PRIM Project : Self-Supervised Training for Hyperspectral unmixing

Supervisor

Christophe Kervazo

Team

Maksim KOCHANOV, Viktoriya ZHUKOVA

Referred articles

1) Palsson, B., Ulfarsson, M. O., & Sveinsson, J. R. (2020). Convolutional autoencoder for spectral–spatial hyperspectral unmixing.
2) Pierre-Antoine Thouvenin, Nicolas Dobigeon, Jean-Yves Tourneret. Hyperspectral Unmixing With Spectral Variability Using a Perturbed Linear Mixing Model.

Key notions

Hyperspectral unmixing (HSU), generative neural network (GNN), spectral-spatial model

Introduction

Hyperspectral imaging is a specific type of image data acquisition that uses several wavelength sensors (from few to hundreds) to extract qualitative information from each pixel of the seen distant object. The main application frame assumes the picture being composed out of a fixed number of materials (called endmembers or sources), each one characterized by a unique specter, defined on the channel range. The final objective is to describe the observed picture through the proportions (called abundances) of every material within every pixel. This process of transforming the rough observation X into endmembers matrix A and abundance matrix S is called unmixing.

As in most image processing tasks, two approaches are available : algorithmic one and network one, and each corresponds to a conceptual viewpoint on the problem. Generally, man can mathematically describe the situation and make assumptions about the matrices X, A, S in order to solve X = AS as precisely as possible. We can resolve it with for example the PALM algorithm. On the other hand, man can see unmixing as an application of the information compression task. Indeed, the goal is to simplify the pixel description from hundreds of channels to a few aboundances. And this corresponds to the compact data representation of autoencoder latent space.

All in all our study pipeline can be described with two parts:

<u>Part 1. Perturbations approach</u>

1. **Getting the first estimation**: Denoting X the hyperspectral image to unmix, the first step consists in applying an unsupervised unmixing algorithm to provide a first estimate of $A*$ and $S*$. In this study we use a neural approach with Autoencoder.
2. **Endmember perturbations**: In this stage we compared two different approaches:
    a. A Random perturbation function is applied N times to each endmember in $A*$ to generate a realistic data-augmentation of the extracted endmembers.
    b. A pixel-wise perturbation in the generated training images to better mimic real-world hyperspectral images.
3. **Autoencoder Training on Corresponding Data.**

Part 2. Variational Autoencoder (VAE) approach

1. **Constructing a Training Dataset for a VAE**: By utilizing the original image, we can obtain approximations of endmembers as a result of the Vertex Component Analysis (VCA) algorithm's operation.

2. **Generation of endmembers**: Having obtained several examples of endmember approximations, we train the VAE. The subsequently generated endmembers via VAE can also be considered somewhat perturbed, as the training dataset consisted of slightly differing approximations of the same endmembers.

3. **Autoencoder Training on Corresponding Data**: Abundances were obtained using Fully Constrained Least Squares (FCLS), followed by the acquisition of the hyperspectral image using a linear model.

As a model for training in all cases we use the Autoencoder model described in the next section. The loss functions are different and precise for each subsection.

In this study we use two of the most widely used hyperspectral data - the Urban and Samson datasets. In the Urban dataset there are 307 x 307 pixels, each of which corresponds to a 2 x 2 m2 area. There are 210 wavelengths ranging from 400 nm to 2500 nm, resulting in a spectral resolution of 10 nm. After the channels 1--4, 76, 87, 101--111, 136--153 and 198--210 are removed (due to dense water vapor and atmospheric effects), we obtain 162 channels. The one we will use contains 6 endmembers (i.e. sources): Grass, Roads, Roofs, Trees, Buildings, Ground.

In the Samson dataset each pixel is recorded at 156 channels covering the wavelengths from 401 nm to 889 nm. The spectral resolution is up to 3.13 nm. A region of 95 x 95 pixels is used. It starts from the (252,332)-th pixel in the original image. This data is not degraded by the blank channel or badly noised channels. Specifically, there are three targets in this image, i.e. Soil, Tree and Water respectively.

