Dimensionality Reduction for Semantic Segmentation in Different Architectures

Lucca Arantes Martins[a], José Luis Seixas Junior[1]

[a]*ELTE University, Budapest, Hungary*

Abstract

Hyperspectral imaging (HSI) captures detailed spectral information across numerous wavelength bands, providing powerful sources of information for pixel-wise classification. However, the high dimensionality and complexity of HSI data pose significant challenges for image segmentation. This study explores the performance of different neural network architectures on hyperspectral image segmentation by employing dimensionality reduction techniques. We utilized Principal Component Analysis (PCA) and Independent Component Analysis (ICA) to create 12 datasets from the Salinas and Indian Pines datasets, each with varying numbers of components (8, 15, and 30). Four neural network models were trained on these datasets: shallow and deep variants of 3D Convolutional Neural Networks (3DCNNs) and Recurrent Neural Networks (RNNs), resulting in 48 experiments. Our results indicate that deep architectures generally outperform shallow ones, particularly with higher-dimensional data, but shallow models can occasionally excel in simpler scenarios. Notably, shallow 3DCNNs exhibited higher susceptibility to convergence issues with higher-dimensional data. Statistical tests, including ANOVA and paired T-Tests, were conducted to assess the significance of performance differences. This study highlights the importance of selecting appropriate model architectures and dimensionality reduction methods based on dataset characteristics, providing insights into the effective segmentation of hyperspectral images.

Keywords: Hyperspectral Images, HSI, Image Segmentation, Semantic Segmentation, Neural Networks, Dimensionality Reduction

1. Introduction

Hyperspectral imaging (HSI) is an advanced imaging technology that captures a wide spectrum of light across contiguous wavelength bands. Unlike conventional imaging, which captures light in the red, green, and blue (RGB) bands, HSI captures hundreds of bands, providing detailed spectral information for each pixel in an image. This rich spectral data enables the identification and classification of materials based on their spectral signatures, making HSI a powerful tool in various fields, including advanced agriculture, chemistry, medicine [1, 2, 3], and many others.

Segmenting HSI involves classifying each pixel into distinct categories based on their spatial and spectral information. The main issue that arises from this task is the high

dimensionality of the data, which can be computationally expensive to process. This issue makes Dimensionality Reduction (DR) very useful for HSI classification [4]. DR methods are used to simplify the data without losing relevant information. More details will be described in Section 2.1.

This project aims to explore the behavior of different neural network architectures in segmenting hyperspectral images by leveraging dimensionality reduction techniques. We employed Principal Component Analysis (PCA) and Independent Component Analysis (ICA) generating datasets with varying numbers of components. Four neural network models were trained on these datasets: two 3D Convolutional Neural Networks (3DCNNs) and two Recurrent Neural Networks (RNNs), each with shallow and deep variants.

Our primary objective is to analyze the performance of these neural network architectures with different dimensionality reduction methods and component sizes. We also aim to test the null hypothesis that there is no significant difference arising from an increase in the number of components or in the depth of a network. The evaluation is based on Overall Accuracy and Intersection over Union (IoU) scores, supplemented by various visualization techniques to gain deeper insights.

This document is structured as follows: Section 2 shows related works that also worked with HSI using similar methods or neural network architectures, as well as it explains how the DR methods work and how is HSI data inputted into 3DCNNs and RNNs. Section 3 exposes our pipeline for processing, training, and classifying the data. In Section 5 we present and discuss the results while highlighting insights that could be retrieved from statistical tests. Finally, 6 concludes our study and defines some of the possible future works that can extend our research.

Code for the experiments and resuls are stored in the following Github repository: **https://github.com/LuccaMartins/DS-Lab-1**, as well as all the trained models, their inferences and notebooks for analysis.

2. Related Work

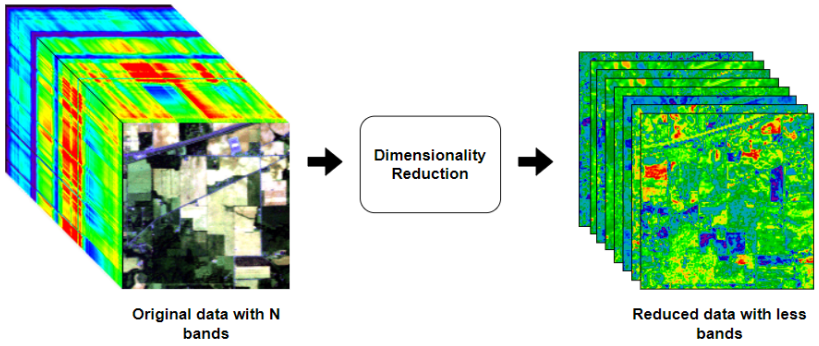
Different approaches have been used for the task of HSI segmentation. In our work, we are focusing on neural networks and how they behave with different Dimensionality Reduction strategies.

2.1. Dimensionality Reduction in HSI data

The high spectral dimensionality of HSI is a strong source of information for recognizing patterns. However, this dimensional complexity becomes a problem regarding computational capability. The Salinas Scene (one of the most widely used datasets for HSI classification), for instance, contains a spatial resolution of 512x217 pixels, and 224 spectral bands. In total, we have 512 (width) x 217 (height) x 224 bands = 24.887.296 data points. This huge amount of information can be challenging for data processing and analysis [5]. Also, some bands are strongly correlated, thus containing redundant information along different bands for the same pixel, which may complicate a classifier performance [6]. For these reasons, Dimensionality Reduction plays a massive role in HSI analysis by simplifying the data into

a more succinct structure, avoiding redundant information, and finding what may help the most in the classification task. Different methods are used for Dimensionality Reduction in the context of HSI, being PCA and ICA the most common ones [4]. The process of dimensional reduction in HSI data is illustrated in the image 1.

Figure 1: Dimensionality Reduction in HSI data (ilustration shows the Indian Pines dataset).



The most commonly used method is PCA [7]. It is a linear technique that projects the original data onto a lower-dimensional subspace, while retaining as much variability as possible. To use the data as input for PCA, the original hypercube of spectral bands is transformed into a matrix in which each row corresponds to a pixel, and each column corresponds to one spectral band. PCA calculates the eigenvectors and eigenvalues of the matrix, where eigenvectors represent the principal components, and eigenvalues indicate the variance explained by each component. All the eigenvectors are ranked by their eigenvalues, so that the first few components captures the most variability in the data. Each component represents a linear combination of the original spectral bands. We can use this rank to select the K principal components of the data.

While PCA focuses on capturing the maximum variance in the data, ICA aims to uncover statistically independent components. In HSI, each pixel contains spectral information across multiple wavelengths, representing a mixture of different materials and sources. ICA seeks to decompose these mixed signals into components corresponding to a distinct source or phenomenon present in the scene. ICA also operates linearly with the same input matrix used for PCA, but aims to find components that are statistically independent rather than orthogonal. While in PCA we rank the components by their respective amount of explained variance, in ICA we rank these components by considering their statistical independence from the other components.

