Evaluating autoencoders for hyperspectral anomaly detection

Ema Galuppini, Julie Laurent, Quentin Senatore

December 16, 2025

Abstract

Hyperspectral imaging is a technique used to collect the spectrum of each pixel in the image of a scene. This type of imaging can be used for anomaly detection and has applications in various sectors such as military or agriculture. Past research, such as statistical methods, has shown limits to accurately detecting anomalies in complex images. Detection methods based on deep learning can be used to improve the performance of the detection. We aim to compare the performance of statistical and deep learning detection algorithms by testing a new method called autoencoders, which are a type of neural network. Different values of hyperparameters (epochs, batch size, learning rate, metrics, number of layers) are tested to determine which combination better detects the anomalies. Additionally, an alternative method called variational autoencoders is implemented. The database used for analysis is composed of two aerial images provided by ONERA. Our algorithms are implemented in Python. We found that optimizing the hyperparameters improves the detection results, which are comparable than state of the art algorithm. Additionally, using variational autoencoders gives comparable results to autoencoders. Next researches could focus on more complex images with mixed background to evaluate the differences between autoencoders and variational autoencoders results.

Key words: Hyperspectral imaging, Autoencoders (AEs), Variational autoencoders (VAEs), Anomaly Detection, Deep Learning

---

1 Introduction

Hyperspectral imagery is a method that emerged during the 1980's, which consists of taking images that capture a large amount of continuous and contiguous spectral bands typically ranging between ultraviolet and infrared. A hyperspectral image can be seen as a cube, each pixel having its own electromagnetic spectrum, stocked in the form of a vector. Each component of this vector is the radiation intensity for a given spectral band. The information in each pixel is consequently more complex than for a classic RGB image. This method gives access to information that would have been impossible to see with the naked eye.

Images produced by this type of technology are used to detect objects, processes or materials [7]. Anomaly detection based on hyperspectral data is an active field of research. We can define an anomaly as something that strongly differentiates itself from its background, or that should not be present naturally in its environment. The use of hyperspectral imagery for anomaly detection is interesting in the case of anomalies that are not visible for the human spectrum.

Many different methods have been developed to detect anomalies. They range from statistical, distance, reconstruction, or subspace methods [2]. Early studies mostly focus on statistic-based methods as shown in [4]. Although the statistical approach gives accurate results, in some complex images it can tend to have more difficulties detecting anomalies. As explained in [9], autoencoders have shown great potential for anomaly detection because they make no strong assumptions about

data distribution and allow non-linear dimensionality reduction.

A different type of autoencoder can be used to detect anomalies on hyperspectral images. They are called a variational autoencoders and were introduced in 2014 by Diederik P. Kingma and Max Welling [3]. Variational autoencoders are an extension of autoencoders which add new constraints on their latent space. They can also be promising methods, their functioning is explained in detail in [8]. To the best of our knowledge, no previous and open-source models have been proposed clearly for hyperspectral anomaly detection using autoencoders and variational autoencoders. In this paper, we compare autoencoder's performances to a statistical method called the Reed-Xiaoli (RX) algorithm [6]

The goal of this study is to compare the autoencoder results with the RX algorithm results to determine which one performs better, by identifying the best combination of hyperparameters (size of epochs, learning rate, distance, number of layers, etc.)

Outline

In Section 2, we introduce the data we work on as well as autoencoders and their structure. We also explain the function of each hyperparameter composing the autoencoder. Section 3 contains detection performance results depending on the variable parameter and an interpretation of the results obtained. In Section 4, we discuss the results of the variational autoencoders applied to an image with unsatisfactory reconstruction performance.

2 Materials and methods

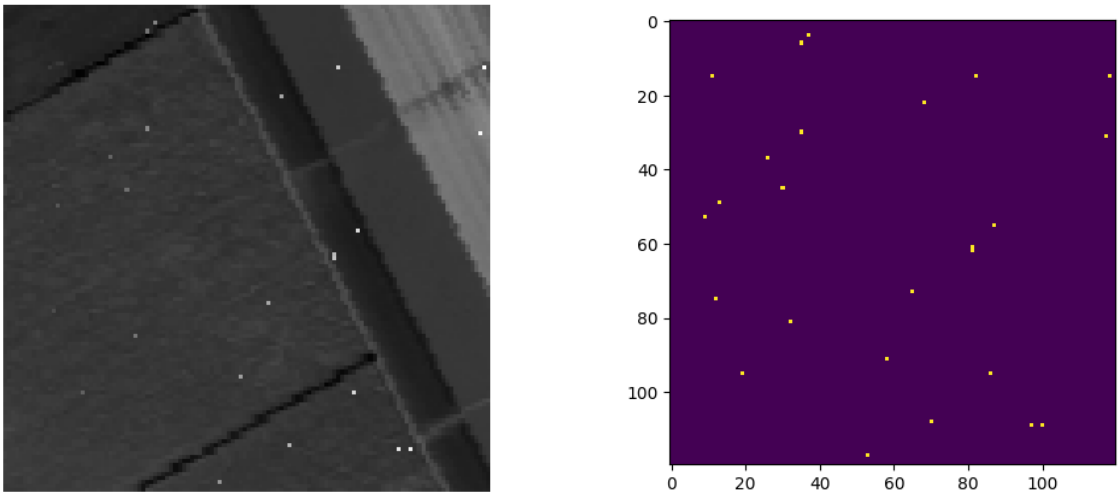
2.1 Data description

For this research, we work on two different images. Figure 1.a and Figure 1.b represent a lake shore from above where anomalies are boats and objects on the beach. Figure 2.a and Figure 2.b is a field with small sparse anomalies (white boards).

(a) Lake shore image provided by ONERA, coming from the hyperspectral sensor ODIN, developed by NEO (Norway) as part of the Sysiphe project (project of the DGA - Direction Générale de l'Armement -France)

(b) Ground truth image of the Lake Shore, provided by ONERA, coming from the hyperspectral sensor ODIN, developed by NEO (Norway) as part of the Sysiphe project (project of the DGA - Direction Générale de l'Armement -France)

Figure 1: Overview of the Lake shore image with its ground truth.



(a) AVIRIS image (field)

(b) Ground truth (AVIRIS)

Figure 2: Overview of AVIRIS image with its ground truth (NASA). Provided by the Jet Propulsion Laboratory and the California Institute of Technology.

2.2 Evaluation methodology

To determine the optimal hyperparameters of the autoencoder to detect anomalies on a specific image, we use an experimentation and trial-error method. This process involves iteratively testing different hyperparameter values and adjusting them according to the obtained results.

To benchmark our autoencoder-based method, we compare it to the Reed-Xiaoli (RX) algorithm which is a standard anomaly detection technique in hyperspectral imaging. RX assumes that the background follows a multivariate normal distribution and computes the Mahalanobis distance D_M between each pixel $\mathbf{x}$ and the estimated background mean μ using the covariance matrix Σ [6]:

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \mu)^\top \Sigma^{-1} (\mathbf{x} - \mu)} \tag{1}$$

Pixels with high RX scores are considered anomalies. This algorithm is widely used as a baseline due to its simplicity and effectiveness under gaussian assumptions .

The different values of the hyperparameters are compared using ROC (Receiver Operating Characteristic) curves. ROC curves show the performance of the model by plotting the true positive rate against the false positive rate at different threshold settings. The true positive rate measures the proportion of actual anomalies that are correctly identified by the model, while the false positive rate indicates the proportion of normal data points that are incorrectly classified as anomalies. In this study, we want to avoid false negatives. Therefore, we prioritize ROC curves that have a high true positive rate while having a low false positive rate. In a graph, we want to have the beginning of the curve at a high level. To deal with the variability of the autoencoder, the average ROC curve on multiple iterations can be computed instead of a single ROC curve.

Distance maps are also used to visualize the results. These maps are generated by computing the reconstruction error which is the pixel-wise difference between the original image and its reconstruction by the autoencoder. Areas with high differences often correspond to anomalies. These visualizations support the interpretation of model outputs and help localize anomalous regions within the image.

Several metrics are used to calculate the reconstruction error. We consider the Mahalanobis distance (1), the Mean Squared Error (MSE) and the Spectral Angle Mapper (SAM). MSE cal-

culates the squared difference between the original and reconstructed pixel values, providing a straightforward measure of reconstruction quality. SAM evaluates the angle between spectral vectors, which makes it less sensitive to illumination changes.

