Hyperspectral image processing through implicit regularization by a neural network

Report IMA 206

Cédric Vincent
Pegah Khayatan
Khalil BRAHAM
William Khieu

June 25, 2023

Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 The article . . . 1

2 Contribution and Novelties **2**
- 2.1 Deep Image Prior Methods . . . 2
- 2.2 Abundance Estimation by a DIP Method . . . 3
  - 2.2.1 Hyperspectral Modelling . . . 3
  - 2.2.2 Abundance Estimation . . . 3

3 Model **5**
- 3.1 Inspiration from Deep Image Prior . . . 5
- 3.2 Network architecture . . . 6

4 Reproduction of the Results **7**
- 4.1 Test images . . . 7
  - 4.1.1 Samson dataset . . . 7
  - 4.1.2 Jasper dataset . . . 7
  - 4.1.3 Urban dataset . . . 8
- 4.2 Results . . . 9
  - 4.2.1 Evaluation Metrics . . . 9
  - 4.2.2 First Results . . . 9

5 Parameters choice **11**
  - 5.0.1 Epoch choice . . . 11
  - 5.0.2 Optimizer and learning rate . . . 12

6 Robustness to Noise **13**
  - 6.0.1 Results on noisy data . . . 13

7 Model Structure **16**
- 7.1 Configurable Model . . . 16
- 7.2 Results: Comparing Different Configurations . . . 17

8 Initialisations **21**
- 8.1 Random initialisation . . . 21
- 8.2 PALM initialisation . . . 21
- 8.3 NMF initialisation . . . 21
- 8.4 Results . . . 23
  - 8.4.1 Quantitative results . . . 23
  - 8.4.2 Qualitative results . . . 24
  - 8.4.3 Conclusion on results . . . 24

9 Conclusion **25**

1 Introduction

1.1 Context

The field of hyperspectral imaging has witnessed significant advancements in recent years, thanks to the increasing availability of high-resolution sensors and the development of sophisticated analysis techniques.
Hyperspectral imaging allows for the capture of information across a wide range of wavelengths, providing valuable insights into the composition, properties, and spatial distribution of materials in a scene.
However, one of the key challenges in hyperspectral data analysis is unmixing, which involves separating the mixed spectral signatures of different materials present in a pixel.
Traditional unmixing methods often rely on linear or non-linear models, but they may struggle with complex spectral mixtures and suffer from computational inefficiencies.
The article "UnDIP: Hyperspectral Unmixing Using Deep Image Prior" presents a promising solution that harnesses the power of deep learning and the concept of image priors to address these challenges.

1.2 The article

The article "UnDIP: Hyperspectral Unmixing Using Deep Image Prior" introduces a novel approach to hyperspectral image analysis. Hyperspectral imaging enables the capture of rich spectral information for each pixel, offering great potential for material identification and characterization.
However, unmixing, the task of decomposing mixed spectral signatures, remains a challenge. Traditional methods often struggle with complex mixtures and computational efficiency.
In this study, the authors propose UnDIP, a method that leverages the concept of deep learning and image priors. By exploiting the inherent structures and features present in hyperspectral images, UnDIP aims to improve the accuracy and efficiency of unmixing.
The article presents the theoretical foundations of UnDIP, experimental results, and comparisons with existing methods, demonstrating its effectiveness in hyperspectral unmixing tasks.
This research contributes to the advancement of hyperspectral analysis techniques and offers promising avenues for applications in various fields, including remote sensing, agriculture, and environmental monitoring.

2 Contribution and Novelties

This work falls into the category of deep image prior *(DIP)* methods. Before exploring further this category for the task in hand, let us take a look at its definition and advantages.

2.1 Deep Image Prior Methods

The core idea behind DIP methods, originally introduced by [UVL18], is that they utilize the power of deep neural networks to learn and reconstruct images directly from the input data, without the need for explicit supervision or ground truth labels. By exploiting the inherent structure and priors present in images, DIP methods can perform various image processing tasks, such as denoising, inpainting, super-resolution, and more.

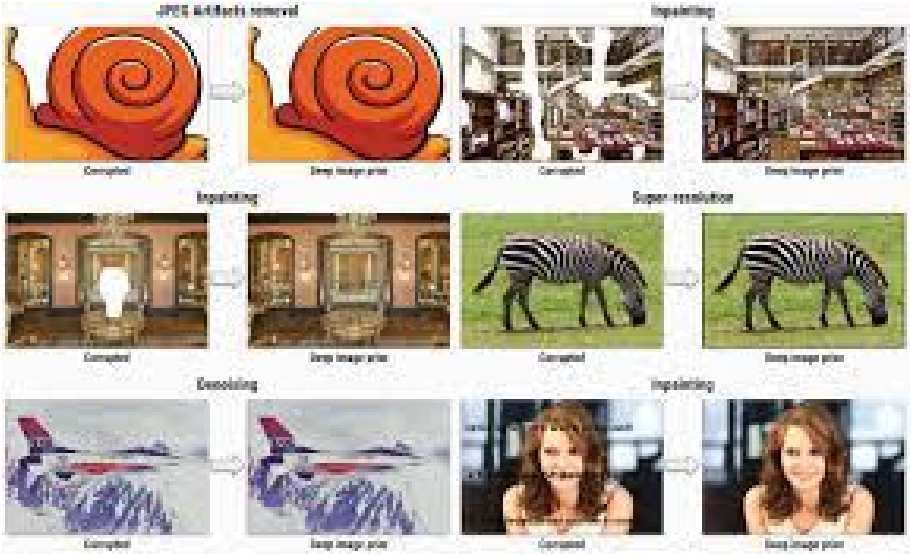

Figure 1: A few applications of deep image prior. Image taken from [UVL18]

DIP methods typically involve training a deep neural network to reconstruct an input image by minimizing a suitable loss function that encourages image fidelity and adherence to the desired priors. The network is initialized randomly or with a noise input, and during the training process, it learns to exploit the structural biases present in the network architecture to generate visually pleasing and high-quality outputs.

Deep image prior methods offer several advantages that make them valuable in various image processing tasks. Firstly, these methods eliminate the need for large-scale annotated training data, which is often a significant challenge in many computer vision applications. Traditional deep learning methods require a massive amount of labeled data for training, making them impractical when labeled datasets are scarce or expensive to obtain. In contrast, deep image prior methods exploit the inherent structure and priors present in the images themselves. By leveraging the structure of the data, these methods can generate high-quality results without relying on extensive labeled data, making them more accessible and cost-effective.

Secondly, deep image prior methods provide a framework for unsupervised learning, enabling them to learn directly from the input data without the need for ground truth labels. This characteristic is particularly advantageous in scenarios where obtaining ground truth annotations is challenging or impossible. Instead of relying on labeled data, deep image prior methods leverage the underlying characteristics of the data to learn and reconstruct high-quality images. This unsupervised approach opens up possibilities for various applications, such as image denoising, inpainting, super-resolution, and more, where high-quality training data may be limited or unavailable.

Overall, the advantages of deep image prior methods lie in their ability to overcome the limitations of traditional deep learning approaches, eliminating the dependency on large-scale annotated datasets and allowing for unsupervised learning.

2.2 Abundance Estimation by a DIP Method

2.2.1 Hyperspectral Modelling

Within the scope of this work, a linear model for Hyperspectral Unmixing is assumed:

$$\mathbf{Y} = \mathbf{X} + \mathbf{N} = \mathbf{EA} + \mathbf{N}$$

Where $\mathbf{Y} \in \mathbb{R}^{p \times n}$ is the observed hyperspectral image and $\mathbf{X} \in \mathbb{R}^{p \times n}$ the unknown image to be estimated. $\mathbf{X}$ is composed of the product of $\mathbf{E} \in \mathbb{R}^{p \times r}$ the endmembers and $\mathbf{A} \in \mathbb{R}^{r \times n}$ the fractionnal abundances. $\mathbf{N} \in \mathbb{R}^{p \times n}$ contains the error of this model, including all noise possibly present in the observed image. n is the number of pixels, p the number of bands in the spectral signature and r the number of endmembers. We assume that $r << p$.

The main objective is to estimate the abundances $\mathbf{A}$, which are the different sources of the signal corresponding to real materials. The image to be estimated $\mathbf{X}$ can be reconstructed using $\mathbf{A}$, either with prior knowledge of the endmembers $\mathbf{E}$ or by estimating or extracting them from $\mathbf{Y}$.

2.2.2 Abundance Estimation

The general concept of Deep Image Prior is generating an image $\mathbf{X}$ from a random initialization $\mathbf{Z}$ by using the neural network as a parametric function: $\mathbf{X} = f_\theta(\mathbf{Z})$ where θ are the parameters of the network f. The network is optimised in an iterative algorithm to obtain the optimal estimated image: $\hat{\mathbf{X}} = f_{\hat{\theta}}(\mathbf{Z})$.

Tasks that can be tackled with DIP, inverse image reconstruction tasks such as denoising, inpainting or super-resolution, can be formulated as optimization problems:

$$\hat{\mathbf{X}} = \underset{\mathbf{X}}{argmin} Q(\mathbf{Y}, \mathbf{X}) + \lambda R(\mathbf{X})$$

Here, the function Q controls the data fidelity and R is a regularization function based on prior knowledge. Such frameworks requires the selection of a good regularizer which depends on the prior knowl-

edge. DIP solves this issue by implicitly replacing the regularizer by a deep neural network:

$$\hat{\theta} = \underset{\theta}{argmin} Q(\mathbf{Y}, f_\theta(\mathbf{Z})) \text{ st. } \hat{\mathbf{X}} = f_{\hat{\theta}}(\mathbf{Z})$$

This implies that the selection of the regularizer is done by optimizing the network. This new optimization problem is solved by the network optimizer, on the network parameters.