A similar study in relation to ours was developed by Kang *et al* [8], analyzing classification methods' performances for different Dimensionality Reduction techniques over the Xiongan dataset [9]. They randomly selected 2000 samples, divided into 20 categories of land cover (which are mostly cash crops), being 100 samples of each category. Their study considered 9 DR methods, being 4 linear methods: PCA, KPCA-linear, ICA, and SVD; and 5 non-linear methods: KPCA-rbf, Isomap, LLE, LE, and Autoencoder. The methods

reduced the spectral bands to 15, based on a trade-off between computational complexity and accuracy with minimum information loss. For the classification, 3 methods were considered: Logistic Regression (LR), Multi-layer Perceptron (MLP), and Random Forest (RF). They used 1000 samples for training, and 1000 for testing the 3 classification methods. The results has shown that the classification accuracy was higher for any of the 3 methods when applying linear DR methods in the data, except for ICA. This former method, as the nonlinear methods, generated poor separability among the data points, leading to poor classification performance. Also, the nonlinear methods had higher computational complexity. The conclusion states that PCA had the best DR performance in spectral and spatial domain.

2.2. Traditional Methods for HSI Classification

In the work of Evgeny [10], 3 classical segmentation techniques were numerically evaluated on HSI data: clustering, region-growing, and watershed transform. PCA was the only method used for Dimensionality Reduction. After the DR and segmentation, an optional region merging procedure was added to avoid over-segmentation. For the clustering technique, the well-known K-Means was adopted and its result was tested varying the number of clusters from 10 to 100. For the region-growing algorithm, the seed points was set as local minimums of the absolute value of a gradient image. For the watershed approach, water sources were placed on each local minimum. After applying the watershed transform to establish region boundaries, connected regions were extracted, and boundary pixels were classified based on nearest neighbor rule. The Salinas and Indian Pines dataset were used in the experiments. As evaluation measures for the segmentation, Global Consistency Error and Rand Index were used. The experiments showed that K-means provided the best results based on GCE, and the region growing approach provided a better RI measure. It was also stated that the used of DR did not give advantages to the segmentation technique based on watershed transform.

2.3. Neural Networks for HSI Classification

In the past few years, neural networks have surpassed conventional techniques in HSI classification tasks, mostly using Convolutional Neural Networks (CNN) [11]. A CNN is composed of a series of layers containing several neurons. In the context of HSI, a convolutional layer extracts low-dimensional features from the input data while preserving the spatial relationship between the input data pixels. When using 1D or 2D kernels, CNNs focus either on spatial or spectral information to classify HSI, not being able to combine both information, limiting the performance. However, 3D kernels can learn from spatial and spectral information altogether, being able to learn more complex spatiotemporal representations [12]. In Chen *et al* [13] it is specially shown that 3DCNNs outperforms 2DCNNs for hyperspectral image classification. Figure 4 shows how HSI data is fed into a 3DCNN.

A convolution operation involves sliding a small matrix, a.k.a. kernel or filter across the input image (or signal). At each position, the kernel is multiplied element-wise with the portion of the input it covers, and the results are summed up into a single value in the output. Multiple convolutions along all the input form a new image, a.k.a feature map.

With the resulting feature maps, CNNs can detect patterns in the input that may indicate specific characteristics of the original image or signal inputted. Image 2 shows how the kernel slides across 1D, 2D, and 3D inputs. The resulting output has the same number of dimensions of the original input, usually with a reduced size. Image 3 shows an example of a 2D convolution in image processing.

Figure 2: 1D, 2D, and 3D convolution processes. *Image source: https://daje0601.tistory.com/198*

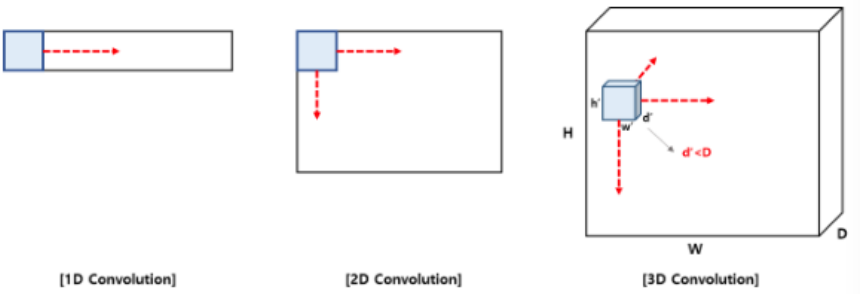
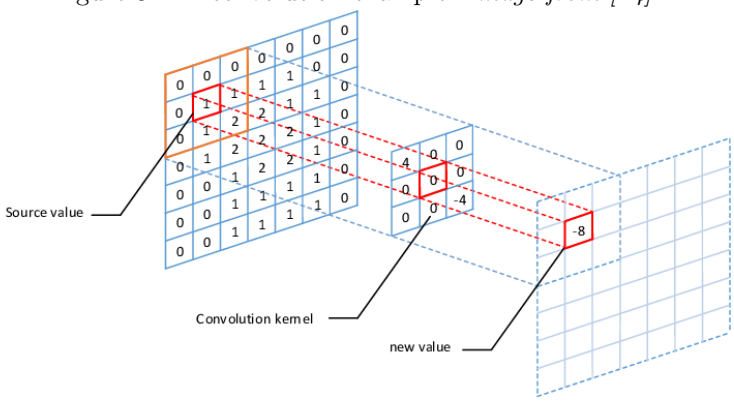
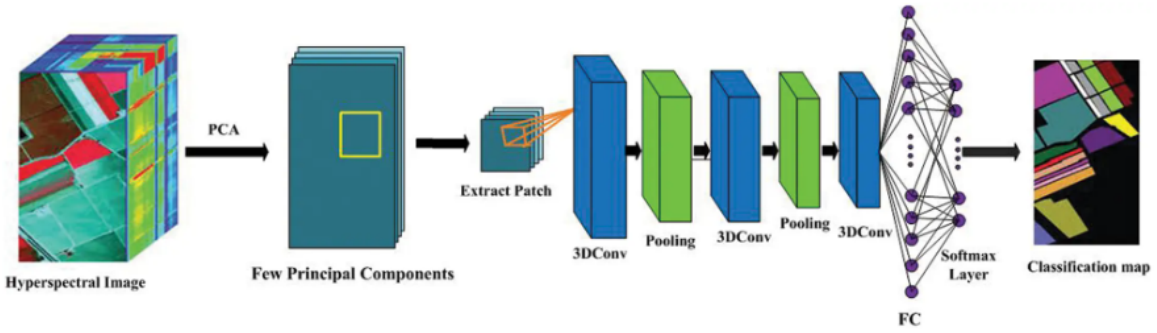


Figure 3: 2D convolution example. *Image from [14]*



An interesting approach merging 2D and 3D CNN was proposed by Roy *et al* [15], using a Hybrid Spectral Convolutional Neural Network (HybridSN) which is basically a spectral-spatial 3DCNN followed by a spatial 2DCNN. According to the authors, it is evident that using each of these CNNs separately had some shortcomings. The 2DCNN alone misses channel relationship information, and a 3DCNN generates high computational complexity and can get confused when classes have similar textures over different spectral bands. The proposed model assembles 2D and 3D convolutional layers utilizing both spectral and spatial feature maps aiming to the maximum possible accuracy. As in other studies, the datasets involved were Indian Pines, Salinas, and Pavia University. PCA is used to reduce spectral redundancy along spectral bands. The results of the HybridSN were compared with the most widely used supervised methods at that time, such as SVM, 2DCNN, 3DCNN, M3DCNN, and SSRN. By using the metrics Overall Accuracy, Average Accuracy, and Kappa coefficient, the authors observed that the proposed model outperforms all compared methods in each