2.3 Autoencoder architecture and functioning

An autoencoder is a neural network architecture commonly used in unsupervised learning which purpose is to reconstruct input data by extracting and manipulating essential features. The architecture of an autoencoder contains three main components: an encoder, a bottleneck, and a decoder (Figure 3).

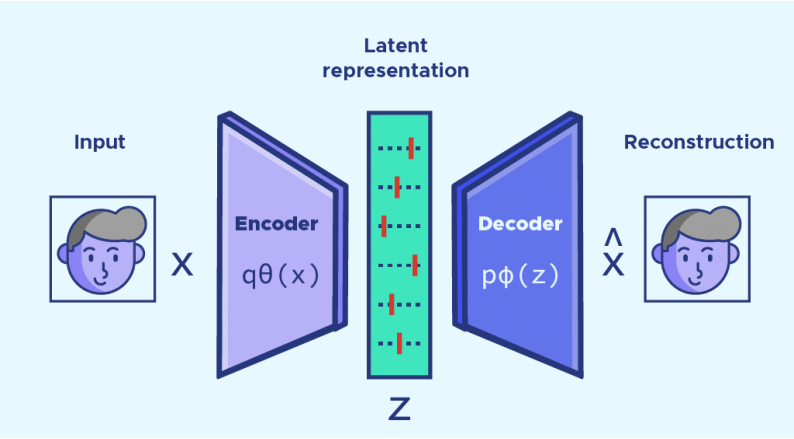


Figure 3: Simplified structure of an autoencoder. The input image x is compressed into the latent space Z, then reconstructed into the output image $\hat{x}$ adapted from [5].

Each part plays a key role in data compression, representation, and reconstruction. The encoder is the first component of the model. It reduces the dimensionality of the input while preserving the most important information. Its purpose is to extract the necessary features while minimizing the size of the data to make the reconstruction easier in later stages. The encoder learns to ignore redundancies and irrelevant details. It transforms raw input data into a condensed form known as the latent space. Structurally, the encoder is a neural network made of several successive layers. The number of neurons decreases progressively through these layers, leading to data compression.

Then, the data passes through the second component: the bottleneck layer. This layer is also called the code or latent representation and is located between the encoder and the decoder. It acts as the output of the encoder and the input of the decoder. The bottleneck is a compact single layer where the number of neurons is supposed to be small. It forces the model to compress the information and has a latent representation of the data which contains the most abstract and essential form of the input data.

The decoder is the final component of the autoencoder and is responsible for reconstructing the original input from the compressed representation. The purpose of the decoder is to restore the data as closely as possible to its initial form, minimizing the reconstruction error. This error makes it possible to measure the difference between the original input and the generated output. The decoder is symmetrical to the encoder, but in contrast to the encoder, the number of neurons increases progressively through its hidden layers. This component reverses the compression process. Starting from the code, the decoder gradually rebuilds the data, increasing its complexity and size until reaching an output similar to the input.

2.4 Mathematical structure of autoencoders

Having outlined the overall architecture of an autoencoder, we now briefly turn to the mathematics that are at the basis of its functioning. This allows us to better understand how to calibrate the different parameters in order to optimize the anomaly detection process.

As every neural network, an autoencoder is composed of multiple layers. They are of the following form:

$$h^{(n)} = \sigma(W^{(n)}h^{(n-1)} + b^{(n)}) \tag{2}$$

Here, the n-th layer takes a $h^{(n-1)}$ vector in input (from the previous layer) and returns a new vector $h^{(n)}$.

-σ is called the activation function
-$W^{(n)}$ is called the weight matrix
-$b^{(n)}$ is called the bias vector

The ReLu function is used as our activation function, as it is frequently done for image processing : $ReLu(x) = max(0, x)$.

By adjusting the size of the weight matrices and bias vectors, the dimension of the vector that passes through the layer can be enlarged or reduced. The encoder and decoder of an autoencoder are built in this way. It is a succession of shrinking layers followed by growing layers.

Generally, the layers are initialized randomly using a normal distribution. Once a vector has been through all the layers, the reconstruction error can be computed using the Mean Square Error (MSE) function:

$$MSE(x, \hat{x}) = \frac{1}{d}\sum_{i=1}^{d}(\hat{x}_i - x_i)^2 \tag{3}$$

where x is the input vector and $\hat{x}$ is the reconstructed vector. They are both of dimension d. Other loss functions can be used,such as the L1 norm.

We want to minimize this loss function. The last layer l of our autoencoder is of the following shape: $\hat{x} = \sigma(z)$ with $z = W^{(l)}h^{(l-1)} + b^{(l)}$.

Using the chain rule we have that:

$$\frac{\partial MSE}{\partial W^{(l)}} = \frac{\partial MSE}{\partial \hat{x}} \cdot \frac{\partial \sigma}{\partial z} \cdot \frac{\partial z}{\partial W^{(l)}} \tag{4}$$

And in the same way:

$$\frac{\partial MSE}{\partial b^{(l)}} = \frac{\partial MSE}{\partial \hat{x}} \cdot \frac{\partial \sigma}{\partial z} \cdot \frac{\partial z}{\partial b^{(l)}} \tag{5}$$

In the last layer, the weight matrix $W^{(l)}$ and the bias vector $b^{(l)}$ can be updated using the gradients we just calculated and the algorithm of our choice. The simplest way would be to perform a classic stochastic gradient descent:

$$W^{(l)} = W^{(l)} - \eta \frac{\partial MSE}{\partial W^{(l)}} \quad \text{and} \quad b^{(l)} = b^{(l)} - \eta \frac{\partial MSE}{\partial b^{(l)}} \tag{6}$$

To improve the efficiency of our autoencoder, we use the Adam optimizer [1] instead of a gradient descent.

With this same method, we can now compute $\frac{\partial MSE}{\partial W^{(l-1)}}$ and $\frac{\partial MSE}{\partial b^{(l-1)}}$. Once those are calculated, we can go back one layer, and so on until the first layer. This technique is called gradient backpropagation and it lies at the core of autoencoder training.

To summarize, our input vector x is first passed through the autoencoder using randomly initialized weights and biases. Then, using a chosen optimization algorithm, these parameters are updated using backpropagation. The vector x is passed again through the autoencoder, and the weights W and biases b are updated accordingly. This process is repeated iteratively until the reconstruction of x becomes sufficiently accurate.

2.5 Mathematical structure of variational autoencoders

A different type of autoencoder can be used to detect anomalies on hyperspectral images. They are called a variational autoencoders and were introduced in 2014 by Diederik P. Kingma and Max Welling [3].

The main difference between variational autoencoders (VAE) and classical autoencoders consists in the fact that VAEs constraint their latent space to have a certain form. In a normal autoencoder, two points that are close in the latent space can give two very different reconstructions. A VAE forces its latent space to be gaussian. The idea behind a VAE is to assign for each vector that will pass through it a mean and a variance in the latent space, rather than a fixed position.

Mathematically, this works by splitting the last layer before the latent space into two branches, a "mean" branch and a "variance" branch. These two branches are defined as follows: $W_\mu x + b_\mu$ and $W_\sigma x + b_\sigma$. They both take as input the vector x from the layer before, and output a vector of the same dimension as the latent space.

In this way, a vector z in the latent space is defined by a normal distribution $q(z|x)$:

$$q(z|x) = \mathcal{N}(z; \mu(x), \sigma(x)^2) \tag{7}$$

To sample a vector from the latent space, a VAE uses what is called the reparameterization trick. The input vector z for the decoder is chosen using the following computation:

$$x = \mu + \sigma \cdot \epsilon \tag{8}$$

where $\epsilon \sim \mathcal{N}(0, I_d)$ and the latent space has a dimension of size d. The reparameterization trick allows gradient backpropagation through the probabilistic latent space.

To constraint the latent space to be gaussian, a term is added to the VAE's loss function called the Kullback–Leibler (KL) divergence. This divergence measures the gap between $q(z|x)$ and a standard gaussian $\mathcal{N}(0, 1)$.

$$KL\left(\mathcal{N}(\mu, \sigma^2) \,\|\, \mathcal{N}(0, 1)\right) = \frac{1}{2} \sum_{i=1}^{d} \left(\mu_i^2 + \sigma_i^2 - \log \sigma_i^2 - 1\right) \tag{9}$$

In this way, the loss function is:

$$\mathcal{L} = MSE(x, \hat{x}) + \beta KL\left(\mathcal{N}(\mu, \sigma^2) \,\|\, \mathcal{N}(0, 1)\right) \tag{10}$$

β is a positive parameter that quantifies to which extent we want to "force" the latent space to follow a gaussian distribution. A higher value of β will penalize a point more strongly who has a mean far from 0 and a variance far from 1. For the first term of the loss function, the $L1$ norm can also be used instead of MSE.