In the case of hyperspectral unmixing, the usual method to estimate the abundances is the fully constrained least squares unmixing (FCLSU), which consists in solving the following optimization problem:

$$\hat{\mathbf{A}} = \underset{\mathbf{A}}{argmin} \frac{1}{2} \parallel \mathbf{Y} - \mathbf{E}\mathbf{A} \parallel_F^2 + \lambda R(\mathbf{A}) \text{ st. } \mathbf{A} \geq 0, \mathbf{1}_r^T \mathbf{A} = \mathbf{1}_n^T$$

The regularizer R has been proven to lead to a better estimation of A. UnDIP uses the principles of DIP to replace this regularizer by a deep network. Hence, abundance estimation using deep image prior consists in solving the following optimization problem of the parameters of the network:

$$\hat{\theta} = \underset{\theta}{argmin} \frac{1}{2} \parallel \mathbf{Y} - \mathbf{E} f_\theta(\mathbf{Z}) \parallel_F^2 \text{ st. } \hat{\mathbf{A}} = f_{\hat{\theta}}(\mathbf{Z})$$

The constraints can be enforced in the structure of the network, by using the softmax activation function in the final layer:

$$softmax(A)_{ij} = \frac{e^{\mathbf{A}_{ij}}}{\sum_{i=1}^{r} e^{\mathbf{A}_{ij}}}$$

3 Model

3.1 Inspiration from Deep Image Prior

The hyperspectral unmixing task of abundance estimation can be performed by DIP. As such the method described in DIP has been adapted to match the current problem.

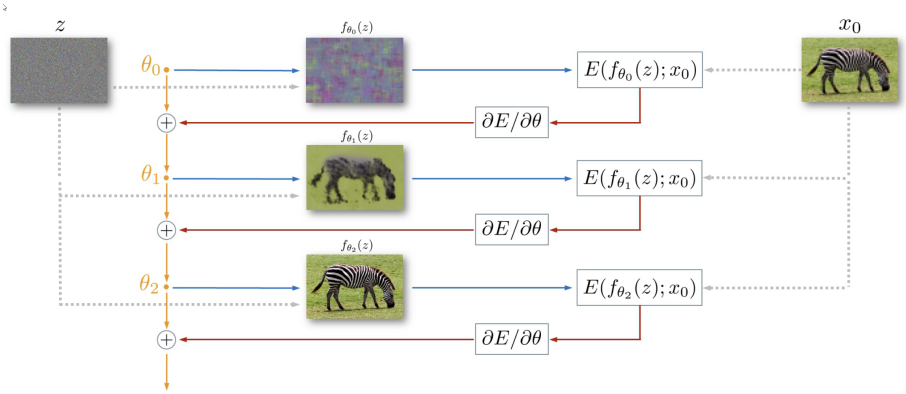

Figure 2: Image restoration using the deep image prior

Starting from a random weights θ_0, DIP iteratively update them to minimize the data fidelity term. At every iteration the weights θ are mapped to an image $x = f_\theta(z)$, where z is a fixed random tensor and f is a neural network with parameters θ. The image x is used to compute the task-dependent loss $E(x, x_0)$. The gradient of the loss w.r.t. the weights θ is then computed and used to update the parameters.

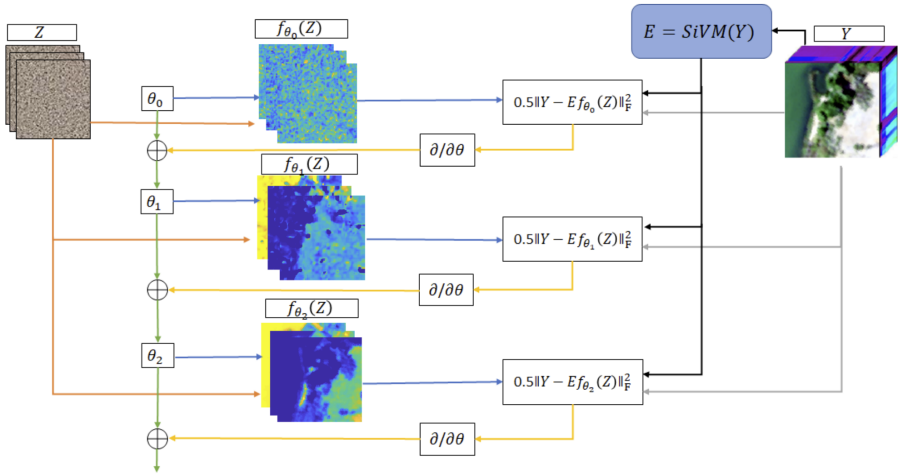

Figure 3: Abundance estimation using the deep image prior model

Using the same method as DIP, UnDIP fixes a random image $\mathbf{Z}$ and maps it to $\hat{\mathbf{A}} = f_\theta(\mathbf{Z})$ using a deep network f_θ. The change happens in the loss function we minimize to ensure the model is adapted to our task. The loss minimized is the reconstruction loss $\frac{1}{2} \parallel \mathbf{Y} - \mathbf{E}f_\theta(\mathbf{Z}) \parallel_F^2$. This loss utilize both the noisy observed hyperspectral image $\mathbf{Y}$ and the endmembers $\mathbf{E}$ extracted from it using $SiVM$

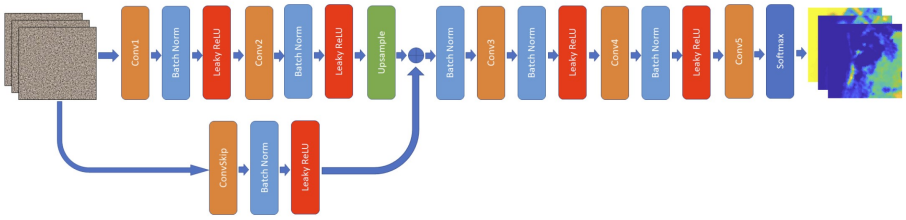

Figure 4: Proposed convolutional network architecture for UnDIP

3.2 Network architecture

The DIP model doesn't explicitely give a network for the hyperspectral unmixing task, but the convolutional encoder-decoder network with some skips was suggested as the best option for DIP and was adapted to work for UnDIP.

Unlike DIP, UnDIP only use one downsampling and upsampling block. This allow to preserve the spatial resolution and improve the convergence rate. The forward pass is composed of 2 parts : one before the addition of the information carried by the skip connections and one after.
Both parts are composed of blocks of 3 layers: a convolution layer (Conv) to learn the spatial features in the image, a batch normalization (BN) layer to speeds up the learning process improve robustness in terms of the hyperparameter selection, and a Leaky ReLU nonlinear activation layer. To solve the vanishing gradient issue and remind the network of what it was trying to learn initialy the UnDIP network use a skip connection which was popularized by resnet.

As said in the paper, Leaky ReLU was used as the activation function (except in the last layer), to speed up the learning process since the derivative is either one or close to zero. Reflection padding was used in the convolution to preserve the size of the image.

HYPERPARAMETERS USED IN THE EXPERIMENTS FOR UNDIP.

Hyperparameters				
	Input Ch.	Ouput Ch.	Filter Size	Stride
Conv1	r	256	3x3	2
Conv2	256	256	3x3	1
Conv3	260	256	3x3	1
Conv4	256	256	1x1	1
Conv5	256	r	1x1	1
ConvSkip	r	4	1x1	1
	Negative Slope			
Leaky ReLU	0.1			
	Scale Factor		Mode	
Upsample	2		Bilinear	
	Type	Learning Rate		Iterations
Optimizer	Adam	0.001		3000

4 Reproduction of the Results

We first implemented the UnDIP algorithm as presented in the paper. The implementation was made using PyTorch. The first model implemented is the one presented in the previous section, with the same parameters as used in the paper. The optimizer is Adam with a learning rate of 0.001. The algorithm was run for 3000 epochs.

The code of this section can be found under the Results and Noise notebook.

4.1 Test images

We tested our implementation for 3 images also used in the paper:

4.1.1 Samson dataset

The Samson dataset is a 95x95 pixels hyperspetral image. There are 3 endmembers corresponding to Soil, Tree and Water. The spectral signature contains 156 bands.

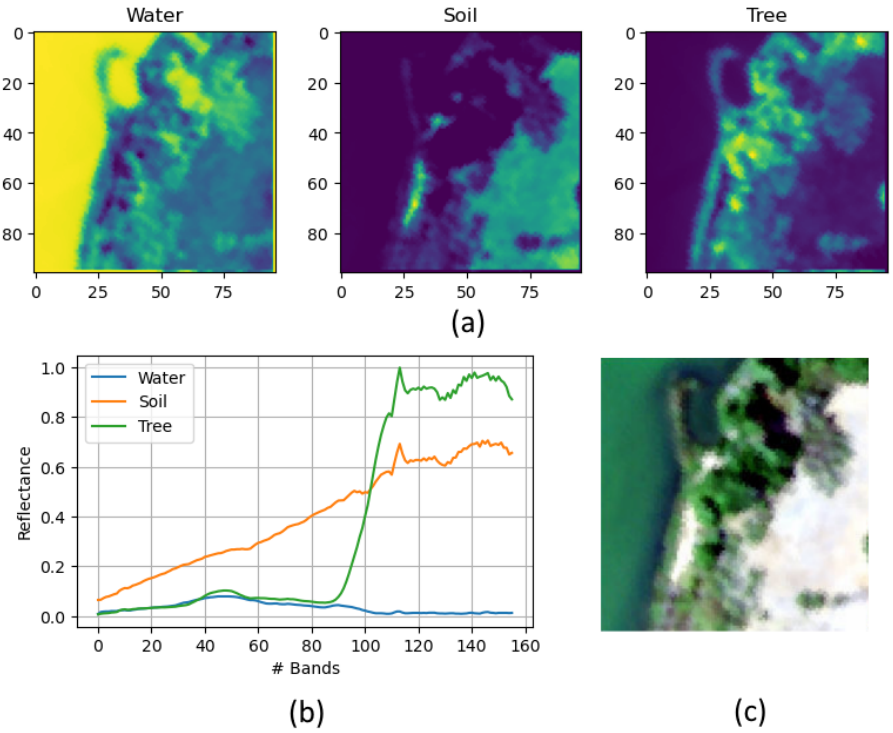

Figure 5: Samson image: (a): Abundances, (b): Endmembers, (c): True-color image

4.1.2 Jasper dataset

The Jasper Ridge dataset is a 100x100 pixels hyperspectral image with 198 channels and 4 endmembers. They correspond to Road, Soil, Water and Tree.

