Stable training of autoencoders for hyperspectral unmixing

Kamil Książek, Przemysław Głomb, Michał Romaszewski, Michał Cholewa, and Bartosz Grabowski

Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, 44-100 Gliwice, Poland
{kksiazek,przemg,mromaszewski, mcholewa, bgrabowski}@iitis.pl

Abstract. Neural networks, autoencoders in particular, are one of the most promising solutions for unmixing hyperspectral data, i.e. reconstructing the spectra of observed substances (endmembers) and their relative mixing fractions (abundances). Unmixing is needed for effective hyperspectral analysis and classification. However, as we show in this paper, the training of autoencoders for unmixing is highly dependent on weights initialisation. Some sets of weights lead to degenerate or low performance solutions, introducing negative bias in expected performance. In this work we present the results of experiments investigating autoencoders’ stability, verifying the dependence of reconstruction error on initial weights and exploring conditions needed for successful optimisation of autoencoder parameters.

Keywords: hyperspectral unmixing · autoencoders · training stability · weights initialisation.

1 Introduction

Hyperspectral imaging (HSI) combines reflectance spectroscopy with image processing – image pixels contain information about hundreds of spectral bands that can characterise chemical composition and properties of visible objects. In HSI the spectra of pixels are often a mixture of different substances [12], as the sensor captures light reflected from nearby objects or aggregated from several sources due to the low spatial resolution. The task of hyperspectral unmixing (HU) is to reconstruct the original spectra of observed substances – endmembers – and their fractional mixture coefficients – abundances. On one hand, HU facilitates further data analysis and improves classification results [8]. On the other hand, correlations between pixels and huge data volume resulting from the fact that every pixel can be treated as an example in a high-dimensional feature space, make neural networks particularly suitable models for HU.

This is why, while a number of machine learning algorithms for HU based on statistical and geometric principles have been developed [2], deep learning models seem to be the most promising solution, with autoencoders (AE) emerging as an architecture of choice. Examples include: a deep AE network [25] which is a sequence of stacked AE followed by a variational AE, generative and encoder models trained using pure pixel information [4] or EndNet architecture [19] with a loss function based
on Kullback-Leibler divergence and the Spectral Angle Distance (SAD). Furthermore, deep convolutional autoencoders are also applied for the HU problem e.g. [22].

One of key elements of training a neural model is a proper initialisation of weights, which prevents the phenomenon of vanishing or exploding gradients. Several weight initialisation methods have been introduced, e.g. in [7] authors derive a solution based on the hypothesis of linear activations while in [10] a method dedicated for rectifier nonlinear functions and deep architectures is proposed. However, while those methods are derived from examining gradient flow principles, they are usually applied without verification of the quality of initial weights. As a result, when training fails there is no way of knowing if the failure results from a combination of training data and hyperparameter choice, or a bad initialisation.

In this work we study the problem of AE training failures, resulting from bad initialisation weights, focusing on the problem of HU. In particular, we present the following contributions:

1. We have experimentally verified the presence of failed trainings of autoencoders in the HU scenarios. We have investigated this effect through \( n = 100000 \) individual autoencoder training sessions across a diverse range of variables, and found that this effect persists across all studied variants of autoencoder architectures, datasets, weight initialisation methods, loss function types, and hyperparameter choices.
2. To our best knowledge, as we were unable to locate a discussion of this phenomenon in the literature, this is the first detailed study of such failures in a standard autoencoder training scenario on a real world hyperspectral dataset. This situation can be a serious problem, as when the final effectiveness of the autoencoder are dependent on the result of a particular initialisation i.e. selection of initial weight values, it would be difficult to select the best parameters for the model.
3. Based on our results, we use the statistical analysis with the Kruskal-Wallis H-test to empirically confirm the thesis that a specific autoencoder initialisation affects the final data reconstruction error from a trained model.
4. To resolve this issue, we propose a simple solution based on retraining, that can improve the reconstruction error in the presence of bad weights.

### 1.1 Related work

An overview of the HU methods can be found in [2]. The approaches range from simple pure-pixel algorithms e.g. Pixel Purity Index (PPI) [3] or N-FINDR [26] to more complex ones e.g. SISAL [1] which work with non-pure pixels and noisy data.