VAEs can be used to search for anomalies in a new way compared to classical autoencoders. Instead of comparing the reconstructed image with the original one, we can directly look into the latent space. Since the latent space is gaussian, the Mahalanobis distance can be used to compute the distance of each point to the background of this latent space. In this way, points with a high distance likely correspond to an anomaly.

2.6 Impact of hyperparameters on autoencoder performance

To determine the optimal hyperparameters in our study, we decide to systematically vary key parameters of the autoencoder and evaluate their impact on anomaly detection performance. Each hyperparameter influences the model's capacity to learn, compress, and reconstruct the input data effectively. Generally, when training a neural network, two common problems are underfitting and overfitting. Overfitting occurs when the model learns not only the underlying patterns in the training data but also its noise or irrelevant details. However, underfitting occurs when the model is too simple to capture the underlying structure of the data. In this case, the autoencoder fails to learn important patterns during training, leading to poor reconstruction performance and low anomaly detection accuracy.

The parameters that we evaluate are described below.

The number of layers defines the depth of the autoencoder. Increasing the number of layers allows the model to capture more complex and abstract features. However, deeper networks are more challenging to train and are more prone to overfitting. Training deep neural networks is also more difficult for several reasons. First, the backpropagation process can suffer from vanishing or exploding gradients, which slows down or destabilizes learning. Second, deeper networks contain more parameters, increasing training time and computational cost.

The number of epochs refers to the number of complete passes through the training dataset during model training. If the number of epochs is too low, the model may be underfitted and not capture essential patterns.

Batch size determines the number of training samples processed before the model's internal parameters are updated. A small batch size can lead to more robust generalization but might increase training time and introduce noise in gradient updates. A large batch size can speed up training but risks converging to poor local minima and reducing the model's capacity to detect anomalies.

The learning rate η governs how much the model's weights are updated during each training step. A high learning rate may speed up convergence but can cause instability or overshooting of the optimal solution. A low learning rate ensures more stable convergence but requires longer training times. Finding an appropriate learning rate is critical for ensuring efficient and stable training.

The size of the latent space defined by the number of neurons in the bottleneck layer controls the degree of data compression. A smaller latent space encourages the model to preserve only essential features, which can enhance its ability to detect anomalies. However, excessive compression may cause loss of important information, leading to poor reconstruction quality. We investigate the influence of different latent space sizes on the trade-off between compression and reconstruction accuracy.

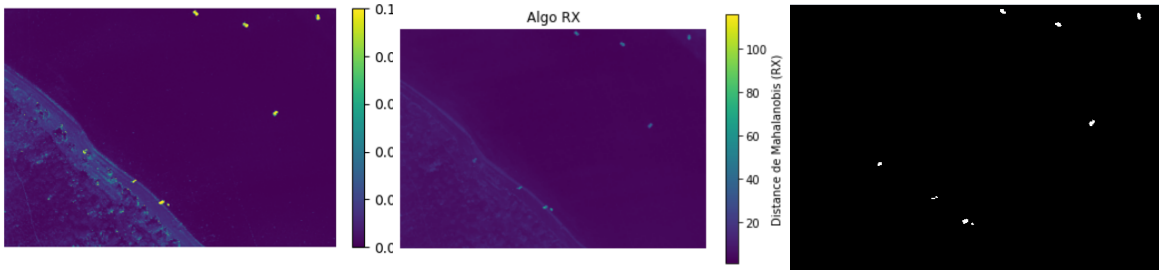
3 Optimal anomaly detection performance of autoencoders

In this section, we present the results of the anomaly detection performance of the autoencoder, achieved by optimizing the hyperparameters. The results show how adjusting each hyperparameter impacts the model's ability to detect anomalies with high accuracy. By fine-tuning these hyperparameters, we are able to identify the optimal settings that led to the best performance in anomaly detection. The following subsections detail the performance metrics and the impact of the selected hyperparameter combinations on the model's detection accuracy.

3.1 Image 1 - Lake shore

3.1.1 Final results

Figure 4 shows the result of the performance detection using autoencoders after choosing the most optimal hyperparameters: 2 layers, 1 epoch, learning rate=0.001, batch size = 32 and distance metric = MSE. This result can be compared to the result of the performance detection using RX algorithm and it appears to be equally performant.



(a) Distance map using autoencoders under fixed hyperparameters: 1 epoch, 2 layers, batch size of 32, learning rate = 1e-3, latent space size = 4, MSE metric

(b) Distance map using RX algorithm

(c) Ground truth image of the Lake Shore

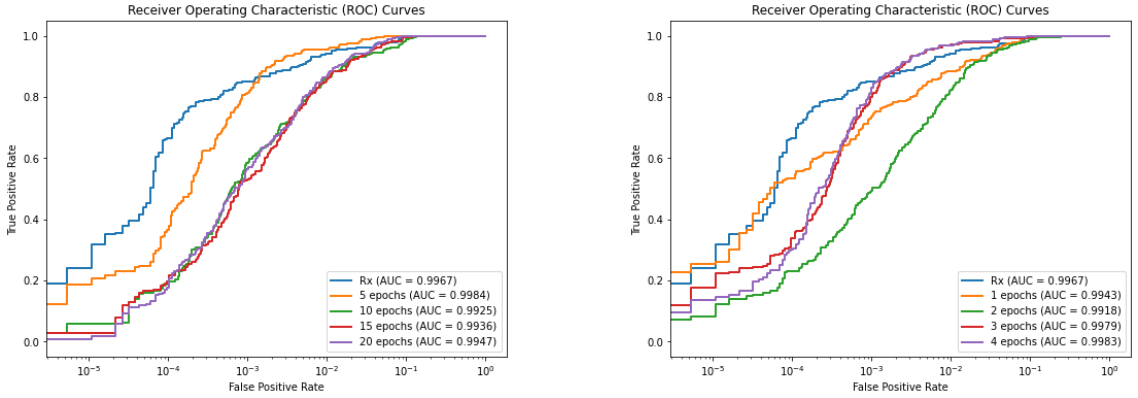
Figure 4: Comparison of distance maps using different methods of detection

3.1.2 Intermediate results

To obtain this result, we have to evaluate the performance of the autoencoder for each hyperparameter. We start our experiment with these hyperparameters values:

- Number of epochs = 5
- Number of layers = 5
- Batch size = 32
- Learning rate = 1e-3
- Size of latent space = 4
- Distance metric = MSE

First, we evaluate the impact of the number of training epochs on the model's ability to detect anomalies by comparing the resulting ROC curves. As the number of epochs increases, the model typically improves its reconstruction quality. As shown in Figure 5, performance stagnates or even decreases due to overfitting beyond a certain point.



(a) (b)

Figure 5: ROC curves for different numbers of training epochs under fixed hyperparameters: 1 epoch for a) , 5 epochs for b) , batch size = 32, learning rate = 1e-3, latent space size = 4.

In general, the RX algorithm shows the best performance.

We assume that the model has already learned the necessary features from the first epoch. Since better results are obtained with only one epoch, this can mean that our model is overfitting the data. As a result, we choose to work with one epoch only.

Trying to simplify our model suggests that reducing the number of layers in the autoencoder could be a promising direction to explore. An interesting test conducted next is to reduce the size of the model.

The average ROC curve on 5 iterations is studied to deal with the variability of the autoencoder with 1 epoch in Figure 6.

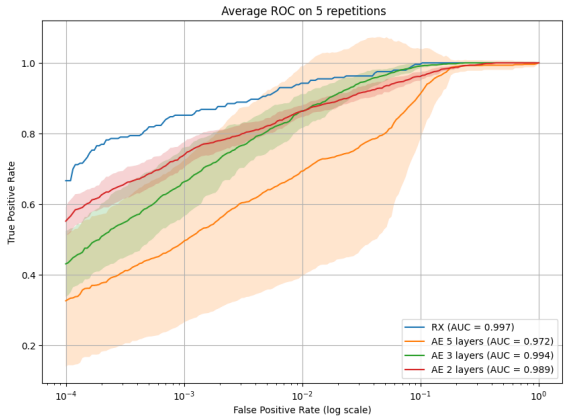


Figure 6: Average ROC curves for different number of layers under fixed hyperparameters: 1 epoch, batch size = 32, learning rate = 1e-3, latent space size = 4

Reducing the number of layers to avoid overfitting is relevant as shown in Figure 6. The model with 2 layers and 1 epoch is more performant than the models with 3 or 5 layers and 1 epoch.