Figure 4: Architecture of a 3DCNN model for HSI classification. *Image from: https://www.techscience.com/CMES/v133n2/48965/html*



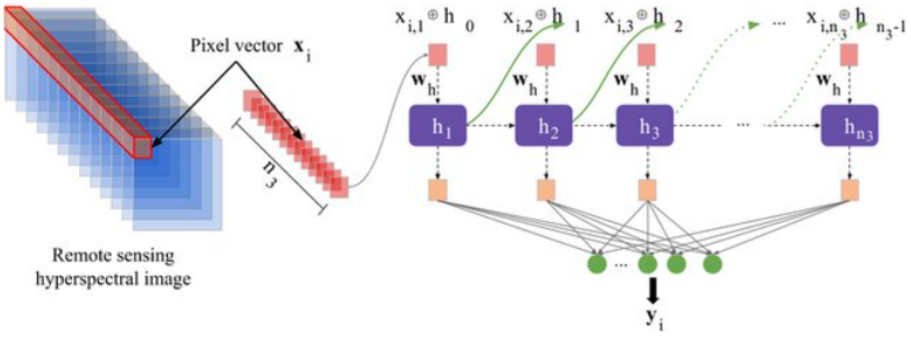
dataset. Curiously, the 2DCNN outperformed the 3DCNN for the Salinas Scene, which the authors believe to be due to 2 classes in this dataset having very similar textures over most spectral bands, confusing the model.

Also using the most popular datasets Salinas, Indian Pines, and Pavia University, Ahmad [16] firstly applied Incremental Principle Component Analysis (iPCA) to reduce redundant bands while maintaining spatial dimensions. Then a 3DCNN model is proposed including multiple convolutional layers to extract spatial-spectral features. The dataset is divided into overlapping 3D spatial patches, with ground labels formed based on central pixels. These patches cover spatial windows and are convolved with 3D kernel functions, followed by activation functions to introduce nonlinearity. A comparative analysis was performed against state-of-the-art methods, including Multi-scale-3D-CNN, 3D and 2D CNNs, showed competitive or improved results for each of the three datasets selected.

Recurrent Neural Networks (RNN) were also implemented for HSI classification considering the data to be sequential along the multiple wavelengths. In this approach, for a given pixel, the information on each wavelength is fed into the recurrent layers one by one. Figure 5 illustrates HSI data as an RNN input. There are 3 types of Recurrent Neural Networks, as categorized in [17]: vanilla RNNs, Long Short-Term Memory (LSTM) [18], and Gated Recurrent Unit (GRU) [19]. In the context of HSI, The vanilla RNN follows the intuition of simply passing information of each band through the recurrent layers, what leads to the vanishing gradient problem [17]. LSTM models introduce a memory cell to regulate the flow of information by selectively retain or drop information over long sequences, what efficiently addresses the vanishing gradient problem. GRU, similarly to LSTM, employs a gating mechanism, resetting and updating gates to control the information flow through the network. In our work, we chose to work with GRU as found in the literature, and also given its simpler architecture that makes it computationally more efficient then LSTM while also addressing the vanishing gradient problem.

The firt suggestion of using RNN for HSI classification appears in Mou *et al* [21]. The architecture is implemented by assuming that recurrent layers can efficiently learn short-term and long-term patterns along the spectral domain (more information about the proposed architecture can be found in 4.1, as it was one of the chosen models for our experiments).

Figure 5: HSI as input for Recurrent Neural Network. *Image from [20]*



Wu *et al* [22] implement a mixed architecture with Convolutional and Recurrent layers, utilizing initial convolutional layers to extract position-invariant middle-level features and recurrent layers to capture spectral-contextual details. Zhou et al. [23] also introduced an integrated fusion network combining CNN and GRU. In this model, CNN is responsible for extracting spatial features, while GRU handles both feature-level and decision-level fusion.

3. Methods

This section presents the data pipeline of our experiments, details about the datasets, and description of the metrics used to evaluate the models.

3.1. Datasets

The Salinas Scene was collected by NASA's Jet Propulsion Laboratory (JPL) using the AVIRIS sensor over Salinas Valley, California. It has a high spatial resolution of 3.7-meter pixels. The area covered comprises 512 lines by 217 samples. The sensor captures 224 bands, out of which 20 water absorption bands were discarded, as usual [15, 16, 24]. The area covered includes vegetables, bare soils, and vineyard fields. The ground-truth for each pixel contains 16 classes, as shown in 6.

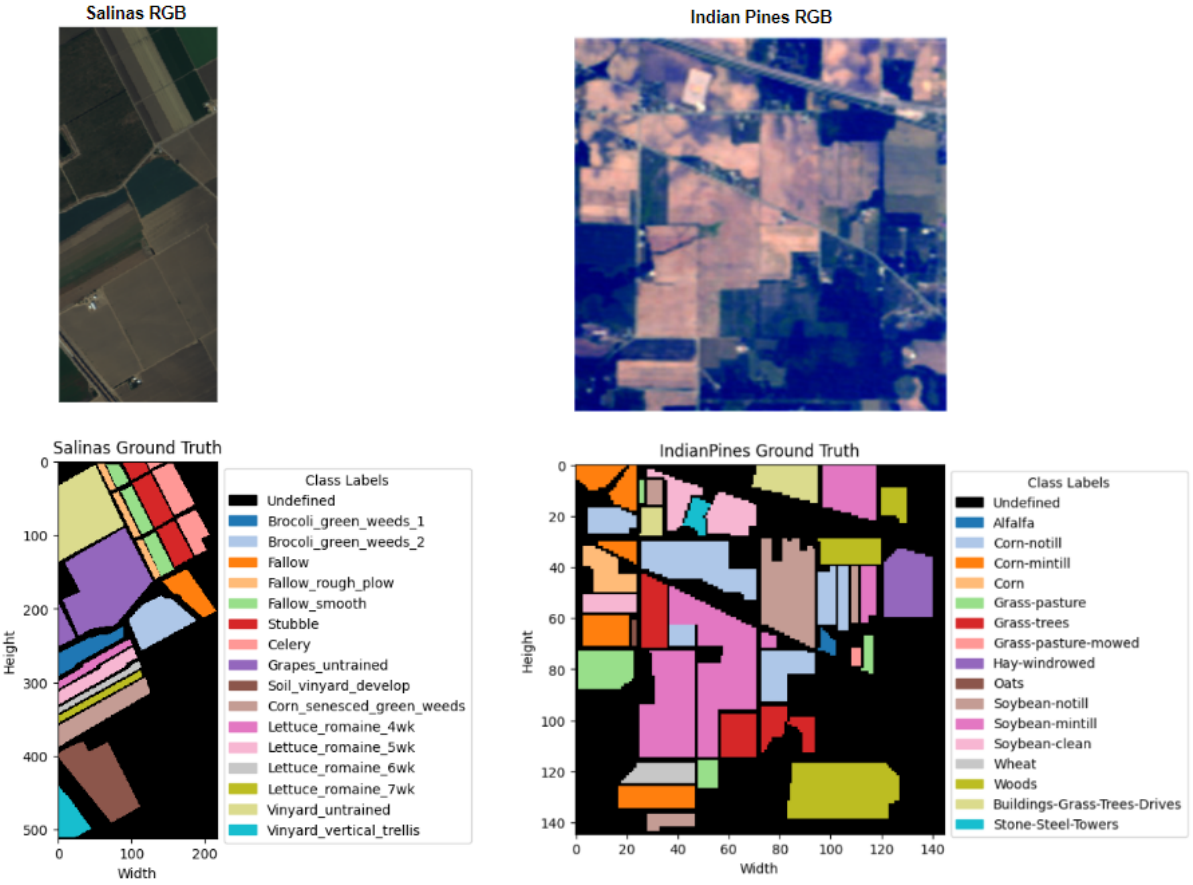
Also collected by JPL using the AVIRIS sensor, the Indian Pines dataset was collected over a single landscape in Indiana, US. The image contains 145×145 pixels. For each pixel, the data set contains 220 spectral reflectance bands, out of which 20 water absorption bands were also discarded. There are also 16 classes in this dataset, as shown in 6.

