Spectral unmixing is central when analyzing hyperspectral data. To accomplish this task, physics-based methods have become popular because, with their explicit mixing models, they can provide a clear interpretation. Nevertheless, because of their limited modeling capabilities, especially when analyzing real scenes with unknown complex physical properties, these methods may not be accurate. On the other hand, data-driven methods using deep learning in particular have developed rapidly in recent years, thanks to their superior capability in modeling complex nonlinear systems. Simply transferring these methods as black boxes to perform unmixing may lead to low interpretability and poor generalization ability. To bring together the best of two worlds, recent research efforts have focused on combining the advantages of both physics-based models and data-driven methods. In this article, we present an overview of recent advances on this topic from various perspectives, including deep neural network (DNN) design, prior capturing, and loss selection. We summarize these methods within a common optimization framework and discuss ways of enhancing our understanding of these methods. The related source codes are made publicly available at http://github.com/xiuheng-wang/awesome-hyperspectral-image-unmixing.

Introduction

Conventional machine vision applications widely use red-green-blue cameras. These cameras are suitable for identifying objects with shapes and colors consistent with the human visual system. As a significant advancement in imaging techniques, hyperspectral cameras measure objects or scenes by recording hundreds of narrow bands across a wide-range spectrum that can cover the nonvisible range of light. In that way, hyperspectral images enable the identification of materials based on their unique spectral signatures, beyond their visible characteristics, coupled with spatial information [1]. Initial hyperspectral imaging applications started with the remote sensing of Earth’s surface using satellite or airborne cameras, e.g., for mineral exploration, vegetation monitoring, and land analysis. Recently, the subsequent technical advancements and
ongoing cost reduction of hyperspectral cameras have made their deployment practical and possible in other emerging applications, such as food safety inspection, medical diagnostics, industrial sorting, drug analysis, biometric forensics, archaeology, and so on.

The spectral content of individual pixels in hyperspectral images is typically a mixture of the spectra of multiple materials [2], [3]. This phenomenon, called spectral mixture, is attributed to multiple factors, including mainly the low spatial resolution of hyperspectral imaging devices, the diversity and intimate interactions of materials in the imaged scenes, and multiple photon reflections from layered objects. Separating the spectra of individual pixels into a set of spectral signatures (called endmembers), and determining the abundance fraction of each endmember, is an essential task in quantitative hyperspectral subpixel analysis. This process, denoted as spectral unmixing or mixed pixel decomposition, is currently used in many applications, such as conventional mineral distribution analysis in remote sensing, biodistribution analysis in fluorescence images, food quality and composition control, and pharmaceutical inspection.

Diverse solutions have been proposed for the spectral unmixing problem. They can be divided primarily into two categories: physics-based and data-driven methods. Physics-based methods exploit physical models to estimate light scattering and interaction mechanisms among multiple materials in the imaged scene. The observed spectra are explicitly related to the endmembers and abundances, albeit with strong assumptions regarding photon-interacting processes. In practice, it is difficult to accurately model real scenes, and sophisticated models with complex mathematics are usually difficult to estimate. By contrast, data-driven unmixing methods can overcome some limitations of the physics-based methods by directly deriving mixture models from the observed hyperspectral data. These techniques are effective in uncovering the underlying relationships in complicated scenes with abundant data. Nevertheless, explicit and useful knowledge of the physical mixing process is often overlooked in naive data-driven approaches. To reap the benefits of both physics-based and data-driven models, it is essential to properly integrate these techniques to achieve a superior unmixing performance with clear interpretation.

The main purpose of this article is to provide an overview of hyperspectral unmixing models and techniques with special attention focused on the integration of physics-based models and data-driven methods. These methods are analyzed from multiple perspectives, including model structure design, prior capturing, and loss function design. In addition to reviewing the materials dispersed in the literature, we establish a mathematical framework that can characterize and relate these methods. This article provides a set of methods that can serve as good examples for understanding the topics and a selection of references that is sufficient to cover the main contributions. The symbols used in this article are provided in Table 1.

**A general problem formulation of the unmixing problem**

This section establishes a general modeling and optimization framework for spectral unmixing. The classical and recent physics-based and data-driven unmixing methods described subsequently are compatible with this framework.

Generally, hyperspectral unmixing consists of three essential steps: the estimation of the number of endmembers present in the imaged scene, their extraction from available data, and the estimation of their respective abundances. Estimating the number of endmembers can be performed with signal dimension estimation techniques or circumvented with sparse regression methods. Estimating the endmembers and their abundances can be performed sequentially or simultaneously. During the unmixing process, endmembers and abundances are often constrained to preserve physical properties and interpretability. The endmembers represent the pure spectra existing in the imaged scene, implying that the so-called endmember nonnegativity constraint (ENC) is usually imposed. The abundances represent the contribution in percentage of each pure component, meaning that the abundance nonnegativity constraint (ANC) and the abundance sum-to-one constraint (ASC) are often considered. The feasible regions of endmembers and abundances are denoted by $\Omega_M$ and $\Omega_a$, with

$$\Omega_M = \{ M : M \succeq 0 \},$$

$$\Omega_a = \{ a : a \succeq 0, a_1 + a_2 + \cdots + a_R = 1 \},$$

where $\succeq 0$ denotes elementwise nonnegativity of its matrix or vector argument. The feasible region $\Omega_M$ is for the ENC, and the feasible region $\Omega_a$ is for the ANC and ASC. The parameter $R$ denotes the number of endmembers available in the imaged scene.

---

**Table 1. The main symbols used in this article.**

| Symbols | Definitions |
|---------|-------------|
| $x, X$  | Scalar      |
| $x$     | Column vector |
| $X$     | Matrix      |
| $X$     | 3D data cube |
| $N$     | Number of pixels |
| $L$     | Number of spectral bands |
| $R$     | Number of pure spectra (endmembers) |
| $\gamma$ | Image with $N$ pixels and $L$ spectral bands |

| Symbols | Definitions |
|---------|-------------|
| $y$     | A general observed pixel |
| $y_i$   | $i$th pixel of the image, with $1 \leq i \leq N$ |
| $m_i$   | $i$th endmember, with $1 \leq i \leq R$ |
| $M$     | Endmember matrix with $R$ pure spectra |
| $a$     | A general abundance vector |
| $a_i$   | Abundance vector of the $i$th pixel |
| $Y$     | Matrix by stacking $\gamma$ with $N$ pixels |
| $A$     | Abundance matrix of $N$ pixels |
An observed pixel spectrum \( y \) can be described by the following general mixture mechanism:

\[
y = F^*(M,a) + \epsilon, \tag{3}
\]

where \( F^* \) defines the inherent photon interaction mechanism parameterized by the endmembers and their abundances. In a nutshell, spectral unmixing aims at determining the endmembers and their respective abundances by considering the inverse process of \( (F^*)^{-1} : y \rightarrow \{M,a\} \) under the constraints defined by \( \Omega_M \) and \( \Omega_a \), provided that this inverse is known or can be estimated. From the perspective of mathematical optimization, the general form of the unmixing problem can be formulated in different terms depending on known information and priors.