Next we analyze the performance depending of the chosen batch size. As we can see in Figure 7, a small batch size presents the best AUC (Area Under Curve) and has larger values at the beginning of the curve. To be sure of choosing the correct batch size, we can analyze the distance map in Figure 8. The distance maps for different batch sizes can be found in the Appendix A.

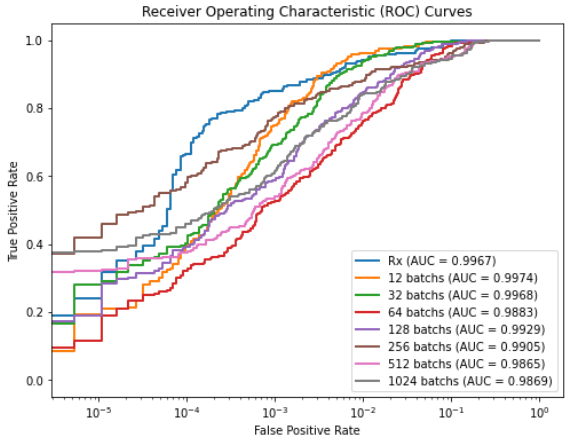
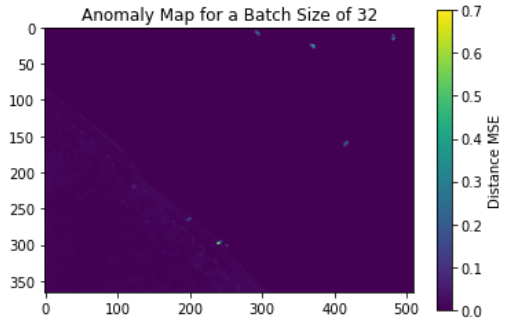


Figure 7: ROC curves for different batch sizes under fixed hyperparameters: 1 epoch, 2 layers, learning rate = 1e-3, latent space size = 4



(a) batch size = 32

Figure 8: Distance maps for batch size of 32 under fixed hyperparameters: 1 epoch, 2 layers, learning rate = 1e-3, latent space size = 4.

We observe that with a high batch size the algorithm detects fewer anomalies than with a smaller batch size. Therefore, we decide to keep a batch size of 32. We did not choose a smaller batch size to keep a reasonable training time.

Additionally, we have determined the most performant learning rate for the Lake shore image. The result is shown in Figure 9.

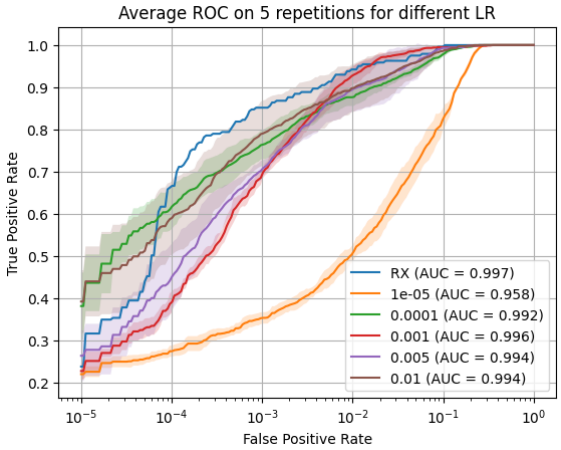


Figure 9: Average ROC curves for different learning rates under fixed hyperparameters: 1 epoch, 2 layers, batch size = 32, latent space size = 4.

Figure 9 shows that learning rate of 0.001 presents the best AUC but learning rate of 0.0001 has larger values on the beginning of the curve. To help us decide on a learning rate, distance maps are used.

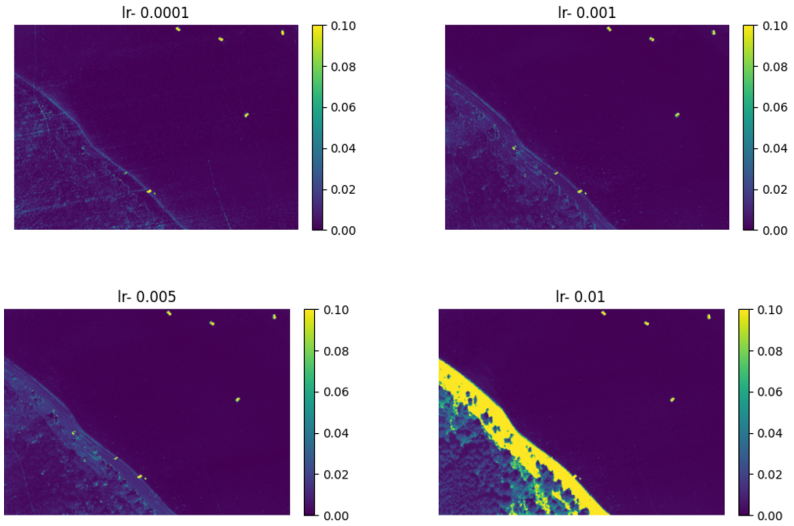


Figure 10: Average distance maps for different learning rates under fixed hyperparameters: 1 epoch, 2 layers, batch size = 32, latent space size = 4.

Based on Figure 10, we have decided to choose a learning rate equal to 0.001 since it has better visual quality, mainly for the anomalies on the shore.

Finally, in Figure 11, Mean Square Error (MSE), Spectral Angle Mapper (SAM), and Mahalanobis metrics are compared to improve our results, which are already satisfactory. The SAM method does not work well and the Mahalanobis metric has the same AUC as the MSE one that is used since the beginning of this paper (see Section 2.2).

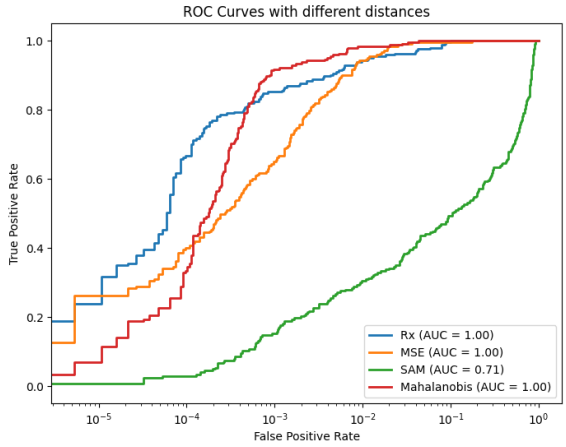
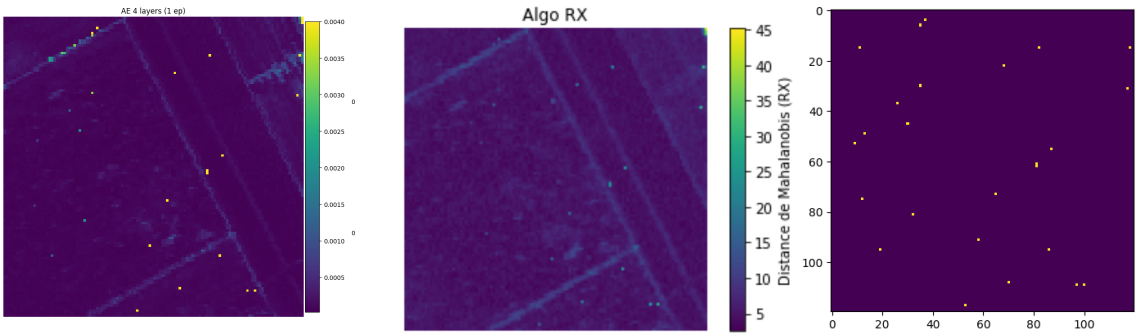


Figure 11: Average ROC curves for different distance metrics under fixed hyperparameters: 2 layers, 1 epoch, batch size = 32, latent space size = 4, learning rate=1e-3

3.2 Image 2 - Aviris

3.2.1 Final results

Figure 12 shows the result of the performance detection using autoencoders after choosing the most optimal hyperparameters. This result can be compared with the performance detection result using the RX algorithm. Therefore, autoencoders can be as performant as the RX algorithm.



