Deep unrolling of the multiplicative updates algorithm for blind source separation, with application to hyperspectral unmixing.

Christophe Kervazo
LTCI, Télécom Paris
Institut Polytechnique de Paris
Palaiseau, France
christophe.kervazo@telecom-paris.fr

Abdelkhalak Chetoui
LTCI, Télécom Paris
Institut Polytechnique de Paris
Palaiseau, France
0009-0001-7198-1059

Jérémy E. Cohen
Univ Lyon, INSA-Lyon, UCBL, UJM, CNRS, Inserm, CREATIS UMR 5220, U1294, F-69621,
Lyon, France
jeremy.cohen@cnrs.fr

***Abstract*—Blind Source Separation (BSS) has gained a large interest in many fields, including hyperspectral unmixing which is broadly used in remote sensing and astrophysics. BSS being an ill-posed problem, many strategies have been proposed to solve it, ranging from model-based to deep-learning ones. While model-based algorithms are in general interpretable, in contrast with neural networks, these algorithms often require a large number of iterations and obtain worse unmixing results that their deep-learning counterparts. To try to obtain the best of both worlds, in this work we unroll the multiplicative updates algorithm, leading to two new algorithms. The first one, NALMU, learns some parameters which are fixed once the training is over. The second one, ALMU, enables the parameters of the unrolled algorithm to be predicted by small neural networks, making the whole algorithm adaptative to the specific datasets considered in the test phase. We conduct experiments on two astrophysic datasets, and show that our approach enables to largely outperform the other unmixing unrolled algorithms, while largely reducing the number of iterations compared to the original multiplicative updates algorithm.**

***Index Terms*—Blind source separation, hyperspectral unmixing, deep unrolling, interpretable deep learning, nonnegative matrix factorization, multiplicative updates.**

I. Blind source separation / unmixing

Blind source separation (BSS) has applications in a large variety of fields, such as astrophysics [1], biomedical imaging [2], chemistry [3] and hyperspectral remote sensing [4], in which case it is known under the name of HyperSpectral Unmixing (HSU). BSS assumes the observed multi-valued signals $\mathbf{X} \in \mathbb{R}^{m \times t}$, with m the number of observation channels and t the number of samples, to have been generated by n unknown elementary signals, called the sources $\mathbf{S}^* \in \mathbb{R}^{n \times t}$. Despite its limitations [5] and due to its simplicity, many BSS algorithms further assume the mixing model to be linear:

$$\mathbf{X} = \mathbf{A}^* \mathbf{S}^* + \mathbf{N},$$

where $\mathbf{A}^*$ are some unknown mixing coefficients and $\mathbf{N}$ stands for potentially any additive noise, although we will assume in the following a white Gaussian noise. In short, BSS aims at recovering the sources $\mathbf{S}^*$ as well as the mixing coefficients $\mathbf{A}^*$ from the data matrix $\mathbf{X}$ [6], up to a scaling and permutation indeterminacy.

Classical BSS approaches: BSS is severely ill-posed, since for any invertible matrix $\mathbf{P} \in \mathbb{R}^{n \times n}$, $\hat{\mathbf{A}} = \mathbf{A}^* \mathbf{P}$ and $\hat{\mathbf{S}} = \mathbf{P}^{-1} \mathbf{S}^*$ are such that $\mathbf{X} = \hat{\mathbf{A}} \hat{\mathbf{S}} + \mathbf{N}$. Therefore, solving it requires introducing further knowledge about the sought-after $\mathbf{A}^*$ and $\mathbf{S}^*$ matrices. In this context, several matrix factorization approaches have been proposed in the last decade to incorporate further handcrafted priors. Among the first ones, independent component analysis [6], [7] assumes the statistical independence of the sources. On the other hand, sparse matrix factorization [8], [9] assume the sources to be sparse in a (potentially transformed) domain. Lastly, Nonnegative Matrix Factorization (NMF) [10], [11] has recently obtained broad success due to its mathematical guarantees and the fact that the nonnegativity prior is met in a large number of applications.