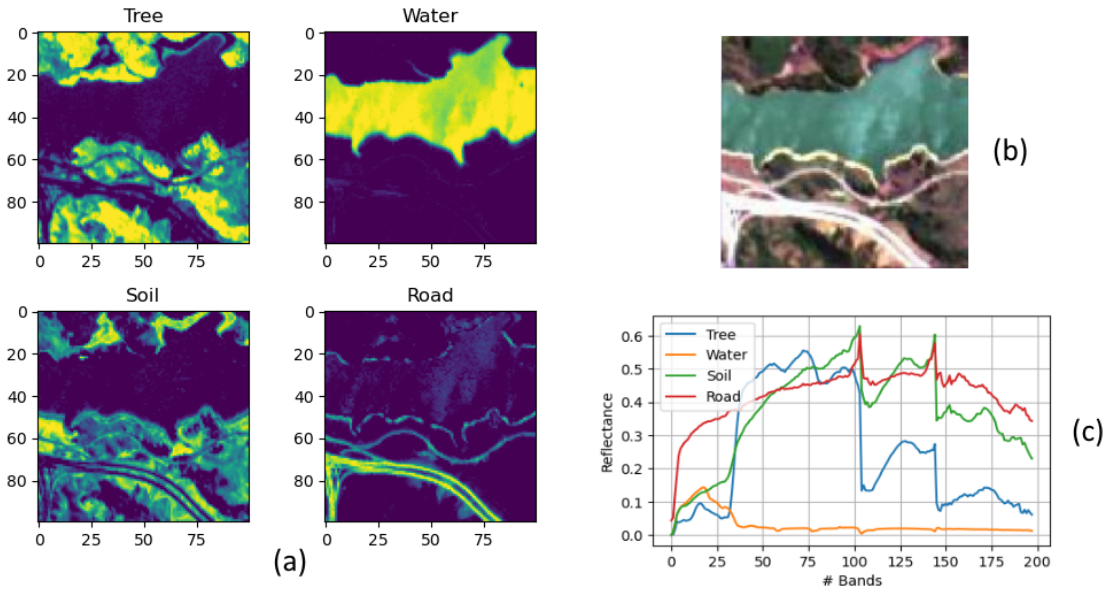

Figure 6: Jasper image: (a): Abundances, (b): True-color image, (c): Endmembers

4.1.3 Urban dataset

The Urban dataset is of size 307x307 pixels. It contains 6 endmembers which correspond to Asphalt, Grass, Tree, Roof, Metal and Dirt. The spectral signature contains 162 bands.

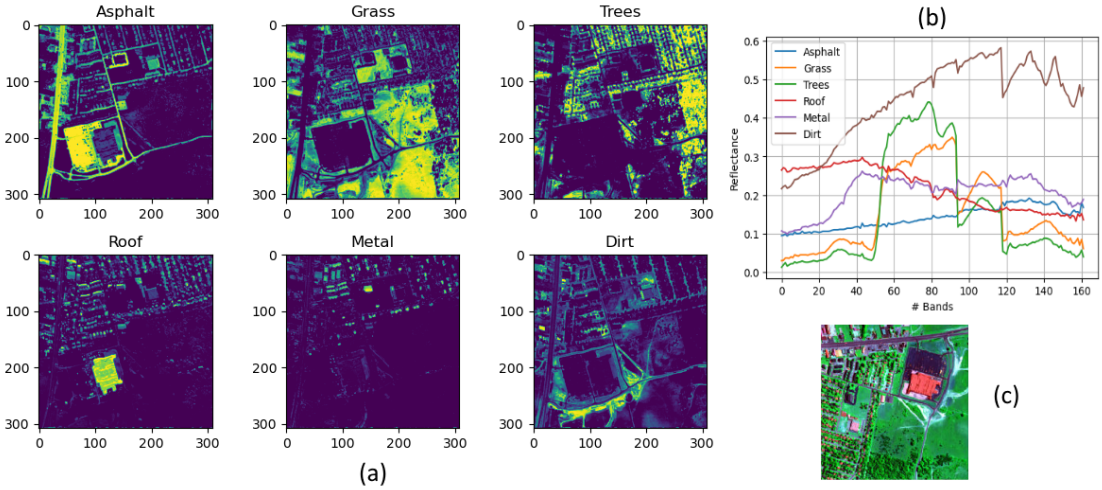

Figure 7: Urban image: (a): Abundances, (b): Endmembers, (c): True-color image

For all of these images, the abundances ground truths were obtained with FCLSU which is the baseline of evaluation. The endmembers ground truths were extracted by Simplex Volume Maximisation (SiVM).

Reflexive padding was added to the Samson and Urban images in order to obtain even dimensions of 96x96 and 308x308 respectively. This was done to keep the coherence of image dimensions before downsampling and after upsampling when passing through the network.

The image $\mathbf{Y}$ was normalized by its l1-norm to have pixel values between 0 and 1. This was done to ensure a similar dynamic between $\mathbf{Y}$ and the ground truths of $\mathbf{EA}$.

4.2 Results

4.2.1 Evaluation Metrics

Results are evaluated with relation to the error when reconstructing $\mathbf{Y}$ and the distance to the ground truth abundances. The two evaluation metrics are the Reconstruction Error (RE) and the Abundances Mean Absolute Error (MAE). The Reconstruction Error is the mean squared error between the reconstructed image from the result of the algorithm, and the observed image $\mathbf{Y}$:

$$RE = \frac{1}{pn} \sum_{j=1}^{p} \sum_{i=1}^{n} ((E\hat{A})_{ji} - Y_{ji})^2$$

The Abundances MAE is the mean absolute difference between the estimated abundances and the estimated abundances:

$$MAE = \frac{1}{rn} \sum_{k=1}^{r} \sum_{i=1}^{n} |\hat{A}_{ki} - A_{ki}|$$

Both metrics are used because the model might overfit (not quite the right term i think) to the imperfection in the observation $\mathbf{Y}$ and have a lower RE with a worse abundance estimation. While the RE is used to train the model, the abundances MAE denotes a better estimation, and a lower RE should only be considered an improvement if the MAE does not increase.

4.2.2 First Results

We first evaluate the algorithm using the same parameters and model as presented in the paper:

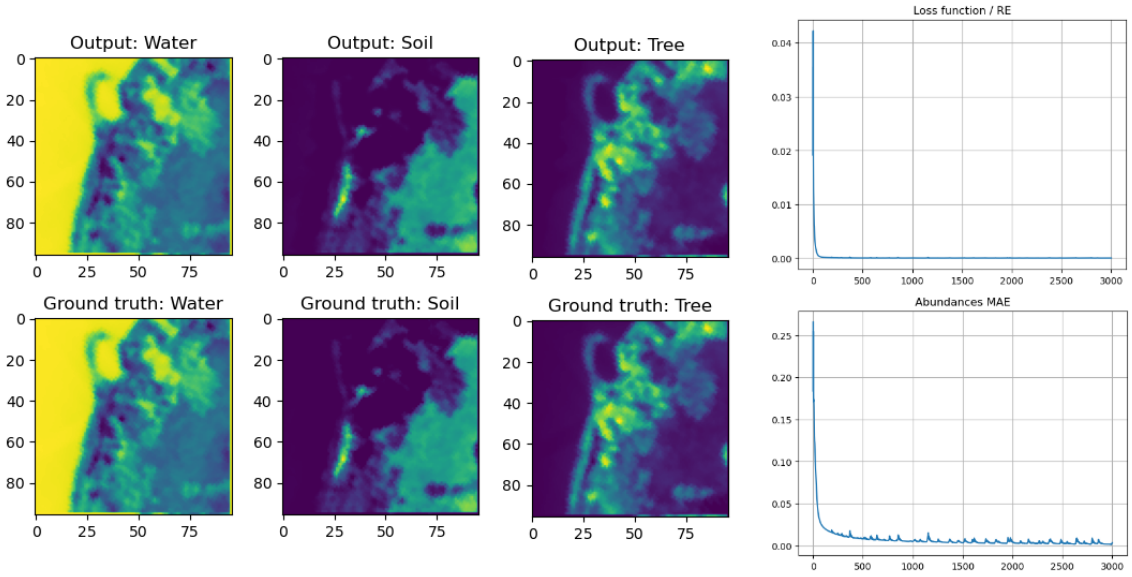

Figure 8: Results of UnDIP on Samson dataset

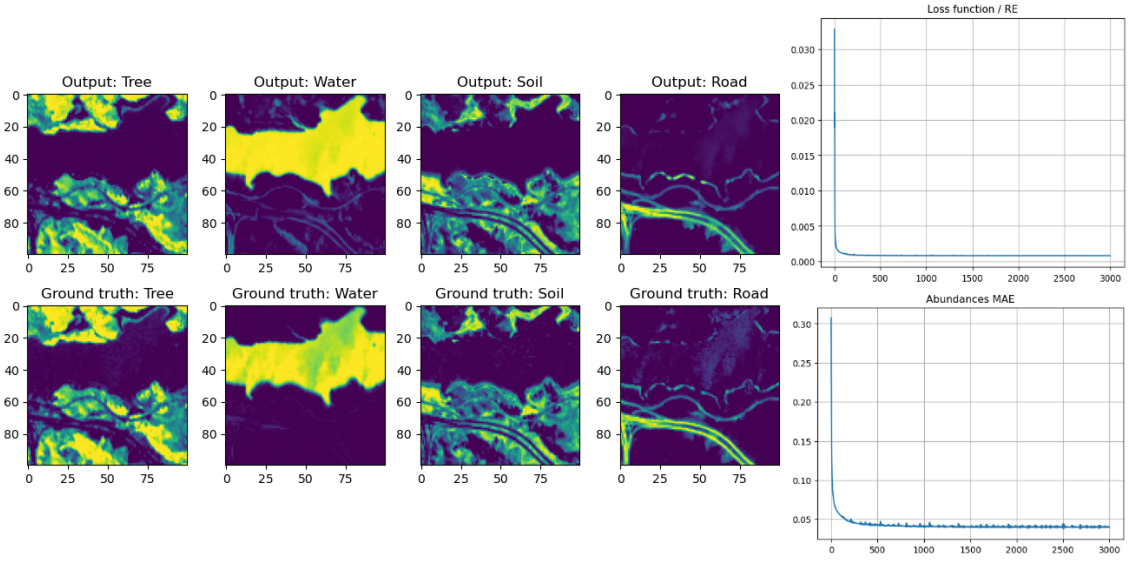

Figure 9: Results of UnDIP on Jasper Ridge dataset

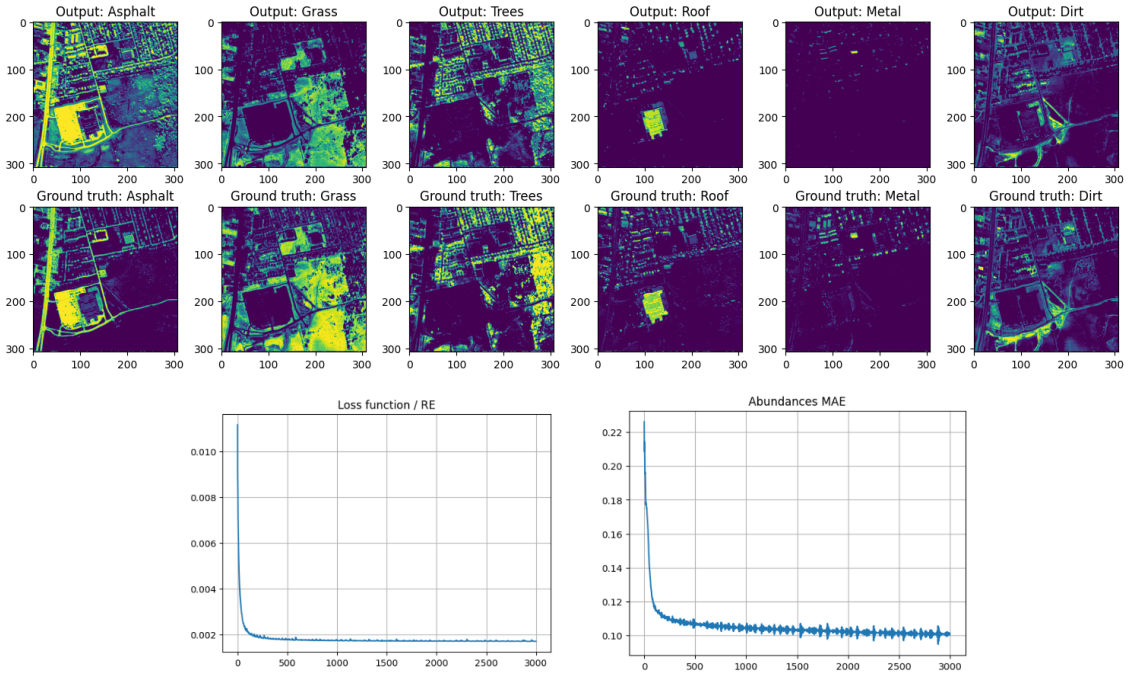

Figure 10: Results of UnDIP on Jasper Ridge dataset

5 Parameters choice

5.0.1 Epoch choice

The code of this section can be found under the Video creation and plots notebook.

As said above the algorithm was run for 3000 epochs. However since UnDIP is an iterative algorithm, the number of iteration is an important hyperparameter to set.

Although having a high enough number of iteration is sure to bring us very close to the minimum solution, it is still possible for the end result to be worst if we run it for just a few more iterations. See fig.

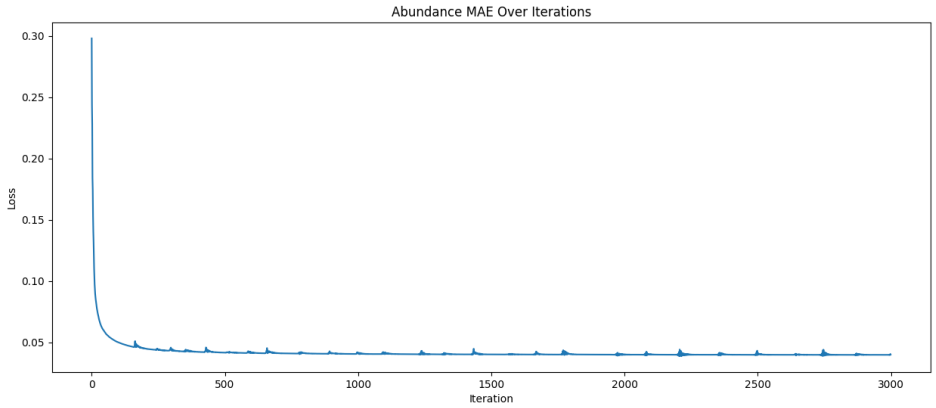

Figure 11: Abundance MAE over iterations, without exponential window for the Jasper Dataset

For example, in the fig above, if the epoch was set to 2500, the abundance map would get noisy

To ensure that the algorithm is robust to this parameter we do an exponential weighted average the outputs. This means that it is necessary to have a higher number of epoch to ensure convergence, but overall average is certain to be close to the minimal solution.

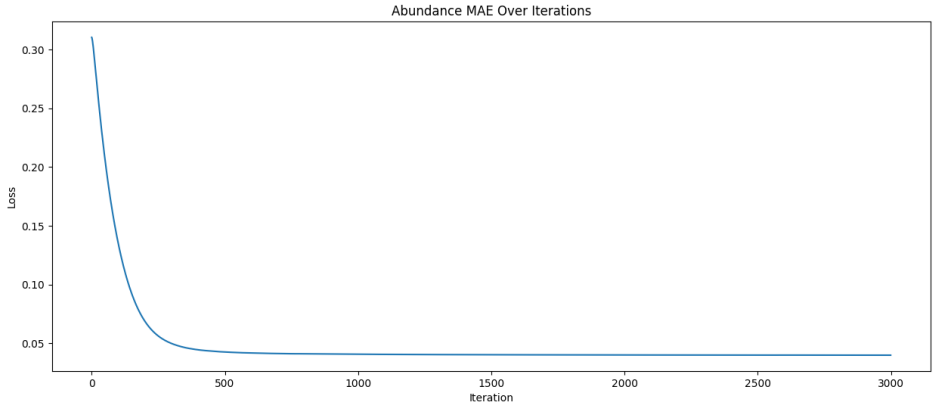

Figure 12: Abundance MAE over iterations, with exponential window for the Jasper Dataset

We can see that the curve is now a lot smoother at the cost of convergence time.

5.0.2 Optimizer and learning rate

We tried different optimizers and learning rate values to see if we can obtain a better or faster convergence. The optimizers and learning rates tested were: Adam with learning rates of 0.01, 0.005, 0.001; NAdam with learning rates of 0.001, 0.0005, 0.0001 and RMSProp with a learning rate of 0.01.
The code of this section can be found under the Results and Noise notebook.

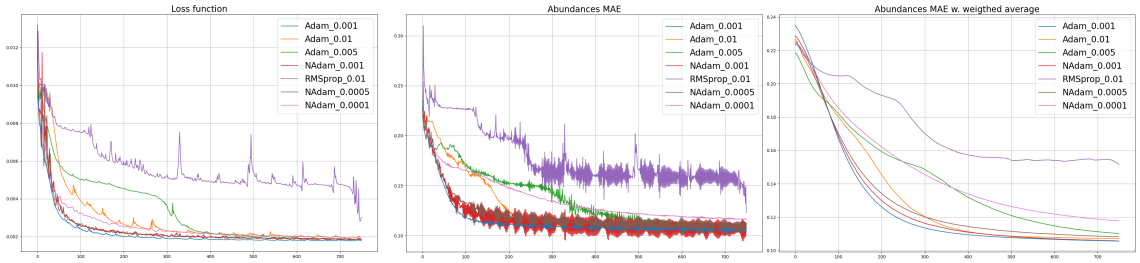

Figure 13: Evolution of metrics for different optimizers during 750 epochs

6 Robustness to Noise

In real-life applications, hyperspectral images often contains non-negligible levels of noise. Furthermore, hyperspectral unmixing methods are robust to noise and can be used as denoisers. The robustness to noise of the method is an important perfomance to evaluate. Besides, because the ground truth was obtained with FCLSU, another similar technique, it is impossible to obtain better results than this baseline. Evaluating the algorithm with relation to its performance on noisy data can lead to better performance comparisons.

We added gaussian noise of mean 0 and different variances to $\mathbf{Y}$ to obtain Signal to Noise Ratio (SNR) values of 20, 30, 40 and 50 dB. With σ the standard deviation of the gaussian noise of mean 0:

$$SNR = \frac{\frac{1}{pn}\sum_{j=1}^{p}\sum_{i=1}^{n} Y_{ji}^2}{\sigma^2}$$

If we write $\tilde{\mathbf{Y}}$ the noise-added $\mathbf{Y}$, we now try to minimize $\| \tilde{\mathbf{Y}} - \mathbf{E}f_\theta(\mathbf{Z}) \|_F^2$ with $\hat{\mathbf{A}} = f_{\hat{\theta}}(\mathbf{Z})$. In addition to the abundances MAE and the reconstruction error RE which is calculated with $\tilde{\mathbf{Y}}$, we can evaluate the robustness to noise of the model using the Spectral MSE, which is the MSE between the estimated image $E\hat{A}$ and the noiseless $\mathbf{Y}$:

$$SpectralMSE = \frac{1}{pn}\sum_{j=1}^{p}\sum_{i=1}^{n}((E\hat{A})_{ji} - Y_{ji})^2 \text{ and } RE = \frac{1}{pn}\sum_{j=1}^{p}\sum_{i=1}^{n}((E\hat{A})_{ji} - \tilde{Y}_{ji})^2$$

6.0.1 Results on noisy data

The code of this section can be found under the Results and Noise notebook.

We first add noise to the previous images:

We can now compare the evolution of the different metrics with relation to the level of noise:

We see that even if adding noise has a strong effect on the Reconstruction Error (or Loss function), the estimated Abundances stay the same even with a SNR of 10dB, for both images. We can compare the final outputs for each SNR values:

This method appears to be very robust to noise, as the abundances are well estimated even with low SNR values. The amount of noise added to $\mathbf{Y}$ does not seem to affect the end result.