Patch technique

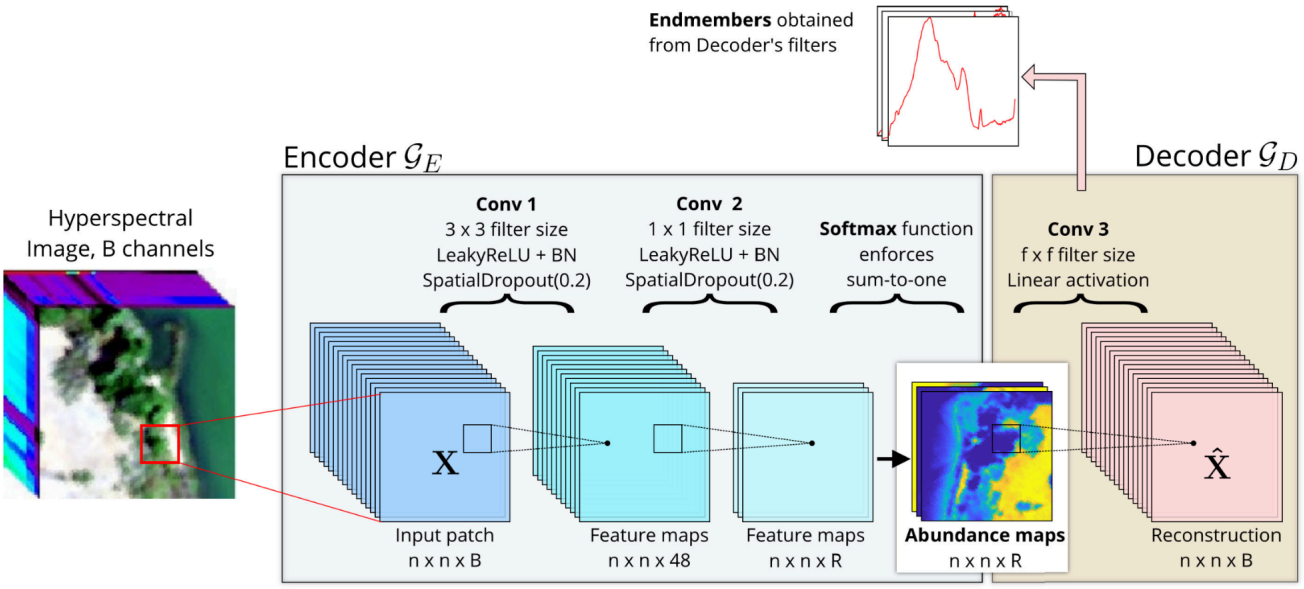
For each training model, we employed two approaches. In the first one, we presented the whole images for the model. In the second, we partitioned the original image into patches of size 40x40x$N_{channels}$ and fed them to the model for training.

Patches are randomly selected using a mask applied to the matrices X, A, S to preserve the pixel sequence within the patches.

We will show next that the approach with patches gives us impressive improvements in all proposed training models.

Perturbations approach

First Estimation. Autoencoder



Encoder

The encoder consists of 2 convolutional layers that are responsible for extracting features from the input data. To introduce non-linearity into the model, LeakyReLu activation functions are employed. Additionally, Batch Normalization (BN) and Spatial Dropout techniques are utilized.

Batch Normalization is a method used to normalize the inputs of a neural network layer, specifically by normalizing the mean and variance of the inputs over a mini-batch of training examples. This helps in stabilizing and accelerating the training process by reducing internal covariate shift, making the optimization process more efficient.

Spatial Dropout is a variation of the traditional Dropout technique that is specifically designed for convolutional neural networks. Dropout randomly sets a fraction of the input units to zero during training, which helps in preventing overfitting and promoting better generalization of the model. Spatial Dropout extends this concept to the spatial dimensions of the input data, effectively dropping entire channels or feature maps, which can be beneficial in reducing the risk of over-reliance on specific spatial features.

Regarding the latent space, it is claimed to correspond to the abundance map of the observed image. In order to ensure that the abundance coefficients sum up to one, proportion coefficients are introduced. The softmax function is utilized for this purpose. However, the text does not provide any further theoretical justification or explanation for this choice of approach or its desired correspondence. The effectiveness of the encoder is evaluated empirically by comparing its results with those obtained from algorithmic methods, thus validating its role in the overall network architecture.

Decoder

More stuff is done while managing the decoder subnetwork. Within only one convolutional layer, it has to reconstruct the X matrix from the A one. Initially, the convolution mask has a size of (R,B), to transform each pixel source representation back to the hyperspectral one, and its weights directly correspond to the coefficients of the S matrix, because of the X = AS HSU equation. Here comes the extension capacities of the network approach. We can basically initialize those coefficients through different ways, exploring the efficiency and the limitations and the model.

The central idea is to change the size of this convolution mask from (R,1,1,B) to (R,f,f,B) in order to consider spatial information among nearby f x f neighbor pixels, characterizing local dependencies of the image pixels. This is the core study of the referred article.

The number and the initialization of the decoder layer weights are the main hyperparameters we can work with.

Loss: Here we use $\sum_{i=1}^{N}(1 - \frac{X_{pred} \cdot X}{\|X_{pred}\| \cdot \|X\|})$

Learning rate: In contrast to the authors of the article, who used a low learning rate for 320 epochs, we used a high learning rate at the beginning with a gradual decrease in it during the learning process. Thus, we were able to achieve approximately the same results, significantly speeding up the training of the neural network.

First Estimation

Using the patch technique described above we train the autoencoder and obtain a first estimation of A∗ and S∗.

Endmember perturbations

Generate synthetic labeled training samples

In the previous step we found the first estimation of endmembers (A) and abundances (S) matrices by applying an unsupervised method to the original image (X). Now the idea is applying a perturbation function on the first estimation of matrix A to derive several other spectra in order to emulate spectral variability in HSI. Naturally it can be explained with spectrum similarity of analogous sources. In other words we suppose that for example spruce, pine and fir have similar spectrums. Consequently we can obtain many images with spectrum perturbations instead of our single original image. In this case we replace the initial unsupervised problem with the supervised problem on synthetically generated images. To do that we apply a random perturbation function N_{train} times to each endmember in A to generate a realistic data-augmentation of the extracted endmembers (where the Perturbation function is a random function explained in the next subsection).