BSS through deep learning: Due to their flexibility, several deep learning BSS neural networks have also been introduced. In the field of hyperspectral unmixing, many such approaches leverage the linear mixture model by using auto-encoders with linear decoders [12]. Going a step further, a few methods enable to cope with non-linear models by dealing with the spectral variabilities. Nevertheless, such approaches often rely on black-box architectures, thus lacking in interpretability. Leveraging a different point of view, some algorithms have tried to bridge model-based and deep-learning algorithms; among them, Plug-and-Play [13] and deep unrolling [14] methods exhibit a higher interpretability than black-box neural networks. Due to its high computational efficiency, we focus in this work on the latter category. Deep unrolling originated from the work of [15], where a reparametrization of the iterative skrinkage thresholding algorithm (ISTA) was learnt from a training set, enabling to drastically reduce the number of iterations. Several works towards this direction have then been conducted [14], [16], which have been extended later on to BSS. Among them, DNMF [17] unrolls the well-known Multiplicative Updates (MU) algorithm for performing NMF. Nevertheless, in that work, unrolling is only performed for the $\mathbf{S}$ factor, without alternating between the $\mathbf{A}$ and $\mathbf{S}$ updates. This might potentially reduce the estimation quality. More focused on hyperspectral unmxing, MNNBU [18] and SNMF

[19] unroll a sparsity-based algorithm, the latter algorithm better enabling to take into account potential spectral variabilities within the image. Very closely related to these two algorithms, the LPALM [20] unrolls a sparsity based BSS algorithm in a transformed wavelet domain. Lastly, the authors of [21] propose to leverage a group sparsity constraint to better take into account the spatial dependencies in hyperspectral images.

Contributions: due to its popularity in the NMF field, we propose in this work to unroll the iterates of the MU algorithm, leading to a first algorithm, the NALMU. This has only been considered in the work [17], which however does not use an alternating framework and uses a different update, which leads to worse separation results (as will be confirmed in the experimental section). In addition and in contrast to what is usually done in deep unrolling, we propose in a second algorithm, coined ALMU, to learn a data-adaptative reparametrization of the MU iterates. This enables a higher flexibility when the test samples exhibit a large degree of variability. The experimental section shows the interest of the two proposed algorithms, which largely outperform two other existing unrolled methods, while enabling a huge reduction in terms of number of iterations compared to the original MU. This is even more interesting since both NALMU and ALMU are trained on simplistic synthetic datasets.

II. Proposed approach

In the following, we briefly review the MU algorithm, which is the starting point of the two methods we propose, detailed in the next subsections.

A. Multiplicative updates

The MU algorithm [22] is a wide-spread algorithm for BSS and HSU, known for its simplicity: the updates only take the form of componentwise multiplications. Starting from positive initializations of $\mathbf{A}$ and $\mathbf{S}$, the MU performs L_{MU} iterations, each of them alternating between updates of the $\mathbf{A}$ and $\mathbf{S}$ factors. An iteration l of the MU writes as:

$$\mathbf{A}^{(l+1)} \leftarrow \mathbf{A}^{(l)} \odot \frac{\mathbf{X}\mathbf{S}^{(l)^T}}{\mathbf{A}^{(l)}\mathbf{S}^{(l)}\mathbf{S}^{(l)^T}}$$
$$\mathbf{S}^{(l+1)} \leftarrow \mathbf{S}^{(l)} \odot \frac{\mathbf{A}^{(l+1)^T}\mathbf{X}}{\mathbf{A}^{(l+1)^T}\mathbf{A}^{(l+1)}\mathbf{S}^{(l)}}.$$

B. Non Adaptative Learned Multiplicative Updates (NALMU)

We propose to apply deep unrolling to the MU algorithm, leading to a first algorithm denoted as NALMU, summarized in Algorithm 1. To this end, we introduce in the MU updates of matrix $\mathbf{A}$ a number L_{NALMU} of trainable matrices $\underline{\mathbf{W}_\mathbf{A}}^{(l)} \in \mathbb{R}^{m\times n}, l \in \{1..L_{NALMU}\}$:

$$\mathbf{A}^{(l+1)} \leftarrow \mathbf{A}^{(l)} \odot \underline{\mathbf{W}_\mathbf{A}}^{(l)} \odot \frac{\mathbf{X}\mathbf{S}^{(l)^T}}{\mathbf{A}^{(l)}\mathbf{S}^{(l)}\mathbf{S}^{(l)^T}}. \quad (1)$$

