Preprocessing Ground-Based Hyperspectral Image Data for Improving CNN-based Classification

Andreas Schliebitz[1], Heiko Tapken[1] and Martin Atzmueller[2,3]

[1]*Osnabrück University of Applied Sciences, Albrechtstr. 30, 49076 Osnabrück, Germany*

[2]*Osnabrück University, Semantic Information Systems Group, Wachsbleiche 27, 49090 Osnabrück, Germany*

[3]*German Research Center for Artificial Intelligence (DFKI), Hamburger Str. 24, 49084 Osnabrück, Germany*

Abstract

Complex data – like hyperspectral image data – requires adequate preprocessing methods for tackling the issue of data quality, as a prerequisite for further machine learning approaches like deep learning. This paper addresses preprocessing in the context of ground-based hyperspectral image data: It presents novel preprocessing methods, and proposes a comprehensive preprocessing pipeline for handling complex hyperspectral image samples. Multiple preprocessing pipelines are applied on a set of hyperspectral images in the context of image classification, analyzing which preprocessing algorithms perform best, in order to draw further conclusions about methods and their combinations in our application context. Our results show trends on the application of specific methods, and indicate that the application of shorter pipelines tends to achieve better results. We also provide empirical evidence suggesting that too intensive dimensionality reduction can have detrimental effects on classifiability, regardless of contamination levels.

Keywords

hyperspectral image analysis, data preprocessing, artificial neural networks, image classification

1. Introduction

In recent years, deep learning-based techniques have advanced significantly, in particular, in the area of computer vision, e. g., [1, 2, 3, 4]. However, one important prerequisite for their successful application is sufficient data quality [5, 6], requiring appropriate preprocessing: in particular, preprocessing needs to be applied e. g., when working with complex data like hyperspectral images [7, 8, 9] or performing advanced sampling approaches for quality estimation, cf. [10, 11].

In this paper, we target preprocessing methods in the context of analyzing hyperspectral image data: We investigate several preprocessing algorithms based on a literature review, provide efficient Python implementations (detached from data acquisition), and combine them according to the recommendations of Vidal et al. [11] into comprehensive preprocessing pipelines. These are applied to artificially contaminated hyperspectral images of potatoes. A hyperspectral image is a data cube consisting of many two-dimensional spatial bands stacked along the spectral dimension. Subsequently, the cleaned datasets are classified using an artificial neural network to assess and identify promising combinations of preprocessing methods [12, 13, 14].

In particular, our application context is given by an agricultural application for visual-based automated quality determination of potatoes [14], which is being developed in the Agri-Gaia research project by Osnabrück University of Applied Sciences in cooperation with Wernsing Feinkost GmbH. It addresses both the use of RGB as well as hyperspectral data described in this paper. Real-time capable RGB cameras are only suitable for determining defects visible on a potato's surface. Therefore, the quality of dirty potatoes cannot be assessed using this approach.

Due to the different penetration depths of different wavelengths of the non-visible electro-magnetic spectrum, it is suspected that hyperspectral cameras can detect defects beneath a thin layer of soil [10]. Hence, hyperspectral data can be a valuable provider of nontrivial features, which may be extracted using convolutional neural networks (CNNs). However, due to multiple contamination hazards typically encountered during data collection, hyperspectral preprocessing is often necessary for assuring data quality. Our core contributions are summarized as follows:

1. We discuss preprocessing, and propose two novel methods, i. e., spectral binning using self-reference deviation (SRD) and spatial binning via a SID-SAM sliding window (SSSW).
2. In our experimentation, we construct different preprocessing pipelines and estimate their performance on a reference dataset in the described real-world context.
3. We provide an open-source implementation for creating preprocessing pipelines in Python using the proposed methods: https://github.com/andreas-schliebitz/hipp

The rest of the paper is structured as follows: Section 2 discusses related work. After that, Section 3 describes our proposed approach in detail. Next, Section 4 presents and discusses our results. Finally, Section 5 concludes with a summary and interesting directions for future work.

2. Related Work

Vidal et al. [11] discuss preprocessing steps for hyperspectral data, as well as individual algorithms within the preprocessing pipeline shown in Figure 1:

- Dead pixels can be detected using thresholding techniques relying on median spectra or more robust methods like genetic or evolutionary algorithms [11, p. 143]. In addition to more sophisticated methods like Minimum Volume Ellipsoid (MVE), defect pixels can also be localized by their abnormal intensity values [11, p. 143].
- Spectral spike points may be removed using median-modified Wiener filters (MMWF), wavelet transforms (Wt), signal derivatives or statistical methods detecting unlikely deviations from the signal's mean [11, p. 144].
- Background removal can often be performed using manual or automatic thresholding techniques [11, p. 142]. Outlier detection should only be attempted if a normal distribution of data points can be assumed [11, p. 145]. If so, methods like Resampling by Half Means (RHM), Smallest Half Volume (SHV) or more robust estimators like MVE or Minimum Covariance Determinant (MCD) can be used [11, p. 145].
- Following the paper of Vidal et al. [11], spectral preprocessing can be split into noise reduction, scatter correction and the reduction of additive baseline shifts. Noise reduction can be performed using 3D or Daubechies wavelets [11, pp. 140-141]. Both spectral denoising and the rectification of additive baseline shifts can be attempted using the Savitzky-Golay filter [11, p. 145]. For scatter correction the use of the Standard Normal

Variate (SNV) is advisable if no reference spectrum is available. Otherwise, if a reference spectrum can be recorded a priori, then Multiplicative Scatter Correction (MSC) should be considered instead [11, p. 145] [15].

- Dimensionality reduction of a hyperspectral datacube can be achieved using variable selection techniques based on genetic algorithms or partial least squares regression (e. g. iPLS) [11, p. 140]. Other methods include spatial and spectral binning as well as factor model approaches such as Principal Component Analysis (PCA) or Multivariate Curve Resolution (MCR) [11, p. 140].