3.2. Pipeline

This section aims to explain the data flow from the original dataset to the performance analysis of the models, which is illustrated in Figure 7. For clearance, the datasets were downloaded from Kaggle, with the 20 water absorption bands already removed.

The first step is to apply a dimensionality reduction method to transform the hyper-dimensional original data into a simplified structure with a reduced shape. With the reduced data, the second step is to split the data into training and test sets. In our experiments, we split the data by taking half of it for each subset. Also, the training set is split into

Figure 6: Classes in Salinas dataset and ground truth image.



a validation set with 5% of the samples. Image 7 illustrates the output of applying a DR method to reduce the Indian Pines dataset to 8, 15, or 30 components, and the data sampling process.

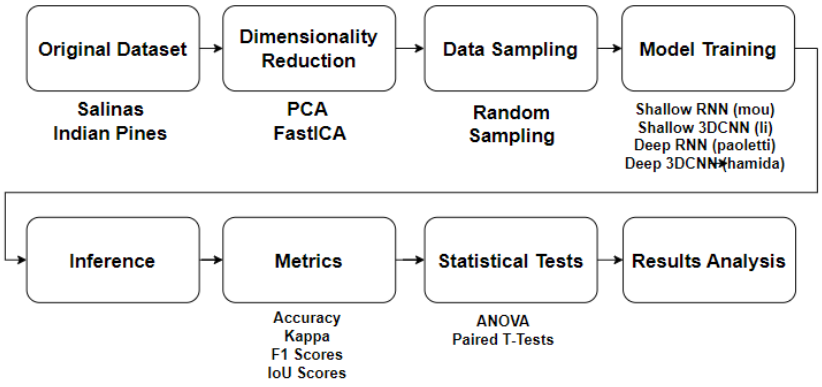
With the training samples, models were trained for 30 epochs (details about the training settings for each model can be found in the following subsections), and tested using the metrics described in 3.3. It is relevant to highlight that the data sampling happens before every training, i.e., each model was trained with different randomly generated subsets of the reduced dataset.

After running all the experiments (detailed in Section 4), the metrics results were analyzed with statistical tests in order to draw conclusions about the models performances. In this stage, accuracy and IoU scores of the inferences were compared using ANOVA and paired T-Tests.

3.3. Metrics

For evaluating performances of the trained models, we collected the Overall Accuracy (OA), Kappa Coefficient, also F1 and Intersection over Union (IoU) scores per class. We

Figure 7: Data flow from the original dataset to the performance analysis of the models.



focused on the two widely used OA and IoU scores [25], which formulas are shown in Figure 8. By analyzing the IoU scores per class, we have a fine-grained evaluation of performance for each class, enabling us to identify specific classes for which the model may be specially performing better or worse.

Figure 8: Formulas for Overall Accuracy (OA) and IoU Score (IoU).

$$OA = \frac{TP + TN}{P + N} \qquad IoU = \frac{TP}{TP + FP + FN}$$

4. Experiments

All experiments were executed in a Dell Inspiron 7572 with Intel(R) Core(TM) i7-8550U CPU of 1.80GHz, 16GB of RAM, and using CUDA with the GeForce MX150 GPU from Nvidia.

For each dataset, we created 6 new datasets by applying PCA and ICA with different sizes i.e., we reduced the dimensionality to 8, 15, and 30 components. Image 9 shows the datasets produced from the original Salinas dataset. The very same was done using the original Indian Pines dataset, totaling 12 datasets. Dimensionality reduction was performed using Scikit-learn's implementation of PCA and ICA (FastICA). Data sampling was made randomly with the proportion as showin in image 10.

Each of the 4 models were trained with each of the 12 datasets produced, totaling 48 experiments, named with their corresponding dataset, dimensionality reduction method, number of components, and model, respectively (e.g., Salinas_PCA_8_hamida).

4.1. Implemented Models

We are analyzing two different architectures of neural networs: Recurrent (RNN), specifically with Gated Recurrent Unit (GRU) layers, and Convolutional (CNN), specifically 3DCNNs. Convolutional Neural Networks are widely used for image tasks, whereas Recurrent

Figure 9: Datasets produced from Salinas.

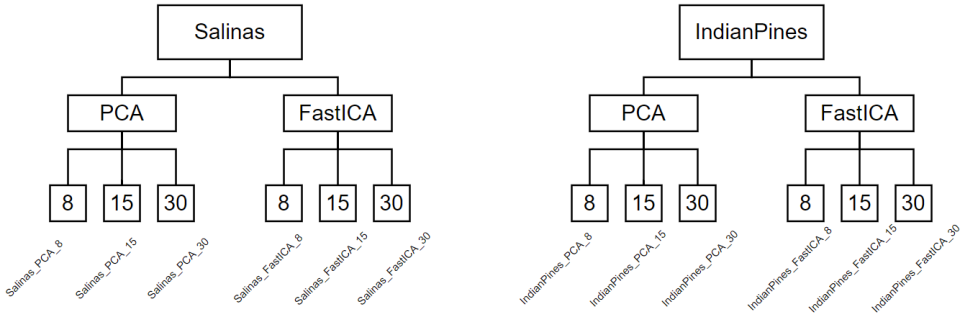
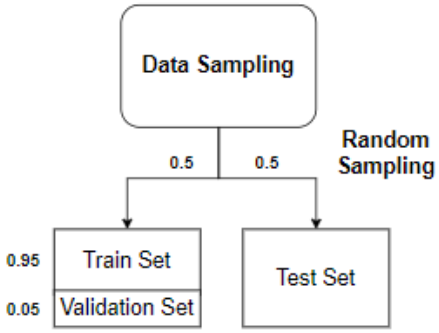


Figure 10: Data sampling.



Neural Networks are more often used for signal data, time-series, and Natural Language Processing [26]. In order to equivalently compare these architectures, we chose 2 shallow, and 2 deep networks of each from the literature. All models were implemented using PyTorch, and based on the DeepHyperX toolbox used in [27]. For all experiments, we used patch size 7x7. Cross Entropy Loss was chosen as criterion.