An autoencoder (AE) is a neural network that through hidden layers compresses an input into a lower-dimensional (latent) space and reconstructs back the original data. Reduction of the input dimensionality makes the AE well-suited for HU, as it can be expected that due to spectra correlations, the actual information in HSI data can be compressed into a low-dimensional subspace of the original feature space.

\[ \text{In such case, standard evaluation of hyperparameters – e.g. through Raytune framework – would contaminate the score of hyperparameter values with bias resulting from unknown bad initialisations.} \]
Autoencoders are thus often used as a base for HU algorithms. In [21], authors analysed fully connected AE-based architectures for blind unmixing in an unsupervised setting. The use of AE for unmixing in a nonlinear case was a focus of [28], where authors showed how in certain situation the linearity assumption will not hold. The approach via convolutional AE was tested in [20] and [24] however, both approaches use spectral and spatial features of data at the same time.

The general problem of random weight initialisation leading to inadequate results was observed in [11], [13] and mitigated by the use of a stacked Restricted Boltzmann Machines (RBMs) to determine the initial weights for AE networks. In [7], a weight initialisation scheme was proposed that maintains activation and back-propagated gradients variance as one moves up or down the network.

To alleviate the vanishing gradients when method from [7] is used for ReLu (nonlinear) function, authors of [10] proposed a method based on zero-mean Gaussian distribution with $\sigma$ dependent on the number of pixels and input channels of a given layer. In addition, Parametric Rectified Linear Unit (PReLU) activation function with adaptive coefficients on the negative part of the real axis was tested.

In [27] authors investigated influence of random initialisation and methods from [7] and [10] on DNN learning. Their analysis yields some insights into learning dynamics of DNNs, but it does not indicate a clear winning method for weight initialisation.

Authors of [15] discussed convergence of back-propagation, using several heuristics to support NN construction. In [23] authors proposed to mitigate the unstability of the unmixing algorithm with multiple initialisations. In [25] a scheme was proposed, where the first layer of a neural network determined the initialisation parameters for the unmixing engine, similarly to [9] where a cascade model was proposed. In [19] the EndNet algorithm was paired with VCA (Vertex Component Analysis) filter or FCLS (Fully Constrained Least Squares) for initialisation. The FCLS was also used in [4]. In [18] authors used results of the first run of the network to re-initialise it in order to improve results.

2 Performance investigation of autoencoders in a hyperspectral unmixing problem

2.1 Linear spectral mixing

In this work we use the Linear Mixing Model (LMM) of the pixel spectra, i.e. a $B$-band pixel $x = [x_1, ..., x_B]^T$ is written as a linear combination of $E$ endmembers with the addition of a noise vector:

$$x = \sum_{j=1}^{E} a_j \cdot w_j + n,$$

where $a = [a_1, ..., a_E]^T$ is a vector of abundances, $W = [w_1, ..., w_E]$ is a matrix of endmembers, $W \in \mathbb{R}^{B \times E}$ and $n = [n_1, ..., n_B]^T$ is a noise vector. To preserve
the physical properties of abundances, it is necessary to ensure that the nonnegativity constraint is fulfilled and the sum of all fractional abundances equals to one: 

\[ \forall j \in \{1, ..., E\} \quad a_j \geq 0, \quad \sum_{j=1}^{E} a_j = 1. \quad (2) \]

For \( M \) pixels in the image \( X \in \mathbb{R}^{B \times M} \) the corresponding abundance matrix is denoted as \( A = [a_1, ..., a_M] \), \( A \in \mathbb{R}^{E \times M} \) and \( N = [n_1, ..., n_M] \), \( N \in \mathbb{R}^{B \times M} \) is the noise matrix, where \( n_i \in \mathbb{R}^{B \times 1} \) is a noise vector, \( i \in \{1, ..., M\} \). With that the Equation 1 can be rewritten as \( X = WA + N \). The aim of the HU process is to estimate the endmembers matrix \( W \) and the abundances matrix \( A \) which provide an estimate of the pure spectra of substances and their fractions present in different pixels of the image.