A classification of hyperspectral data cleaned with a selection of these methods is performed by Amigo et al. [16], co-author of [11]. The authors aim at distinguishing flame retardants contained in six different types of plastics using partial least squares regression. In contrast to this paper, the authors do not attempt a classification via artificial neural networks. In the context of hyperspectral image classification, neural networks, especially convolutional types, can be considered as successors of kernel-based methods like PCA and Support Vector Machines (SVM) [17]. Yu et al. [18] demonstrate the superior classification ability of convolutional neural networks (CNN) in a comparative study involving k-nearest neighbor (KNN) and SVM variations. The results show that the CNN-based classifier performs best on all three hyperspectral datasets, being Indian Pines, PaviaU and Salinas. The literature today includes a number of CNN architectures specifically designed for hyperspectral image classification. Ausdebert et al. implement ten of these CNN-based classifiers in their DeepHyperX toolbox [19, 20], of which the S-CNN architecture [21] performs best in our experiments (see Table 4).

3. Methods

A preprocessing pipeline can be considered as a composition of functions which takes a hyperspectral data cube (*hypercube*) as input and applies a set of transformations to it. Each preprocessing step can affect the spatial and spectral dimensions of the resulting hypercube, which will serve as input of the subsequent step. Therefore, algorithms in a preprocessing pipeline must be able to adapt to changes in input dimensionality. Experiments have shown that careless composition of algorithms within preprocessing pipelines can have a negative impact on their effectiveness. Figure 1 (adapted from Vidal et al. [11]) depicts a reasonable sequence of individual preprocessing steps also used by the pipelines compiled in this paper. Below, the findings of a literature review are used to instantiate several such preprocessing pipelines with selected algorithms. The task of these pipelines is to remove unwanted noise from hyperspectral images, for three contamination levels (low, medium and high). After this preprocessing phase, the cleansed hyperspectral images are used for training a neural classifier in order to draw conclusions about effective combinations of different preprocessing algorithms. The selected preprocessing algorithms are summarized in Table 1.
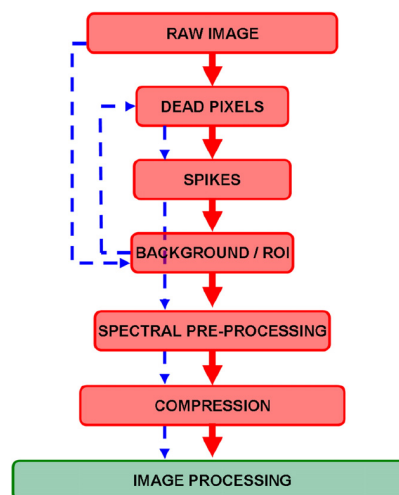

Figure 1: Structure of a hyperspectral preprocessing pipeline, cf. [11]

Table 1
Overview of algorithms used for creating preprocessing pipelines.

Preprocessing step	Algorithm	Abbr.	Ref.
Dead pixel detection	Standard Deviation Threshold	SDT	[11, sec. 6]
Spike detection	Standard Deviation Factor	SDF	[11, sec. 7]
Outlier detection	Stochastic Outlier Selection	SOS	[22]
	Copula Based Outlier Selection	COPOD	[23]
	Reed-Xiaoli Detector	RXD	[24]
Noise reduction	Savitzky-Golay Filter	SGF	[25]
	Daubechies Wavelet Filter	Wt	[26]
	Minimum Noise Fraction	MNF	[27]
Scatter correction	Standard Normal Variate	SNV	[28]
	Multiplicative Scatter Correction	MSC	[29]
Dimensionality reduction	Principal Component Analysis	PCA	[30]
	Spectral Binning	SRD	—
	Spatial Binning	SSSW	—

3.1. Methods for Dimensionality Reduction

3.1.1. Spectral Binning using Self-Reference Deviation (SRD)

For hyperspectral data, spectral binning is a dimensionality reduction technique, which merges similar bands along the spectral dimension z of the hypercube $\mathbf{H} \in \mathbb{R}^{y \times x \times z}$ where y and x denote the spatio-temporal and spatial dimensions respectively. More generally, y represents the height and x the width of each spectral band in pixels. Binning can be done manually for obvious groups of similar bands by specifying static bins. Since this approach is rarely practical for all samples in a dataset, a new automatic spectral binning procedure is presented below.

   The SRD method is based on the average distance $\overline{d} \in \mathbb{R}_{>0}$ of the intensity values inside the averaged spectrum $\mathbf{I}^* \in \mathbb{R}^z$ (self-reference) of the hypercube. For dynamic creation of bins, SRD expects a user defined maximum deviation $\varepsilon \in (0, 1]$ from this average distance $\overline{d}$. In summary, the SRD algorithm works as follows:

1. Calculate the reference spectrum $\mathbf{I}^*$ of $\mathbf{H}$, where the i-th entry of $\mathbf{I}^*$ is the mean $\overline{b}_i$ of the i-th spectral band of $\mathbf{H}$.
2. Calculate the mean distance $\overline{d}$ of all neighboring intensity values in $\mathbf{I}^*$, adding the maximum allowed percentage deviation: $\overline{d} := \overline{d} \cdot (1 + \varepsilon)$
3. Obtain the averages $\overline{b}_i$ from the reference spectrum $\mathbf{I}^*$ with $1 \leq i \leq z$ and identify the bins using the following case distinction:
   a) If $i = 1$ or the distance from the current mean $\overline{b}_i$ to the mean of the bands within the current bin is less than $\overline{d}$: Add the i-th band from $\mathbf{H}$ to the current bin. A small deviation from the current bin's mean indicates a high degree of similarity.
   b) Otherwise: Close the current bin and start with a new one.
4. Create a reduced hypercube $\mathbf{H}' \in \mathbb{R}^{y \times x \times z'}$ by averaging the spectral bands of each bin.