The main insight in doing so is that, by learning the $\underline{\mathbf{W}_\mathbf{A}}^{(l)}$ matrices on a training dataset, the resulting NALMU updates might be more adapted to the test dataset at hand, reducing the number of required iterations compared to the MU and potentially improving the separation quality. Introducing the above updates form is motivated by several observations:

1) setting all the entries of the $\underline{\mathbf{W}_\mathbf{A}}^{(l)}$ matrices to 1 for every layer, we recover the original MU — as such, the algorithm should in theory be able to outperform the original MU, provided that the number of iterations L_{NALMU} is large enough. This is in contrast to the DNMF [17] algorithm, in which the $\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$ matrices are replaced by two trainable matrices, which are thus highly dependent on the input data.
2) From our empirical observations, the $\underline{\mathbf{W}_\mathbf{A}}$ matrix in (1) might behave as a masking matrix, enabling to enhance the unmixing by focusing during the first iterates on the least mixed rows of $\mathbf{A}$ and setting to zero the other ones. Moreover, using different matrices $\underline{\mathbf{W}_\mathbf{A}}^{(l)}$ for each layer (untied weights) enables to empirically obtain better estimates of $\mathbf{A}^*$ and $\mathbf{S}^*$.
3) We could in principle follow a similar update rule as (1) for $\mathbf{S}$, leading to:

$$\mathbf{S}^{(l+1)} \leftarrow \mathbf{S}^{(l)} \odot \underline{\mathbf{W}_\mathbf{S}}^{(l)} \odot \frac{\mathbf{A}^{(l+1)^T}\mathbf{X}}{\mathbf{A}^{(l+1)^T}\mathbf{A}^{(l+1)}\mathbf{X}},$$

with $\underline{\mathbf{W}_\mathbf{S}}^{(l)} \in \mathbb{R}^{m\times t}$. We did not do so, because it would require the $\mathbf{S}^*$ matrices in the training set and in the test set to have the same sizes, which is a strong assumption in the case of the images we will consider in the experimental section. Moreover it largely increases the number of trainable parameters, thus requiring more training samples to obtain good results.

Algorithm 1 NALMU

Require: $\mathbf{X}, L_{NALMU}$
  Initialize $\mathbf{A}^{(1)}$ and $\mathbf{S}^{(1)}$ with positive coefficients
  for $l \in \{1..L_{NALMU}\}$ **do**
    $\mathbf{A}^{(l+1)} \leftarrow \mathbf{A}^{(l)} \odot \underline{\mathbf{W}_\mathbf{A}}^{(l)} \odot \frac{\mathbf{X}\mathbf{S}^{(l)^T}}{\mathbf{A}^{(l)}\mathbf{S}^{(l)}\mathbf{S}^{(l)^T}}$
    $\mathbf{S}^{(l+1)} \leftarrow \mathbf{S}^{(l)} \odot \frac{\mathbf{A}^{(l+1)^T}\mathbf{X}}{\mathbf{A}^{(l+1)^T}\mathbf{A}^{(l+1)}\mathbf{S}^{(l)}}$.
  end for
  return $\mathbf{A}^{(L+1)}$ and $\mathbf{S}^{(L+1)}$

C. Adaptative Learned Multiplicative Updates (ALMU)

A limitation of the above NALMU algorithm is that the $\underline{\mathbf{W}_\mathbf{A}}^{(l)}$ matrices are the same for all $\mathbf{X}$ samples in the (training and test) datasets. While such approach already lead to interesting results as will be shown in the experimental section, we propose here to make the $\underline{\mathbf{W}_\mathbf{A}}^{(l)}$ matrices adaptative to the considered data $\mathbf{X}$ matrix, enabling improved results when the considered $\mathbf{X}$ matrices have a large variability in the train and test datasets. A difficulty is however that the $\mathbf{X}$ matrices can be high dimensional. To reduce the computational burden, we thus rather preferred to make the $\underline{\mathbf{W}_\mathbf{A}}^{(l)}$ matrices depend on a first guess of $\mathbf{A}^*$, obtained using NALMU and denoted as $\mathbf{A}_{NALMU}$. The adaptative $\mathbf{W_A}^{(l)}$ matrices, now denoted as $\mathbf{W_A}^{(l)}(\mathbf{A}_{NALMU})$ to emphasize their depency on $\mathbf{A}_{NALMU}$,

are in practice parametrized using Multi-Layer Perceptrons (MLPs). There are L_{ALMU} such matrices. The resulting algorithm, called Adaptative LMU (ALMU) is summarized in Algorithm 2.