### 2.2 Architectures of autoencoders for hyperspectral unmixing

We focus on the architecture presented in [21] (see Figure 1). Its encoder part consists of multiple linear layers that transform the input data into a latent space, according to the pattern: \( \text{Enc}(X_b) : \mathbb{R}^{B \times b_s} \rightarrow \mathbb{R}^{E \times b_s} \), where \( X_b \in \mathbb{R}^{B \times b_s} \) is a batch of input data, \( b_s \) is a batch size. Then, in the decoder part, input spectra are reconstructed based on a latent representation: \( \text{Dec}(\text{Enc}(X_b)) : \mathbb{R}^{E \times b_s} \rightarrow \mathbb{R}^{B \times b_s} \).

We denote the reconstructed image by \( \hat{X} = \text{Dec}(\text{Enc}(X)) \). The goal of the autoencoder is to minimise difference between \( X \) and \( \hat{X} \). The design of this architecture allows it to be used for HU; as a decoder has only one layer, its neurons’ activations on the last encoder layer can be treated as abundance vectors while weights connecting the encoder part with the output of the autoencoder can be considered as endmembers.

The authors of [21] studied several architectures with different number of layers and activation functions. We focus on the one that achieved one of the highest efficiencies, i.e. a version with sigmoid activation function. This architecture consists of four linear layers in the encoder part, having \( 9E, 6E, 3E \) and \( E \) neurons, respectively. After that, a batch normalisation (BN) layer is applied. Then, a dynamical soft thresholding (ST) is used which can be written as

\[ x_{ST}^i = \max(0, x_{BN}^i - \alpha) \]

where \( i \in \{1, ..., b_s\} \), \( X_{ST} = [x_{ST}^1, ..., x_{ST}^{b_s}] \), \( X_{BN} = [x_{BN}^1, ..., x_{BN}^{b_s}] \) are matrices with all batch pixels after soft thresholding or batch normalisation, respectively; \( 0 \) is a zero vector and \( \alpha \) is a vector of trainable parameters. Then, to ensure that the sum to one constraint from Equation 2 is met, each pixel vector is normalised i.e. \( \forall i \in \{1, ..., b_s\} \quad x_{norm}^i = x_{ST}^i / \left( \sum_{k=1}^{E} x_{ST}^{i,k} \right) \),

where \( x_{norm}^i \) is the \( i \)-th batch vector after normalisation, \( x_{ST}^{i,k} \) is the \( k \)-th coordinate of the \( i \)-th vector. Gaussian Dropout (GD) is applied as a last encoder layer, but only during training. Finally, the single decoder layer reconstructs the signal from the latent space to the input space. We denote this architecture as the \textit{original}.

We have also prepared a simplified version of the described architecture, denoted \textit{basic}, where an encoder has two linear layers in which ReLU activation function is used. The number of neurons in the first hidden layer is a hyperparameter and is equal to \( n_1 E \), where \( E \) is the number of endmembers, while the second hidden layer ending
the encoder part has $E$ neurons. There are no BN, ST or GD layers. A normalisation layer is left to ensure that the sum of fractional abundances per each pixel is equal to one. A decoder part has one linear layer.

2.3 Performance evaluation

Datasets We used two well-known datasets for HU: Samson and Jasper Ridge [29].

Samson is an image with dimensions $95 \times 95 \times 156$, spectral range of $401 - 889$ nm and spectral resolution $\sim 3.13$ nm. Pixels spectra are a mixture of three endmembers: water, trees and soil.

Jasper Ridge is an image with dimensions $100 \times 100 \times 198$. Originally, there were 224 bands covering the range of $380 - 2500$ nm but bands $1 - 3$, $108 - 112$, $154 - 166$ and $220 - 224$ were removed. Endmembers represent: trees, water, soil and road.

Experiment The goal of the experiment is to confirm the impact of weight initialisation on the final reconstruction error of the AE. For each of $N = 50$ randomly initialised AE, we have performed $k = 50$ separate training sessions for each AE initialisation with the same dataset and hyperparameter values, resulting in $n_a = 2500$ trained models. This has been repeated across ten different hyperparameter sets (see Table 1) and four methods of weight initialisation, bringing the total number of models to $n_b = 100000$.