This operation reduces the hypercube's depth by merging multiple similar bands into one. The storage gain G_{SRD} achieved by spectral binning is calculated from the product of the depth difference $z - z'$, the unchanged spatial resolution $y \times x$ and the constant storage size k of an intensity value in bits: $G_{\text{SRD}} = (z - z') \cdot y \cdot x \cdot k$ where $0 < z' \leq z$. Here, the storage gain is large if the difference $z - z'$ is large, that is, the number of bins generated is small.

3.1.2. Spatial Binning using a SID-SAM Sliding Window (SSSW)

Spatial binning, in contrast to spectral binning, achieves dimensionality reduction by merging similar point spectra located in the xy plane of the hypercube $\mathbf{H} \in \mathbb{R}^{y \times x \times z}$. The spatial binning method presented in this section compares multiple point spectra in a $s \times t$ pixel wide window, which is moved over the xy-plane of the hypercube with a horizontal step size of s pixels. After reaching the right image boundary, the window is moved t pixels down and repositioned at the left edge of the image. For each point spectrum within a window region, a similarity score is calculated by comparing it to the hypercube's averaged spectrum $\mathbf{I}^*$ using the SID-SAM similarity measure [31]. In doing so, only those spectra are merged that exhibit a high similarity to $\mathbf{I}^*$ and thus contribute comparatively little to the hypercube's entropy. SSSW can be used with all types of hyperspectral data cubes regardless of the window size, since the median spectrum $\mathbf{I}^*$ does always exist. The similarity matrix $\mathbf{R}^{y \times x}$ stores the SID-SAM score for each point spectrum. An entry of $\mathbf{R}$ is small if the associated point spectrum is similar to $\mathbf{I}^*$. Afterwards, the matrix $\mathbf{R}$ is subdivided into the original $s \times t$ wide window regions. The SID-SAM values inside these smaller matrices are averaged to obtain a single similarity measure tied to a specific image area. Subsequently, all averaged SID-SAM values are associated with their corresponding point spectra in the xy-plane of the hypercube. In doing so, a maximum of $u = (\lfloor y/s \rfloor + 1) \cdot (\lfloor x/t \rfloor + 1)$ non-overlapping image regions are considered for spatial binning. After sorting the averaged SID-SAM values in ascending order, the point spectra eligible for merging are determined through the use of a user defined percentile $\theta \in [0, 1]$. In this case, it is sufficient to associate the smallest $\lfloor \theta \cdot u \rfloor$ SID-SAM values with their corresponding window regions and merge the contained spectra by spectral averaging. The storage gain G_{SSSW} achieved by the SSSW algorithm increases with the window size $s \times t$ and the percentile θ used for binning: $G_{\text{SSSW}} \approx \lfloor \theta \cdot u \rfloor \cdot (s \cdot t - 1) \cdot z \cdot k$ bits. The value k represents the constant storage size of an intensity value in bits. Strictly speaking, this storage gain calculation is rather an approximation, since the sliding window procedure will generate irregular window sizes if either the image width or height is not a multiple of s or t respectively. Finally, since merging multiple point spectra creates gaps in the original data cube, a compression is performed by densely rearranging the binned point spectra resulting in different output dimensions. This procedure has the disadvantage of separating spatial features in the xy-plane of the hypercube.

3.2. Preprocessing Pipelines

The combinatorial complexity associated with instantiating various alternative preprocessing pipelines is significantly reduced by using the fixed order of the individual preprocessing steps depicted in Figure 1. The number of eligible pipeline combinations is further limited by purposely leaving out MSC, since the linearly related SNV [32] is faster to compute. Furthermore, due to the absence of a reference spectrum, there are no advantages justifying the use of MSC in this paper. Additional pipeline combinations are eliminated by always applying SDT and SDF at the beginning of each preprocessing pipeline. This results in the generated pipelines differing only in their preprocessing algorithms for outlier detection, noise reduction, and dimension reduction (see Figure 2). The input and output degrees of all inner nodes of this graph are initially identical by definition, since each node produces an output and passes it as input to all descendant nodes.

For reasons of clarity, the three outgoing edges of each noise reduction node (MNF, Wt, SGF) in Figure 2 are merged into a single edge. The same applies to the outgoing edges of the SNV node. The number of pipelines constructed in this way can be determined by counting all possible paths from the root (SDT) to the leaf nodes (SRD, PCA, SSSW) of the graph. Since preprocessing steps with only one algorithm do not create additional pipelines, only three of the six preprocessing steps have to be considered with three alternatives each. Therefore, the graph shown in Figure 2 produces a total of $3^3 = 27$ different pipeline combinations.
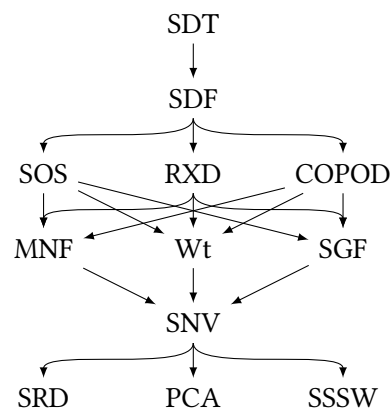

Figure 2: Preprocessing pipelines with algorithms from Table 1.