Algorithm 2 ALMU

Require: $\mathbf{X}, L_{NALMU}, L_{ALMU}$
$\mathbf{A}_{NALMU}, \mathbf{S}_{NALMU} = \text{NALMU}(\mathbf{X}, L_{NALMU})$
$\mathbf{A}^{(1)} = \mathbf{A}_{NALMU},\ \mathbf{S}^{(1)} = \mathbf{S}_{NALMU}$
for $l \in \{1..L_{ALMU}\}$ **do**
$\mathbf{A}^{(l+1)} \leftarrow \mathbf{A}^{(l)} \odot \underline{\mathbf{W}_{\mathbf{A}}}^{(l)}(\mathbf{A}_{NALMU}) \odot \frac{\mathbf{X}\mathbf{S}^{(l)T}}{\mathbf{A}^{(l)}\mathbf{S}^{(l)}\mathbf{S}^{(l)T}}$
$\mathbf{S}^{(l+1)} \leftarrow \mathbf{S}^{(l)} \odot \frac{\mathbf{A}^{(l+1)T}\mathbf{X}}{\mathbf{A}^{(l+1)T}\mathbf{A}^{(l+1)}\mathbf{S}^{(l)}}$.
end for
return $\mathbf{A}^{(L+1)}$ and $\mathbf{S}^{(L+1)}$

D. Training

The proposed algorithms, NALMU and ALMU, are trained on a supervised dataset containing D matrices ${}^{(d)}\mathbf{X}$ and their associated groundtruths ${}^{(d)}\mathbf{A}$ and ${}^{(d)}\mathbf{S}, d \in \{1..D\}$. The NALMU is trained end-to-end while the ALMU is trained in two stages: the NALMU is first pretrained and then its parameters are frozen while the parameters of the adaptive part of ALMU are updated. We use in the loss the Spectral Angular Distance (SAD), which is defined between two matrices $\mathbf{B} \in \mathbb{R}^{b\times c}$ and $\mathbf{C} \in \mathbb{R}^{b\times c}$ as

$$\text{SAD}(\mathbf{B}, \mathbf{C}) = \frac{1}{b}\sum_{k=1}^{b} < \mathbf{b}_k | \mathbf{c}_k > . \tag{2}$$

After several tests, and in agreement to [12], we indeed found the results to be better when using the SAD rather than the normalized Mean Square Error. Precisely, the loss we aim at minimizing is given by:

$$\mathcal{L} = -\sum_{d=1}^{D}\sum_{l=1}^{L} v^{(l)} \left(\text{SAD}({}^{(d)}\mathbf{A}^{(l)}, {}^{(d)}\mathbf{A}^{*}) + \text{SAD}({}^{(d)}\mathbf{S}^{(l)}, {}^{(d)}\mathbf{S}^{*})\right), \tag{3}$$

with ${}^{(d)}\mathbf{A}^{(l)}, {}^{(d)}\mathbf{S}^{(l)}$ the estimates of the considered learned MU algorithm at the l-th iteration when given as input the ${}^{(d)}\mathbf{X}$ matrix. L is the total number of iterations ($L = L_{NALMU}$ or $L = L_{ALMU}$). The $v^{(l)}$ weights are used are used to put more emphasis on the estimates given by the last iterations. Such an approach is already used in [19], [20]. In this work, we chose weights increasing linearly: $v^{(l)} = \frac{1}{L-1}(l-1)$.