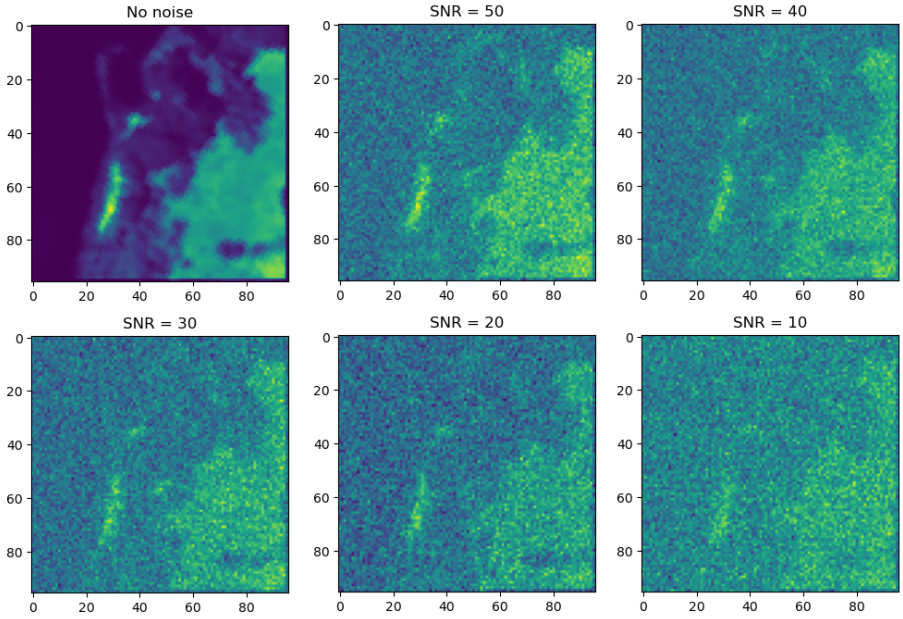

Figure 14: Noise of different SNR on Samson dataset (one slice of Y)

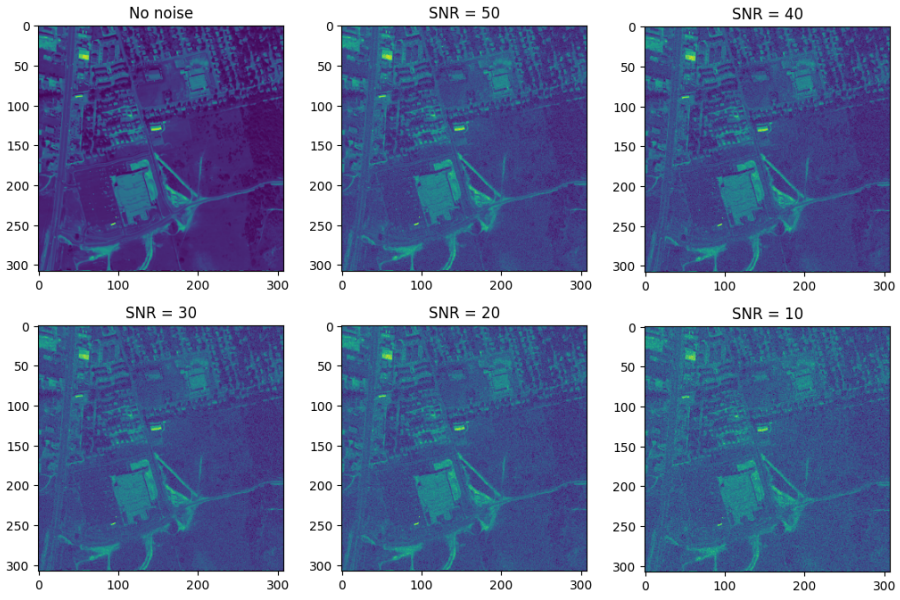

Figure 15: Noise of different SNR on Urban dataset (one slice of Y)

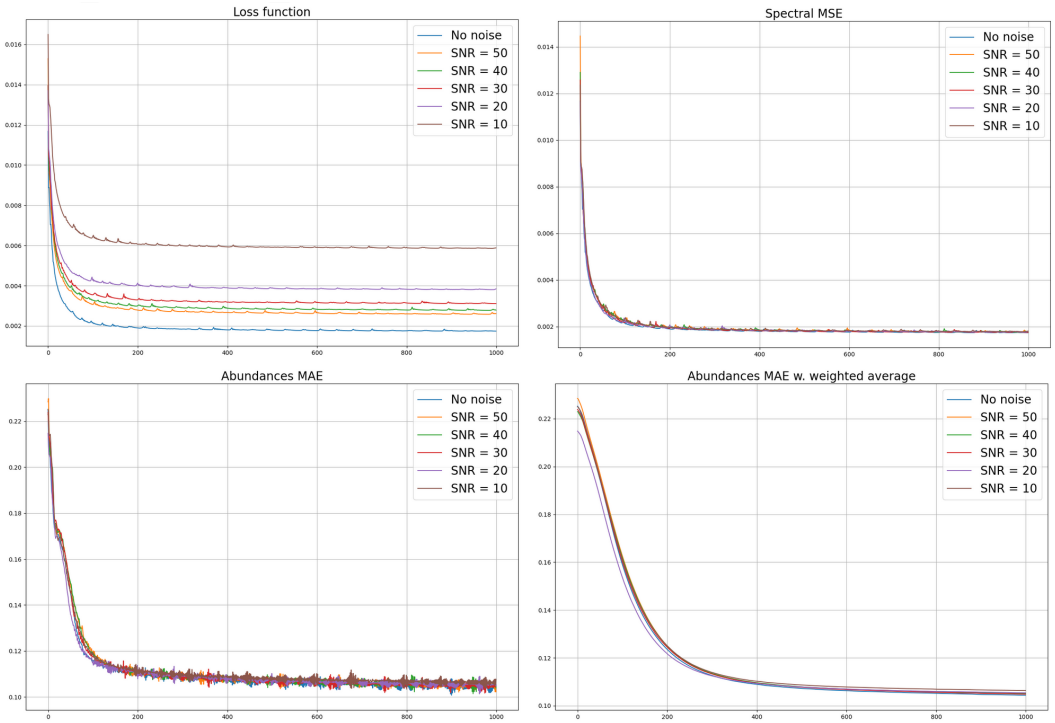

Figure 16: Evolution of metrics on Urban dataset with 1000 epochs

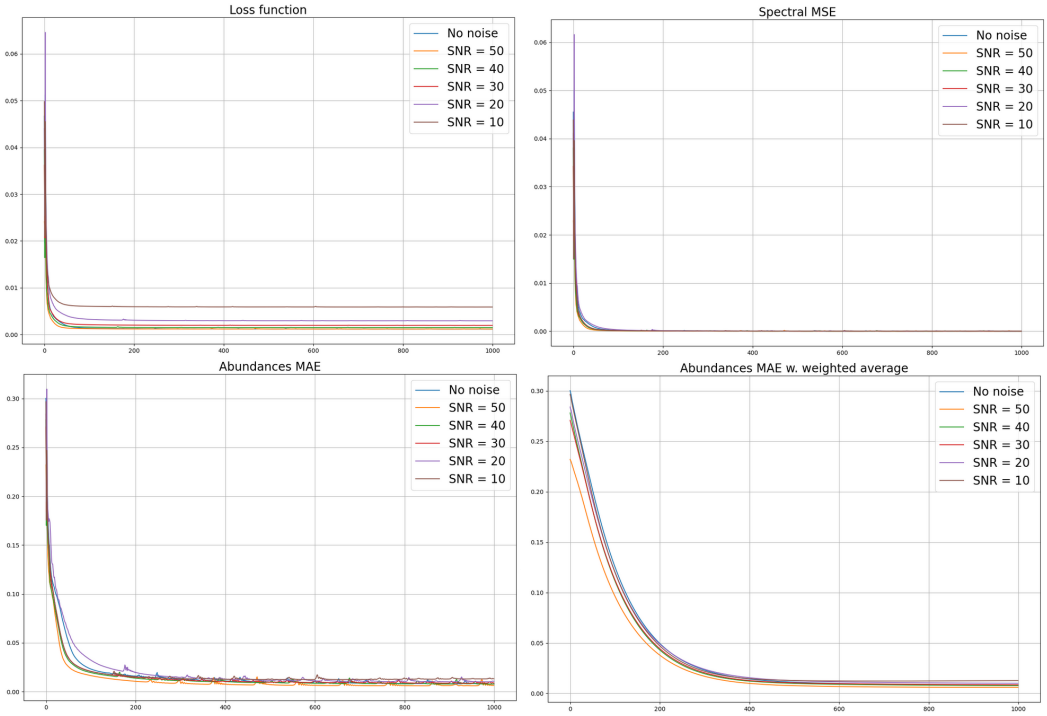

Figure 17: Evolution of metrics on Samson dataset with 1000 epochs

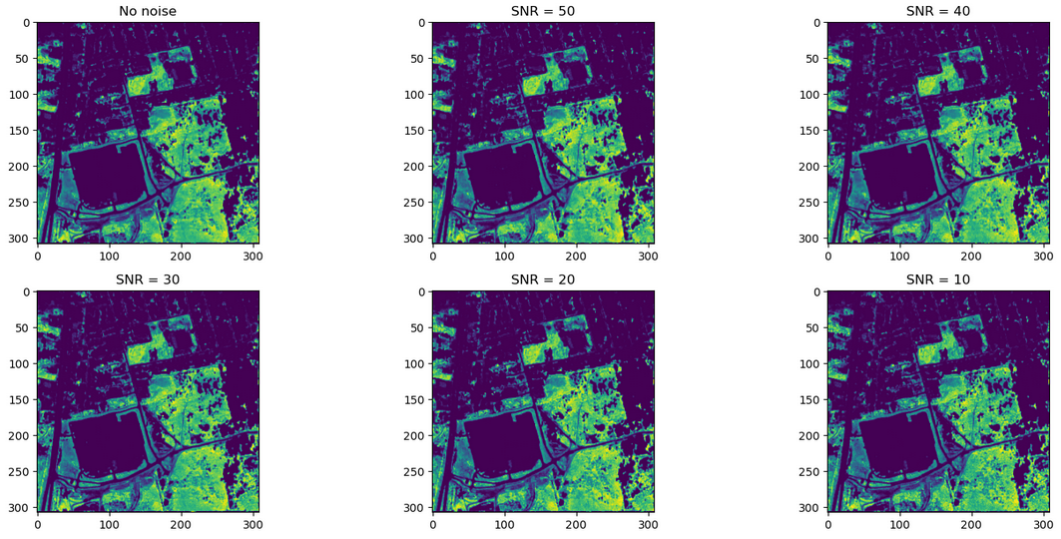

Figure 18: Grass abundance of Urban dataset as estimated from different SNR values

7 Model Structure

*The code of this section can be found under the*Configurable model notebook .

