received the B.S. degree in mathematics and physics in 1999, the B.S. degree in energy and environmental technology in 2012, and the M.Sc. degree in electrical and computer engineering in 2021, all from the University of Iceland, Reykjavik, Iceland, where he is currently working toward the Ph.D. degree in electrical engineering.

His research interests include hyperspectral unmixing and classification in remote sensing and the development and applications of deep-learning-based methods in that field.

Johannes R. Sveinsson (Senior Member, IEEE) received the B.S. degree from the University of Iceland, Reykjavík, Iceland, and the M.S. and Ph.D. degrees from Queen's University, Kingston, ON, Canada, all in electrical engineering.

He is currently a Professor with the Department of Electrical and Computer Engineering, University of Iceland, where he was with the Laboratory of Information Technology and Signal Processing from 1981 to 1982 and with the Engineering Research Institute and the Department of Electrical and Computer Engineering as a Senior Member of research staff and a Lecturer, respectively, from 1991 to 1998. He was a Visiting Research Student with the Imperial College of Science and Technology, London, U.K., from 1985 to 1986. At Queen's University, he held teaching and research assistantships. His current research interests include systems and signal theory.

Dr. Sveinsson is a recipient of the Queen's Graduate Awards from Queen's University. He is a co-recipient of the 2013 IEEE Geoscience and Remote Sensing Society Highest Impact Paper Award.

Magnus O. Ulfarsson (Senior Member, IEEE) received the B.S. and M.S. degrees in Electrical and Computer Engineering from the University of Iceland, Reykjavík, Iceland, both in 2002, and the Ph.D. degree in Electrical and Computer Engineering from the University of Michigan, Ann Arbor, MI, USA, in 2007.

In 2007, he joined the University of Iceland, where he is currently a Professor and the Chair of the Faculty of Electrical and Computer Engineering. Since 2013, he has been with deCODE Genetics, Reykjavík. His research interests include statistical signal processing, image processing, machine learning, remote sensing, medical imaging, and genomics.

Dr. Ulfarsson has been an Associate Editor for IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING since 2018.