3.3. Reference Dataset

The raw dataset used in this paper consists of hyperspectral images depicting a total of 728 clean potatoes, which were manually acquired using a Helios Core 0.9-1.7 XF-PA100 240×320/CII/H330 line scan camera. Table 2 includes 18 more potatoes for a grand total of 746 since some potatoes are annotated with multiple classes. During dataset creation, samples annotated with multiple classes are duplicated to each assigned quality class. To avoid methodological errors in training and evaluation, potatoes with multiple defect classes are not included in the contiguous reference dataset. Additionally, the raw dataset is restricted to the four largest defect classes from Table 2 to ensure the availability of sufficient test spectra during evaluation. These measures reduce the size of the raw dataset from 728 to initially 622 and after deduplication to finally 606 potatoes with an average of approximately 13 700 point spectra each.

Table 2
Quality classes of the hyperspectral potato dataset.

#	Quality class	Quantity	Σ	#	Quality class	Quantity	Σ
1	green	264		7	growth deformity	17	
2	dry rot	171		8	blue spots	17	
3	no defect	117	622	9	withered	14	124
4	growth crack	70		10	wet rot	9	
5	mechanical damage	43					
6	scab	24				**Total**	746

3.4. Dataset Contamination

The spectra of the reference dataset are artificially contaminated with varying amounts of noise to investigate the effectiveness of different preprocessing pipelines. This procedure became necessary due to an unknown amount of in-camera preprocessing that was applied to the raw sensor data at acquisition time. Contamination is performed in three stages using additive and multiplicative scattering effects, Gaussian noise, spike points and dead pixels. Table 3 quantifies the degree of artificial contamination for all three contamination levels using statistical indicators applied to the whole set of intensity values contained within each dataset.

The strength of additive and multiplicative scattering effects is dynamically calculated as a function of contamination level and applied to random spectra of a fixed user-defined percentile (24 %). As with the other types of contamination, the spectra are increasingly influenced by additive and multiplicative

Table 3: Statistical characteristics of the three contamination levels.

Contam.	Avg.	SD (σ)	Var. (σ^2)
low	0.374	0.241	0.058
medium	0.353	0.246	0.061
high	0.333	0.256	0.066

scattering effects at higher contamination levels. Additive scattering effects can manifest themselves not only in a positive but also negative shift of intensity values. In contrast to additive influences, multiplicative scattering affects the slopes within a spectrum's wavelength profile. In this paper, this phenomenon is simulated by multiplying a spectrum with by factor between 0.5 and 0.9. This ensures that the intensity values of the spectrum are not shifted out of their normalized range of values [0,1]. Since multiplying a spectrum by a number close to one has little effect on its slopes, the highest scaling factor is used at the lowest contamination level. Gaussian noise is applied to every sample with increasing variance at higher contamination levels (0.0004, 0.0012, 0.0020). Simulation of both spike points (max. 1.5 %) and defect pixels (max. 2.4 %) is implemented by randomly setting maximum (1) or minimum (0) intensity values. The influence of defect pixels is simulated within the i-th band $\mathbf{B}_{\lambda_i} \in \mathbb{R}^{y \times x}$ by randomly replacing columns, such that they consist either of the minimum or maximum value $\mathbf{0}, \mathbf{1} \in \mathbb{R}^y$, respectively.

3.5. Classification of Preprocessed Datasets

3.5.1. Dataset Sampling

Sampling of both the reference dataset and the three contaminated datasets is performed using a special sampling method that guarantees the spatial disjointness of training, test, and validation spectra. The purpose of this sampling procedure is to separate the point spectra of a spatially contiguous dataset in such a way that no two spectra of the same potato appear in more than one of the three dataset splits. The developed sampling strategy achieves this by dividing the ground truth mask of the reference dataset into three overlap-free regions (see Figure 3). This trisection is performed along the horizontal axis of the contiguous hypercube, since a vertical decomposition could result in a class imbalance within the partial datasets. The reason for this is that the potatoes inside the contiguous hypercube are sorted according to their size in order to save space by reducing the amount of empty background voxels. However, in our dataset, green potatoes were on average significantly smaller than potatoes of other quality classes and therefore located in the lower third of the ground truth mask. Performing vertical partitioning along the horizontal axis showed an improvement in the distribution of potato sizes and classes in each dataset split.
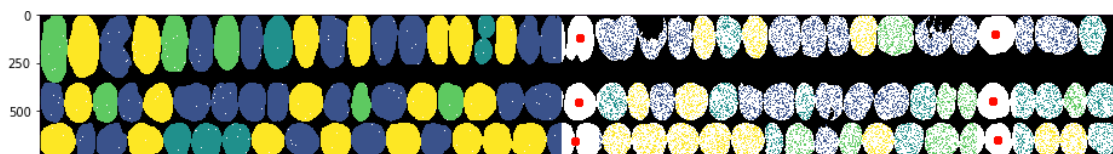

Figure 3: Sampling of training, testing, and validation data from three non-overlapping regions.

3.5.2. Network Architecture Selection

Classification of the contaminated datasets as well as those cleaned by the preprocessing pipelines is performed using the DeepHyperX toolbox [19, 20]. For the selection of a suitable classifier, the reference dataset consisting of 8.3 million point spectra is limited to a training and test dataset of 200 000 spectra each. The training is monitored with a validation split of 20 000 spectra. The goal of these initial training runs is to determine a neural network architecture that has a low training duration with comparatively high accuracy. Excessively long training durations are unacceptable from a practical point of view, since up to 45 training and testing iterations must be performed with the pipelines constructed in Section 3.2. The seven neural network architectures evaluated in advance to this paper are part of the DeepHyperX toolbox and trained in six epochs with a batch size of 100. Other settings and hyperparameters like learning rate, its scheduler, optimizer and loss function are left at their defaults. All training runs are performed on two Tesla V100 graphics cards of an NVIDIA DGX Station. The comparatively low amount of training spectra is due to the long training duration (4 h 20 min) of the best performing CNN architecture developed by Sharma et al. (see Table 4).