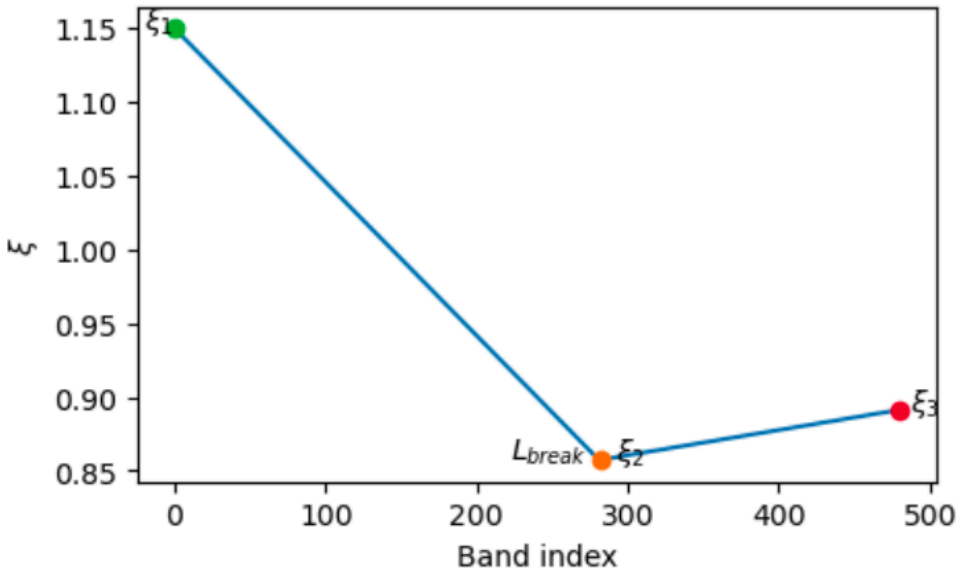
Using this endmember data-augmentation, we can then generate new synthetic data-sets using the Linear Mixture Model:

$$^{(i)}_{\text{pert}}X = {}^{(i)}_{\text{pert}}A^* \ {}_{\text{SNPA}}S^* + {}^{(i)}N, \forall i \in \{1, ..., N_{train}\}$$

where N is generated as an additive gaussian noise matrix. The tuples $\left({}^{(i)}_{\text{pert}}X,\ {}^{(i)}_{\text{pert}}A^*,\ {}_{\text{SNPA}}S^*\right)$ can then be used as labeled datasets for training any model in a supervised way.

Perturbation procedure describing

The Perturbation function enables the controlled generation of spectral variability. The endmember signature at the function's input undergoes multiplication by a randomly generated piece-wise affine function, as illustrated in the example function depicted in the figure.



To elaborate further, when the user selects a certain level of variability c_var, piece-wise affine functions (refer to Figure 1) are created using four parameters:

- $\forall i \in \{1,2,3\}, \xi_i \sim \mathcal{U}_{[1-c_{var}/2, 1+c_{var}/2]}$.
- $L_{break} = \lfloor \lfloor L/2 \rfloor + \lfloor LU/3 \rfloor \rfloor \in \{1, ..., L\}$ is the spectral band marking the non-linearity, $U \sim \mathcal{N}(0,1)$ and $\lfloor . \rfloor$ is the floor function.

Balanced Loss (supervised VS unsupervised)

Now, having *N* tuples of the form *(X, A, S)*, we can transform the loss in our autoencoder model by incorporating the components A, S, which are known to us. In this way, we combine unsupervised and supervised losses in proportions determined by the parameter *alpha* and (1-*alpha*) and then will call it Balanced loss. We will explore the optimal selection of the coefficient in the next sections.

Balanced loss:

$$alpha * \left(\sum_{i=1}^{N}\left(1 - \frac{X_{pred} \cdot X}{\|X_{pred}\| \cdot \|X\|}\right)\right) +$$

$$+ \quad (1 - alpha) * \left(\sum_{i=1}^{N}\left(1 - \frac{A_{pred} \cdot A}{\|A_{pred}\| \cdot \|A\|}\right) + \sum_{i=1}^{N}\left(1 - \frac{S_{pred} \cdot S}{\|S_{pred}\| \cdot \|S\|}\right)\right)$$

It is worth noting that we have transformed the original SAD-Loss proposed in the article, as it involved computing the arccosine, which posed difficulties in handling corner cases. Instead, we employ a cosine-similarity-based function expressed as (*1 - similarity)*.