The shallow CNN considered here was proposed by Li *et al* [28], containing two sequential 3D convolutional layers with 16 3x3 kernels each, followed by a ReLu activation, and a final fully connected layer. As mentioned by the authors, the learnable 3D kernels convolve the input data at each convolution layer reducing the samples size to 1x1, thus being enough for the model to have two convolutional layers. The Stochastic Gradient Descent (SGD) optimizer was used with learning rate 0.01, as in the toolbox.

The chosen shallow RNN was proposed by Mou *et al* [21], and contains 1 GRU layer with hidden size of 64, followed by 1D batch normalization, and the so called Parametric Rectified Tanh activation function (PRTanh), and finally a fully connected layer. In our case, we used the PyTorch Tanh activation layer instead of the PRTanh. Also, instead of using Adadelta as an optimizer, we chose Adam with learning rate 0.001, as it empirically shown better performance in our case.

As a deep 3DCNN, we chose the model proposed by Hamida *et al* [29]. In this model,

blocks of CNN layers are stacked on top of each other ensuring an efficient representation of the image. The first block contains 2 3D convolutional layers interleaved by 2 convolutional pooling layers. The second block contains 2 more 3D convolutions layers followed by a fully connected layer. The first convolutional layer contains 20 3x3 kernels, the second layer contains 35 3x3 kernels, and the third and fourth contain 35 1x1 kernels. As for the shallow 3DCNN, here we also used SGD with learning rate of 0.01.

In Paoletti *et al* [17], different architectures are implemented and compared. The proposed deep RNN architecture is implemented with regular RNN layer, GRU or Long Short-Term Memory (LSTM). Given that we have a shallow RNN which uses GRU, we implemented the deep RNN also with GRU layers. The only difference to the shallow RNN is one extra GRU layer with Tanh activation function right after the first one. We must highlight here that we are following the agreement established to distinguish shallow and deep architectures in [30], whereby architectures with two or more hidden layers are considered as Deep Neural Networks. Adam optimizar with learning rate 0.001 was also used here.

5. Results and Discussion

Our experiments summed up to 48 results for analysis (4 models trained with 12 different datasets). This section aims discuss the behavior of each neural network architecture with each dimensionality reduction method, while addressing our null hypothesis regarding the significance of the difference between shallow and deep architectures. Also, common difficulties raised from characteristics of each dataset will be highlighted, as well as models that did not converge. The analysis is based on Overall Accuracy and IOU Scores, aligned with different visualization methods.

Figure 11 shows a rank of accuracies for all the experiments in each dataset. In general, for experiments with the Salinas dataset, all models performed regularly well in datasets originated from DR with PCA. Thus, we may be able to say that the number of components did not significantly affect each model's performance, which aligns with the null hypothesis regarding the number of components (i.e., performances' accuracies are not proportional to the number of components presented in the dataset). The same does not happen with other sets of experiments, in which some models does not converge. However, this type of statements can only be confirmed with statistical tests, as describe in Section 5.3.

5.1. Models that did not converge

From image 11 we can see that in 5 experiments the models did not converge (3 with Salinas and 2 with Indian Pines). Only shallow architectures did not converge, and it only happened with a higher number of components (15 or 30). This suggests that shallow architectures may not be able to learn from higher-dimensional data.

The experiments that did not converge were: *Salinas_FastICA_15_li*, *Salinas_FastICA_30_li*, *Salinas_FastICA_30_mou*, *IndianPines_PCA_30_li*, and *IndianPines_FastICA_30_li*. As we can see, 4 out of 5 experiments that did not converge had the shallow 3DCNN as model. This observation suggests a potential susceptibility of shallow 3DCNNs to failing to learn

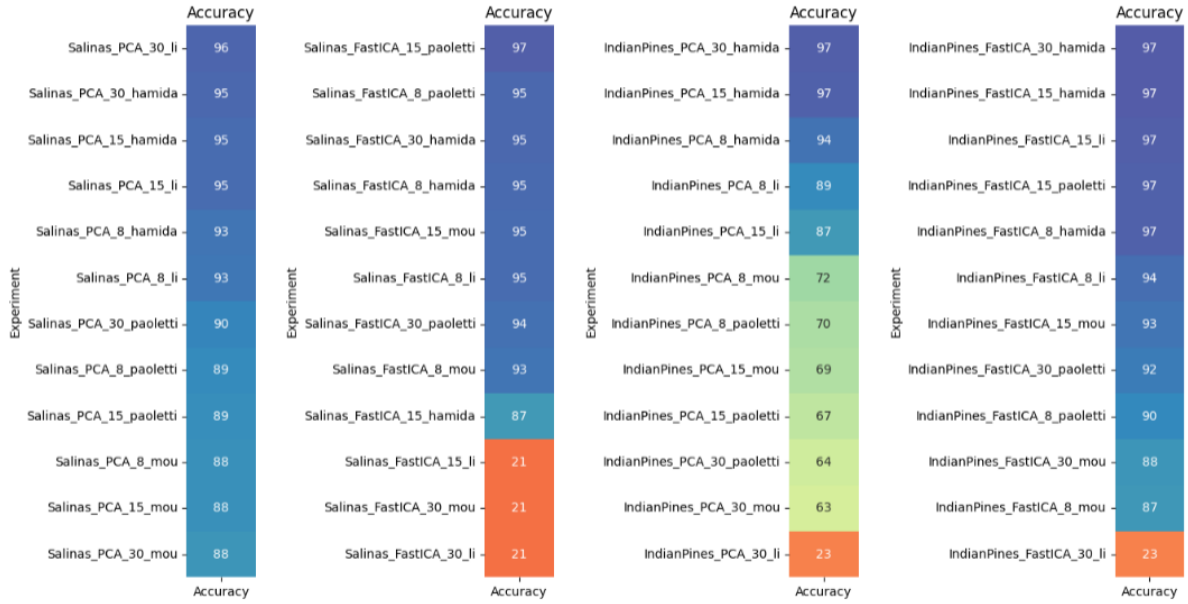


Figure 11: Rank of accuracies in each dataset by DR method

from higher dimensions. This may be led by the lack of depth necessary to capture underlying patterns in complex data, which is not a problem for deep networks of both architectures.

Figure 12 shows two examples of Confusion Matrices of performances from models that did not converge. As we can see, the misclassifications are accumulated in the majority class *Grapes_untrained*. This may have arose from the fact that the classes are very unbalanced, thus the model tends to be biased towards the majority class. This issue can be solved with Class Balancing, which may be included in future works.

5.2. Overlapping of Classes

Two classes in the Salinas dataset are normally a source of confusion for the models, they are the *Grapes_untrained* and *Vinyard_untrained*. The confusion raises from the fact the these types of land are actually very similar, thus some misclassifications are actually expected. Figure 13 shows how all the models tend to mix these classes. The same aspect can be visualized with the confusion matrices as in Figure 15. The example shows models that achieved a very good accuracy (around 90%), and had the great majority of misclassifications with the classes cited.

For the sake of completeness, Figure 14 shows the classification with the same configuration on the Indian Pines dataset. We can also note that the classes *Soybean-mintill* and *Corn-notill* are often misclassified between themselves, but this is not so evident as it is for the classes in the Salinas dataset.