Table 4
Classification accuracies and training durations of a hyperspectral CNN (`sharma`) and multilayer perceptron (`nn`) on spectra of the reference dataset.

Arch.		Train loss	Val. acc.	Test acc.	Train t	Test t
sharma	[21]	0.228	59.7 %	64.7 %	4:20:08	0:14:57
nn	[19]	0.999	49.4 %	52.8 %	0:02:26	0:06:51

Due to the long training duration of the `sharma` architecture, the second ranked `nn` architecture is chosen as the classifier to evaluate the performance of the different preprocessing pipelines, serving as a prototypical instance towards more advanced architectures. The `nn` architecture is not a convolutional neural network, but rather a simple multilayer perceptron (MLP) consisting of only four fully connected layers, each with 2048 neurons.

4. Results

4.1. Experimental Setup

The nn architecture is trained on approximately 4.5 million spectra from each of the contaminated datasets in 12 epochs with a batch size of 100. Training is monitored at each epoch with approximately 225 000 validation spectra while the learning rate is dynamically adjusted by the `ReduceLROnPlateau` scheduler. Cross entropy is used as the loss function in conjunction with the Adam optimizer. As a regularization mechanism, Dropout is applied with a dropout probability of $p = 0.5$. The evaluation of a trained model is performed on approximately 900 000 test spectra. Since the training, test, and validation datasets are created using the spatially disjoint sampling method from Section 3.5.1, no spectra from the same potato can appear in both the training and test datasets. The computed performance metric is the classification accuracy of each model on its respective test dataset.

4.2. Reference Dataset

The nn classifier trained on the reference dataset achieves an accuracy of 53.61 % on the first four defect classes (cf. Table 2), which is used as a baseline for performance comparison below.

4.3. Contaminated Datasets

As expected, the classification accuracy of the nn architecture decreases monotonically but not uniformly as the contamination level of the dataset increases. A test accuracy of 40.77 % was reported for low, 37.53 % for medium and 36.36 % for high contamination. The observed degradation spans 4.41 %, with a gap of 3.24 % from second to first place. This difference is almost three times larger than from third to second rank (1.17 %). Compared to the classification results obtained from the reference dataset, the nn architecture exhibits a drop in test accuracy of 12.84 % at low contamination. Hence, the generalization ability of the nn architecture is found to be significantly weaker on the contaminated datasets.

4.4. Preprocessed Datasets

The classification results listed in Tables 5 and 7 correspond to the pipeline combinations implied by Figure 2. Also, the 36 rows of the first table together with the 9 rows of the second table constitute the 45 preprocessing runs mentioned in section 3.5.2. Before analyzing the reported classification accuracies, it is important to note that the tables presented in this section do not explicitly list the first two preprocessing algorithms dealing with dead pixel (SDT) and spike detection (SDF). As mentioned in Section 3.2, these procedures are always applied in this order at the beginning of each preprocessing pipeline, such that they are omitted for better readability of the tabular listings. Table 6 quantifies the performance differential between the various dimensionality reduction techniques observed in Table 7. The left half of this listing includes the mean classification accuracies of the first 19 pipeline combinations using spectral binning, which are grouped by their respective contamination level and sorted by test accuracy. This ordering also applies to the right half of Table 6, which summarizes the same metrics for the last 17 pipelines using PCA for dimensionality reduction. Table 7 reports the classification results of those preprocessing pipelines that discard the spatial features of their inputs. This is because, unlike in Table 5, outliers are detected using the COPOD algorithm and dimensionality reduction is performed through spatial (SSSW) rather than spectral binning (SRD). This leads to a new peak classification accuracy of 44.52 %, i. e., 1.81 % higher than reported in Table 5.

4.5. Discussion

The classification results listed in Table 5 show a clear dichotomy in their final processing step. The first 19 pipelines largely perform spectral binning as their dimensionality reduction, whereas the last 17 pipelines apply PCA. All pipeline combinations in the upper half of Table 5 (rows 1 to 19) were able to improve the classifiability of datasets with medium and high contamination levels. None of the last 17 pipelines succeeded in achieving the same objective. The pipelines which showed the largest improvement in classifiability of their cleaned datasets are located in the first two rows of Table 5 (+6.35 %, +5.03 %).

Table 5
Classification results of preprocessed datasets with retained spatial features.

#	Contam.	SOS	RXD	SGF	Wt	MNF	SNV	PCA	SRD	Test acc.
1	high	–	✓	–	–	✓	✓	–	✓	42.71 %
2	medium	✓	–	–	–	✓	✓	–	✓	42.56 %
3	high	✓	–	–	–	✓	✓	–	✓	40.74 %
4	medium	–	✓	–	–	✓	✓	–	✓	40.68 %
5	low	✓	–	–	–	✓	✓	–	✓	39.71 %
6	low	✓	–	✓	–	–	✓	–	✓	39.64 %
7	low	–	✓	–	–	✓	✓	–	✓	39.56 %
8	medium	–	✓	✓	–	–	✓	–	✓	39.53 %
9	low	–	✓	✓	–	–	✓	–	✓	39.39 %
10	low	–	✓	–	✓	–	✓	–	✓	38.79 %
11	high	✓	–	✓	–	–	✓	–	✓	38.71 %
12	low	✓	–	–	✓	–	✓	–	✓	38.62 %
13	medium	–	✓	–	–	✓	✓	✓	–	38.49 %
14	high	–	✓	–	✓	–	✓	–	✓	38.43 %
15	medium	✓	–	✓	–	–	✓	–	✓	38.32 %
16	high	✓	–	–	✓	–	✓	–	✓	38.13 %
17	medium	–	✓	–	✓	–	✓	–	✓	38.10 %
18	medium	✓	–	–	✓	–	✓	–	✓	37.88 %
19	high	–	✓	✓	–	–	✓	–	✓	37.64 %
20	low	–	✓	–	–	✓	✓	✓	–	35.59 %
21	low	✓	–	–	–	✓	✓	✓	–	35.45 %
22	medium	✓	–	–	–	✓	✓	✓	–	35.24 %
23	high	✓	–	–	✓	–	✓	✓	–	35.08 %
24	medium	✓	–	–	✓	–	✓	✓	–	34.25 %
25	high	–	✓	–	✓	–	✓	✓	–	33.71 %
26	low	✓	–	–	✓	–	✓	✓	–	33.20 %
27	low	–	✓	–	✓	–	✓	✓	–	32.91 %
28	high	✓	–	–	–	✓	✓	✓	–	32.41 %
29	high	✓	–	✓	–	–	✓	✓	–	32.32 %
30	medium	–	✓	–	✓	–	✓	✓	–	32.11 %
31	high	–	✓	–	–	✓	✓	✓	–	31.92 %
32	low	✓	–	✓	–	–	✓	✓	–	30.91 %
33	high	–	✓	✓	–	–	✓	✓	–	30.67 %
34	low	–	✓	✓	–	–	✓	✓	–	30.30 %
35	medium	✓	–	✓	–	–	✓	✓	–	30.25 %
36	medium	–	✓	✓	–	–	✓	✓	–	29.10 %