Pixel-wise endmember perturbations

As spectral variabilities are inherent in real-world datasets, the model tends to overfit to the linear mixture model if training as explained earlier with simple endmember perturbations. To avoid this issue, we introduce an additional step in the simulation process by introducing pixel-wise perturbations in the generated training images, aiming for a more accurate mimic real-world hyperspectral images. Specifically, instead of relying on the LMM for mixture generation, we employ a spectral variability model of the following form:

$$\forall i \in \{1, ..., N_{train}\}, \forall k \in \{1, ..., Nb\}, \forall j \in \{1, ..., p\},$$
$$\overset{(i,k)}{\text{SV}} A_j^* = Perturbation(\overset{(i)}{\text{pert}} A_j^*)$$
$$\overset{(i)}{\text{SV}} X_k = \overset{(i,k)}{\text{SV}} A^* \ \text{SNPA} S_k^* + {}^{(i)} N_k$$

Precisely, instead of applying a perturbation function to all spectrum in the whole image, we will apply a perturbation function to all spectrum in each pixel of the image. It means that now we have different matrices A for each pixel.

It's true that now we have much more parameters than earlier and it seriously impacts computations performance but we expect better results.

As loss function we use the balanced loss explained in the previous section but with one remark: a matrix for i-th endmember determines as the mean between all image pixels spectrums: $\frac{1}{N}\Sigma_{k=1}^{Nb} (i,k)\, A^{(i)}{}_k$.

Metrics

Endmembers cosine similarity: $\frac{<A,\, A_{pred}>}{||A| \cdot ||A_{pred}||}$,

Endmembers MSE: $||A - A_{pred}||^2$,

Abundances MSE: $||S - S_{pred}||^2$,

Endmembers cosine similarity: $\frac{<S,\, S_{pred}>}{||S|| \cdot ||S_{pred}||}$

Experiments

Models describing

Seven models were performed on two datasets.

Models without perturbations:

1) original no patches (Autoencoder, unsupervised loss)
2) original with patches (Autoencoder, unsupervised loss) - **first estimation**

Models with simple perturbations:

3) simple perturbations no patches
4) simple perturbations with patches

Models with pixel-wise perturbations:

5) pixel-wise perturbations no patches
6) pixel-wise perturbations with patches
7) pixel-wise perturbations with random patches - experimental part which avoid any image savings and potentially may have the same result as the 6th model. The idea is skip the part with total image generating and use only randomly generated patches from an initial image.

Urban dataset

Two first models are unsupervised and we choose the best of them for the first estimation which will be used in the next models. We can see that the second model is better in both metrics: +2.5% to endmembers similarity and +58.5% to abundance similarity. This is obtained thanks to patch technique which provides more variability to the autoencoder.

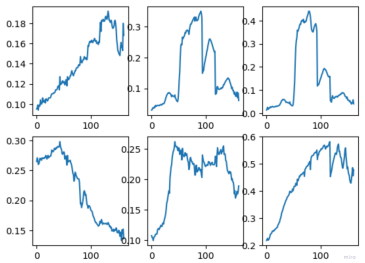
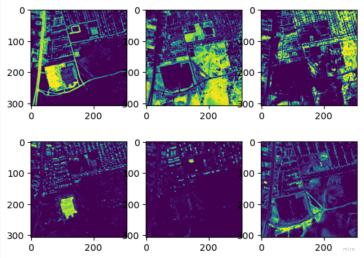
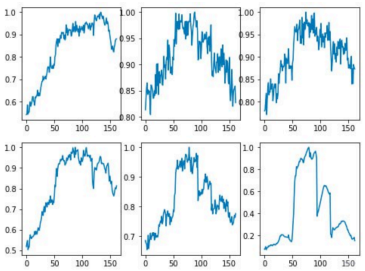
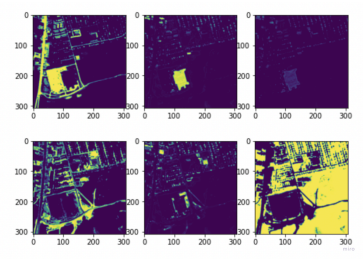
Going further using the second model as the first estimation we perform data generation with simple endmembers perturbations we obtain for both third and forth models a little improvements of endmembers and abundances metrics. Visually and theoretically we choose the forth model here as the best one.