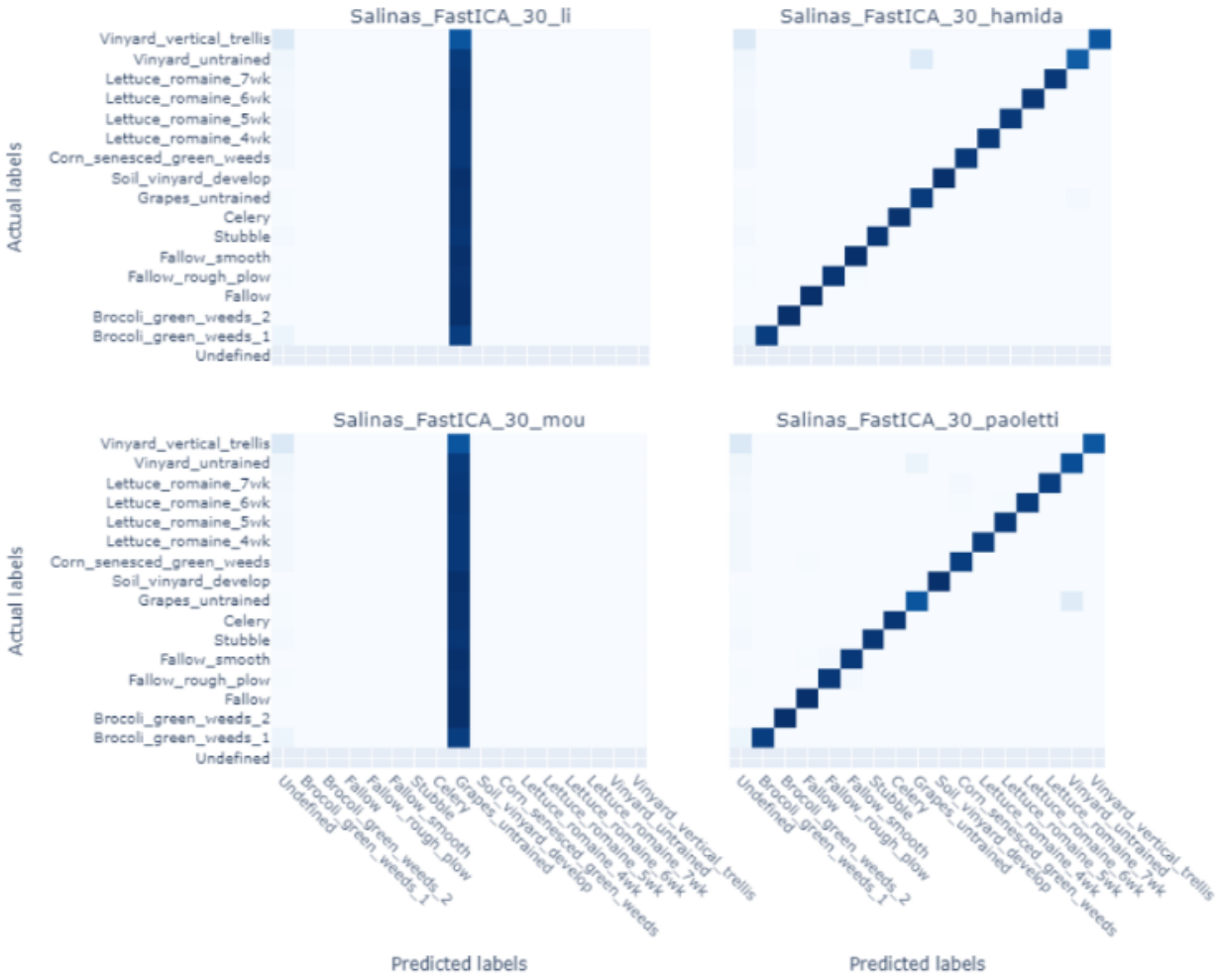
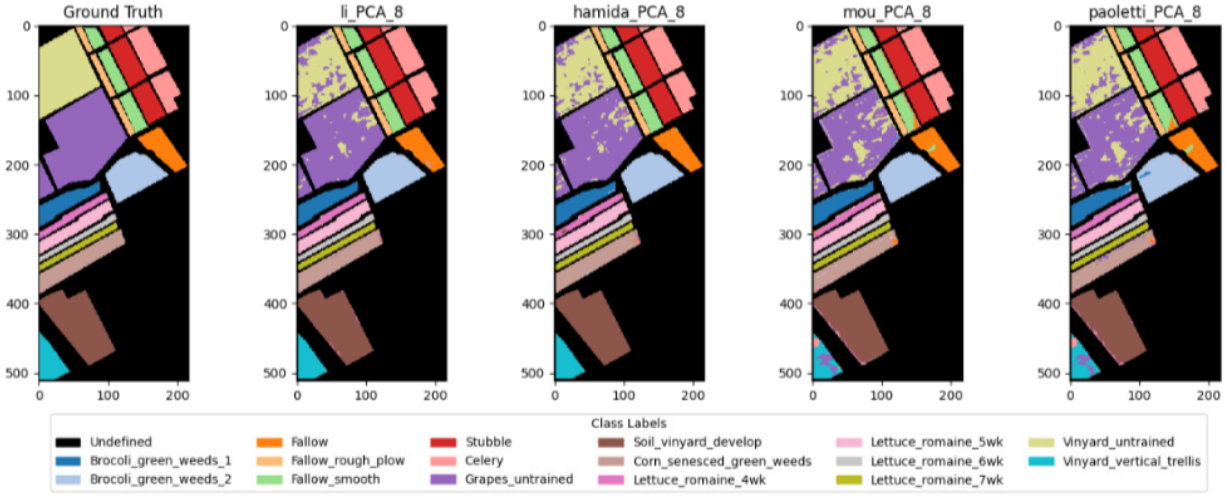


Figure 12: Example of Confusion Matrices for models that did not converge

Figure 13: Example of models confusion with the classes *Grapes_untrained* and *Vinyard_untrained* in the Salinas dataset.