Table 6
Mean test accuracies and their changes related to the first 19 and last 17 pipelines from Table 5.

#	Contam.	∅ Test acc.	Δ Acc. (Sec. 4.3)	#	Contam.	∅ Test acc.	Δ Acc. (Sec. 4.3)
1	high	39.39 %	+3.03 %	1	low	33.06 %	-7.71 %
2	medium	39.37 %	+1.48 %	2	high	32.69 %	-3.67 %
3	low	39.29 %	-1.48 %	3	medium	32.19 %	-5.34 %

By examining the actual dimensionality reductions performed in Table 5, we observe that the highest ranked pipeline (42.71 % acc.) uses spectral binning to perform an average dimensionality reduction of 81 bands per sample. After replacing the spectral binning with PCA, the test accuracy drops by 10.79 % to 31.92 % (rank 31). The dimensionality reduction carried out by the PCA decreases the average depth of the respective dataset from 252 to only 3 bands. Based on

Table 7
Classification results of preprocessed datasets with discarded spatial features.

#	Contam.	COPOD	SGF	Wt	MNF	SNV	SSSW	Test acc.
1	low	✓	✓	–	–	✓	✓	44.52 %
2	high	✓	–	–	✓	✓	✓	42.40 %
3	medium	✓	✓	–	–	✓	✓	42.35 %
4	medium	✓	–	–	✓	✓	✓	41.39 %
5	high	✓	✓	–	–	✓	✓	40.89 %
6	low	✓	–	✓	–	✓	✓	38.62 %
7	high	✓	–	✓	–	✓	✓	35.78 %
8	medium	✓	–	✓	–	✓	✓	35.41 %
9	low	✓	–	–	✓	✓	✓	34.64 %

this observation, it can be concluded that a drastic dimensionality reduction using PCA can have detrimental effects on the classifiability of hyperspectral datasets. This claim is further substantiated by rows 20 to 36 of Table 5. For reference, the PCA carried out in this paper was set to obtain 99.8 % of a sample's variance.

In Table 6, we observe that the contamination levels are reversed in comparison to the classification results of the contaminated datasets (see Section 4.3). On average the best improvements in test accuracy can be obtained on the most contaminated datasets (+3.03 %). Our experiments indicate, that preprocessing datasets with low contamination levels may even lead to a deterioration of their classifiability ($\varnothing$ -1.48 %). In our tests, this observation holds true for each individual pipeline applied to a dataset with low contamination. The best classification result of a lightly contaminated dataset is 39.71 % (rank 5) behind two more contaminated datasets. The second half of the table shows the negative impact of PCA on the classification results. The greatest losses occur for datasets of the lowest contamination level, whose test accuracy decreases by 7.71 % on average. In second place are the most contaminated datasets, whose mean classification accuracy is least affected by PCA (-3.67 %). Regarding noise reduction, the first five rows of Table 5 indicate that the MNF algorithm is superior to SGF and Wt. This coincides with both the visual appearance of the denoised samples and the fact that MNF was specifically designed for hyperspectral noise reduction. Even the popular Savitzky-Golay filter, which seems to work well with the Reed-Xiaoli detector, cannot outperform MNF. In terms of outlier detection, the combinations of SOS and MNF (3) and RXD and MNF (2) occur almost equally often in the Top-6 results of Table 5. However, the highest ranked preprocessing pipeline uses RXD rather than SOS for outlier detection. The importance of RXD increases after the rankings are extending to the first eleven results, where RXD is used by six pipelines and SOS only by five. In the lower half of Table 5 we observe that SGF performs poorly in conjunction with PCA, whereas the wavelet transform appears to be more robust. Similar advantages emerge in the midfield, where wavelet filtering is on par with SGF and even outperforms on datasets with higher contamination levels.

Overall, the best classification accuracy is achieved by a pipeline which discards features in the spatial-temporal domain (see Table 7). This pipeline preprocesses a dataset of low contamination – classified with an accuracy of 44.52 %. This result is 1.81 % higher than the previous peak of 42.71 % (see Table 5). The second best accuracy was previously obtained by a combination of RXD, MNF and spectral binning. For the results shown in Table 7, there is no superior method for noise reduction in the context of COPOD and spatial binning. Both MNF and SGF show promising results in spectral and spatial smoothing, while wavelet transform clearly underperforms.