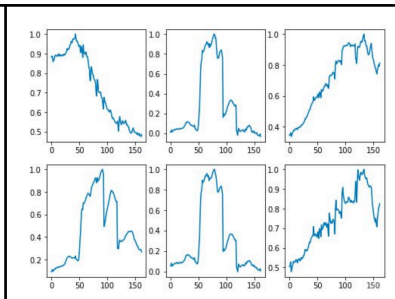
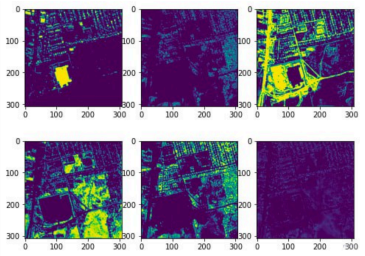
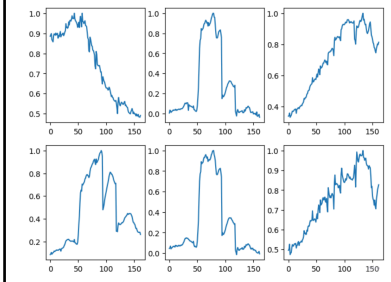
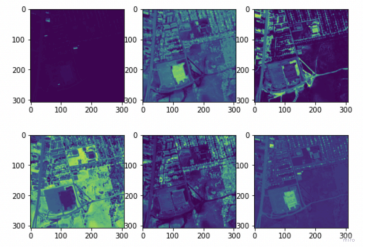
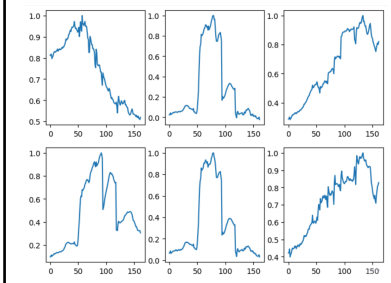
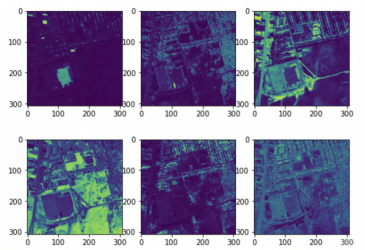
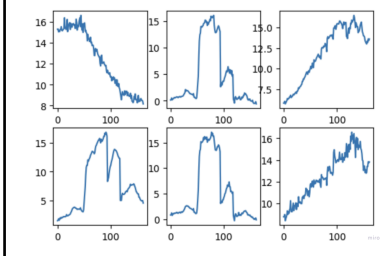
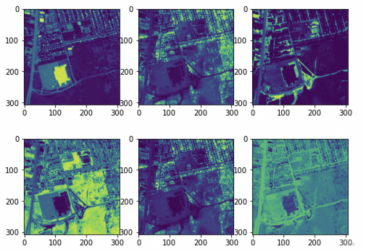
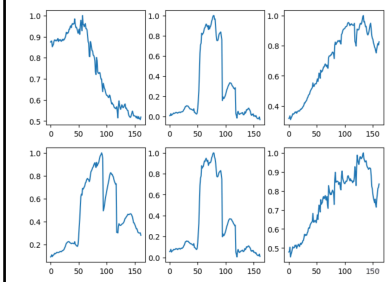
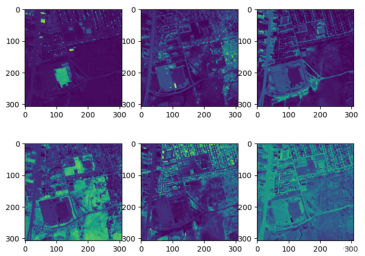
In the next three models we still used the second model as the first estimation for data generation. But the generating process was changed: instead of perturbing the endmember matrix once for each pixel we performed the pixel-wise perturbations described in previous sections. Considering the time the 6th model is the best here, it takes 3 hours to image generation and 1 hour for training. It also obtained good metrics: 0.9247 - for cosine similarity (endmembers) and 0.0439 - MSE (abundances). The 7th model has not too bad results too but the time is unexpectedly long even with many computation resources.

Model name	Parameters Subtitle	Time	Type	Metric cosine similarity (endmembers)	Metric MSE (abundances)	Metric cosine similarity (abundances)
1) original no patches	num_epochs=100 BATCH_SIZE = 10 NUMBER_IMAGES = 1	~5min	Local Apple M1	0.9006	0.0926	0.6464
2) original with patches	num_epochs=200 BATCH_SIZE = 10 PATCH_SIZE = 40 PATCH_NUMBER = 100 NUMBER_IMAGES = 1	~14min	Local Apple M1	0.9231	**0.0385**	**0.8439**
3) simple perturbations no patches	num_epochs=50 BATCH_SIZE = 10 PATCH_SIZE = 40 NUMBER_IMAGES = 70 NOISE_STD = 0.03	~50min	Google Collab T4	0.9232	0.0450	0.7710
4) simple perturbations with patches	num_epochs=30 BATCH_SIZE = 10 PATCH_SIZE = 40 PATCH_NUMBER = 50 NUMBER_IMAGES = 70 NOISE_STD = 0.03	~42min (gen) + 50 min (train)	Google Collab T4	0.9235	0.0518	0.7399
5) pixel wise perturbations no patches	num_epochs=30 BATCH_SIZE = 10 PATCH_SIZE = 40 NUMBER_IMAGES = 50 NOISE_STD = 0.03	~16 hours	Tesla V100 32Gb + Intel(R) Xeon(R) Gold 6278C CPU @ 2.60GHz	0.9223	0.0424	0.7887

6) pixel wise perturbations with patches	num_epochs=50 BATCH_SIZE = 10 PATCH_SIZE = 40 PATCH_NUMBER = 50 NUMBER_IMAGES = 70 NOISE_STD = 0.03	~3h (gen) + 1h (train)	Google Collab T4	**0.9247**	0.0439	0.7852
7) pixel wise perturbations with random patches	num_epochs=30 BATCH_SIZE = 10 PATCH_SIZE = 40 PATCH_NUMBER = 40 NUMBER_IMAGES = 50 NOISE_STD = 0.03	~11.5h	Tesla V100 32Gb + Intel(R) Xeon(R) Gold 6278C CPU @ 2.60GHz	0.9234	0.0686	0.6323