Each model is evaluated as follows. Let $D$ be a HSI image and let $V_{GT}$ be a set of correct endmembers for image $D$. Let also $D_{GT}$ be ground truth for fractional abundances, i.e. correct abundances for endmembers $V_{GT}$. The AE $A_{i,j}, i = 1, \ldots, N, j = 1, \ldots, k$ is trained with a set of vectors from dataset $D$. The endmembers $V_{i,j}$ and abundance vectors are then extracted; the endmembers set $V_{i,j}$ is matched to $V_{GT}$. The match used is the permutation of endmembers with smallest distance to $V_{GT}$. Abundance vectors are compared to ground truth image $D_{GT}$ to calculate error $E_{a_{i,j}}$ in terms of RMSE, like in [25], while endmembers are compared to their counterparts from $V_{GT}$ and the value of $E_{e_{i,j}}$ is calculated according to the SAD function. For a given experiment mean endmembers error is calculated, as follows: $E_{e} = \sum_{i=1}^{N} \sum_{j=1}^{k} \frac{E_{e_{i,j}}}{N_k}$. Mean abundances error $E_{a}$ is calculated analogously.
Table 1: The detailed description of performed experiments with all hyperparameters used. An encoder means the number of neurons on the first hidden layer and it concerns only the basic architecture. Gaussian Dropout (GD) is applied only in the case of the original architecture.

| experiment ID | architecture | origin | loss | dataset | encoder | batch size | learning rate | GD |
|---------------|--------------|--------|------|---------|---------|-------------|--------------|----|
| 1             | original     | RayTune | MSE  | Samson  | 100     | 0.01        | 0            |    |
| 2             | original     | RayTune | SAD  | Samson  | 100     | 0.01        | 0            |    |
| 3             | original     | article | SAD  | Samson  | 20      | 0.01        | 0.1          |    |
| 4             | basic        | RayTune | MSE  | Samson  | 10E     | 4           | 0.0001       |    |
| 5             | basic        | RayTune | SAD  | Samson  | 20E     | 4           | 0.0001       |    |
| 6             | original     | RayTune | MSE  | Jasper Ridge | 100 | 0.01 | 0.        |    |
| 7             | original     | RayTune | SAD  | Jasper Ridge | 100 | 0.01 | 0.        |    |
| 8             | original     | article | MSE  | Jasper Ridge | 5   | 0.01 | 0.1      |    |
| 9             | original     | article | SAD  | Jasper Ridge | 5   | 0.01 | 0.1      |    |
| 10            | basic        | RayTune | MSE  | Jasper Ridge | 10E | 20  | 0.001    |    |

2.4 Parameters

We have explored the effect of weight initialisation with a range of hyperparameters (see Table 1). We have used: two architectures (original and basic), two datasets, two loss functions (MSE and SAD). The rest of hyperparameters (e.g. learning rate, batch size, GD parameter etc.) have been tuned using RayTune optimisation library [17]. For the comparison, we also included hyperparameters indicated by the authors of the original architecture [21]. In all experiments we have used the Adam optimiser. Additionally, we have used four initialisation algorithms from [10] and [7], each with uniform and normal distribution. The source code necessary for replication of experiments is publicly available on the github repository

2.5 Statistical verification

To confirm a non-uniform behaviour of weights (the existence of initialisations resulting in high reconstruction error) we have used the Kruskal-Wallis H-test for a one-way analysis of variance [14]. We have not used the ANOVA parametric test since the assumption of variance equality among the populations was not met according to results of the Levene’s test [16].

The test is performed as follows: our models are treated as $N$ different populations where every population corresponds to a single set of initial weights and samples correspond to error estimates of consecutive training runs. The hypothesis of a H-test are as follows:

$H_0$: All population means are equal, i.e. $\mu_{A_1} = \mu_{A_2} = \ldots = \mu_{A_N}$.

$H_1$: At least one population has a statistically significantly different mean than the others.

2 With [10] method using uniform initialisation we have used the version from the PyTorch library, which differs from the paper with: 1) biases are not initialised to 0; 2) bounds of a uniform distribution are constant and not dependent on the number of connections.