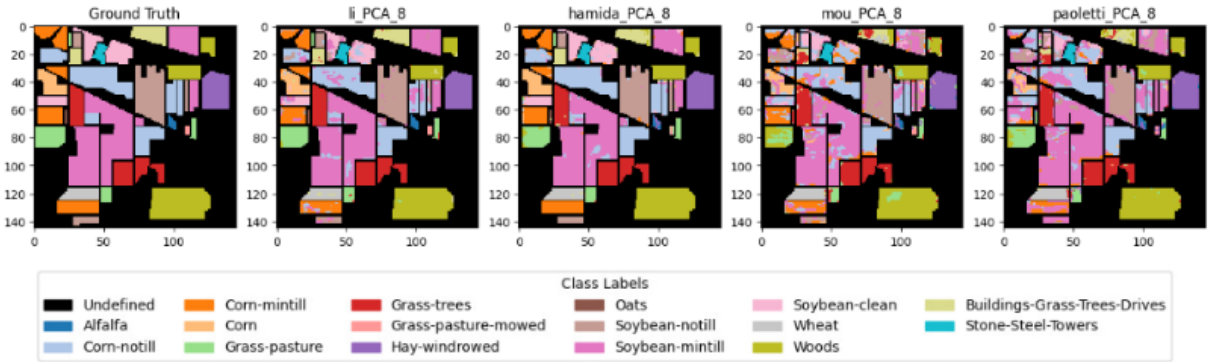
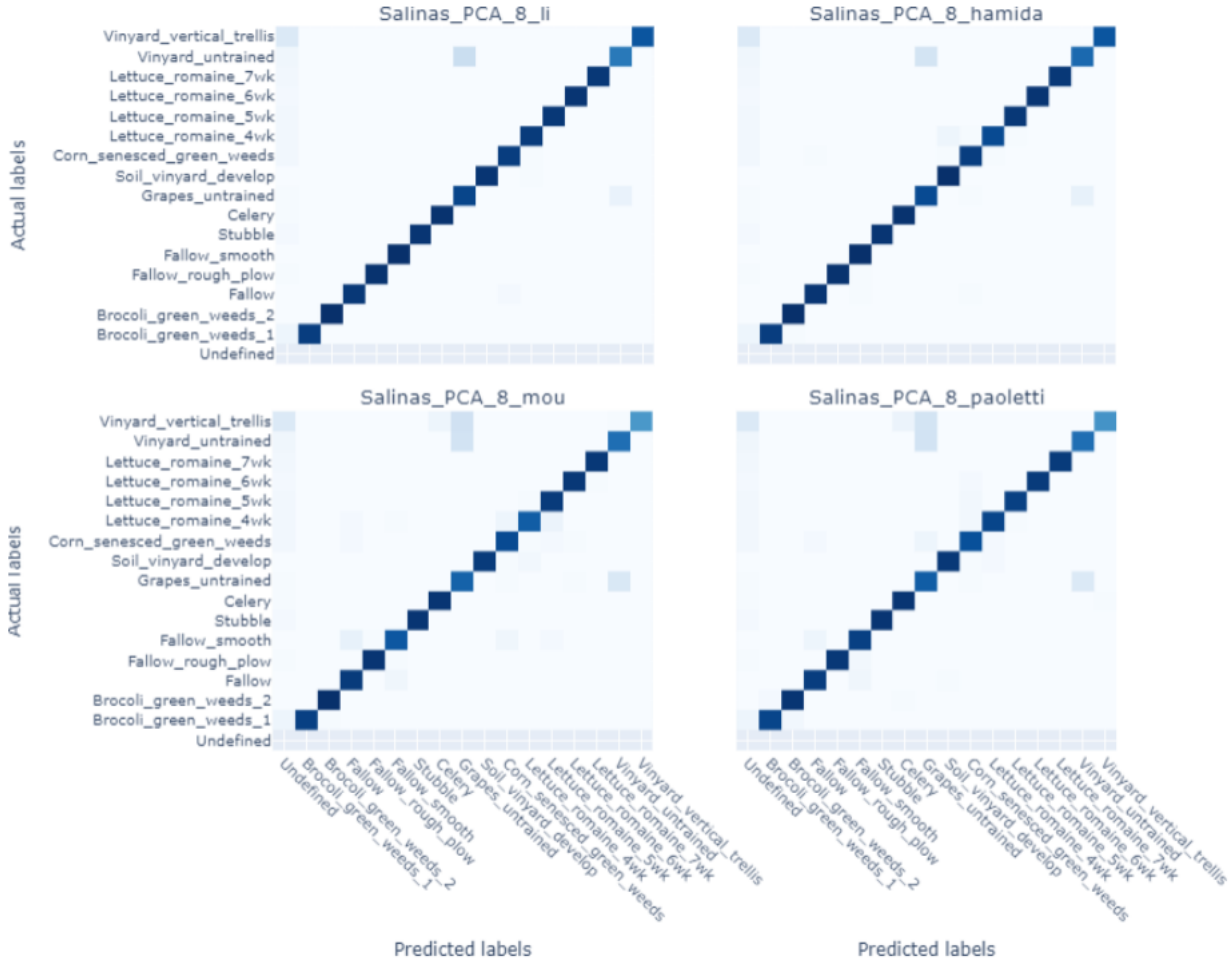


Figure 14: Predictions of each model on Indian Pines dataset with PCA applied resulting on 8 components.

Figure 15: Example of models confusion with the classes *Grapes_untrained* and *Vinyard_untrained* in the Salinas dataset shown in confusion matrices.



5.3. Statistical Tests

In order to establish whether there is or not a statistically significant difference in performance of different sets of experiments, we used ANOVA test and paired T-Tests.

5.3.1. ANOVA

ANOVA was applied to 4 different sets of experiments (grouped by dataset and DR method) in order to compare the models based on their accuracies.

The test applied to the models' performances with datasets originated from the Salinas dataset after DR with PCA concluded that there was a significant difference in favor of 3DCNNs in relation to RNN, with both shallow or deep networks. Nonetheless, this difference is not our focus because we are not comparing one architecture against the other, but analyzing how each of them behave with different DR methods and sizes. The test applied to performance in the Salinas dataset with FastICA did not confirm any significant difference.

For performances with the Indian Pines dataset after DR with PCA, the ANOVA test showed a significant superiority of the deep 3DCNN (hamida) in relation to both shallow and deep RNNs (mou and paoletti), which is also not really relevant for us. Nevertheless, ANOVA test on performances with FastICA concluded a significant superiority of the deep RNN over the shallow RNN.

5.3.2. Paired T-Tests

Paired T-Tests were applied firstly to compare shallow and deep networks of the same architecture based only on IOU scores per class. Tables 1, 2, 3, and 4 show which models were found to be superior or if no significant difference was found (expressed by an "=" symbol). We will analyze each of these tables separately.

Firstly, we analyze the 3DCNNs in the Salinas dataset. In Table 1 we can see that the deep 3DCNN (Hamida) had a statistically superior performance over the shallow 3DCNN (Li) when ICA was used with 15 and 30 components. This happened because the shallow 3DCNN did not converge on those cases. However, the shallow 3DCNN is shown to be superior when PCA was used with 30 components. For the Indian Pines dataset, the superiority of the deep 3DCNN appears only on higher-dimensional data produced with both DR methods, see Table 2. It is important to highlight that the shallow 3DCNN did not converge in Indian Pines with 30 components, when PCA or ICA was used. Thus, there was 6 cases in which the deep 3DCNN performed significantly better, 5 cases with no significant difference, and 1 case in which the shallow 3DCNN performed better. These paired T-Tests show that a deeper architecture not necessarily leads to a better performance.

Table 1: Paired T Test of Li (Shallow 3DCNN) and Hamida (Deep 3DCNN) on Salinas Dataset

	8	15	30
PCA	=	=	li
FastICA	=	hamida	hamida

Table 2: Paired T Test of Li (Shallow 3DCNN) and Hamida (Deep 3DCNN) on Indian Pines Dataset

	8	15	30
PCA	=	hamida	hamida
FastICA	=	hamida	hamida

Analyzing the T-Tests with RNNs, we can see that the deep RNN (Paoletti) was the only one to show superiority. For the Salinas dataset, the deep RNN was not superior only when using PCA with 15 components and ICA with 8 components. In these cases, no significant difference was found, see Table 3. For the Indian Pines dataset, the deep RNN showed superiority only in higher-dimensional data where ICA was applied. Thus, it is shown that when PCA was used, no significant difference was found for any number of components.

For the sake of completeness, we also applied paired T Tests to compare the shallow models and the deep models separately. Although this is not the focus, it is interesting to analyze if 3DCNNs really showed a superior performance over RNNs in general or not.