Table 1: Metric and computational results for the Urban dataset

Model name	Endmembers Urban	Abundances Urban
ground truth		
1) original no patches		

2) original with patches		
3) simple perturbations no patches		
4) simple perturbations with patches		
5) pixel wise perturbations no patches		
6) pixel wise perturbations with patches		

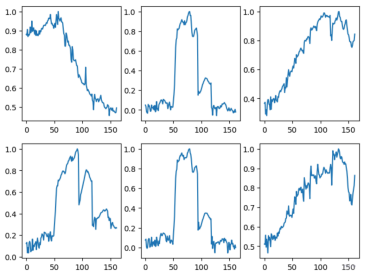
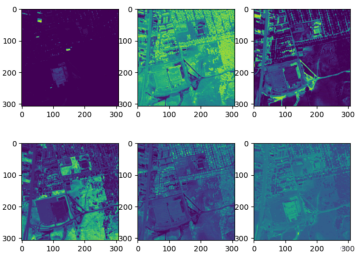
7) pixel wise perturbations with random patches		

Table 2: Endmembers and abundances results for the Urban dataset

Three best models were chosen in each category: original, with simple perturbations, with pixel-wised perturbations. The models were chosen not only with metrics values but also with time characteristics. These three models (number 2,4,6) are also used for training on the Samson dataset.

Samson dataset

The Samson dataset is simpler than the Urban dataset so we expect high metrics here. We don't change any parameters for the training model but the computational time for all models is much faster. We have achieved more than 0.99/1 metric cosine similarity for all models in this dataset. The fourth model performed best, which uses simple perturbations to generate data.

Model name	**Parameters** Subtitle	Metric cosine similarity	Metric MSE
2) original with patches	num_epochs=200 BATCH_SIZE = 10 PATCH_SIZE = 40 PATCH_NUMBER = 100 NUMBER_IMAGES = 1	0.9955	0.0485
4) simple perturbations with patches	num_epochs=30 BATCH_SIZE = 10 PATCH_SIZE = 40 PATCH_NUMBER = 50 NUMBER_IMAGES = 70 NOISE_STD = 0.03	**0.9971**	**0.0319**
6) pixel-wise perturbations with patches	num_epochs=50 BATCH_SIZE = 10 PATCH_SIZE = 40 PATCH_NUMBER = 50 NUMBER_IMAGES = 70 NOISE_STD = 0.03	0.9914	0.0842

Table 3: Metric and computational results for the Samson dataset

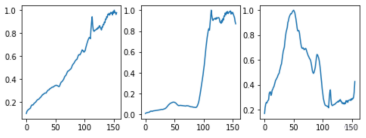
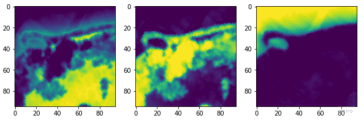
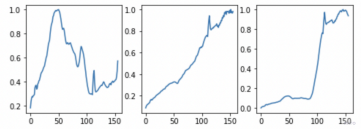
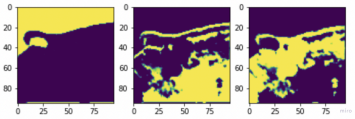
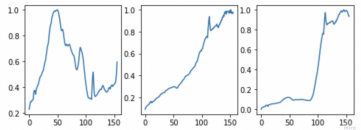
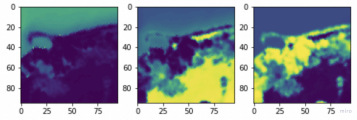
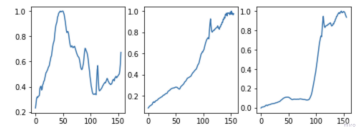
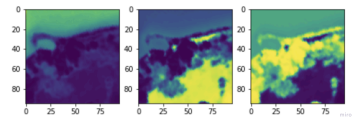
Model name	Endmembers Samson	Abundances Samson
ground truth		
2) original with patches		
4) simple perturbations with patches		
6) pixel wise perturbations with patches		