The model structure, or architecture, of a neural network influences the results it can achieve. The depth and width of the architecture also impact performance. Deeper architectures can capture hierarchical representations and learn more abstract features, while wider architectures provide more capacity for modeling complex relationships. However, both deeper and wider networks require more computational resources and need careful regularization to avoid overfitting. Hence, a *configurable architecture* has been designed to study the effectiveness of different architectures in terms of convergence speed and training time.

In the following, we shortly discuss the design of this configurable architecture, which slightly diverges from the one suggested in the original paper, and subsequently, we discuss the observations made by experimenting a few different structures.

7.1 Configurable Model

The configurable model follows an overall U-Net [RFB15] structure: The U-Net architecture is a widely used neural network structure for semantic segmentation tasks, commonly applied in medical image analysis and computer vision. It is called ”U-Net” due to its U-shaped architecture, characterized by an encoder-decoder design.

The U-Net consists of two main parts: the contracting path (encoder) and the expansive path (decoder). The contracting path gradually reduces the spatial dimensions of the input data while capturing high-level features, and the expansive path then recovers the spatial information and generates the output. The contracting path typically consists of multiple convolutional layers with a downsampling operation, such as max pooling or strided convolutions. This process allows the network to capture increasingly abstract representations by hierarchically aggregating information from the input.

The expansive path, which is the decoder part, aims to recover the spatial resolution and generate fine-

grained segmentation outputs. It consists of upsampling operations, such as transposed convolutions or bilinear interpolation, combined with skip connections. These skip connections link corresponding encoder features to the decoder, enabling the integration of both low-level and high-level features at different scales.

We follow these specificities in the design:

1. Down-sampling layers are learnable convolutions with a stride of 2.
2. Up-sampling layers are transposed convolutions.
3. The observed HSI is padded in the preprocessing, so that its height and width are divisible by 8. This latter is done in order to unify the size of kernels and strides in transposed convolutions across different datasets.

In our design, certain parameters of the model can be modified:

1. **Number of encoder and decoder layers**: One can modify the number of encoder and decoder layers. The design of the model is only valid when the number of encoder and decoder layers is equal, but they have been considered separately for the reason of clarity. Increasing the number of encoder layers in a U-Net architecture allows for capturing more abstract and high-level features from the input data. Similarly, increasing the number of decoder layers helps in recovering finer spatial details and generating more accurate segmentation outputs. However, adding more layers to both the encoder and decoder parts increases the model's complexity and computational requirements.

2. **Layers with skip connections**: One can decide the layers on which to add skip connections. As mentioned above, adding skip connections allows for the integration of both low-level and high-level features at different scales.

3. **Number of convolution layers on each skip connection**: One can add or modify the number of convolution layers in each of the skip connections. Adding convolutions allows for further feature transformation and adaptation within the skip connections themselves. This can help refine and enhance the information transmitted through the skip connections, potentially improving the integration of low-level and high-level features.

4. **Limit the number of down-scaling layers**: One can limit the number of times the input gets down-scaled (and up-scaled similarly).

7.2 Results: Comparing Different Configurations

Six distinct architectures were compared (for three different datasets):

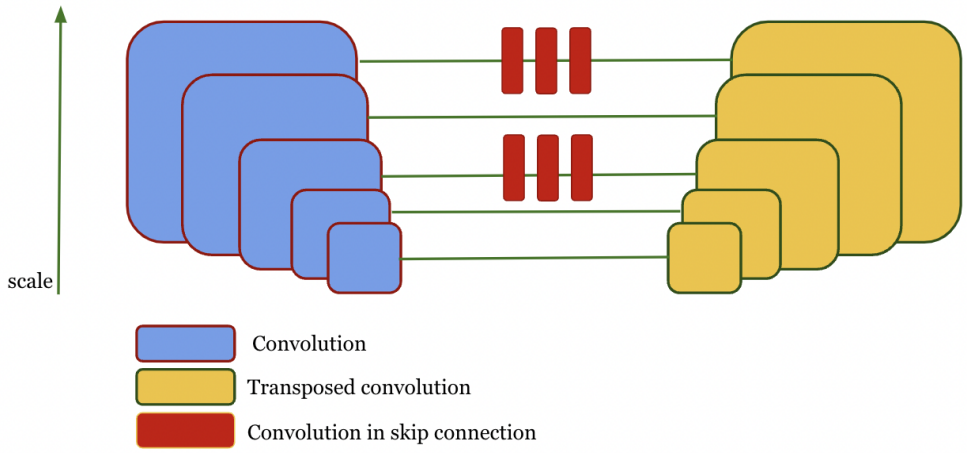

Figure 19: A schema of the configurable model. The number of encoder/decoder layers and skip connections can be modified.

1. Only two encoder/decoder layers, with no skip connection.
2. Two encoder/decoder layers, with a direct (no convolution) skip connection.
3. Two encoder/decoder layers, with a convolutional skip connection.
4. Four encoder/decoder layers, with direct skip connections at all the layers.
5. Four encoder/decoder layers, with convolutional (one convolution layer) skip connections at all the layers.
6. Four encoder/decoder layers, with convolutional (two convolution layers) skip connections at all the layers.

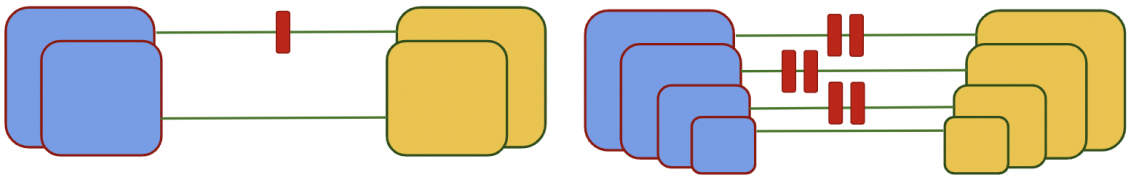

Figure 20: Two examples of experimental architectures, corresponding to the third and sixth architectures respectively from left to right.

Here are general observations made during these experiments:

1. Convergence becomes slightly slower with the number of learnable parameters increased. However, the converged result can be better.

2. Training time is increased with the number of learnable parameters increased.

3. Adding skip connections contributes to the final score and also facilitates training and convergence.

In the following, structures are compared quantitatively, by their abundance MAE and training loss evolution, and qualitatively, by their final fractional abundance maps.

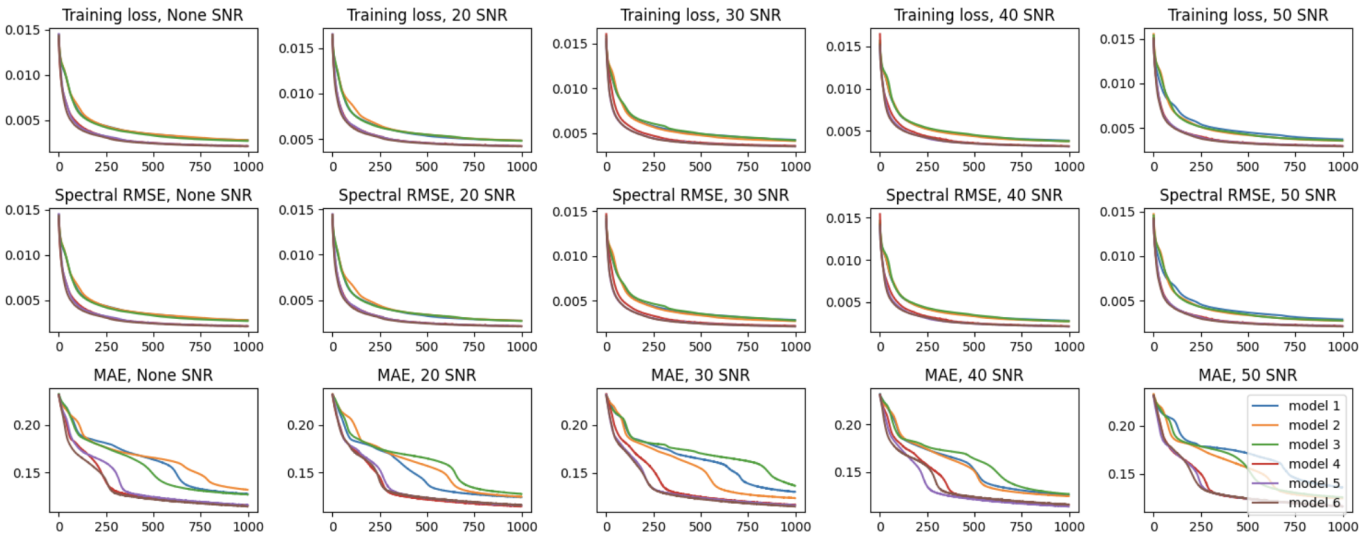

Figure 21: Model structure : Comparing models at different levels of noise. Experiment on Urban image: different models have been tested with different levels of noise on the observed HSI, in order to study the robustness of different models and their convergence speeds. The figure illustrates three different metrics, at five different levels of noise, for six models. Findings: when increasing the level of the noise, the difference between simple models (only two encoder/decoders) and more complicated models (four encoder/decoders and more convolution in becomes more visible. Produced by the code at Configurable model notebook