**Problem formulation 1**

Consider a scene with \( N \) pixels \( \{y_i\}_{i=1}^N \). Under the assumption that the mixture mechanism is defined by an explicit transformation \( F \), the optimization formulation of the unmixing problem is given by

\[
\big[M, \{a_i\}_{i=1}^N\big] = \arg\min_{M,\{a_i\}_{i=1}^N} \sum_{i=1}^{N} L(y_i, \hat{y}_i) + R(M,\{a_i\}_{i=1}^N) \quad \text{with} \quad \hat{y}_i = F(M,a_i) \quad \text{s.t.} \quad M \in \Omega_M \quad \text{and} \quad a_i \in \Omega_a \quad \text{for} \quad \forall i, \tag{4}
\]

where

- \( L(\cdot, \cdot) \) is a loss function defining the dissimilarity between its two arguments
- \( R \) is a regularizer that incorporates priors on endmembers \( M \) and abundances \( \{a_i\}_{i=1}^N \).

Extensive research has been conducted to derive explicit mathematical formulations for describing mixture mechanisms. These physics-based models are constructed with different assumptions regarding how photons interact with scene materials. The works in [2] and [4] provide thorough reviews on physical mixture models and associated unmixing methods. Some typical models are presented here as examples.

**Linear mixture model**

In the linear mixture model (LMM), the macroscopically pure components are assumed to be homogeneously distributed in separate patches within the view field. Each photon interacts with only one pure material before reaching the observer, without being affected by any other material. The LMM is expressed as follows:

\[
y = a_1 m_1 + a_2 m_2 + \cdots + a_N m_N + \epsilon = M a + \epsilon, \tag{5}
\]

that is, the observed spectrum \( y \) is a linear combination of \( a_j m_j \), where \( m_j \) is the spectrum of the \( j \)-th pure material weighted by its abundance \( a_j \). An additive noise is denoted by \( \epsilon \). A general form of the LMM considers different endmembers at each pixel due to, e.g., spectral variability. It is formulated as

\[
y_i = M_i a_i + \epsilon_i, \tag{6}
\]

where \( M_i \), having the same structure as \( M \) defined in Table 1, is the endmember matrix at the \( i \)-th pixel.

**Bilinear model**

Bilinear models [5] generalize the LMM by introducing second-order reflection signatures to capture, e.g., photon scatterings caused by multiple vegetation layers in agricultural scenarios. The mixture model is described by

\[
y = Ma + \sum_{j=1}^{g} \sum_{i=1}^{g} \gamma_{ij} m_i \odot m_j + \epsilon, \tag{7}
\]

where \( \odot \) denotes the Hadamard product, and \( \gamma_{ij} \) stands for the nonlinear contribution of the \( i \)-th and \( j \)-th endmembers. The expression of \( \gamma_{ij} \) allows us to define different bilinear models. For instance, \( \gamma_{ij} = \beta_{ij} a_i a_j \) yields the generalized bilinear model in [5], where \( \beta_{ij} \) controls the interaction strength between the \( i \)-th and \( j \)-th endmembers.

**Hapke model**

The Hapke model has been developed to characterize intimate mixtures with complex photon interactions [6]. The full Hapke model is complex and requires a number of parameters that are generally unavailable in real-world scenes. Under certain reasonable assumptions and simplifications, such as spherical particles and isotropic scatters, the relationship between the bidirectional reflectance \( y \) and the single-scattering albedo (SSA) \( w \) becomes

\[
y = S(w) = \frac{w}{(1 + 2 \mu \sqrt{1 - w})(1 + 2 \mu_0 \sqrt{1 - w})}, \tag{8}
\]

where the division is the elementwise operator, and \( \mu_0 \) and \( \mu \) are the cosines of the incoming and outgoing radiation angles, respectively. In spite of being nonlinear in terms of reflectance, the intimate mixture model is linear in the SSA domain and can be expressed as follows:

\[
w = \sum_{j=1}^{g} a_j w_j. \tag{9}
\]

Unmixing problems with physics-based models frequently fall under Problem formulation 1 when \( F \) is explicitly defined. We now give two examples of linear unmixing problems with known and unknown endmembers, respectively.

**Example 1 (fully constrained least-squares unmixing)**

Under the linear model assumption \( F(M,a) = Ma \) with known \( M \), considering independent pixels and using the squared \( \ell_2 \)-norm error \( L(y, \hat{y}) = \| y - \hat{y} \|^2 \), problem (4) reduces to the following fully constrained least-squares (FCLS) problem [7]:

\[
\hat{a} = \arg\min_{a} \| y - M a \|^2 \quad \text{s.t.} \quad a \in \Omega_a. \tag{10}
\]
Example 2 [regularized nonnegative matrix factorization (NMF) unmixing]

Under the linear model assumption $Y = MA$, for $i = 1, \ldots, N$, and using $L(y_i, \hat{y}_i) = \|y_i - \hat{y}_i\|^2$ with a regularization term, problem (4) becomes

$$\{M, \{a_i\}_{i=1}^N\} = \arg\min_{M, \{a_i\}_{i=1}^N} \sum_{i=1}^N L(y_i, \hat{y}_i) + R(M, \{a_i\}_{i=1}^N)$$

s.t. $M \in \Omega_M$, and $a_i \in \Omega_a$. \hfill (11)

where we notice that $\|Y - MA\|^2 = \sum_{i=1}^N \|y_i - Ma_i\|^2$. The design of regularizers $R$ for enhancing the unmixing performance has been extensively studied in the literature. For example, a minimum volume-based term is usually introduced as a geometrical constraint on the endmember estimation [8]. Total variation, sparsity, and low rankness are usually used to regularize the abundance estimation [9].

Though the physics-based model and unmixing with Problem formulation 1 have been widely used, it is clear that the real mixture mechanism may not be explicitly known or accurately assumed, and the actual physics-based models have a limited nonlinearity modeling capacity. Simultaneously learning the function $F$ from data while determining the endmembers and abundances appears to be a remedy to this problem.

**Problem formulation 2**

Under the assumption that the mixture mechanism $F$ is unknown or partially unknown, the optimization formulation that learns $F$ from the data can be formulated as

$$\{M, \{a_i\}_{i=1}^N, F\} = \arg\min_{M, \{a_i\}_{i=1}^N, F} \sum_{i=1}^N L(y_i, \hat{y}_i) + R(M, \{a_i\}_{i=1}^N)$$

s.t. $y_i = F(M, a_i)$

$$with \arg\min_{M \in \Omega_M, a_i \in \Omega_a} \|\hat{y}_i - F(M, a_i)\|.$$ \hfill (12)

where $\mathcal{H}$ denotes a functional space where $F$ lies. Typical nonlinear unmixing problems with conventional machine learning techniques often fall into this category [10], [11].