5. Conclusions

This paper considered classification methods on hyperspectral image data, specially presenting novel preprocessing methods, which were integrated into a comprehensive preprocessing pipeline. We performed an analysis of multiple preprocessing pipelines in the context of image classification, in order to analyze which preprocessing algorithms perform best.

Our results indicate that dimensionality reduction using principal component analysis can potentially have negative effects on the classifiability of hyperspectral data. Furthermore, both the Reed-Xiaoli detector and the MNF transform are preferable as dedicated hyperspectral preprocessing algorithms to general-purpose methods such as SOS or wavelet filtering. The Savitzky-Golay filter, widely used in signal processing, can also be used successfully for smoothing noisy spectra. The results obtained with this method can be considered satisfactory as long as no dimensional reduction is performed with PCA. For this reason, it is recommended that dimensionality reduction should be removed from a preprocessing pipeline and not applied across the board to all samples. If data reduction is still required, then the less invasive but also less effective methods of spatial and spectral binning can be used as an alternative to PCA. Furthermore, a step-by-step calculation of the results listed in the Table 5 and 7 showed that shorter pipelines correlate with better classification results. Therefore, preprocessing pipelines should contain as few algorithms as possible, focusing on those that are tailored to the type of contamination at hand. In order to determine which preprocessing steps are really useful, an examination of the contaminated dataset is recommended.

In our experimentation, advantages in favoring preprocessing pipelines that preserve spatial features were not observed in terms of classification accuracy using a multilayer perceptron. However, it can be conjectured that convolutional neural networks can extract generalizable features from coherent spatio-temporal regions, which is supported by the results shown in Table 4.

In summary, for classifying hyperspectral image data we recommend the following strategies:

1. The preprocessing algorithms should be as domain-specific as possible.
2. A minimal number of pre-processing algorithms should be used.
3. The preprocessing algorithms should address the specific impurities of the dataset.
4. The algorithms should be carefully parameterized and arranged according to Figure 1.
5. For further classification, a CNN designed for hyperspectral data should be used.

For future work, we aim to extend the analysis of the applied neural network architectures. Also, for explainability and interpretability of the applied method [33, 34, 35, 36] both specialized preprocessing approaches as well as respective feature generation, e. g., using declarative or neuro-symbolic techniques [37, 38, 39] provide interesting directions for future research.

Acknowledgments

References

[1] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al., Recent advances in convolutional neural networks, Pattern recognition 77 (2018) 354–377. doi:10.1016/j.patcog.2017.10.013.

[2] A. Khan, A. Sohail, U. Zahoora, A. S. Qureshi, A survey of the recent architectures of deep convolutional neural networks, Artificial Intelligence Review 53 (2020) 5455–5516. doi:10.1007/s10462-020-09825-6.

[3] D. Hemanth, V. Estrela, Deep Learning for Image Processing Applications, Advances in Parallel Computing, IOS Press, 2017.

[4] L. Jiao, J. Zhao, A Survey on the New Generation of Deep Learning in Image Processing, IEEE Access 7 (2019) 172231–172263. doi:10.1109/ACCESS.2019.2956508.

[5] S. E. Whang, J.-G. Lee, Data collection and quality challenges for deep learning, Proceedings of the VLDB Endowment 13 (2020) 3429–3432. doi:10.14778/3415478.3415562.

[6] S. E. Whang, Y. Roh, H. Song, J.-G. Lee, Data collection and quality challenges in deep learning: A data-centric ai perspective, The VLDB Journal 32 (2023) 791–813. doi:10.1007/s00778-022-00775-9.

[7] A. Plaza, J. A. Benediktsson, J. W. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, A. Gualtieri, et al., Recent advances in techniques for hyperspectral image processing, Remote sensing of environment 113 (2009) S110–S122. doi:10.1016/j.rse.2007.07.028.

[8] M. Manley, Near-infrared spectroscopy and hyperspectral imaging: non-destructive analysis of biological materials, Chemical Society Reviews 43 (2014) 8200–8214. doi:10.1039/c4cs00062e.

[9] J. M. Amigo, I. Martí, A. Gowen, Hyperspectral Imaging and Chemometrics: A Perfect Combination for the Analysis of Food Structure, Composition and Quality, in: Data handling in science and technology, volume 28, Elsevier, 2013, pp. 343–370. doi:10.1016/B978-0-444-59528-7.00009-0.

[10] B. Boldrini, W. Kessler, K. Rebner, R. Kessler, Hyperspectral Imaging: A Review of Best Practice, Performance and Pitfalls for in-line and on-line Applications, Journal of Near Infrared Spectroscopy 20 (2012) 438–508. doi:10.1255/jnirs.1003.

[11] M. Vidal, J. M. Amigo, Pre-processing of hyperspectral images. essential steps before image analysis, Chemometrics and Intelligent Laboratory Systems 117 (2012) 138–148. doi:10.1016/j.chemolab.2012.05.009.

[12] L. Dale, A. Thewis, C. Boudry, I. Rotar, P. Dardenne, V. Baeten, J. Fernández Pierna, Hyperspectral Imaging Applications in Agriculture and Agro-Food Product Quality and Safety Control: A Review, Applied Spectroscopy Reviews 48 (2013) 142. doi:10.1080/05704928.2012.705800.

[13] C. Wang, B. Liu, L. Liu, Y. Zhu, J. Hou, P. Liu, X. Li, A review of deep learning used in the hyperspectral image analysis for agriculture, Artificial Intelligence Review 54 (2021) 5205–5253. doi:10.1007/s10462-021-10018-y.

[14] A. Schliebitz, H. Graf, T. Wamhof, H. Tapken, A. Gertzen, KI-basiertes Computer-Vision-System zur Qualitäts- und Größenbestimmung von Kartoffeln, 43. GIL-Jahrestagung, Resiliente Agri-Food-Systeme (2023).