(a) Distance map using autoencoders under fixed hyperparameters: 1 epoch, 4 layers, batch size of 12, learning rate = 5e-3, latent space size = 4, MSE metric

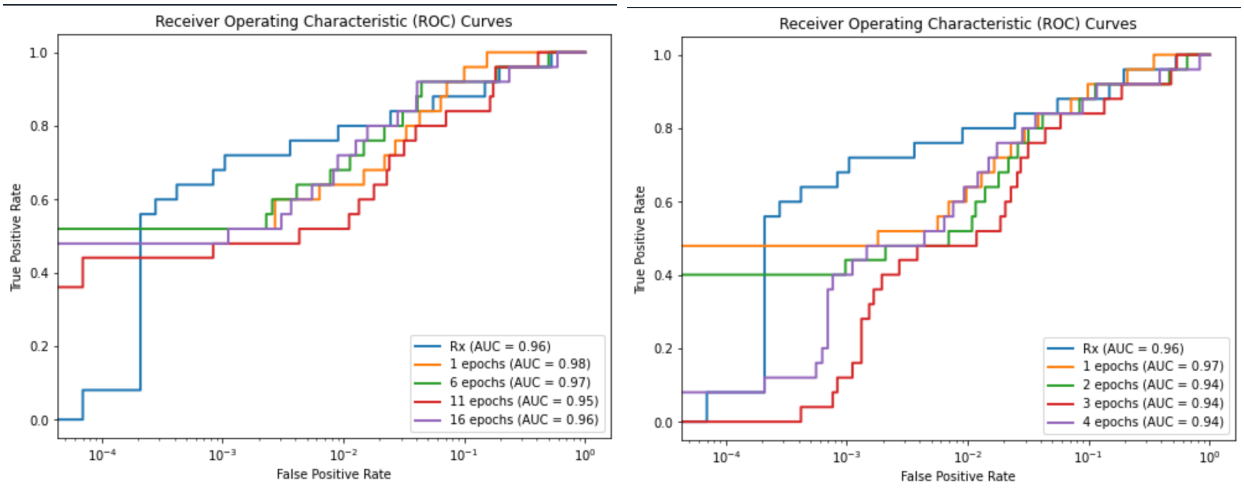
(b) Distance map using RX algorithm

(c) Ground truth image

Figure 12: Comparison of distance maps using different detection methods

3.2.2 Intermediate results

The results above are obtained using the same method as used in Section 3.1.
First we compare the performance results depending on the number of epochs chosen. Figure 13 shows similar results to those for Image 1. We can see the model overfitting as the number of epochs increases, but this time, the RX algorithm is less performant than for the first image. As a result, we choose to work with one epoch only.



(a) (b)

Figure 13: ROC curves for different numbers of training epochs under fixed hyperparameters: 5 layers, batch size = 32, learning rate = 1e-3, latent space size = 4.

On the Aviris image, a grid search experiment is conducted to determine the optimal number of layers and epochs. All the possible combinations of layers and epochs are computed. The results (ROC curves) are shown in Figures 14. Average distance maps for different number of layers can be found in Appendix B (Figure 27).

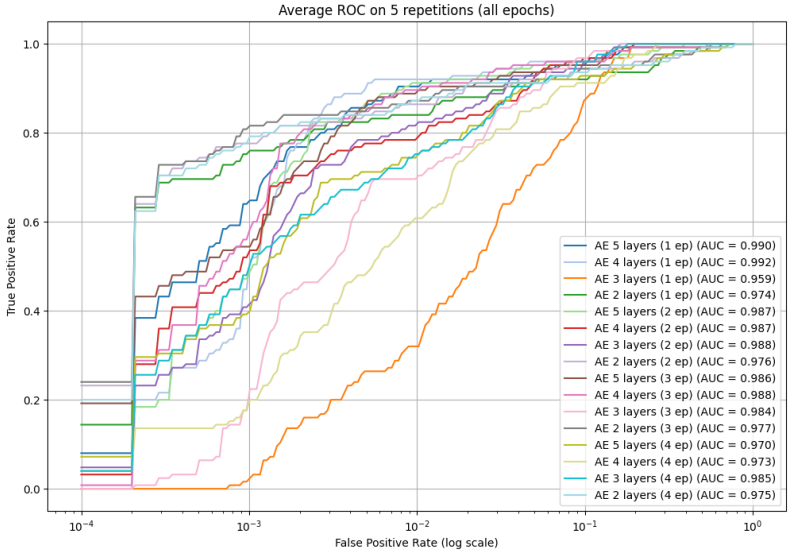


Figure 14: Average ROC curves for different number of layer and epoch under fixed hyperparameters: batch size = 32, latent space size = 4, learning rate=1e-3

As shown in Figure 15, the model with 4 layers and 1 epoch already provides competitive results.

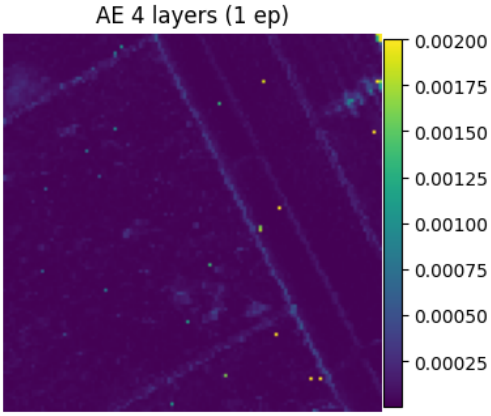


Figure 15: Average distance map: 4 layers, 1 epoch, batch size = 32, latent space size = 4, learning rate=1e-3

We can now compare the performance results depending of the batch size chosen.

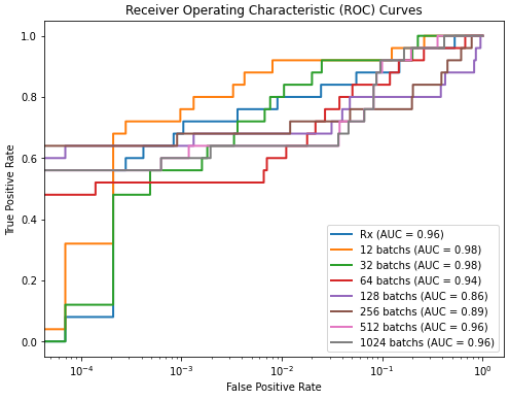
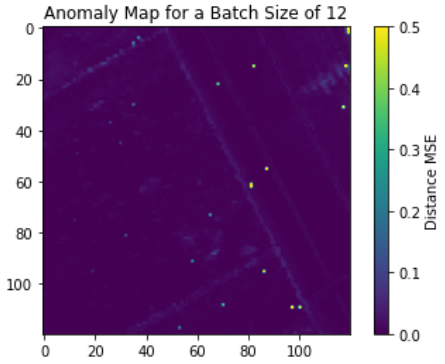


Figure 16: ROC curves for different batch sizes under fixed hyperparameters: 1 epoch, 4 layers, learning rate = 1e-3, latent space size = 4

Figure 16 shows that it is difficult to choose the best batch size as some batch sizes have high AUC but a low curve at the beginning of the graphic. Therefore, we analyse the detection performance and the best visualization of the anomalies on the map in Figure 17. The distance maps for different batch sizes can be found in Appendix B (Figure 28).



(a) batch size = 12

Figure 17: Distance map for a batch size of 12 under fixed hyperparameters: 1 epoch, 2 layers, learning rate = 1e-3, latent space size = 4.

As a result, we decide to choose a batch size of 12.

By analysing Figures 18 and 19, we determine that the most optimal learning rate for anomaly detection on the AVIRIS image is 0.005.

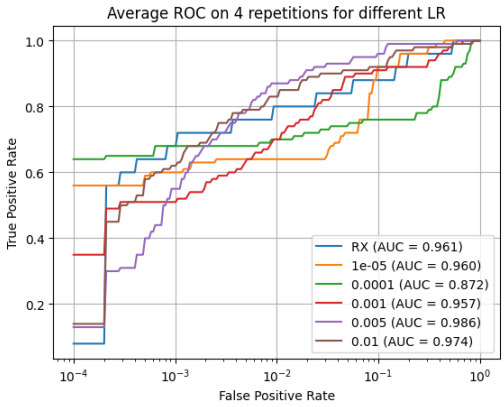
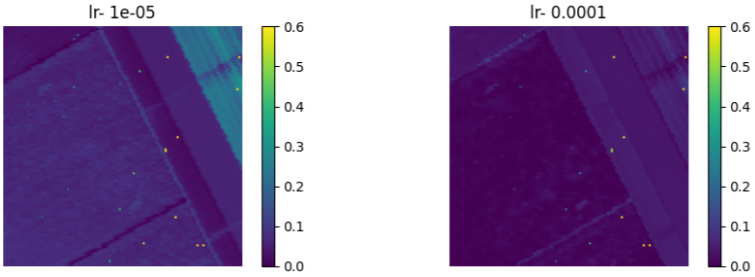
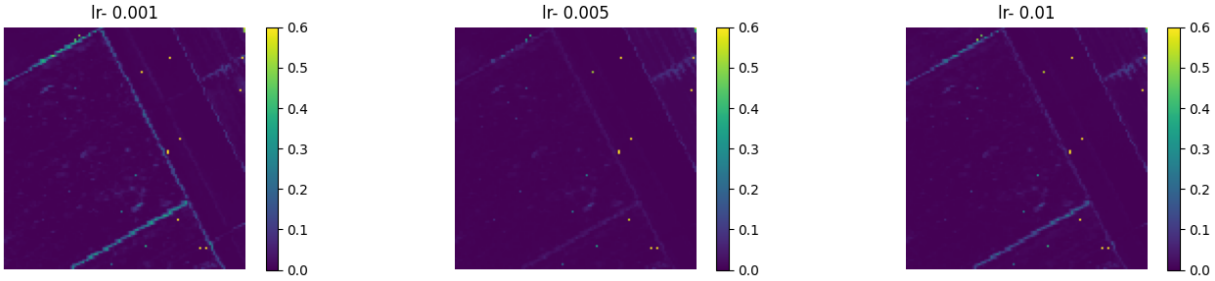