Learning $F$ from data opens the possibility of determining mixture mechanisms that are difficult to formulate using explicit descriptions and expressions. However, this strategy may fail if $F$ is not sufficiently parameterized and jointly constrained with physics-based models. Some researchers have specified several forms for $F$ to make it more interpretable. In addition to the trivial linear form, the additive nonlinear model and postnonlinear model are two major categories that have been extensively investigated.

Additive nonlinear models

Models in this category consider that the mixture consists of a linear mixture with an additive nonlinear component involving the endmembers and their abundances; that is,

$$y = Ma + F_{add}(Ma) + \epsilon.$$ \hfill (13)

Several subcategories, depending on specific forms for $F_{add}$, have been investigated to make the problem tractable, including

- **The linear-mixture/nonlinear-fluctuation model**: This model assumes that the mixture consists of two components, namely, a linear mixture component and a nonlinear fluctuation depending on the interaction between the endmembers; that is,

$$y = Ma + F_{add}(m_1, \ldots, m_N) + \epsilon.$$ \hfill (14)

Though useful and attractive, this model has some limitations: the additive nonlinear fluctuation function of this model does not depend on the abundances, and all endmembers contribute uniformly to the nonlinearity of the model.

- **The generalized linear-mixture/nonlinear-fluctuation model**: This form suggests that the contribution of each material to the nonlinear interaction is proportional to its abundance; that is,

$$y = Ma + F_{add}(a_1m_1, \ldots, a_Hm_H) + \epsilon.$$ \hfill (15)

This assumption appears reasonable given that a material with insignificant abundance may contribute minimally to both the linear and nonlinear mixing components in $y$ and vice versa.

Postnonlinear models

Models in this category consider that a spectrum is produced by distorting the LMM with a nonlinear function, that is,

$$y = F_{post}(Ma) + \epsilon.$$ \hfill (16)

A specific form can be constructed by setting $F_{post} = I + F_{add}$, leading to the following additive postnonlinear model:

$$y = Ma + F_{add}(Ma) + \epsilon,$$ \hfill (17)

which can also be viewed as a special case of (15) by setting $F_{add}(a_1m_1, \ldots, a_Hm_H) = F_{add}(a_1m_1 + \cdots + a_Hm_H)$.

Figure 1 shows the hierarchical structure of the preceding categories and how they cover typical physics-based mixture models. The following examples illustrate the relationship between Problem formulation 2 and two typical nonlinear unmixing techniques.

Example 3 (K-Hype and regularized K-Hype unmixing)

The derivation of the K-Hype algorithm is based on a linear-mixture/nonlinear-fluctuation model of the form (14). The nonlinear fluctuation function $F_{add}$ is constrained to be a member of a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_{RKHS}$ and expressed with kernels. Considering pixels independently and assuming that $M$ is known, the optimization problem is given by

$$\{a, F_{add}\} = \arg\min_{a, F_{add} \in \mathcal{H}_{RKHS}} \frac{1}{2} \sum_{i=1}^L \epsilon_i^2 + \frac{1}{2} \left(\left\|F_{add}\right\|^2 + \|a\|^2\right)$$

with $\epsilon_i = y_i - m_i^T a - F_{add}(m_i)$ \hfill (18)
s.t. $a \in \Omega_a$.
where $m_{ij}$ denotes the $i$th row of $M$, i.e., all reflectances at band $\lambda_i$. This problem can be solved using the Lagrangian method, which yields an explicit expression relating $\hat{a}$ and the dual variables and expands $\mathcal{F}_{\text{add}}$ as a sequence of kernels parameterized by $\{m_{ij}\}_{i=1}^N$. The K-Hype formulation, along with its sparse variant [10] and its spatially regularized variant in [11], clearly falls into Problem formulation 2.

Example 4 (unmixing with manifold learning/geodesic distance)

The geodesic distance can be calculated to conduct data unmixing on a manifold rather than in a linear space. This process considers a nonlinear but continuous bijective mapping $C$ between the linear space of abundance coefficients and a manifold in the spectral space,

$$y = C(Ma) + \epsilon,$$

which is of the form (16) as shown in [12] and [13]. More specifically, defining a K-nearest neighbor graph with the dataset, the geodesic distance can be approximated as the length of the shortest path between two points. The endmembers $\{m_i\}_{i=1}^N$ span the simplex with the largest volume on the data manifold, and the abundance coefficients of each data to be unmixed can be expressed as volume ratios in this manifold. The manifold formulation also falls into Problem formulation 2 as $C$ is learned from the data.

Problem formulations 1 and 2 cover physics-based methods or conventional machine learning techniques with handcrafted regularizers. See “Implications of Problem Formulations 1 and 2”. This article focuses more on the integration of physics-based methods with recent machine learning techniques powered by DNNs and other data-driven technologies. Problem formulation 3 encompasses these new methods by introducing additional hierarchical structures on endmembers and abundances.

**Problem formulation 3**

Under the assumption that the mixture mechanism $\mathcal{F}$ is unknown or partially unknown, the optimization formulation that 1) jointly learns $\mathcal{F}$ from data and 2) considers hierarchical reparameterization of the endmembers and abundances can be formulated as

$$\begin{align*}
\{M, \Theta_M, \{\hat{a}_i\}_{i=1}^N, \Theta_a, \mathcal{F}\} = \\
\arg\min_{M, \Theta_M, \{\hat{a}_i\}_{i=1}^N, \Theta_a, \mathcal{F} \in \mathcal{H}} \sum_{i=1}^N \mathcal{L}(y_i, \hat{y}_i) + \mathcal{R}(M, \{a_i\}_{i=1}^N), \\
\text{s.t. } M \in \Omega_M, \text{ and } a_i \in \Omega_a.
\end{align*}$$

(20)

**Implications of Problem Formulations 1 and 2**

Problem formulations 1 and 2 imply that the following three steps are essential for achieving a good unmixing performance:

- Determining a mixture model $\mathcal{F}$ that matches the photon interaction mechanism
- Defining a loss function that characterizes the dissimilarity between the reconstructed and the observed spectra
- Constructing a regularizer that captures the prior information on the variables.
where $\Theta_{M}$ and $\Theta_{a}$ are parameters that allow us to determine the endmember matrix $M$ and the abundances $\{a_{i}\}_{i=1}^{N}$, respectively, and $H$ denotes a given function space. For example, in recent DNN structures, $\Theta_{a}$ can represent the encoder in an autoencoder unmixing framework, and $\Theta_{M}$ can represent the decoder part that generates the endmembers. Recent physically inspired data-driven methods will be reviewed in the following three sections. Their connections to Problem formulation 3 will also be established. In contrast to conventional machine learning-based unmixing methods, which mostly focus on learning mixture models from data, recent data-driven methods have shown their superiority in improving a variety of aspects including model design, prior capturing, and loss function design.