3 The source code is available at the following link: [https://github.com/iitis/AutoencoderTestingEnvironment](https://github.com/iitis/AutoencoderTestingEnvironment)
In all but one experiment

Results of experimental evaluation of our thesis are presented in Table 2 as Kruskal-

3 Results

p

n

t

value of data reconstruction error. This leads to the question of how many times does

different random initialisations to choose the best network based e.g. on the lowest

2.6 Minimisation of reconstruction error

Since rejection of the null hypothesis does not give an answer which population

differs from others, a post-hoc analysis is performed using the Conover-Iman test [5]

Table 2: Results of experiments for different weight initialisation methods: He [10] with

normal / uniform distribution and Glorot [7] with normal / uniform distribution. For

each experiment a H-test statistic and a logarithm of p-value are presented. The signifi-

cance level α is equal to 0.05. The ‘ph’ column corresponds to the ratio of p-values < α

in post-hoc analysis. Bold font indicates experiment with p-value > α. In the case of

very small p-values, log p-values are denoted ‘− inf’.

| exp. ID | H-stat log p-val ph | H-stat log p-val ph | H-stat log p-val ph | H-stat log p-val ph |
|--------|---------------------|---------------------|---------------------|---------------------|
| 1      | 200.9 −45.07 0.65 | 243.2 −61.76 0.70 | 78.8 −5.41 0.55 | 70.4 −3.72 0.54 |
| 2      | 891.2 −355.38 0.84 | 443.2 −147.76 0.77 | 625.9 −231.04 0.81 | 660.3 −246.96 0.82 |
| 3      | 763.8 −295.33 0.83 | 267.8 −71.82 0.69 | 239.1 −60.11 0.69 | 180.9 −57.49 0.66 |
| 4      | 2185.8 −inf 0.93 | 2025.3 −inf 0.85 | 1997.6 −inf 0.87 | 2141.5 −inf 0.87 |
| 5      | 1954.3 −inf 0.93 | 2093.4 −inf 0.94 | 1840.8 −inf 0.92 | 1777.2 −inf 0.91 |
| 6      | 134.0 −20.95 0.61 | 75.8 −4.79 0.55 | 76.3 −4.90 0.55 | 98.8 −10.32 0.57 |
| 7      | 903.8 −361.36 0.85 | 761.1 −294.07 0.82 | 953.8 −385.10 0.84 | 871.0 −345.85 0.83 |
| 8      | 77.3 −5.69 0.55 | 75.4 −4.71 0.54 | 78.4 −5.33 0.55 | 93.3 −8.88 0.57 |
| 9      | 69.2 −3.50 0.54 | 66.2 −2.98 − | 74.2 −4.46 0.54 | 47.9 −0.65 − |
| 10     | 1767.3 −inf 0.91 | 2041.9 −inf 0.90 | 1344.6 −572.45 0.84 | 1155.1 −481.29 0.85 |

By performing the pairwise comparison for all population pairs, we can conclude which differences between populations are statistically significant.

2.6 Minimisation of reconstruction error

When unfortunate initial weights of a neural network may result in its low effectiveness

after training, a simple way to deal with this is to repeat the training several times for

different random initialisations to choose the best network based e.g. on the lowest

value of data reconstruction error. This leads to the question of how many times does

the experiment need to be repeated.

Given some desired, low value of the reconstruction error t, we can treat a single

initialisation of the AE as a random variable X with Bernoulli distribution, where the

probability P_{eq} = P(X = 1) = p is the probability that the reconstruction error of the trained

network is lower than the value t. The probability of obtaining at least one success in

n trials is P(X ≥ 1) = 1 - (1 - p)^n. What follows, given the required value of probability p_{req} = P(X ≥ 1) (e.g. p_{req} = 0.95), the required number of trials n_{req} can

be expressed as n_{req} = \left\lfloor \frac{\log(1-p_{req})}{\log(1-p)} \right\rfloor. The probability p can be estimated empirically, based on results of experiments.

3 Results

Results of experimental evaluation of our thesis are presented in Table 2 as Kruskal-

Wallis H-test statistics and corresponding p-values for the significance level α = 0.05.