Figure 18: Average ROC curves for different learning rates under fixed hyperparameters: 4 layers, 1 epoch, batch size = 12, latent space size = 4, learning rate=1e-3



(a)



(b)

Figure 19: Average distance maps for different learning rates under fixed hyperparameters: 2 layers, 1 epoch, batch size = 12, latent space size = 4.

The Mean Square Error metric is the most efficient for anomaly detection on AVIRIS image (Figure 20).

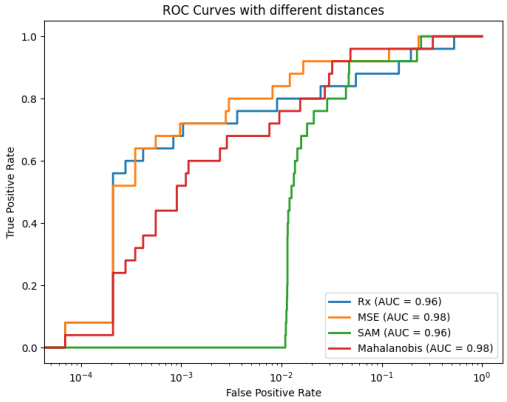


Figure 20: ROC curves for different distance metrics under fixed hyperparameters: 1 epoch, 4 layers, batch size=12, learning rate = 5e-3, latent space size = 4

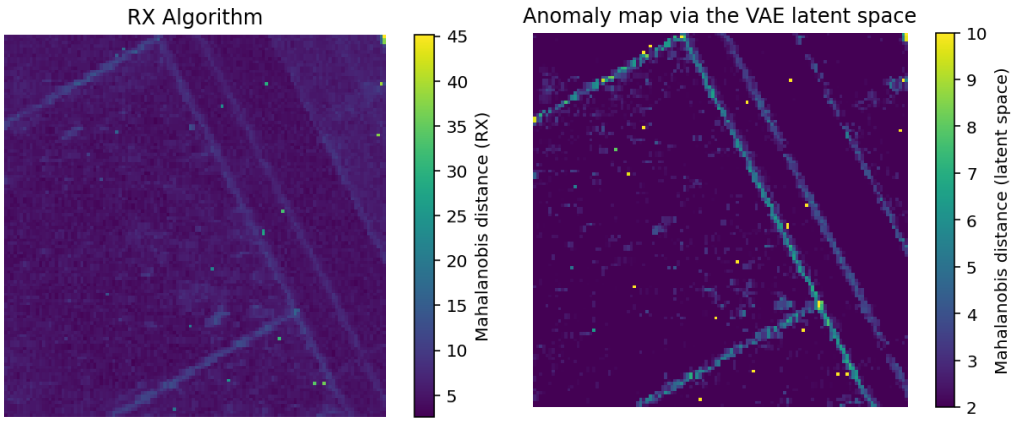
As not all of the anomalies have been detected even using autoencoders, we decided to try different variant called variational autoencoders.

4 Anomaly detection performance of VAE algorithms

As quality results are obtained with the lake shore image, we choose to train our VAE on the AVIRIS image.

4.1 Final results

Figure 21 presents the detection results obtained with the optimal combination of hyperparameters for the VAEs used in this study.



(a) Distance map using RX algorithm

(b) Distance map using VAE under fixed hyperparameters: 5 epochs, batch size of 128, learning rate = 1e-3, $\beta = 5$

Figure 21: Comparaison of distance maps between VAE with optimal hyperparameters and RX

On this image, the VAE clearly outperforms the RX algorithm. This result is discussed at the end of Section 4.2.

4.2 Intermediate results

The VAE is built with one encoder layer, one decoder layer and a latent space of dimension 2. The encoder and decoder are each of size 16. We start by trying to detect the anomalies with the following set of hyperparameters:

- Number of epochs = 1
- Batch size = 32
- Learning rate = 1e-3
- $\beta = 1$

The latent space is visualized in Figure 22.

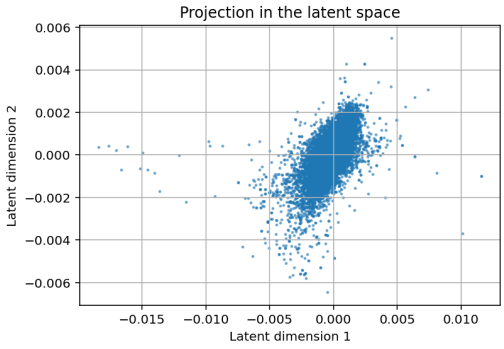
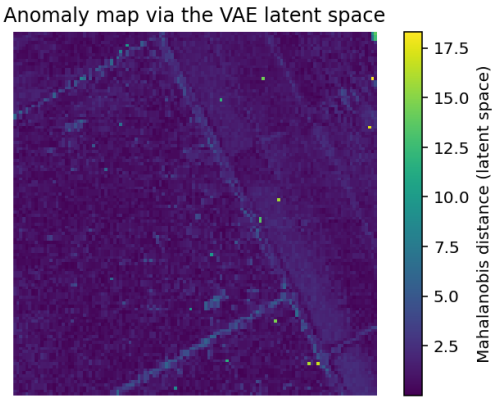


Figure 22: Latent space of dimension 2 for 1 epoch, batch size of 32, learning rate = 1e-3, $\beta = 2$

We clearly see the gaussian distribution of the points, centred in 0. The anomalies that will be detected are the points far from the main cluster.

Anomaly detection works differently with VAEs. We do not use a reconstruction function, as we seek for anomalies directly in the latent space. By using the Mahalanobis distance, we can draw a distance map to compute where the furthest points from the cluster in the latent space are situated in the AVIRIS image. The results are shown in Figure 23.



(a) Distance map using VAE under fixed hyperparameters: 1 epoch, batch size of 32, learning rate = 1e-3, $\beta = 2$

Figure 23: Distance map for latent space of dimension 2, 1 epoch, batch size of 32, learning rate = 1e-3, $\beta = 2$

It seems that the VAE is visually also performant as the RX algorithm, but also induces some noise in the anomaly detection map. ROC curves of each algorithm are compared in Figure 23 to better evaluate the precision of each algorithm.

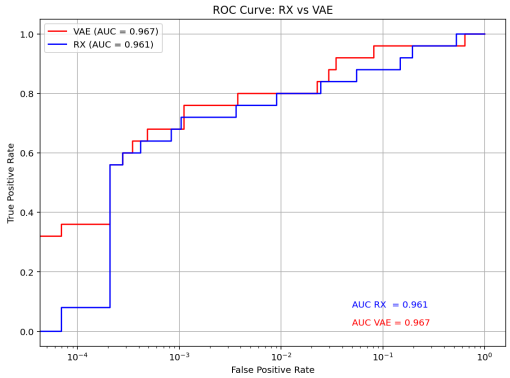


Figure 24: ROC curves for 1 epoch, batch size of 32, learning rate = 1e-3, $\beta = 2$

The curves are similar, even though the VAE one has a larger value at the beginning. The Area Under Curve is slightly larger for VAE than RX (0,967 vs 0,961). As a VAE has an inherently random nature, this process needs to be repeated a sufficient number of times to make sure that the result is consistent. We do not display all the graphs in this paper for the sake of readability, but globally VAE slightly underperform compared to RX on this image with this set of hyperparameters.