**Integrating physics-based models into DNN design**

In practice, it appears natural to model $F^{-}$ in (3) using DNNs since they offer an enhanced modeling capacity compared to conventional nonlinear modeling techniques. Instead of using DNNs as black boxes, elegant designs of network structures for modeling (15)–(17) are indispensable for preserving physical interpretation and then facilitating the extraction of abundances and endmembers. Deep autoencoder networks are considered as the most appropriate ones and are intensively studied for this purpose. Essentially, an autoencoder consists of an encoder network and a decoder network. The encoder part seeks a low-dimensional representation (e.g., the abundances) to reconstruct the input image. Given an input $y$,

$$q = f_{\text{enc}}(y),$$  

(21)

where $f_{\text{enc}}$ is the function representing the encoder with $\Theta_{\text{enc}}$ denoting all network parameters. The decoder part reverses the functionality of the encoder by decompressing the hidden representation vector to reconstruct the original input data, namely,

$$\hat{y} = f_{\text{dec}}(q),$$

(22)

where $f_{\text{dec}}$ is the function representing the decoder with $\Theta_{\text{dec}}$ denoting all network parameters. For instance, an autoencoder can be designed to provide an estimate $q$ of the abundances $a$ and to estimate the endmembers and the mixture mechanism with the decoder network. Then, identifications $\Theta_{a} \sim \Theta_{\text{enc}}$ and $F \sim f_{\text{dec}}$ can be considered to fit these structures into Problem formulation 3.

In the following, rather than simply presenting the original autoencoders introduced in existing works as a whole, we explore the designs of encoders and decoders separately to demonstrate their extra flexibility and the potential of combining different designs. The basic structure of an autoencoder and typical encoder and decoder designs for unmixing are summarized in Figure 2. We start with decoders that are, in fact, associated with mixture models. Then, we proceed with the encoders that implicitly benefit from data priors.

**Design of structured decoders to learn models from data**

Starting with the decoder of a single linear layer, we progressively demonstrate that the mixture models learned by an autoencoder are actually dependent on the decoder.

**Linear decoder**

The relationship between the input and output of a fully connected linear layer decoder is as follows:

$$\hat{y} = f_{\text{dec}}(\rho) = W\rho,$$

(23)

As shown in D1 of Figure 2, this form is consistent with the LMM defined in (5). Consequently, the learning process of the autoencoder, which minimizes the gap between $\hat{y}$ and $y$, leads to $\rho$ as an estimate of $a$ and $W$ as an estimate of the endmember matrix $M$. Despite the fact that the linear unmixing problem has been extensively studied with various classes of methods [14], [15], [16], [17], [18], [19], this structure motivates the other sophisticated designs that learn structured models from the data.

**Postnonlinear decoder**

With the linear decoder at hand, one can connect it to additional nonlinear layers to create the following postnonlinear structure:

$$\hat{y} = f_{\text{dec}}(\rho) = \Psi(W\rho),$$

(24)

where $\Psi$ denotes an implicit mapping function represented by the subsequent network layers with nonlinear activation functions. Parameters $\Theta_{\text{dec}}$ can be determined by training the autoencoder so that the nonlinear function is learned from the data. Compared with the postnonlinear mixture model defined in (16), $\Psi$ is learned to mimic the generative model $F_{\text{post}}$, and the endmembers are obtained from the weights of the first layer of the decoder [20]. The structure that forms this process is shown by D2 in Figure 2.

**Additive nonlinear decoders**

Instead of adding subsequential nonlinear layers, a network branch can be added in parallel to the linear layer for constructing additive nonlinear structures. However, the parameters coupled in the linear and the nonlinear components should be carefully addressed.

- **Additive postnonlinear decoder**: Passing the linear component through a nonlinear network and adding the output to the linear component leads to the following result:

$$\hat{y} = W\rho + \Psi(W\rho).$$

(25)

The structure that conducts this process is depicted in D3 of Figure 2. Comparing the output of this decoder to the model in (17), the endmember can be estimated from the network weights of the linear component, and $\Psi$ mimics $F_{\text{add}}$ in (17) once the network is learned [21].

- **Generalized additive nonlinear decoder**: Constructing a decoder that mimics the generalized linear-mixture/nonlinear-fluctuation model (15) requires more specific designs. First, a block diagonal matrix $W_{b} = \text{bdiag}[w_{1}, \ldots, w_{R}]$ is
considered for the weights of the first linear layer, which is then sparsely connected. Consequently, the output of this layer is a collection of $w_i$ weighted by $u_i$, that is, $u_i w_i$.\(^{(26)}\)

On one hand, the linearly mixed component of the model can be obtained by adding all $R$-spaced entries of $W_b \varphi$ such that

$$\mathcal{T}(W_b \varphi) = W \varphi$$ \(\text{(27)}\)

with $\mathcal{T} : \mathbb{R}^{L \times L} \rightarrow \mathbb{R}^L$ defined as $\mathcal{T}(x) = [\sum_{i=1}^{R} x_{R(i-1)+1}, \sum_{i=1}^{R} x_{R(i-1)+2}, \ldots, \sum_{i=1}^{R} x_{R(i-1)+L}]$. On the other hand, the weighted nonlinear fluctuation component is obtained by feeding $W_b \varphi$ to a nonlinear network representing $\Psi$. Finally, the sum of these two components,

$$\hat{y} = \mathcal{T}(W_b \varphi) + \Psi(W_b \varphi),$$ \(\text{(28)}\)

is consistent with the model in (15). The endmembers are estimated by $W$, and the nonlinear mixing function $F_{\text{add}}$ is estimated by $\Psi$ from training data [22]. The network

---

**FIGURE 2.** The structure of an autoencoder for hyperspectral unmixing and a variety of selections of encoder and decoder designs. FCN: fully connected network; CNN: convolutional neural network.
structure that achieves this design is depicted in D5 of Figure 2.

- **Bilinear decoder**: Designing a decoder to mimic the bilinear model defined in (7) leads to an autoencoder formulated as

$$\hat{y} = W\varphi + Db. \quad (29)$$

where $D = [d_1, d_2, \ldots, d_c] \in \mathbb{R}^{k \times c}$ is a cross-product matrix involving all pairs of endmembers, i.e., $d_{ij} = w_i \odot w_j$ with $c = R(R-1)/2$, and $b = [b_1, b_2, \ldots, b_c] \in \mathbb{R}^c$ denotes the interaction abundance matrix. The structure of this model can be seen from D4 in Figure 2. Referring to (7), the weights of the decoder provide estimates of the endmembers and their cross products. Vector $b$ provides estimates of the cross products of all pairs of abundances $b = [\varphi_1 \varphi_2, \varphi_1 \varphi_3, \ldots, \varphi_i \varphi_j]$ or the latent features learned from the data [23], [24].