III. Experiments

A. Experimental setting

Considered datasets: The experimental assessment of the NALMU and ALMU algorithms is performed on astrophysical data, in which there are $n = 4$ sources to unmix. The mixing matrices $\mathbf{A}^*$ come from astrophysical simulations, described in [23], which have been derived from real astrophysical data: the Cassiopea A supernovae remnant as observed by the X-ray space telescope Chandra[1]. Every $\mathbf{A}^*$ matrix is composed of 4 different spectra of size 65 each: i) a synchrotron emission spectrum, ii) a thermal emission spectrum that is composed of various emission lines, and iii) two line emission spectra that are related to a single atomic component (e.g. iron) but with different redshifts due to the Doppler effect. Examples of generated $\mathbf{A}^*$ matrices are shown in Figure 1, in which we can see that the spectra exhibit a large variability. Concerning the $\mathbf{S}^*$, which corresponds to the spatial repartition of the different emissions, we have only access to a single dataset, which will be used for testing the algorithms. Precisely, we used in this work three datasets:

- *Training set*: 750 datasets $\mathbf{X}$ are generated in the following way: 750 realistic $\mathbf{A}^*$ matrices are first simulated as mentionned above. Due to the lack of training data for the $\mathbf{S}^*$ matrices, natural images coming from the tiny ImageNet dataset are used. Specifically, we randomly concanate 4 such images to obtain $\mathbf{S}^* \in \mathbb{R}^{4\times 16384}$. Lastly, mixtures are generated adding some white Gaussian noise with a SNR of 60 dB.
  It is interesting to note that the synthetically generated $\mathbf{S}^*$ matrices are quite different from the real repartion maps in supernovae remanants. Nevertheless, we will show that such a simple training method enables to obtain good results on the realistic test dataset.
- *Simplistic test dataset:* 150 datasets $\mathbf{X}$ are generated using exactly the same generation process as for the training set, but different $\mathbf{A}^*$ and $\mathbf{S}^*$ matrices are used.
- *Realistic test dataset:* 150 datasets $\mathbf{X}$ are generated, first by simulating 150 $\mathbf{A}^*$ matrices. Nevertheless, in contrast to the simplistic test dataset, we here assess LPALM using a real $\mathbf{S}^*$coming from a true image of supernovae remnant. A white Gaussian noise is added so that the SNR is 60dB.

Implementation details: the NALMU algorithm was implemented with $L_{NALMU} = 50$ iterations. To assess the ALMU algorithm with a similar total number of iterations, we used $L_{NALMU} = 25$ iterations in the NALMU preprocessing step, and $L_{ALMU} = 25$ in the adaptative step. Concerning the ALMU, each of the L_{ALMU} MLPs used for predicting the $\mathbf{W}_{\mathbf{A}}(\mathbf{A}_{NALMU})^{(l)}, l = 1..L_{ALMU}$ has three linear layers and a ReLU activation function is used. The layers respectively have 265, 130 and 75 dimensional inputs, with ReLU activations (except for the last layer). This architecture was not fine-tuned. The loss in (3) was optimized for 2000 epochs using the ADAM optimizer with a learning rate of 10^{-5}. We used a mini-batch size of 25.

Evaluation metric: to assess the methods, we compute using definition (2) the SADs between each estimated and ground truth component.

B. Results on the simplistic test set

Both NALMU and ALMU are first assessed on the simplistic dataset. The results are reported in Table I, in which the two algorithms are further compared with the original MU algorithm using a large number of iterations, $L_{MU} = 10000$. To better assess the variability of the ALMU algorithm, we

[1] chandra.harvard.edu

TABLE I
QUANTITATIVE EVALUATION OF THE TWO PROPOSED LEARNED MU AND COMPARISON WITH MU ON THE SIMPLISTIC DATASET.

	MU	NALMU	ALMU (averaged)
SAD($\mathbf{A}^*, \mathbf{A}$)	0.9850	0.9626	$0.9986 \pm 2.9620 \times 10^{-5}$
SAD($\mathbf{S}^*, \mathbf{S}$)	0.9951	0.9873	$0.9989 \pm 2.5043 \times 10^{-5}$

have run the adaptive part of the algorithm three times and we display the average results and the standard deviation.

Several remarks are in order. First, all the algorithms perform quite well on this simple dataset. In particular, while the NALMU algorithm obtain the worst results, it is interesting to see that its estimates are only slightly worse than the ones of the MU, while it performs 200 times fewer iterations. On the contrary, the ALMU algorithm obtains the best results, both for $\mathbf{A}^*$ and $\mathbf{S}^*$ estimates. This shows that making the $\mathbf{W}_\mathbf{A}$ matrices adaptative to the considered dataset is crucial to make the unrolled algorithm able to outperform the original MU. In addition, it can be seen that the standard deviation of the ALMU results is quite small, meaning that the training of the $L_{ALMU} = 25$ MLPs for predicting the $\mathbf{W}_\mathbf{A}(\hat{\mathbf{A}})^{(l)}$ matrices is stable enough.