The values of the hyperparameters can now be changed in order to find better performances for the VAE. Since the optimization process for the hyperparameters of a classical autoencoder has

already been exposed in Section 3, in this section the tests will only be done for the β parameter. After many tests done on "classical" hyperparameters, the variation of β is evaluated with the following fixed hyperparameters that are optimized for the VAE:

- Number of epochs = 5
- Batch size = 128
- Learning rate = 1e-3
- Loss function = L1 norm

Figure 25 shows the ROC curves for different values of β.

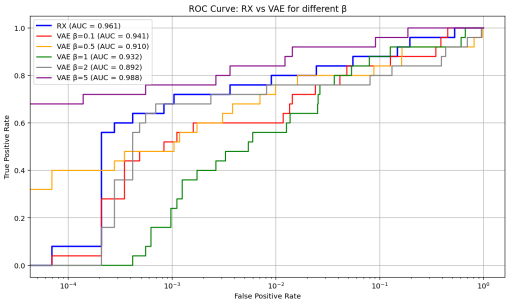


Figure 25: ROC curves for different values of β with 5 epochs, batch size of 128, learning rate = 1e-3

Figure 25 shows that a VAE with β=5 outperforms the RX algorithm. We can visualize the latent space associated with this value of β in Figure 26.

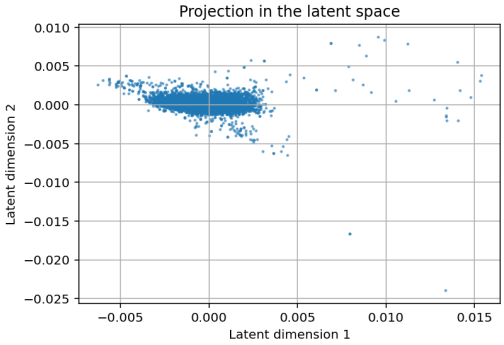


Figure 26: VAE Latent space under fixed hyperparameters: 5 epochs, batch size of 128, learning rate = 1e-3, β = 5

The effect of a strong value of β is directly visible: in the latent space, the point cluster is significantly more compact. Anomaly points can easily be seen on this graph.

It is important to emphasize on the random nature of autoencoders. The results shown in this section were chosen to try to represent at best what happens most of the time, but each test with the same hyperparameters produces a different result. Sometimes, a VAE can underperform compared to the RX algorithm. To be sure to have a trustworthy result, the VAE must be trained multiple times on the same image.

5 Analysis and Conclusion

This study investigated the performance of autoencoders and variational autoencoders for hyperspectral anomaly detection. Our main findings show that, when properly tuned, both AEs and VAEs can have the same performance level as the RX algorithm. In particular, they sometimes demonstrate highly satisfactory results on certain images, confirming their potential for capturing complex spectral structures.

Autoencoders are random by nature. Therefore, the same autoencoder will give different results when given the same image multiple times. This can be helpful compared to the RX algorithm who always returns the same result. Anomalies that are never detected by the RX algorithm have a chance to be found by autoencoders. On the other side, autoenconders may also not detect anomalies that are found each time by the RX algorithm.

Consequently, several limitations emerged. First, both autoencoders and variational autoencoders models exhibit high variability in their outputs, making the training process non-deterministic and harder to reproduce. Second, the selection of hyperparameters proved to be a critical and time-consuming task. The optimal configuration varies significantly from one image to another, which means that these models are far from being plug-and-play solutions.

Furthermore, to evaluate the performance of autoencoders, other benchmarks can be used. In this paper we chose the RX algorithm, but as said in Section 1, many other anomaly detection methods exist, to which we could have compared autoencoders to. The construction of the distance map can also be improved to better fit the random nature of autoencoders, by building a maximum distance map on different trainings for example.

Several questions remain open. While the autoencoder and variational autoencoder models show promise, it is not yet clear how they generalize across different types of hyperspectral scenes. Their sensitivity to initialization and hyperparameter settings still needs to be better understood and controlled.

Future work will also aim to extend this evaluation to more complex scenes, such as ground-level images with mixed backgrounds and multiple types of anomalies. This would help assess the robustness of deep learning methods in more challenging environments.

References

[1] J. B. Diederik P., Kingma, *Adam: A method for stochastic optimization*, 3rd International Conference for Learning Representations, (2014).

[2] M. W. Diederik P., Kingma, *Auto-encoding variational bayes*, ArXiv, (published in 2013, revised in 2022).

[3] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 3rd International Conference for Learning Representations, (2014).

[4] D. Manolakis, E. Truslow, M. Pieper, T. Cooley, and M.Brueggeman, *Detection algorithms in hyperspectral imaging systems: An overview of practical algorithms*, IEEE Signal Process. Mag., 31 (2014), p. 24–33.

[5] F. Moutarde, *Mines paristech lecture notes*, (2021).

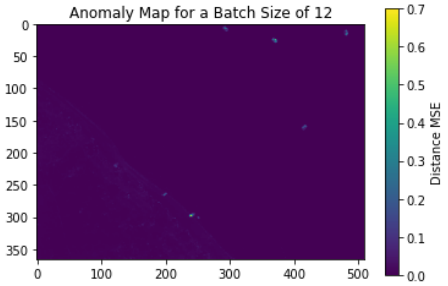
[6] I. S. Reed and X. Yu, *Adaptive multiple-band cfar detection of an optical pattern with unknown spectral distribution*, IEEE Transactions on Acoustics, Speech, and Signal Processing, 38 (1990), pp. 1760–1770.

[7] M. D. S. Matteoli and J. Theiler, *An overview of background modeling for detection of targets and anomalies inhyperspectral remotely sensed imagery*, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens., 7 (2014), p. 2317–2336.

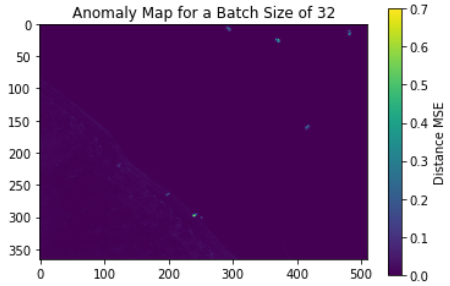
[8] H. Su, Z. Wu, H. Zhang, and Q. Du, *Hyperspectral anomaly detection a survey*, IEEE GEOSCIENCE AND REMOTE SENSING MAGAZINE, (2022).

[9] B. S. L. C. T. L. Zhaomin Chen, Chai Kiat Yeo, *Autoencoder-based network anomaly detection*, IEEE, (2018).

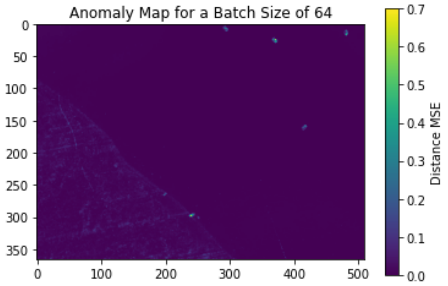
A Additional results for the lake shore image



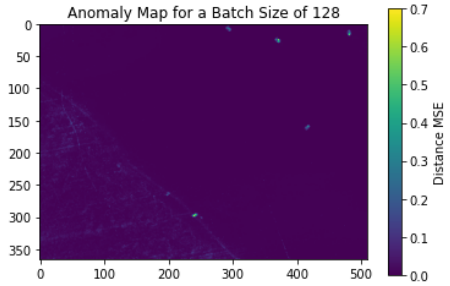
(a) batch size = 12



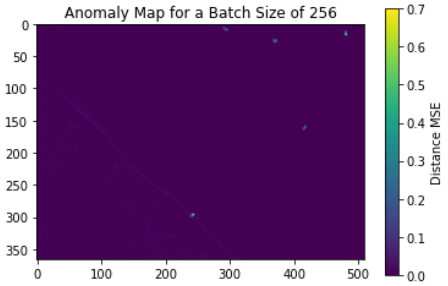
(b) batch size = 32



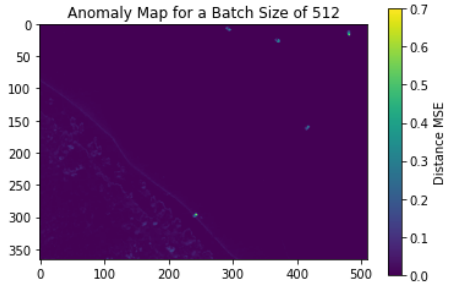
(c) batch size = 64



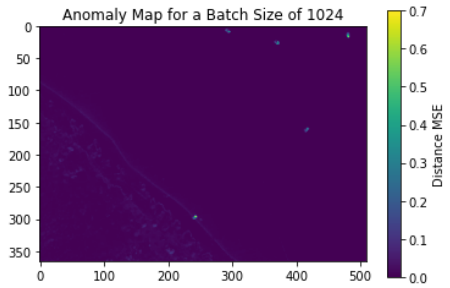
(d) batch size = 128



(e) batch size = 256



(f) batch size = 512



(g) batch size = 1024

Figure 27: The distance maps for different batch sizes under fixed hyperparameters: 1 epoch, 2 layers, learning rate = 1e-3, latent space size = 4.