Table 4: Endmembers and abundances results for the Samson dataset

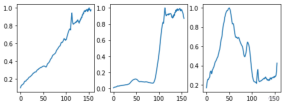
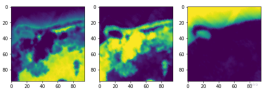
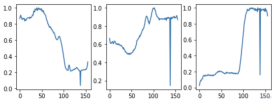
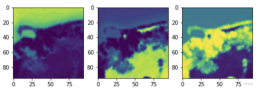
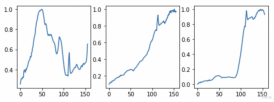
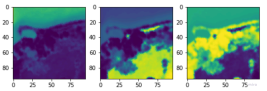
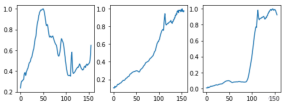
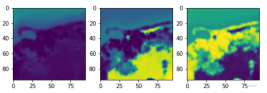
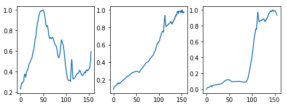
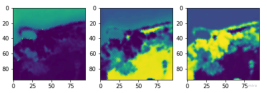
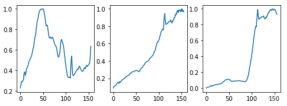
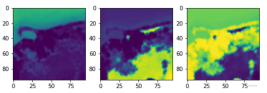
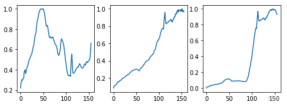
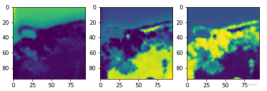
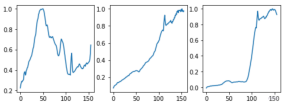
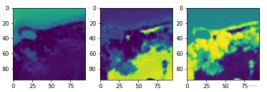
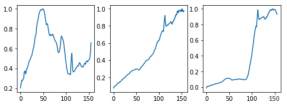
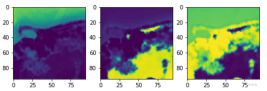
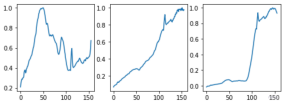
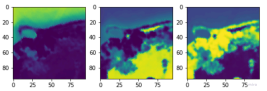
Visually we can observe good endmembers and abundances imitation by all three models. The second model has a specific behavior in abundances image - it is almost binary. This specificity can be explained by the simplicity of the model (unsupervised model here) and the image. Continuing to speak about abundances we observe that the forth model better works with contrast tha six one that's why it obtains better metrics. But visually the 6th model is more accurate with details.

Alpha dependencies in Balanced Loss

In the balanced loss we have a coefficient alpha which regularizes unsupervised and supervised parts. Bigger alpha means bigger influence of the unsupervised part:

$$\textit{alpha} * \left(\sum_{i=1}^{N}\left(1 - \frac{X_{pred} \cdot X}{\|X_{pred}\| \cdot \|X\|}\right)\right) + (1 - \textit{alpha}) * \left(\sum_{i=1}^{N}\left(1 - \frac{A_{pred} \cdot A}{\|A_{pred}\| \cdot \|A\|}\right) + \sum_{i=1}^{N}\left(1 - \frac{S_{pred} \cdot S}{\|S_{pred}\| \cdot \|S\|}\right)\right)$$

Let's consider experiments with various alphas for the best model (simple perturbations with patches) from the previous subsection.

Alpha value	Metric cosine similarity	**Endmembers Samson**	Metric MSE	**Abundances Samson**
ground truth				
0	0.9555		0.0452	
0.1	0.9923		0.0841	
0.2	0.9929		0.0912	
0.3	0.9955		0.0319	
0.4	0.9939		0.1192	
0.5	0.9929		0.0261	
0.6	0.9919		0.0448	
0.7	0.9928		0.1065	
0.8	0.9897		**0.0220**	

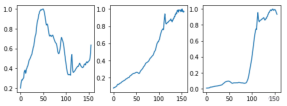
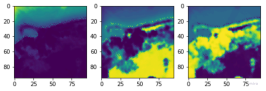
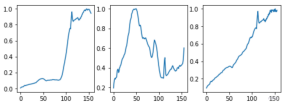
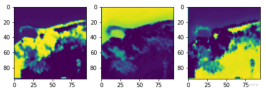
0.9	0.9921		0.0402	
1	**0.9966**		0.0472	

Table 5: Metric results for different alpha levels

As we see, all results are quite similar; however, we can notice several interesting observations.

When alpha equals zero (meaning that we use only supervised loss), it leads to poor results with endmembers but produces good enough results with abundances. When alpha equals 1 (indicating the use of only unsupervised loss), it results in the best outcome with endmembers. The optimal result with abundances is achieved when alpha equals 0.8. The least favorable outcome with abundances occurs when alpha equals 0.4.

So there is not one alpha that can refine both metrics at the same time but we can find a balance depending on our needs. In all previous experiments we use alpha equals 0.3 that show a good balance between endmembers and abundances similarities.

VAE approach

Training dataset for VAE. VCA

The vertex component analysis (VCA) is a method for unsupervised endmember extraction from hyperspectral data [Nascimento 2005] assuming that the data is a linear mixture of pure components (spectra). VCA assumes the presence of pure pixels in the data. It extracts a predefined number of pure components and estimates both the pure spectra and the concentration maps of the found components.

We applied VCA 100 times to Samson and obtained 100 slightly different approximations of the endmembers. A prerequisite for the correct operation of FCLS and further training of VAE was the normalization of the resulting endmembers.

VAE

Network Architecture: The VAE architecture comprises an encoder and a decoder. The encoder consists of two fully connected layers followed by Leaky ReLU activation functions, culminating in the prediction of the mean and log variance of the latent space. Symmetrically, the decoder mirrors this structure to reconstruct the input data from the sampled latent space representation.