C. Results on the realistic test dataset

The proposed algorithms are now tested on the realistic dataset, in which the $\mathbf{S}^*$ matrix is a real map coming from a true supernovae remanent. Qualitative results are presented in Figure 1, in which we can see that both the estimated $\mathbf{A}$ and $\mathbf{S}$ are of high quality. Quantitative results are given in Table II, in which the algorithm is compared with the two recent unmixing unrolled methods SNMF [19] and DNMF [17]. Due to the way these methods are designed, it is necessary to retrain the corresponding neural networks for each new test image, which we did. For DNMF, a λ hyperparameter is to be tuned by the user. We tried the 3 proposed values in the original article ($\lambda \in \{0, 1, 2\}$) and took the best results. Lastly, the Coordinate Descent (CD) NMF method of scikit-learn [24], which is often recognized as superior to the MU in unmixing problems with Gaussian noise [10], is also used as a benchmark.

First, both the NALMU and ALMU algorithms outperform the other unrolled algorithms. Concerning DNMF, this is likely to be linked to the fact that this method does not leverage alternations between the updates of $\mathbf{A}$ and $\mathbf{S}$. In addition, in contrast to us, the update of $\mathbf{S}$ is unrolled (not the one of $\mathbf{A}$) and the $\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$ matrices are used as end-to-end-learned variables. Concerning SNMF, the results for $\mathbf{A}$ are acceptable but the results for $\mathbf{S}$ are the worst. Note that our approach differs quite a lot from this method, which is mostly based on sparsity.

Compared to the MU and CD, the ALMU algorithm obtain the best results for estimating the $\mathbf{A}^*$ matrix. However, the model-based algorithms perform quite well for estimating the $\mathbf{S}^*$ matrix. In contrast, the matrices $\mathbf{S}^*$ in the ALMU training set may be too far from the test data, slightly deteriorating the results. The experiment we performed in the previous subsection, in which the ALMU algorithm was outperforming the MU algorithm when the test matrices $\mathbf{S}^*$ were generated from tiny ImageNet seems to confirm this hypothesis. Therefore, a path for future improvement might be to find a database containing closer-to-real supernovae remanant $\mathbf{S}^*$ matrices for the training.

In the last experiment, we compare in Figure 2 the results of MU and ALMU as a function of the number of MU iterations. Even with several orders of magnitude fewer iterates, the ALMU outperforms the MU for estimating $\mathbf{A}^*$. Concerning the estimation of $\mathbf{S}^*$, the ALMU obtains, using only 50 iterations, competitive results with the MU for up to 4000 iterations. Therefore, using the proposed method as a preprocessing of the MU seems to be a valid approach for speeding-up the unmixing.

CONCLUSION

In this work, we perform deep unrolling of the multiplicative updates for BSS, leading to two new algorithms, NALMU and ALMU. The ALMU method, by enabling the unrolling parameters to depend on the considered data matrix through a parametrization by small neural networks, largely outperforms the other tested unrolled algorithms on the considered astrophysics dataset. Future work include improving the $\mathbf{S}^*$ training matrices, testing the algorithm on different kind of BSS problems and deriving mathematical guarantees for our algorithm.

ACKNOWLEDGMENT

This work is supported by ANR JCJC project LoRAiA ANR-20-CE23-0010. We would like to thank F.Acero and J. Bobin for having provided the Chandra simulations

REFERENCES

[1] R. C. Gertosio, J. Bobin, and F. Acero, "Semi-blind source separation with learned constraints," *Signal Processing*, 2023.

[2] E. Dereure, C. Kervazo, J. Seguin, A. Garofalakis, N. Mignet, E. Angelini, and J.-C. Olivo-Marin, "Sparse non-negative matrix factorization for preclinical bioluminescent imaging," in *2023 IEEE 20th ISBI*, 2023.