**Endmember-generating decoder**

Instead of being extracted from the weights of a decoder, the endmembers can also be estimated from a subnetwork; that is,

$$\hat{y} = f_{\Theta_{\text{end}}}(z) \varphi = \Psi(z) \varphi. \quad (30)$$

where $f_{\Theta_{\text{end}}}$ represents the endmember generative model, with $\Theta_{\text{end}}$ denoting the network parameters. This subnetwork can be a regular or variational autoencoder (VAE) that maps pixels to endmembers [25], [26] or a pretrained decoder to generate the endmembers from a latent vector [27]. The variable $z$ denotes the input of this subnetwork, which can be pixels, candidate endmembers, or latent representations. This decoder is also a powerful tool to cope with spectral variability [26], [27]. Compared with Problem formulation 3, we identify $\Theta_M = \Theta_{\text{end}}$, leading to a hierarchical structure $M(\Theta_M)$.

**Design of encoders to integrate priors from data**

The encoder part is designed to map the input spectra into a latent representation space, which is often considered as the abundance space. The encoder implicitly acts as a regularizer relating abundances and spectra and imposes a hierarchical structure on $a$ with parameters $\Theta_a$, i.e., of the form $a(\Theta_a)$. The encoder part has been designed in the following ways in the literature.

- **Fully connected network-based encoders**

  Early works [14], [15], [16], [17] employ fully connected layers to construct the encoder. The latent representation of the $k$th layer is given by

$$h_k = \sigma_k(U_k h_{k-1} + b_k) \quad \text{with } h_0 = y \quad (31)$$

where $\sigma_k$ represents the nonlinear activation function, and $U_k$ and $b_k$ stand for the weights and bias of the $k$th layer, respectively. The output of the final layer is denoted by $\varphi$ and is connected to the decoder. Some variants have also been designed based on this structure. For example, in [14] and [15], a sparse encoder is used to enforce abundance sparsity. The work [15] applies a denoising encoder to address the problem of noisy data and outliers in hyperspectral data.

- **Convolutional neural network-based encoders**

  Convolutional neural networks (CNNs) have also been used for designing the encoder [18], [19], [22]. Encoder inputs can be the entire image or overlapping patches. A 2D or 3D convolution operator operates on 3D image cubes $Y_n$ as follows:

$$H_k = \sigma_k(\mathcal{H}_{k-1} \ast U_k + B_k)$$

with $\mathcal{H}_0 = Y_n. \quad (32)$

where $\mathcal{H}_k$ denotes a convolution kernel, and $B_k$ represents the bias parameters. As 2D convolutional kernels may cause local spectral distortion, 3D-CNN encoders are used in [19] and [22] to jointly learn the spatial structures and local spectral features of 3D input images.

- **Recurrent neural network encoders**

  Recurrent neural networks, such as long short-term memory (LSTM) networks, benefit from memory cells and are better suited for processing sequences. With their recurrent hidden states depending on previous steps, they can take full advantage of the sequential characteristics in hyperspectral data. The abundances are estimated with an LSTM to encode spectral correlation and band-to-band variability information [21].

- **Kernel-layer-based encoders**

  Another class of approaches uses kernel layers to process hyperspectral data. Such layers can map the input pixels into a manifold and estimate the abundances according to a Riemannian metric. For example, in [24], the kernel layer is defined as

$$a_i = \exp(- \beta \cdot \|y_i - c_i\|^2). \quad (33)$$

where $\beta$ is the kernel parameter and $c_i$ is the centroid related to the $i$th endmember. The smaller the distance between the mixed pixel and the endmember centroid, the larger the corresponding abundance.

- **Variational encoders**

  A variational encoder encodes each input pixel with a distribution in the latent space as opposed to a single vector. In practice, it is common to consider Gaussian distributions parameterized by their means and covariance matrices. The work in [16] uses a variational encoder to infer abundances. In [26], a probabilistic generative model is used to address endmember variability and provide more accurate abundance and endmember estimates.

- **Multibranch encoders**

  Some works use multibranch encoders to capture different features by sharing weights or concatenating features. In [28],
an endmember-guided encoder is used to transfer the endmember information into a parallel abundance estimate encoder. The work [29] uses a dual-branch encoder, where one branch consists of fully connected layers to extract spectral information, and the other branch consists of 2D convolution layers to capture spatial information.

Remark
The ANC and ASC are usually addressed at the last layer of the encoder either by using specific activation functions [e.g., a rectified linear unit (ReLU), a sigmoid, or an absolute operator] to enforce the ASC followed by a normalization layer to enforce the ANC or by using SoftMax to enforce the ANC and ASC simultaneously. They can also be addressed approximately via a penalty term in the loss function.

Example 5 [a 3D-CNN-based autoencoder network for additive nonlinearity unmixing]
Considering the 3D-CNN-based encoder (E2 in Figure 2) and the generalized additive nonlinear decoder defined in (28) (D5 in Figure 2) leads to the algorithm proposed in [22].

Example 6 [a denoising autoencoder with sparsity for spectral unmixing]
Considering the denoising and sparse encoder and the linear decoder defined in (23) (E1 and D1 in Figure 2) leads to the method proposed in [15].

Example 7 [a kernelization and cross-product layer-based autoencoder for bilinear and postnonlinear unmixing]
Considering the kernel-layer-based encoder (E4 in Figure 2), the bilinear nonlinear decoder defined in (29) (D4 in Figure 2), and the postnonlinear decoder defined in (25) (D3 in Figure 2) yield the method proposed in [24].

Example 8 [a probabilistic generative model-based autoencoder for spectral unmixing with spectral variability]
A two-stream autoencoder framework with a probabilistic generative model (E3 and D6 in Figure 2) that copes with spectral variability leads to the method proposed in [26].

Example 9 [a multitask-based autoencoder network for bilinear spectral unmixing]
The combination of two deep autoencoders with a multitask learning framework leads to the method in [23]. One of the autoencoders models the linear components and estimates the endmembers as well as their abundances, while the other one models the bilinear components defined in (29) and estimates interaction abundances.

In addition to the regular autoencoders that have been described so far, some variants of autoencoder frameworks also have been proposed for spectral unmixing. For instance, in [16], a group of autoencoders is first applied to address issues regarding outliers and generate a good initialization. Then, an autoencoder with a single-layer linear decoder is used to perform the process of unmixing. To improve the unmixing performance, two cascaded autoencoders are utilized in an end-to-end fashion to address cycle-consistency constraints in [30]. In [31], the endmembers are extracted using a geometric endmember extraction method, and the abundances are estimated using a deep image prior based on a CNN (depicted by E5 in Figure 2).

In [32], a two-stream network, with one stream for spectral information and the other one for exploiting spatial information, is collaboratively learned to make full use of the spectral and spatial information. The work [33] constructs an attention network to fuse the information of lidar data into the unmixing process. See “Unmixing With Deep Autoencoder Networks” for summarized insights for the roles of the encoder and decoder in the unmixing.

Prior information learning with data-driven approaches
In addition to modeling the spectral mixture mechanism, DNNs are superior to conventional machine learning techniques in extracting information from data priors. It is intriguing to use DNNs to explicitly learn prior knowledge from data and use it with physical model-based inversion algorithms. Instead of designing sophisticated regularizers, the plug-and-play paradigm uses a denoiser with deep architectures to learn an image prior and then incorporates the denoiser into an iterative optimization of the problem in (4). On the other hand, deep unrolling algorithms aim to unfold an iterative optimization of the problem in (4) with a specific regularizer into a trainable end-to-end deep network.