B Additional results for the aviris image

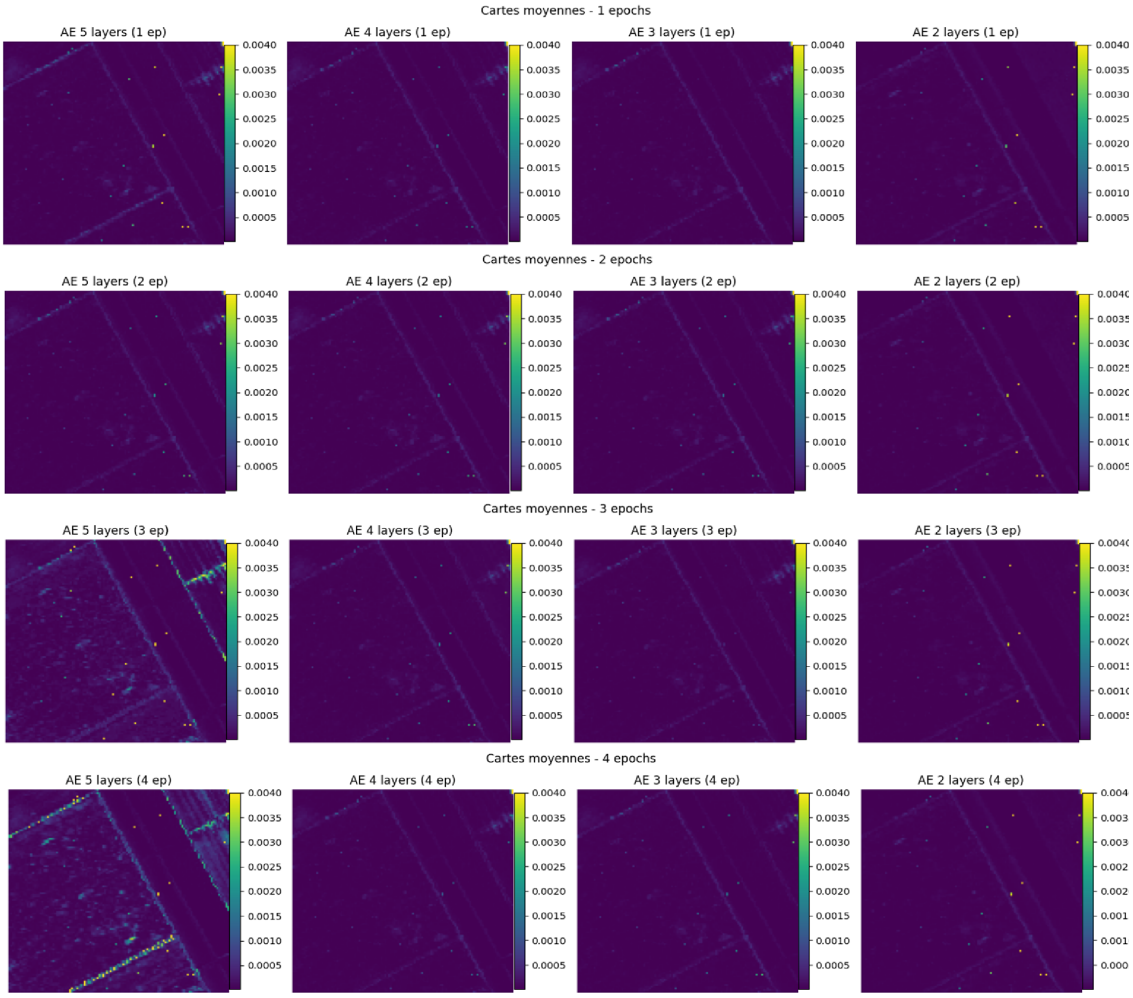
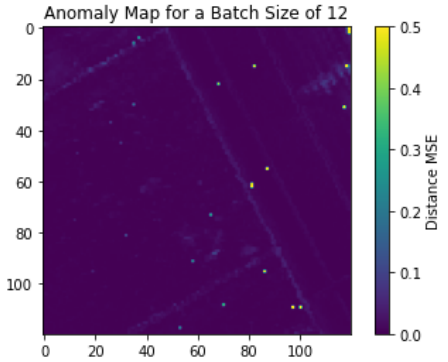
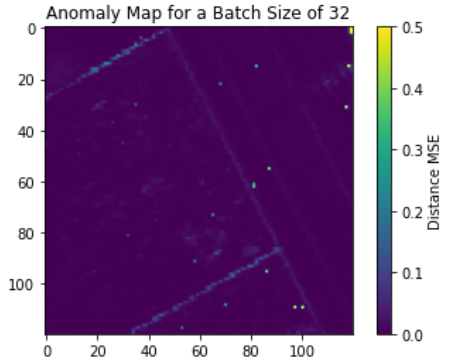


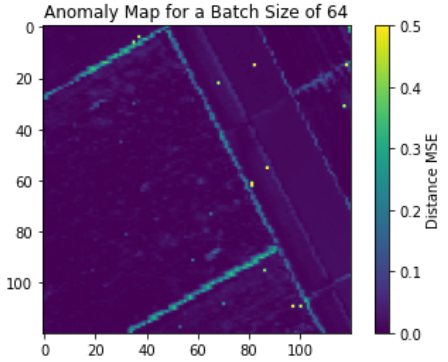
Figure 28: Average distance maps for different number of layers and epochs under fixed hyperparameters: batch size = 32, latent space size = 4, learning rate=1e-3



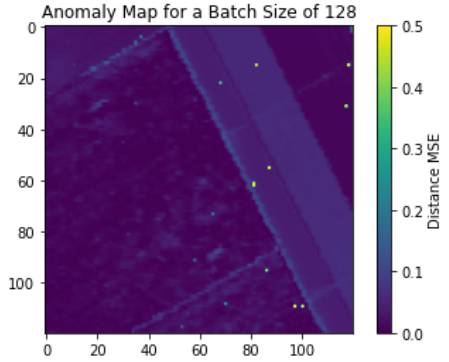
(a) batch size = 12



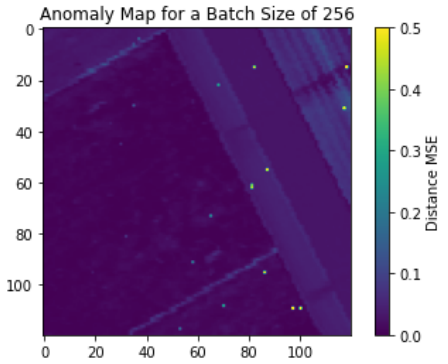
(b) batch size = 32



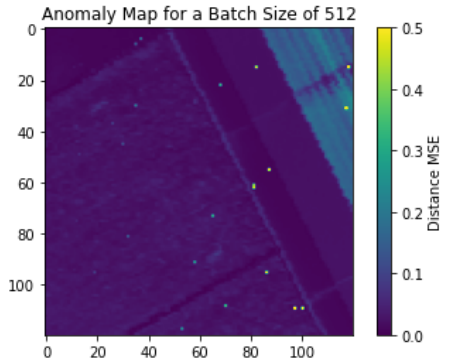
(c) batch size = 64



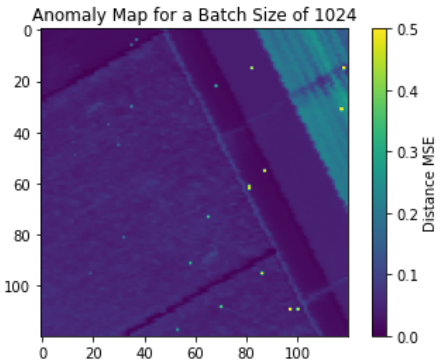
(d) batch size = 128



(e) batch size = 256



(f) batch size = 512



(g) batch size = 1024

Figure 29: The distance maps for different batch sizes under fixed hyperparameters: 1 epoch, 2 layers, learning rate = 1e-3, latent space size = 4.