[3] C. Kervazo, J. Bobin, and C. Chenot, "Blind separation of a large number of sparse sources," *Signal Processing*, 2018.

[4] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches," *IEEE journal of selected topics in applied earth observations and remote sensing*, 2012.

[5] C. Kervazo, N. Gillis, and N. Dobigeon, "Provably robust blind source separation of linear-quadratic near-separable mixtures," *SIAM Journal on Imaging Sciences*, 2021.

[6] P. Comon and C. Jutten, *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press, 2010.

[7] A. Hyvärinen, I. Khemakhem, and H. Morioka, "Nonlinear independent component analysis for principled disentanglement in unsupervised deep learning," *Patterns*, 2023.

[8] M. Zibulevsky and B. A. Pearlmutter, "Blind source separation by sparse decomposition in a signal dictionary," *Neural computation*, 2001.

[9] C. Kervazo, J. Bobin, C. Chenot, and F. Sureau, "Use of palm for ℓ_1 sparse matrix factorization: Difficulty and rationalization of a two-step approach," *Digital Signal Processing*, 2020.

[10] N. Gillis, *Nonnegative matrix factorization*. SIAM, 2020.

[11] N. Nadisic, N. Gillis, and C. Kervazo, "Smoothed separable nonnegative matrix factorization," *Linear Algebra and its Applications*, 2023.

[12] B. Palsson, J. R. Sveinsson, and M. O. Ulfarsson, "Blind hyperspectral unmixing using autoencoders: A critical comparison," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2022.