Incorporating data-driven priors with denoisers
Benefiting from the variable splitting technique, the plug-and-play framework makes it possible to exploit the priors learned from training data to solve various image restoration problems. State-of-the-art denoising algorithms, especially fast-speed and powerful CNN-based denoisers, are usually plugged into this framework as the proximity operator that captures the intrinsic spatial and spectral structures of hyperspectral images. By leveraging the flexibility of model-based optimization and the powerful learning

Unmixing With Deep Autoencoder Networks
Remarks on unmixing with deep autoencoder networks are as follows:
- The underlying unmixing model used in an autoencoder network is determined by the structure of the decoder, independently of the encoder.
- The encoder implicitly incorporates the inference relation from the observed spectra to the abundances and can thus be considered as a prior for unmixing.
- Unmixing with an autoencoder falls into Problem formulation 3 with nonlinear function $f^*$ and parameters $\Theta^*$ being associated with the decoder network, and parameters $\Theta^*$ learned by the encoder network.
ability of CNN, this framework has shown great potential in performing unmixing tasks. Among the three problem formulations, the form based on Problem formulation 1 is considered here for illustration because of its simplicity. In the plug-and-play framework, an auxiliary variable $Z$ is introduced in (4) to replace $A$ in the regularization term, with an extra constraint $Z = A$ to ensure the equivalence of the optimization problems. Using the alternating direction method of multipliers (ADMM) results in the following iterative updates:

\[
\begin{align*}
\{M^{(k+1)}, A^{(k+1)}\} &= \arg\min_{M, A} \sum_{i=1}^{N} \mathcal{L}(y_i, \hat{y}_i) + \frac{\rho}{2} \| A - Z^{(k)} + V^{(k)} \|_F^2 \\
& \quad \text{s.t. } M \in \Omega_M, A_i \in \Omega_{A_i}, \quad (34a) \\
Z^{(k+1)} &= \arg\min_{Z} \frac{\rho}{2} \| Z - (A^{(k+1)} + V^{(k)}) \|_F^2 + \mathcal{R}(Z), \quad (34b) \\
V^{(k+1)} &= \nabla \mathcal{L}(y_i, \hat{y}_i) + A^{(k+1)} - Z^{(k+1)}, \quad (34c)
\end{align*}
\]

where the superscript $(k)$ refers to the iteration index, $\rho$ is the penalty parameter, and $V^{(k)}$ is the dual variable to be updated at iteration $k$. In this formulation, the regularizer $\mathcal{R}(A)$ can be implicitly defined by a plugged denoising operator Denoiser, and (34b) can be seen as the denoising procedure of $(A^{(k+1)} + V^{(k)})$ and replaced by

\[
Z^{(k+1)} = \text{Denoiser} (A^{(k+1)} + V^{(k)}).
\]

Generally, Denoiser can be any off-the-shelf denoising operator. This offers the opportunity of incorporating CNN-based denoisers with powerful prior learning ability into the physical model-based iterative optimization. In particular, the work [34] considers the LMM assumption in (5) with known $M$ and incorporates conventional or deep learning-based denoisers with two strategies, namely, a direct regularizer on $A$ and a regularizer on reconstructed spectra, i.e., $MA$. Considering the case of an unknown $M$, the work in [35] jointly estimates $M$ and exploits a handcrafted sparsity prior with the regularizer $\| A \|_{2,1}$ and a denoiser prior with implicit regularizer $\mathcal{R}(A)$ in the plug-and-play framework to produce more stable unmixing results. The work in [36] proposes a nonlinear plug-and-play unmixing method based on the general bilinear model in (7), and both traditional and CNN-based denoisers are plugged into the iterative optimization.

Unrolling iterative optimization into learnable deep architectures

Unlike plug-and-play algorithms, the deep unrolling paradigm unfolds and truncates iterative optimization algorithms into trainable deep architectures. Considering the LMM in (5) with known $M$, deep unrolling methods solve the unmixing problem in (10) with the $\| A \|_p$ sparsity regularizer using iterative algorithms, such as the iterative soft thresholding algorithm (ISTA) and the ADMM algorithm. In [37], by introducing an auxiliary variable $z$ such that $z = a$ along with a dual variable $v$, the optimization problem is iteratively solved by the ADMM with the following steps:

\[
\begin{align*}
ad^{(k+1)} &= W_y + B(z^{(k)} + v^{(k)}), \quad (36a) \\
z^{(k+1)} &= \max(\text{soft}(a^{(k)} - v^{(k)}, \lambda \rho), 0), \quad (36b) \\
v^{(k+1)} &= v^{(k)} - \eta(a^{(k)} - z^{(k)}), \quad (36c)
\end{align*}
\]

where $\lambda$ is the regularization parameter, $\rho$ denotes the penalty parameter, $\eta$ is a parameter offering additional flexibility, and $W = (M^T M + \mu I)^{-1} M^T$ and $B = (M^T M + \mu I)^{-1} \rho$. Operator soft$(\cdot)$ in (36b) is the soft-threshold operator given by soft$(\cdot, \lambda \rho) = \text{sign}((\cdot) - \lambda \rho_+ \cdot)$. The key feature of deep unrolling unmixing algorithms is to leverage a reparameterization of these iterative update steps through the consecutive DNN layers (denoted as $f_a, f_z,$ and $f_v$) with learnable parameters \{$W^{(k+1)}, B^{(k+1)}, \theta^{(k+1)}, \eta^{(k+1)}\}$, respectively:

\[
\begin{align*}
a^{(k+1)} &= f_a(z^{(k)}, v^{(k)}, y, W^{(k+1)}, B^{(k+1)}) \\
z^{(k+1)} &= \text{ReLU}(a^{(k+1)} - v^{(k)} - \theta^{(k)}) I, \\
v^{(k+1)} &= f_v(a^{(k+1)}, z^{(k+1)}, v^{(k)}; \eta^{(k+1)}) \\
&= v^{(k)} - \eta^{(k+1)}(a^{(k+1)} - z^{(k+1)}),
\end{align*}
\]

where $\theta^{(k+1)}$ is a learnable parameter that plays the role of $\lambda \rho$, and $\text{ReLU}(\cdot)$ is a componentwise ReLU operation. These specifically designed layers are then cascaded into an entire DNN architecture for unmixing, which can be trained with supervised loss functions in an end-to-end manner. For the blind unmixing problem in (11), the work in [38] builds a nonconvex sparse NMF model using $\| A \|_p$ with $0 < p < 1$ and designs an interpretable sparsity constrained NMF network based on the ISTA to jointly estimate $M$ and $a$.

Integrating loss learned from data into physics-based inverse problems

The loss functions $\mathcal{L}$ in (4), (12), and (20) also play a crucial role in determining the unmixing performance. In addition to the data-driven model, data-learned loss functions have been proposed recently to improve the unmixing performance, particularly for Problem formulation 3. The subsequent review focuses on autoencoder-based works as they mostly refer to Problem formulation 3.