[15] D. Pelliccia, Two scatter correction techniques for NIR spectroscopy in Python, Online, 2018. URL: https://nirpyresearch.com/two-scatter-correction-techniques-nir-spectroscopy-python/, retrieved: 2023-07-15.

[16] J. M. Amigo, H. Babamoradi, S. Elcoroaristizabal, Hyperspectral image analysis. A tutorial, Analytica Chimica Acta 896 (2015) 34–51. doi:10.1016/j.aca.2015.09.030.

[17] G. Camps-Valls, L. Bruzzone, Kernel-based methods for hyperspectral image classification, IEEE Transactions on Geoscience and Remote Sensing 43 (2005) 1351–1362. doi:10.1109/TGRS.2005.846154.

[18] S. Yu, S. Jia, C. Xu, Convolutional neural networks for hyperspectral image classification, Neurocomputing 219 (2017) 88–98. doi:10.1016/j.neucom.2016.09.010.

[19] N. Audebert, DeepHyperX, Online, 2018. URL: https://github.com/nshaud/DeepHyperX, retrieved: 2023-07-15.

[20] N. Audebert, B. Le Saux, S. Lefèvre, Deep Learning for Classification of Hyperspectral Data: A Comparative Review, IEEE Geoscience and Remote Sensing Magazine 7 (2019) 159–173. doi:10.1109/MGRS.2019.2912563.

[21] V. Sharma, A. Diba, T. Tuytelaars, L. Van Gool, Hyperspectral CNN for Image Classification & Band Selection, with Application to Face Recognition, Technical report KUL/ESAT/PSI/1604, KU Leuven, ESAT, Leuven, Belgium (2016).

[22] J. Janssens, Outlier Selection and One-Class Classification, Ph.D. thesis, 2013. Series: TiCC Ph.D. Series Volume: 27.

[23] Z. Li, Y. Zhao, N. Botta, C. Ionescu, X. Hu, COPOD: Copula-Based Outlier Detection, in: 2020 IEEE International Conference on Data Mining (ICDM), 2020, pp. 1118–1123. doi:10.1109/ICDM50108.2020.00135.

[24] I. Reed, 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) 1760–1770. doi:10.1109/29.60107.

[25] A. Savitzky, M. J. E. Golay, Smoothing and Differentiation of Data by Simplified Least Squares Procedures., Analytical Chemistry 36 (1964) 1627–1639. doi:10.1021/ac60214a047.

[26] I. Daubechies, Ten Lectures on Wavelets, SIAM, 1992. doi:10.1137/1.9781611970104.

[27] A. A. Green, M. Berman, P. Switzer, M. D. Craig, A Transformation for Ordering Multispectral Data in Terms of Image Quality with Implications for Noise Removal, IEEE Transactions on geoscience and remote sensing 26 (1988) 65–74. doi:10.1109/36.3001.

[28] R. J. Barnes, M. S. Dhanoa, S. J. Lister, Correction to the Description of Standard Normal Variate (SNV) and De-Trend (DT) Transformations, NIR news 5 (1994) 6–6. doi:10.1255/jnirs.21.

[29] T. Isaksson, T. Næs, The Effect of Multiplicative Scatter Correction (MSC) and Linearity Improvement in NIR Spectroscopy, Applied Spectroscopy 42 (1988) 1273–1284. doi:10.1366/0003702884429869.

[30] K. Pearson, LIII. On lines and planes of closest fit to systems of points in space, The London, Edinburgh, and Dublin philosophical magazine and journal of science 2 (1901) 559–572. doi:10.1080/14786440109462720.

[31] Y. Du, C.-I. Chang, et al., New hyperspectral discrimination measure for spectral characterization, Optical engineering 43 (2004) 1777–1786. doi:10.1117/1.1766301.

[32] M. S. Dhanoa, S. J. Lister, R. Sanderson, R. J. Barnes, The Link between Multiplicative Scatter Correction (MSC) and Standard Normal Variate (SNV) Transformations of NIR Spectra, J. Near Infrared Spectrosc. 2 (1994) 43–47. doi:`10.1255/jnirs.30`.

[33] O. Biran, C. Cotton, Explanation and Justification in Machine Learning: A Survey, in: IJCAI-17 Workshop on Explainable AI, 2017.

[34] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, Information Fusion 58 (2020) 82 – 115. doi:`10.1016/j.inffus.2019.12.012`.

[35] L. Bertossi, F. Geerts, Data quality and explainable AI, Journal of Data and Information Quality (JDIQ) 12 (2020) 1–9. doi:`10.1145/3386687`.

[36] S. Vollert, M. Atzmueller, A. Theissler, Interpretable Machine Learning: A Brief Survey From the Predictive Maintenance Perspective, in: Proc. IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2021), IEEE, 2021. doi:`10.1109/ETFA45728.2021.9613467`.

[37] M. Atzmueller, Declarative Aspects in Explicative Data Mining for Computational Sensemaking, in: D. Seipel, M. Hanus, S. Abreu (Eds.), Proc. International Conference on Declarative Programming, Springer, Heidelberg, Germany, 2018, pp. 97–114. doi:`10.1007/978-3-030-00801-7_7`.

[38] A. Holzinger, From machine learning to explainable AI, in: 2018 world symposium on digital intelligence for systems and machines (DISA), IEEE, 2018, pp. 55–66. doi:`10.1109/DISA.2018.8490530`.

[39] T. Wu, M. Tjandrasuwita, Z. Wu, X. Yang, K. Liu, R. Sosic, J. Leskovec, ZeroC: A Neuro-Symbolic Model for Zero-shot Concept Recognition and Acquisition at Inference Time, Advances in Neural Information Processing Systems 35 (2022) 9828–9840. doi:`10.48550/arXiv.2206.15049`.