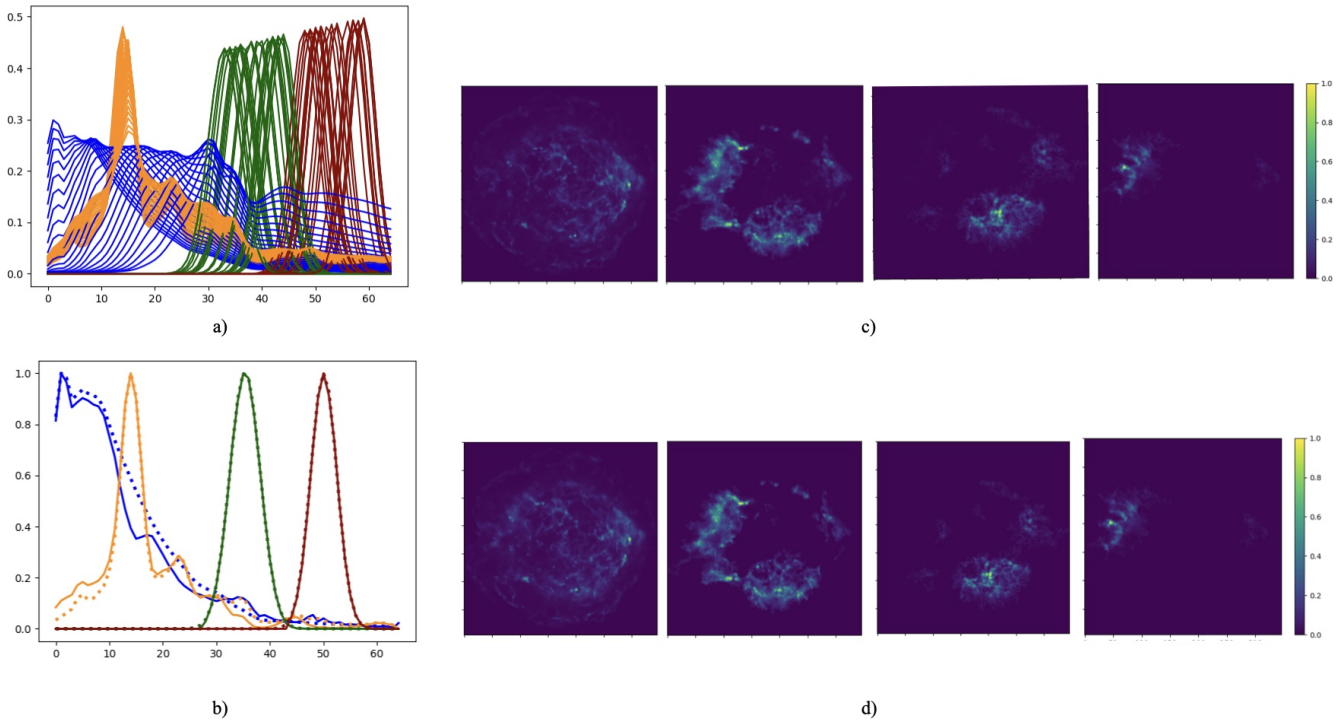

Fig. 1. a) 25 examples of ground-truth synthetic emission spectral (blue: synchrotron, orange: thermal, green: iron 1, red: iron 2); b) example of a estimated mixing matrix $\mathbf{A}$ (dotted line) in comparison with ground-truth (plain line); c) example of ground-truth $\mathbf{S}^*$; d) example of estimated $\mathbf{S}$.

TABLE II
QUANTITATIVE EVALUATION OF THE TWO PROPOSED LEARNED MU ALGORITHMS AND COMPARISONS ON THE REALISTIC TEST SET

	MU	DNMF ($\lambda = 0$)	SNMF	CD	NALMU	ALMU (averaged)
SAD($\mathbf{A}^*, \mathbf{A}$)	0.9650	0.7008	0.7767	0.9815	0.9233	0.9878
SAD($\mathbf{S}^*, \mathbf{S}$)	0.9751	0.8085	0.3353	0.9868	0.9390	0.9681

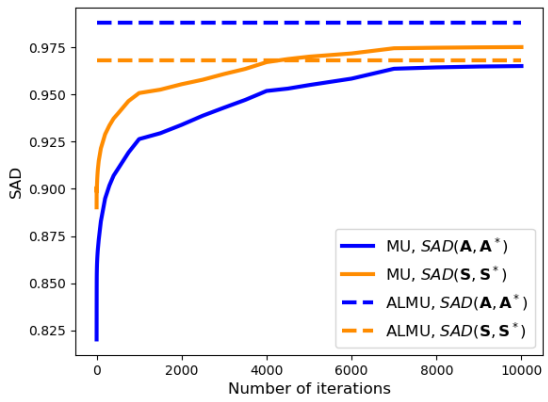

Fig. 2. Evolution of the MU results as a function of the number of iterations. The ALMU results are also plotted for the sake of comparison.

[13] M. Zhao, J. Chen, and N. Dobigeon, "Ae-red: A hyperspectral unmixing framework powered by deep autoencoder and regularization by denoising," *IEEE Transactions on Geoscience and Remote Sensing*, 2024.

[14] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Processing Magazine*, 2021.

[15] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *ICML*, 2010.

[16] X. Chen, J. Liu, Z. Wang, and W. Yin, "Theoretical linear convergence of unfolded ista and its practical weights and thresholds," *Advances in Neural Information Processing Systems*, 2018.

[17] R. Nasser, Y. C. Eldar, and R. Sharan, "Deep unfolding for non-negative matrix factorization with application to mutational signature analysis," *Journal of Computational Biology*, 2022.

[18] Y. Qian, F. Xiong, Q. Qian, and J. Zhou, "Spectral mixture model inspired network architectures for hyperspectral unmixing," *IEEE Transactions on Geoscience and Remote Sensing*, 2020.

[19] F. Xiong, J. Zhou, S. Tao, J. Lu, and Y. Qian, "Snmf-net: Learning a deep alternating neural network for hyperspectral unmixing," *IEEE Transactions on Geoscience and Remote Sensing*, 2021.

[20] M. Fahes, C. Kervazo, J. Bobin, and F. Tupin, "Unrolling palm for sparse semi-blind source separation," in *ICLR*, 2022.

[21] C. Cui, X. Wang, S. Wang, L. Zhang, and Y. Zhong, "Unrolling non-negative matrix factorization with group sparsity for blind hyperspectral unmixing," *IEEE Transactions on Geoscience and Remote Sensing*, 2023.

[22] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, 1999.

[23] A. Picquenot, F. Acero, J. Bobin, P. Maggi, J. Ballet, and G. W. Pratt, "Novel method for component separation of extended sources in x-ray astronomy," *Astronomy & Astrophysics*, 2019.

[24] A. Cichocki and A.-H. Phan, "Fast local algorithms for large scale nonnegative matrix and tensor factorizations," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 2009.