General geometric distances as cost functions

Geometric distances, which are usually used as the essential loss in autoencoder-based unmixing methods, enforce the network to reconstruct the input image. The works [14], [15], and [16] use the mean-square error (MSE) to measure the similarity between the input and reconstructed pixels; that is,
In contrast to the MSE, which is sensitive to the scale of spectra, the spectral angle distance (SAD) and spectral information divergence (SID) are scale invariant. The SAD considers the angle between two spectra as a spectral similarity measure:

$$
L_{\text{SAD}} = \frac{1}{N} \sum_{i=1}^{N} \arccos \left( \frac{\langle y_i, \hat{y}_i \rangle}{\| y_i \| \| \hat{y}_i \|} \right).
$$

(39)

The SID models the spectra as a probability distribution and considers the band-to-band spectral variability as the uncertainty existing in random variables to lower the gap between input and reconstructed pixels,

$$
L_{\text{SID}} = \frac{1}{N} \sum_{i=1}^{N} p_i \log \left( \frac{p_i}{\hat{p}_i} \right),
$$

(40)

where \( p = (y^T y) \) and \( \hat{p} = (\hat{y}^T \hat{y}) \) represent the probability distribution vector of the input and estimated pixels, respectively. Nevertheless, these conventional losses are defined at the pixel level based on basic statistical assumptions that disregard inherent image structures.

**Deep learning metrics as cost functions**

Generative adversarial networks (GANs) have recently been used to address the limitations of geometric distance loss functions. A GAN consists of two components, a generator and a discriminator, and it models the distribution of data using DNNs. The generator \( G \) is trained to generate samples that are similar to real data to confuse the discriminator. The discriminator \( D \) receives samples from the real data distribution (positive data) and samples from the generator (negative data) and attempts to distinguish between the real and generated samples. These two networks engage in a min–max adversarial game during the training process that can be expressed as follows:

$$
\min_{G} \max_{D} E_{z \sim q(z)} [\log D(z)] + E_{x \sim p(x)} [\log (1 - D(G(z)))].
$$

(41)

The architecture of unmixing methods using adversarial loss usually contains two terms: an autoencoder and a discriminator. The former, considered to be the generator, aims to estimate the abundances and extract the endmembers. The latter is a deep binary classifier and is used to distinguish between generated and real samples. Thus, the discriminator may be able to transfer intrusive properties of real data that are useful for unmixing. The loss function utilized for generator training is always composed of two terms,

$$
L_{G} = L_{\text{AE}} + L_{\text{adv}},
$$

(42)

where \( L_{\text{adv}} = \frac{1}{N} \sum_{i=1}^{N} \log D(z_i) + \frac{1}{N} \sum_{i=1}^{N} \log (1 - D(\hat{z}_i)) \) is the loss function that is used to generate samples \( \hat{z} \) that confuse the discriminator, and \( L_{\text{AE}} \) is the reconstruction loss function, e.g., the geometric cost function. The loss function for training the discriminator is defined as \( L_{D} = -L_{\text{adv}} \). The work [39] introduces the Wasserstein distance calculated from the discriminator as a regularization term to characterize the distribution similarity between the input and reconstructed spectra. In [40], an adversarial network is utilized to model the abundance distribution.

Recent works have introduced the perception mechanism to discover the discrepancy between reconstructed pixels and their corresponding ground truth (input pixels). This strategy allows us to relax pixel-level reconstruction and enhance the unmixing performance using an end-to-end process. The perceptual similarity can be viewed as a feature-matching operator that compares the perceptual features of the reconstructed output pixels to those of the input pixels. The expression for the perceptual loss defined in a feature domain is

$$
L_{\text{Perceptual}} = \frac{1}{N} \sum_{i=1}^{N} \| \mathcal{P}(y_i) - \mathcal{P}(\hat{y}_i) \|_2^2,
$$

(43)

where \( \mathcal{P} \) stands for a feature extractor. For example, in [30], a cycle-consistency strategy is used to further refine the detailed information in the unmixing process. In [39], the features extracted from the hidden layers of a discriminator are used to ensure the consistency of high-level representations.

**Discussion and perspectives**

This article reviewed hyperspectral unmixing methods with special attention focused on the integration of physics-based and data-driven methods. A mathematical framework was established to introduce a variety of existing works with a unified formalism. As summarized in this article, notable advances have been recently achieved in this topic. Table 2 lists some features of the surveyed methods and summarizes some of their pros and cons. However, there are still several important aspects that merit further discussion and investigation.

**Performance evaluation**

Evaluating the performance of hyperspectral unmixing algorithms has always been nontrivial since few research works have been devoted to producing ground-truth data [41]. Performance evaluation becomes even more difficult when considering data-driven methods. Preliminary comparison results of several autoencoder-based unmixing methods with synthetic and real datasets have been provided recently in [42]. This study demonstrates the significance of using appropriate loss functions, nonlinear methods, spatial information, and proper initialization. However, thorough and objective evaluation studies are still missing.

**Model selection**

Real data usually involve multiple mixing models in complex scenes. It is anticipated that data-driven models, such as the generalized linear-mixture/nonlinear-fluctuation model, will
Table 2. Features of the surveyed methods.