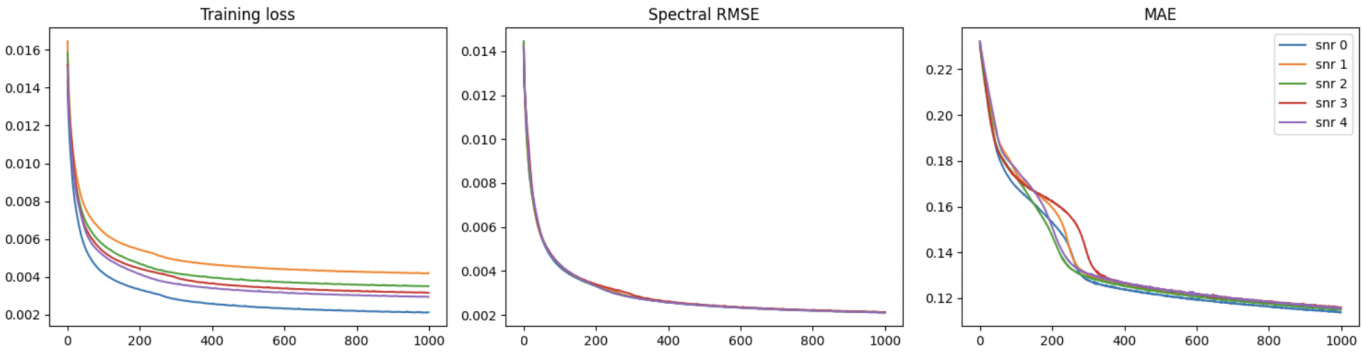

Figure 22: Model structure : Studying the effect of noise on the convergence of a single model. Experiment on Urban image: We chose the most complicated model considered above (the sixth one), and performed training at different levels of noise on the observed HSI. The figure illustrates three different metrics for the same model, at different levels of noise. SNR 0 corresponds to no noise, and the consequent levels are respectively 20,30,40, and 50. Findings: The training loss is more affected by the added noise, and the obtained fractional abundances map is less sensible to this latter. Produced by the code at Configurable model notebook

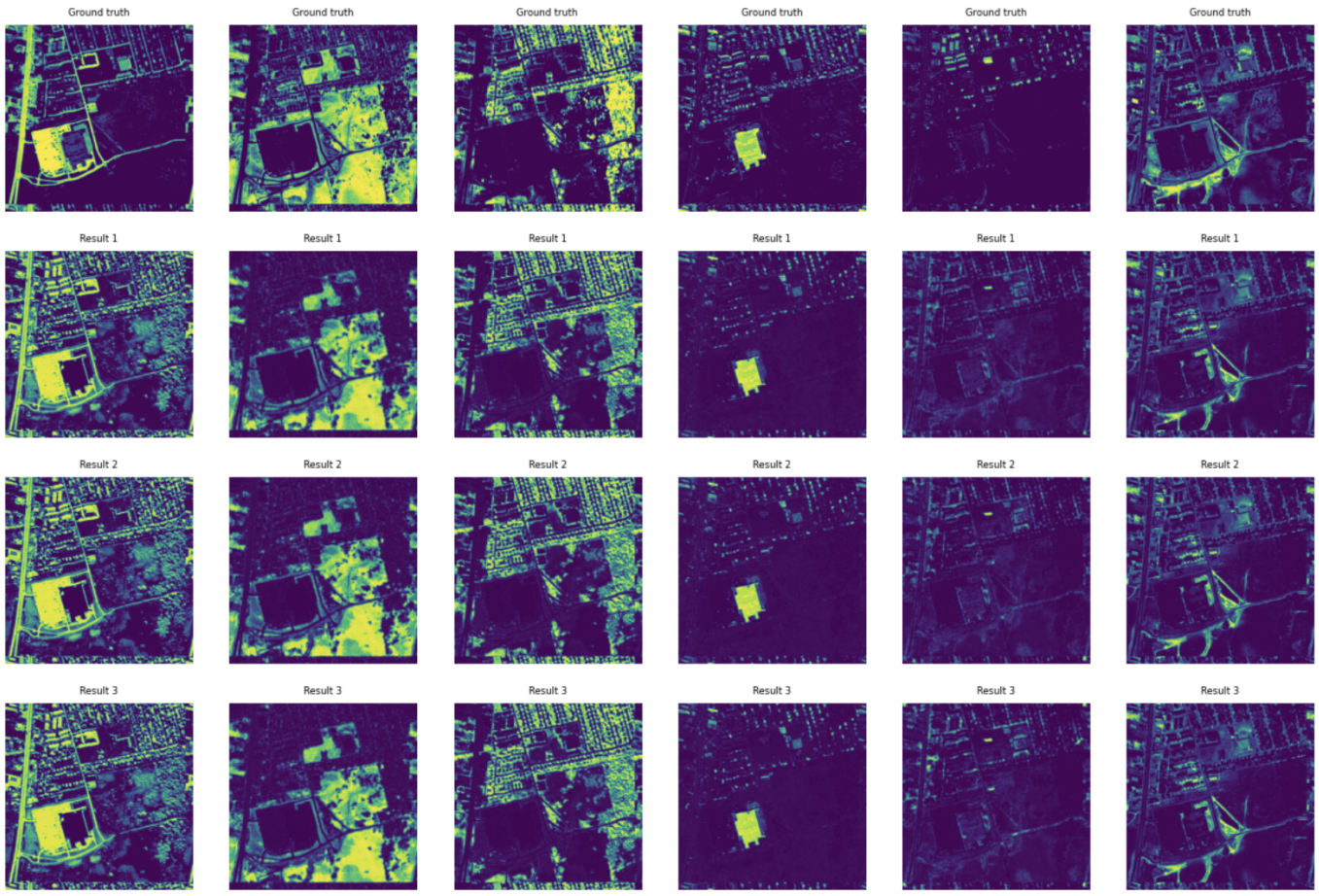

Figure 23: Model structure : Comparing the fourth, fifth, and sixth models qualitatively, on Urban image. The hyperparameters remain the same as in other experiments (learning rate and number of epochs). Produced by the code at Configurable model notebook

8 Initialisations

In order to explore the impact of different initializations on the performance of the neural network, three different initialization methods were evaluated: random normal initialization, initialization with the PALM algorithm, and initialization with non-negative matrix factorization (NMF) unmixing.

8.1 Random initialisation

This method initializes the weights and biases of the neural network using random values drawn from a normal distribution. The purpose of this initialization is to introduce diversity in the initial parameters, allowing the network to explore different regions of the parameter space during training.

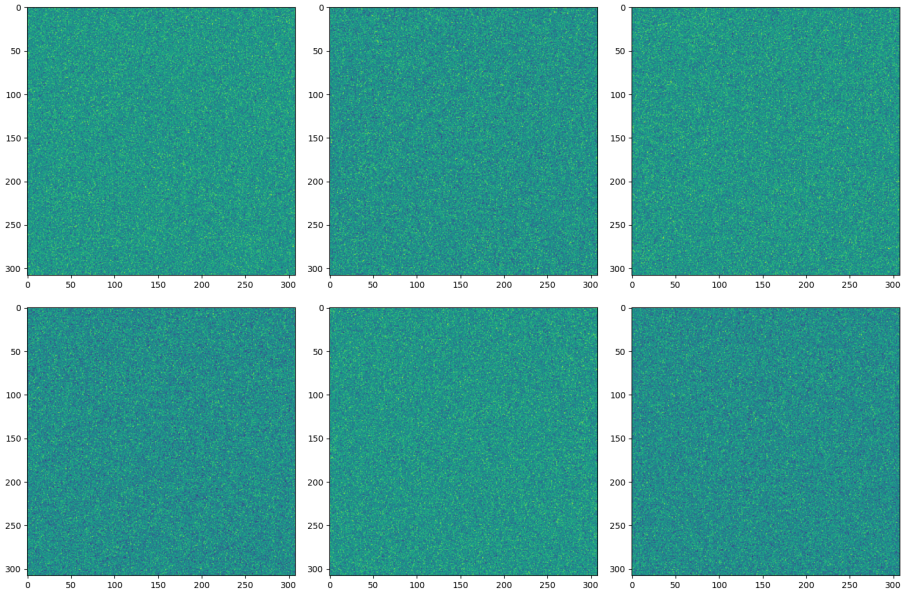

Figure 24: Random initialisation of the input

8.2 PALM initialisation

The PALM algorithm stands for Proximal Alternating Linearized Minimization and is a popular method for solving sparse unmixing problems. In this initialization approach, the PALM algorithm is applied to estimate the sparse abundances of the hyperspectral data. The resulting sparse abundance maps are then used to initialize the corresponding layers of the neural network.

8.3 NMF initialisation

Non-negative Matrix Factorization (NMF) is a widely used technique for spectral unmixing in hyperspectral imaging. NMF decomposes a hyperspectral image into a set of non-negative spectral endmembers and their corresponding abundance fractions. In this initialization method, NMF unmixing is performed on the hyperspectral data to obtain the initial estimates of the spectral endmembers and abundance fractions, which are used as the initial parameters for the neural network.