In all but one experiment H_0 was rejected due to p-values lower than 0.05. An alterna-
tive hypothesis H_1 states that at least one initialisation of initial weights resulted in the
value of reconstruction error that was significantly different from the rest. This confirms
that the reconstruction error of the trained network depends on weight initialisation.

To compare individual initialisations, a post-hoc analysis with the Conover-Iman
test was performed. Example results are presented in Figure 2 where heat-maps where each
cell represents the statistical significance of difference between RMSE of a pair of ex-
periments. NS denotes a statistically insignificant difference. An analysis of these ma-
trices reveals that for some experiments e.g. the Experiment 1, there exists a subset of
outlying initialisations while e.g. for the Experiment 4 the majority of initialisation
pairs are significantly different. A summary of post-hoc analysis is presented in Table 2.
Values in ‘ph’ columns are ratios of statistically significant ($p < 0.05$) differences be-
tween individual experiments in accordance to the Conover-Iman test. Results indicate
that using KHN initialisation slightly increased the number of outlying initialisations.

3.1 Discussion

We observed that optimisation using MSE as a loss function also minimises the recon-
struction error in terms of SAD function, to some extent. This relationship is particularly
evident in the case of basic architecture. Indeed, if the value of MSE is close to 0, then
also the SAD has to be close to 0. However, the opposite is not true, because the opti-
misation using SAD function does not reduce error in the sense of MSE. Moreover,
in most cases, after training with SAD function, all or almost all reconstructed points
are outside of the simplex designated by endmembers. This phenomenon occurs be-
cause SAD function is scale invariant which means that only the spectral angle between
input and output points is minimised. It is not necessarily related to reduction of the
Euclidean distance between spectra. Despite this, abundances error in terms of RMSE
and endmembers SAD error are comparable or even lower than when training with the
MSE function which can be seen in the detailed results in Table 3.
have started while for better performing model gradients decreased gradually. For less efficient model, gradients vanished on two encoder layers shortly after the training.

The graphs show gradient values for two hidden encoder layers. First, Figure 4 presents mean gradient values for first 1000 training iterations of the two exemplar AE models. A problem with the gradient flow during backpropagation steps emerged. Figure 3 presents mean gradient values for basic architecture, on the Samson dataset and MSE loss function. We found out that for some basic reconstruction error, we examined models generated during the Experiment 4, using uniform distribution. Bold font indicates experiment with the lowest value of error.

Regarding the loss function, unmixing results for corresponding pairs of experiments, i.e. experiments with the same architecture and on the same dataset but with different loss functions are comparable. For pairs 1 – 2, 6 – 7 and 8 – 9, and for all weight initialisation methods, both abundances and endmembers errors were slightly smaller when the autoencoder was trained using the SAD loss, compared to the MSE. The lowest average error values are usually achieved for XGU initialisation. Furthermore, for all experiments, this weight initialisation method led to lower mean endmembers reconstruction error than in the case of KHN initialisation. Overall, Glorot initialisation technique seems to be on average better than He methods, when considering the abundances / endmembers error values.

### 3.2 Stabilisation of autoencoder training

The analysis of the conducted experiments reveals a clear relationship between the value of reconstruction error and initialisation. The results for the basic architecture form clear groups of low or high scores. In the histogram (Fig. 3a) it corresponds to three characteristic ‘spikes’. Results for architecture original (in Fig. 3b) do not have distinct groups and the number of results is inversely, exponentially proportional to the reconstruction error. Despite its simplicity, in most of the experiments, architecture basic achieved a lower minimum reconstruction error than architecture original.

To investigate the cause of the differences between various models in terms of the reconstruction error, we examined models generated during the Experiment 4, using basic architecture, Samson dataset and MSE loss function. We found out that for some models a problem with the gradient flow during backpropagation steps emerged. Figure 4 presents mean gradient values for first 1000 training iterations of the two exemplary AE models. The graphs show gradient values for two hidden encoder layers. First two columns illustrates the gradient flow for one of the most efficient models in terms of RMSE (in the end – 0.0067) while two last columns are connected with one of the least effective models achieving RMSE of 0.1244. We observed that in the case of the less efficient model, gradients vanished on two encoder layers shortly after the training have started while for better performing model gradients decreased gradually.
Fig. 3: Plots (a), (b): example histograms of RMSE for two architectures (Experiments 1 and 4). Histograms present results of all repetitions and initialisations in the logarithmic scale. Plots (c), (d): Number of independent Bernoulli trials required to reach at least the given value of the RMSE in Experiments 1 and 4 with 95% confidence.