| Reference | Architecture | Model | Loss | Blind | Initialization | Prior | General Pros and Cons |
|-----------|--------------|-------|------|-------|----------------|-------|------------------------|
| [14]      | Shallow layer | LMM   | MSE+SAD | √     | VCA            | $\ell_1$ on abundances, $\ell_1$ on endmembers | **Pro 1:** Integrating physics-based models in decoder design enables the methods to model complex mixture mechanism with physical characteristics. |
| [15]      | Untied denoising autoencoder | LMM   | MSE | √     | VCA+FCLS       | $\ell_1$ on encoder weights | **Pro 2:** Convolutional filters are able to capture the spatial correlation. |
| [16]      | Stacked autoencoder, VAE | LMM   | MSE | √     | VCA+FCLS       | Minimum volume |
| [17]      | Deep encoder | LMM   | SAD | √     | Randomly       | N/A |
| [18]      | CNN encoder | LMM   | SAD | ×     | Randomly       | N/A |
| [19]      | 3D-CNN encoder | LMM   | SAD | √     | VCA            | TV on abundances |
| [20]      | Postnonlinear decoder | NLMM  | MSE | √     | VCA            | TV on abundances |
| [21]      | LSTM encoder, additive postnonlinear decoder | NLMM  | MSE | √     | VCA            | TV on abundances |
| [22]      | 3D-CNN encoder, generalized additive nonlinear decoder | NLMM  | MSE | √     | VCA            | TV on abundances, $\ell_1$ weight decay |
| [23]      | Multitask learning, bilinear decoder | NLMM  | MSE | √     | VCA+FCLS       | Graph regularizer on abundances |
| [24]      | Kernel-layer-based encoder, bilinear decoder, postnonlinear decoder | NLMM  | MSE | √     | VCA            | N/A |
| [25]      | Two-stream autoencoder | LMM   | SAD-SAD | √     | Region-based VCA | TV on pixel |
| [26]      | VAE | Spectral variability | MSE+KL+SAD | √     | VCA            | Minimum volume |
| [27]      | VAE | MSE+KL | √     | FCLS     | TV on abundances |
| [28]      | Multibranch encoder | NLMM  | MSE | √     | VCA            | $\ell_1$ weight decay |
| [29]      | Dual-branched encoder | LMM   | Squared sine distance | √     | VCA            | $\ell_{10}$ on abundances |
| [30]      | Cascaded autoencoder | LMM   | Cycle-consistency loss | √     | VCA            | ASC constraint |
| [31]      | Deep image prior | LMM   | MSE | √     | VCA            | N/A |
| [32]      | Two-stream autoencoder, collaborative learning | LMM   | SAD | √     | VCA+FCLS       | $\ell_{10}$ on abundances |
| [33]      | Attention networks, multimodal feature encoder | LMM   | SAD | √     | VCA            | $\ell_{10}$ on abundances |
| [34]      | Plug-and-play | LMM   | MSE | ×     | FCLS           | Denoiser prior |
| [35]      | Plug-and-play | LMM   | MSE | √     | VCA+FCLS       | Denoiser prior, $\ell_{10}$ on abundance |
| [36]      | Plug-and-play | NLMM  | MSE | ×     | FCLS           | Denoiser prior |
| [37]      | Unroll ADMM | LMM   | MSE+SID | √     | VCA            | $\ell_1$ on abundances |
| [38]      | Unroll NMF | LMM   | MSE | √     | VCA+FCLS       | $\ell_0$ on abundances |
| [39]      | Deep learning metric loss | LMM   | SAD+Wasserstein distance+perceptual loss | √     | Randomly       | N/A |
| [40]      | Adversarial network | LMM   | SAD+adversarial loss | √     | VCA+FCLS       | N/A |

NLMM: nonlinear mixture model; VCA: vertex component analysis
exhibit superior scene-specific adaptability. Nevertheless, it is challenging, if not impossible, to preselect an optimal model in advance. Using a mixing model with parallel multiple branches and an attention module to balance the contribution of each mixing model is a possible strategy to address this issue. Utilizing superpixel-based strategies and adopting distinct models within each superpixel and at its borders is an alternative solution.

**A fully integrated framework**

This article reviews the use of autoencoders to learn mixture models from data, the plug-and-play method to learn priors from data, and perception dissimilarity characterized by features learned from data. Existing works include one or two of these techniques. Investigating an elegant strategy combining physics-based and data-driven methods that benefits from all of these factors is still missing. This requires a deeper understanding and original methods in both aspects of optimization and neural network design.

**Fast implementation**

Compared to classical optimization methods, data-driven approaches using DNNs require significantly higher computational resources, especially when multiple data-driven modules collaborate to perform unmixing. Fast implementation must be considered for the practical use of unmixing algorithms. The formulation of the optimization problems, the design of the network structures, and the way of integrating them are crucial for enhancing the convergence speed of the learning process. A trained encoder can be used for supervised or semisupervised unmixing when the computation resource is limited. Further, a Compute Unified Device Architecture implementation can also be useful for executing parallel optimization-solving modules.

**Acknowledgment**

The work of J. Chen was supported in part by the Natural National Science Foundation of China under Grants 62192713 and 62171380, Guangdong International Cooperation Project 2022A0505050020, Shenzhen Research Grant JCYJ20220530161606014, Shaanxi Key Industrial Innovation Chain Project 2022ZDLGY01-02, and Xi’an Technology Industrialization Plan XA2020-RGZNTJ-0076. The work of C. Richard was supported by the French government through the 3IA Côte d’Azur Investments in the Future project with reference number ANR-19-P3IA-0002. Jie Chen is the corresponding author.

**Authors**

**Jie Chen** (dr.jie.chen@ieee.org) received his Ph.D. degree from Troyes University of Technology (UTT), France, in 2013. He received his B.S. degree in 2006 from Xi’an Jiaotong University (XJTU), China, and his M.S. and Dipl.-Ing. degrees in 2009 from XJTU and UTT, respectively, all in information and telecommunications engineering. From 2013 to 2014, he was with the University of Nice Sophia Antipolis, France, and from 2014 to 2015, he was with the University of Michigan, United States. He is currently a professor at the School of Marine Science and Technology at Northwestern Polytechnical University, Xi’an 710072 China. His current research interests include adaptive signal processing and distributed optimization with applications to hyperspectral image processing and acoustic signal processing. He is a Senior Member of IEEE.

**Min Zhao** (minzhao@mail.nwpu.edu.cn) received her B.S. degree in electronic and information engineering and her M.S. degree in signal and information processing from Northwestern Polytechnical University, Xi’an 710072, China, in 2017 and 2020, where she is currently pursuing her Ph.D. degree. Her research interests include hyperspectral image analysis and object detection. She received the People’s Choice Award for the Three Minute Thesis Competition at the IEEE International Geoscience and Remote Sensing Symposium 2020 and the (with her team) Champion of Grand Challenges on Near-Infrared Image Colorization at the IEEE International Conference on Visual Communications and Image Processing 2020. She is a Student Member of IEEE.

**Xiuheng Wang** (xiuheng.wang@oca.eu) received his B.S. degree in electronic and information engineering and his M.S. degree in signal and information processing from Northwestern Polytechnical University, Xi’an, China, in 2018 and 2021, respectively. He is currently pursuing his Ph.D. degree at the Université Côte d’Azur, Nice 06000 France. His research interests include inverse problems in image processing, machine learning, and change point detection. He was a corecipient of the Champion of Grand Challenges on Near-Infrared Image Colorization at the IEEE International Conference on Visual Communications and Image Processing 2020. He is a Student Member of IEEE.

**Cédric Richard** (cedric.richard@unice.fr) received his Dipl.-Ing. and M.S. degrees in 1994 and his Ph.D. degree in 1998, all in electrical and computer engineering from the Compiègne University of Technology, France. He is a full professor at Université Côte d’Azur, Nice 06000 France. He is a senior chair at the Interdisciplinary Institute for Artificial Intelligence Côte d’Azur, in the chapter Artificial Intelligence for Smart and Secure Territories. His current research interests include statistical signal processing and machine learning. He is the author of more than 300 journal and conference articles. He was distinguished as a junior member of the Institut Universitaire de France in 2010–2015. He is a Senior Member of IEEE.

**Susanto Rahardja** (susantorahardja@ieee.org) received his Ph.D. and M.Eng. degrees from Nanyang Technological University, Singapore, and his B.Eng. degree from the National University of Singapore; all in the field of electrical and electronic engineering. He is a Ph.D. advisor at Northwestern Polytechnical University Xi’an 710072 China and a professor at the Singapore Institute of Technology. His...