Latent Space Representation: The latent space, with a dimensionality of 2 in this implementation, serves as a compact and expressive representation of the input data. The mean and log variance parameters predicted by the encoder govern the distribution of points in this latent space, enabling the model to capture the underlying structure of the data distribution.

Reparameterization Trick: The reparameterization function plays a pivotal role in sampling latent space points during training. By decoupling stochasticity from the network parameters, this technique ensures smooth gradient propagation and facilitates efficient optimization through backpropagation.

Activation Functions: Leaky ReLU activation functions are employed throughout the encoder and decoder networks, promoting robust training by allowing the propagation of gradients even for negative input values.

Loss Function: During training, the VAE optimizes a combination of two loss functions: the reconstruction loss and the Kullback-Leibler (KL) divergence loss. While the reconstruction loss guides the model to faithfully reproduce the input data, the KL divergence loss regularizes the latent space distribution, ensuring it approximates a desired prior distribution.

Training Autoencoder using VAE-enriched data

We utilized the trained Variational Autoencoder (VAE) to generate 100 endmembers of the Samson dataset, following which we employed Fully Constrained Least Squares (FCLS) to derive abundances from these endmembers. With both the endmembers and abundances in hand, the construction of hyperspectral images was straightforward through the application of a linear model. Additionally, we applied the patch technique while Autoencoder training, consistent with our approach in other models, to enhance the robustness and effectiveness of our method.

Experiments

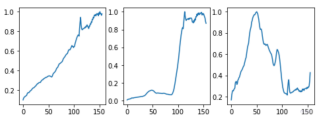
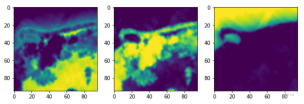
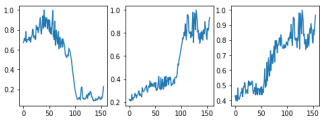
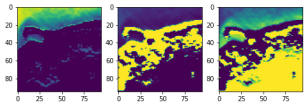
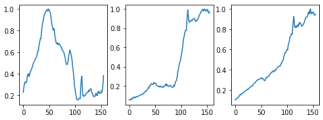
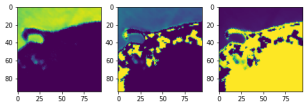
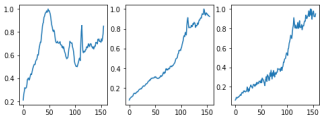
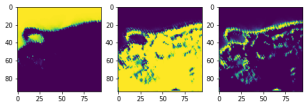
Alpha value	Metric cosine similarity	**Endmembers Samson**	Metric MSE	**Abundances Samson**
ground truth				
0	0.9454		0.1065	
0.3	0.9936		0.1368	
1	0.9592		0.1632	

Table 6: Metric results for different alpha levels of VAE approach

The best metric result for endmembers is obtained with alpha equal to 0.3. At the same time, alpha equal to 0 is the best for abundances. It's notable to mention that the disparities between the results of the different alphas are not substantial. As a result, to strike a balance between them, we opt for an alpha value of 0.3.

The VAE approach demonstrates similar performance to the Perturbation approach when it comes to identifying endmembers. However, it significantly underperforms (10 times worse) in accurately estimating abundances.

Conclusion

Within the context of our work, we carefully read the articles and implemented in total twenty five models based on the autoencoder deep neural network to compare different approaches of endmember perturbation, patch techniques and VAE data enrichment.

We used two datasets, Urban and Samson, to train the models and collect results. The best model on the Urban dataset was the one with pixel-wise perturbations that used patches. The best model on the Samson dataset was the one with simple perturbations, which also used patches. We explain this fact by the simplicity of the Samson dataset; for the more complicated Urban dataset, we needed a more intricate method to obtain the best results. However, for Samson, this complexity is redundant.

Moreover, we conducted an experiment in which we determined the best alpha for the loss function to achieve the highest quality of endmembers/abundances during retrieval: 0.8 for abundances and 1 for endmembers. Noting these results we can suppose that the unsupervised part has more impact on hyperspectral unmixing problem but the presence of supervised part provides a more detailed abundance image.

Comparing Perturbations and VAE approaches we can claim that the first one is more efficient for the chosen data. However, the VAE approach is significantly faster in training.

Division of work in the group

During the course of the project, two students from the SD Option Intern program participated: Viktoriya Zhukova and Maksim Kochanov. The research advisor: Christophe Kervazo.

The collaborative work between the students was divided into two absolutely equal parts, with all tasks executed jointly, including the report writing, coding and presentation preparation.

Each of us had specific areas of responsibility:

Viktoriya Zhukova: Implementation of the models with simple and pixel-wise perturbations, making a pipeline for the Samson dataset, integration of the patch separation function into the models, implementation of the metrics.

Maksim Kochanov: Implementation of the autoencoder model, conducting experiments with hyper-parameters in loss function, making a pipeline for the Urban dataset, implementation of the loss functions and implementation of the VAE approach.