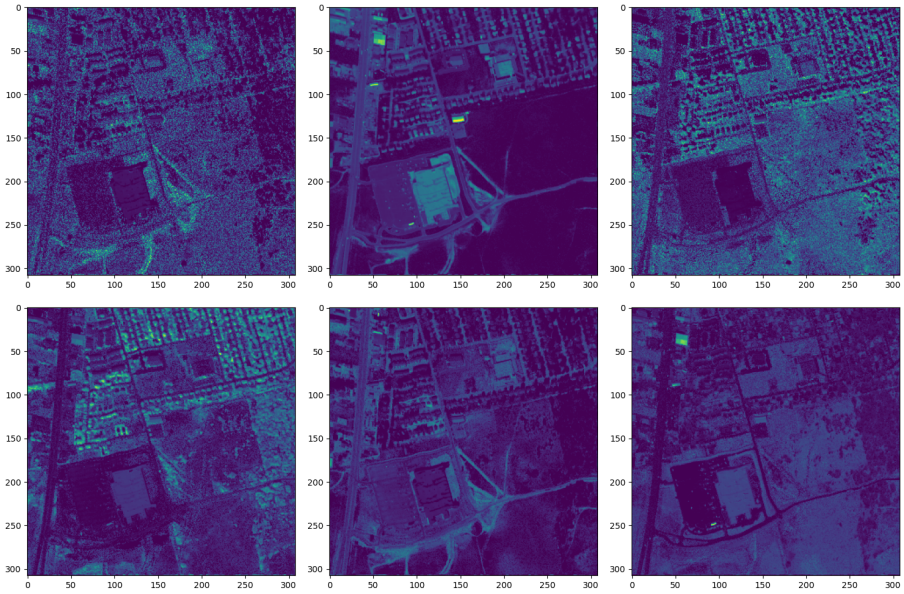

Figure 25: PALM algorithm initialisation of the input

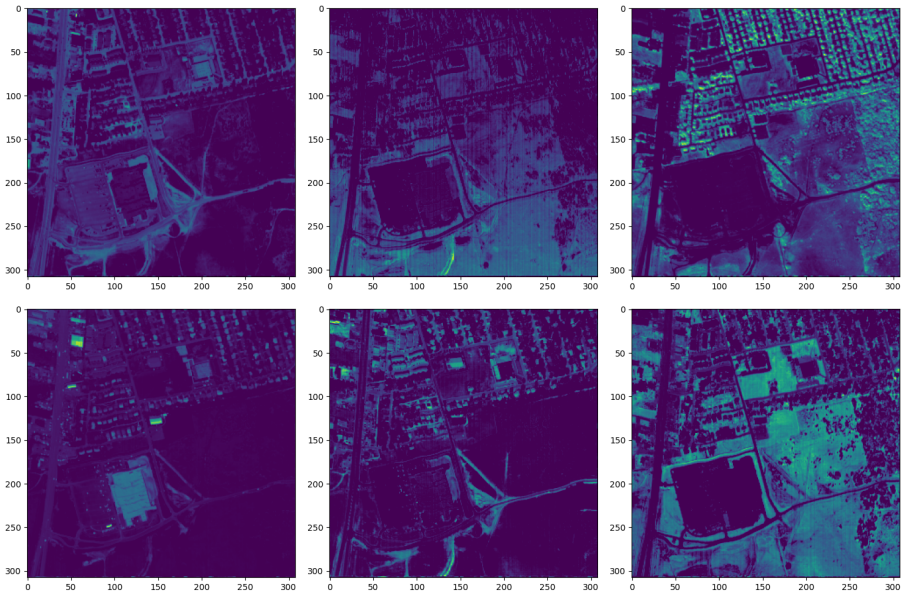

Figure 26: NMF algorithm initialisation of the input

8.4 Results

8.4.1 Quantitative results

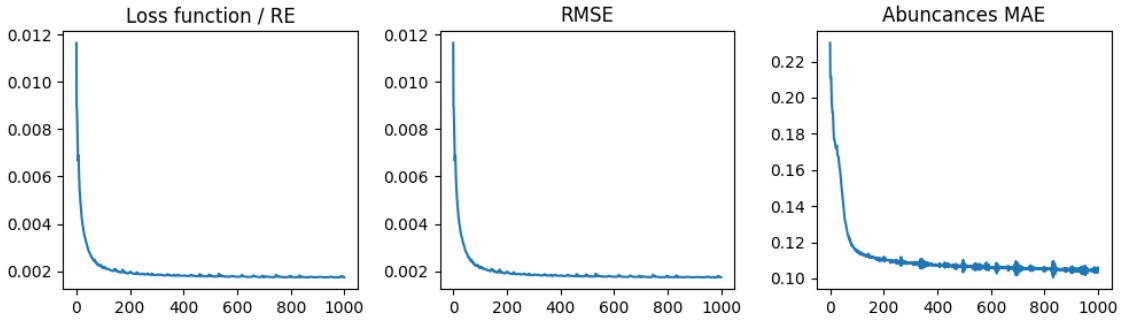

Figure 27: Loss plots with the random initialisation

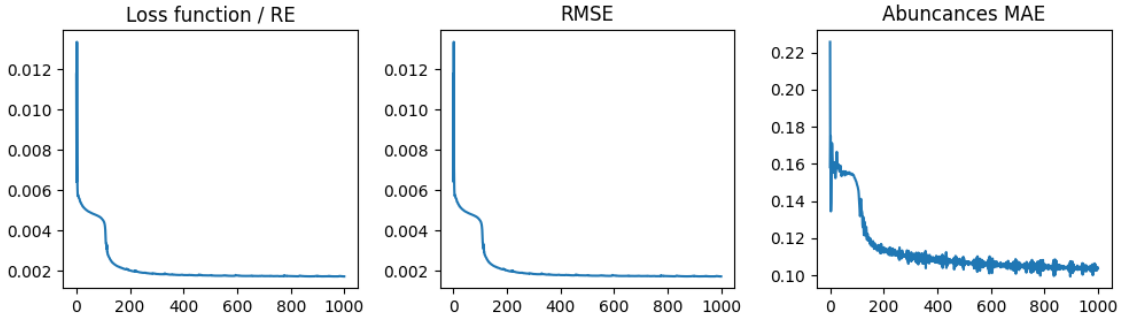

Figure 28: Loss plots with the PALM algorithm initialisation

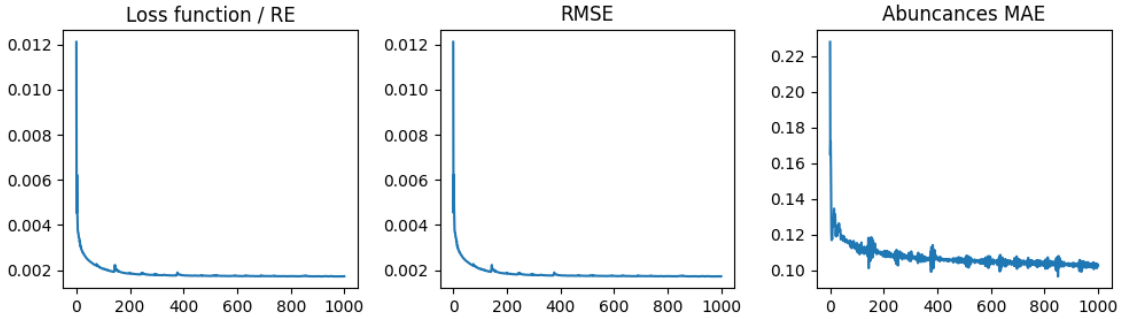

Figure 29: Loss plots with the NMF algorithm initialisation

8.4.2 Qualitative results

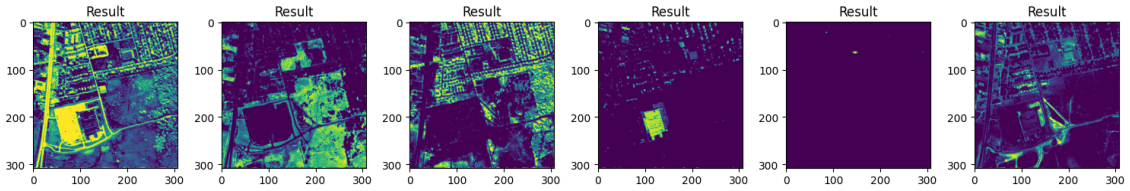

Figure 30: Result with the random initialisation

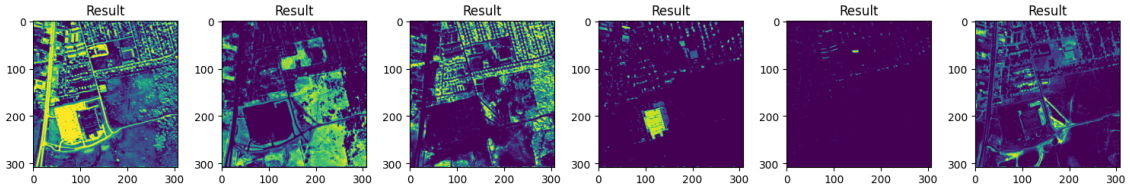

Figure 31: Result with the PALM algorithm initialisation

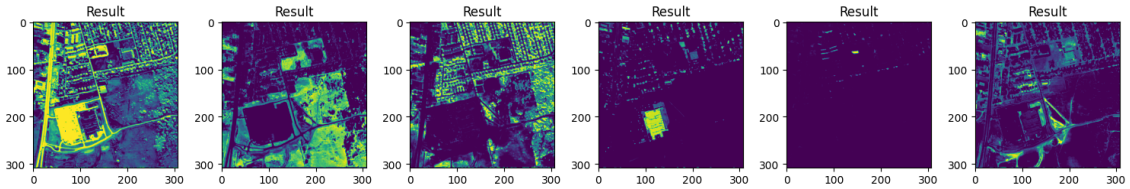

Figure 32: Result with the NMF algorithm initialisation

8.4.3 Conclusion on results

By trying a model initialised with the output of the PALM algorithm and NMF algorithm, we expected a more efficient model and a more rapid convergence which was the case with the NMF initialisation but not the case with the PALM initialisation that showed an unexpected curve in the first epochs.

Qualitatively, these different initilisation do not show a big difference in the output, except for the fourth image.

The quantitative and qualitative results show that the initialisation of the neural network does not change much in the output. Hence, adding them to our pipeline is not worth their execution time.

9 Conclusion

In this report, we managed to explore and explain the technical components of the deep prior unmixing technique UnDIP. Starting from the inspiration, we detailed the architecture of the deep learning model in order to reproduce the results of the paper which we succeeded in.
We took a step further by experimenting with different hyperparameters such as the network architecture and components, the number of epochs, different initialisation of the input and different levels of noise. These experiments were done separately in order to have a deeper understanding of the model and its strong and weak points.

We observed, by comparing different model structures, that even a simple architecture permits obtaining a satisfying result. however, it may affect the convergence speed and also robustness to noise. Overall, we conclude that increasing the number of encoder/decoder layers may accelerate the convergence and also robustness of noise.
A higher number of epochs is sure to improve the result as long as an exponential weighted average is employed. Indeed, it is pretty much mandatory since it minimizes the impact of the choice of the number of epochs on the end result. The best optimizer and learning rate was the one proposed in the paper.
We were able to conclude that this technique is highly robust to noise, easy to implement, and independent of the initialisation.

References

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer. 2015, pp. 234–241.

[UVL18] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. “Deep image prior”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9446–9454.