Fig. 4: Mean gradient values with standard deviations for first two layers of the basic autoencoder, trained in the Experiment 4, using KHU initialisation. Model (a) achieved RMSE of 0.0067 while model (b) achieved RSME equal to 0.1244.

The heuristic described in Sec. 2.6 can be used to partially avoid the problem of weak network initialisations. An example of the number of repetitions, calculated from the Eq. (2.6), is shown in Fig 3. Since the probability of achieving the desired reconstruction error was calculated based on the analysis of the experimental results, there is a correlation between the distributions of errors and the number of trials in Fig 3. For most experiments, XGN initialisation gave the smallest number of trials required to obtain a sufficiently low error.

Table 4 shows the number of trials required to achieve a low reconstruction error with 95% confidence. Threshold values were determined subjectively, with the lowest being approximately the 10th percentile of all scores. The results are presented for the average RMSE for experiments with MSE loss function and the average SAD error for experiments with SAD loss function. Based on our results, 6 − 10 trials seem like a reasonable rule of thumb to achieve a low reconstruction error for a given experiment.

4 Conclusions

Parameter optimisation is crucial for effectiveness of a machine learning model. Typically, optimisation is performed by evaluation of models trained with different parameter values on some validation set. This approach assumes that the training processes are stable and produce convergent results. In the case of random appearance of undertrained models there is a significant risk of bias in optimised parameter values.
Table 4: Number of independent Bernoulli trials to reach the value of RMSE/SAD reconstruction error given by the threshold \( t \) with 95\% confidence. Presented results are for experiments with XGN initialisation and parameters optimised with RayTune.

| Dataset     | Architecture | No. trials (95\% confidence) | Loss: MSE / Error: RMSE | Loss: SAD / Error: SAD |
|-------------|--------------|------------------------------|--------------------------|------------------------|
|             |              |                              | \( t = 0.01 \)           | \( t = 0.015 \)        |
| Samson      | basic        | 6                            | 6                        | 2                      |
| Samson      | original     | 106                          | 10                       | 1                      |
| Jasper      | basic        | 69                           | 4                        | 2                      |
| Jasper      | original     | -                            | 126                      | -                      |

We have explored this phenomenon for the case of HU using autoencoders. We have observed cases of the vanishing gradient in first AE layers and confirmed that weight initialisation has a crucial impact on final AE performance. A weak initialisation leads to high reconstruction or endmembers errors, despite proper values of hyperparameters selected e.g. by RayTune \([17]\). The problem was observed under a range of hyperparameter values, datasets, architectures, and initialisation methods. The phenomenon was confirmed by statistical verification, based on a large set of training experiments. Finally, we have presented estimations of the number of training repeats to achieve confidence in results of the training. Our results are possibly applicable beyond the HU problem, into all domains where similar autoencoder architectures are used.

References

1. Bioucas-Dias, J.M.: A variable splitting augmented lagrangian approach to linear spectral unmixing. In: 2009 First Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing. pp. 1–4 (2009)
2. Bioucas-Dias, J.M., Plaza, A., Dobigeon, N., Parente, M., Du, Q., Gader, P., Chanussot, J.: Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 5(2), 354–379 (2012)
3. Boardman, J., Kruse, F.A., Green, R.: Mapping target signatures via partial unmixing of AVIRIS data. In: Summaries of the Fifth Annual JPL Airborne Earth Science Workshop. Volume 1: AVIRIS Workshop (1995)
4. Borsoi, R.A., Imbiriba, T., Bermudez, J.C.M.: Deep generative endmember modeling: An application to unsupervised spectral unmixing. IEEE Transactions on Computational Imaging 6, 374–384 (2020)
5. Conover, W.J.: Practical nonparametric statistics, Third Edition, vol. 350. John Wiley & Sons (1998)
6. Conover, W.J., Iman, R.L.: On multiple-comparisons procedures (1979)
7. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y.W., Titterington, M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 9, pp. 249–256. JMLR Workshop and Conference Proceedings, Chia Laguna Resort, Sardinia, Italy (13–15 May 2010)
8. Guo, A.J., Zhu, F.: Improving deep hyperspectral image classification performance with spectral unmixing. Signal Processing 183, 107949 (2021)
9. Guo, R., Wang, W., Qi, H.: Hyperspectral image unmixing using autoencoder cascade. In: 2015 7th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS). pp. 1–4 (2015)

10. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: IEEE International Conference on Computer Vision (ICCV). pp. 1026–1034 (2015)

11. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (2006)

12. Keshava, N., Mustard, J.F.: Spectral unmixing. IEEE Signal Processing Magazine 19(1), 44–57 (2002)

13. Krizhevsky, A., Hinton, G.E.: Using very deep autoencoders for content-based image retrieval. In: ESANN. vol. 1, p. 2. Citeseer (2011)

14. Kruskal, W.H., Wallis, W.A.: Use of ranks in one-criterion variance analysis. Journal of the American Statistical Association 47(260), 583–621 (1952)

15. Lecun, Y., Bottou, L., Orr, G., Müller, K.R.: Efficient backprop (08 2000)

16. Levene, H.: Robust tests for equality of variances. Contributions to probability and statistics. Essays in honor of Harold Hotelling pp. 279–292 (1961)

17. Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J.E., Stoica, I.: Tune: A research platform for distributed model selection and training. arXiv preprint arXiv:1807.05118 (2018)

18. Lv, J., Shao, X., Xing, J., Cheng, C., Zhou, X.: A deep regression architecture with two-stage re-initialization for high performance facial landmark detection. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3691–3700 (2017)

19. Ozkan, S., Kaya, B., Akar, G.B.: Endnet: Sparse autoencoder network for endmember extraction and hyperspectral unmixing. IEEE Transactions on Geoscience and Remote Sensing 57(1), 482–496 (2019)

20. Palsson, B., Ulfarsson, M.O., Sveinsson, J.R.: Convolutional autoencoder for spatial-spectral hyperspectral unmixing. In: IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium. pp. 357–360 (2019)

21. Palsson, B., Sigurdsson, J., Sveinsson, J.R., Ulfarsson, M.O.: Hyperspectral unmixing using a neural network autoencoder. IEEE Access 6, 25646–25656 (2018)

22. Palsson, B., Ulfarsson, M.O., Sveinsson, J.R.: Convolutional autoencoder for spectral–spatial hyperspectral unmixing. IEEE Transactions on Geoscience and Remote Sensing 59(1), 535–549 (2021)

23. Plaza, A., Chang, C.: Impact of initialization on design of endmember extraction algorithms. IEEE Transactions on Geoscience and Remote Sensing 44(11), 3397–3407 (2006)

24. Ranasinghe, Y., Herath, S., Weerasooriya, K., Ekanayake, M., Godaliyadda, R., Ekanayake, P., Herath, V.: Convolutional autoencoder for blind hyperspectral image unmixing (2020)

25. Su, Y., Li, J., Plaza, A., Marinoni, A., Gamba, P., Chakravortty, S.: Daen: Deep autoencoder networks for hyperspectral unmixing. IEEE Transactions on Geoscience and Remote Sensing 57(7), 4309–4321 (2019)

26. Winter, M.E.: N-FINDR: an algorithm for fast autonomous spectral end-member determination in hyperspectral data. In: Descour, M.R., Shen, S.S. (eds.) Imaging Spectrometry V. vol. 3753, pp. 266 – 275. International Society for Optics and Photonics, SPIE (1999)

27. Yun, S., Wong, A.: Where should we begin? a low-level exploration of weight initialization impact on quantized behaviour of deep neural networks. In: 6th Annual Conference on Computer Vision and Intelligent Systems (CVIS) (2020)

28. Zhao, M., Wang, M., Chen, J., Rahardja, S.: Hyperspectral unmixing via deep autoencoder networks for a generalized linear-mixture/nonlinear-fluctuation model (2019)

29. Zhu, F.: Hyperspectral unmixing: Ground truth labeling, datasets, benchmark performances and survey (2017)