Table 3: Paired T Test of Mou (Shallow RNN) and Paoletti (Deep RNN) on Salinas Dataset

	8	15	30
PCA	paoletti	=	paoletti
FastICA	=	paoletti	paoletti

Table 4: Paired T Test of Mou (Shallow RNN) and Paoletti (Deep RNN) on Indian Pines Dataset

	8	15	30
PCA	=	=	=
FastICA	=	paoletti	paoletti

The paired T-Tests for deep architectures showed superiority on performance of the deep 3DCNN for most cases, see Table 5 and 6. The only exceptions happens with FastICA in the Salinas dataset, in which no significant difference was found with 8 components, and the deep RNN was superior with 15 components.

Table 5: Paired T Test of Hamida (Deep 3DCNN) and Paoletti (Deep RNN) on Salinas Dataset

	8	15	30
PCA	hamida	hamida	hamida
FastICA	=	paoletti	hamida

Table 6: Paired T Test of Hamida (Deep 3DCNN) and Paoletti (Deep RNN) on Indian Pines Dataset

	8	15	30
PCA	hamida	hamida	hamida
FastICA	hamida	hamida	hamida

For the shallow models things are more balanced. As seen in Tables 7 and 8. In the Salinas dataset, the shallow 3DCNN was superior with any number of components when PCA was used, but the same did not happen with ICA, in which the shallow RNN was superior with 15 components (given that the shallow 3DCNN did not converge), and no significant difference was found for 8 or 30 components. For the Indian Pines dataset, the shallow 3DCNN was superior to the shallow RNN with 8 and 15 components for both PCA and ICA, while the shallow RNN was superior with 30 components.

Table 7: Paired T Test of Li (Shallow 3DCNN) and Mou (Shallow RNN) on Salinas Dataset

	8	15	30
PCA	li	li	li
FastICA	=	mou	=

Table 8: Paired T Test of Li (Shallow 3DCNN) and Mou (Shallow RNN) on Indian Pines Dataset

	8	15	30
PCA	li	li	mou
FastICA	li	li	mou

6. Conclusion and Future works

As we could analyze with the statistical tests, it is fair to say that, in general, deeper architectures tend to get better results, though this does not happen in every case. Also, by checking the accuracies we could see that the number of components does not proportionally affects the models' performances. In other words, having more components does not necessarily leads to a better performance, confirming our null hypothesis.

Regarding the comparison between different architectures, 3DCNNs outperformed RNNs in general, though it was not true for every case. As a future work, other architectures can be included in the study, such as Autoencoders, as in [31, 32]. Also, the issue regarding unbalanced classes can be addresses with different class balancing techniques, which opens the way for a different research branch.

References

[1] L. C. Garcia Peraza Herrera, C. Horgan, S. Ourselin, M. Ebner, T. Vercauteren, Hyperspectral image segmentation: a preliminary study on the oral and dental spectral image database (odsi-db), Computer Methods in Biomechanics and Biomedical Engineering: Imaging amp; Visualization 11 (4) (2023) 1290–1298. doi:10.1080/21681163.2022.2160377.
URL http://dx.doi.org/10.1080/21681163.2022.2160377

[2] J. Sellner, S. Seidlitz, A. Studier-Fischer, A. Motta, B. Özdemir, B. P. Müller-Stich, F. Nickel, L. Maier-Hein, Semantic segmentation of surgical hyperspectral images under geometric domain shifts (2023). arXiv:2303.10972.

[3] S. Seidlitz, J. Sellner, J. Odenthal, B. Özdemir, A. Studier-Fischer, S. Knödler, L. Ayala, T. J. Adler, H. G. Kenngott, M. Tizabi, M. Wagner, F. Nickel, B. P. Müller-Stich, L. Maier-Hein, Robust deep learning-based semantic organ segmentation in hyperspectral images, Medical Image Analysis 80 (2022) 102488. doi:10.1016/j.media.2022.102488.
URL http://dx.doi.org/10.1016/j.media.2022.102488

[4] D. Lupu, I. Necoara, T. C. Ionescu, L. Ghinea, J. Garrett, T. A. Johansen, Dimensionality reduction of hyperspectral images using an ica-based stochastic second-order optimization algorithm, in: 2023 European Control Conference (ECC), 2023, pp. 1–6. doi:10.23919/ECC57647.2023.10178317.

[5] P. Ghamisi, N. Yokoya, J. Li, W. Liao, S. Liu, J. Plaza, B. Rasti, A. Plaza, Advances in hyperspectral image and signal processing: A comprehensive overview of the state of the art, IEEE Geoscience and Remote Sensing Magazine 5 (4) (2017) 37–78. doi:10.1109/MGRS.2017.2762087.

[6] C. Deepa, A. Shetty, D. Narasimha, Quality assessment of dimensionality reduction techniques on hyperspectral data: A neural network based approach, ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLIII-B3-2020 (2020) 389–394. doi:10.5194/isprs-archives-XLIII-B3-2020-389-2020.

[7] D. Ruiz Hidalgo, B. Bacca Cortés, E. Caicedo Bravo, Dimensionality reduction of hyperspectral images of vegetation and crops based on self-organized maps, Information Processing in Agriculture 8 (2) (2021) 310–327. doi:https://doi.org/10.1016/j.inpa.2020.07.002.
URL https://www.sciencedirect.com/science/article/pii/S221431732030189X

[8] L. Kang, X. Hu, C. Zhong, K. Zhang, Y. Jiang, Comparative study of different dimensionality reduction methods in hyperspectral image classification, Journal of Physics: Conference Series 2024 (1) (2021) 012009. doi:10.1088/1742-6596/2024/1/012009.
URL https://dx.doi.org/10.1088/1742-6596/2024/1/012009

[9] Z. L. Cen Y, et al., Aerial hyperspectral remote sensing classification dataset of xiongan new area (matiwan village), Journal of Remote Sensing (2020).

[10] E. Myasnikov, Hyperspectral image segmentation using dimensionality reduction and classical segmentation approaches, Computer Optics 41 (2017) 564–572. doi:10.18287/2412-6179-2017-41-564-572.

[11] G. Farooque, L. Xiao, J. Yang, A. B. Sargano, Hyperspectral image classification via a novel spectral–spatial 3d convlstm-cnn, Remote Sensing 13 (21) (2021). doi:10.3390/rs13214348.
URL https://www.mdpi.com/2072-4292/13/21/4348

[12] N. Noshiri, M. A. Beck, C. P. Bidinosti, C. J. Henry, A comprehensive review of 3d convolutional neural network-based classification techniques of diseased and defective crops using non-uav-based hyperspectral images, Smart Agricultural Technology 5 (2023) 100316. doi:https://doi.org/10.1016/j.atech.2023.100316.
URL https://www.sciencedirect.com/science/article/pii/S2772375523001454

[13] Y. Chen, H. Jiang, C. Li, X. Jia, P. Ghamisi, Deep feature extraction and classification of hyperspectral images based on convolutional neural networks, IEEE Transactions on Geoscience and Remote Sensing 54 (2016) 1–20. doi:10.1109/TGRS.2016.2584107.

[14] Z. Guyu, G. Huang, H. He, H. He, J. Ren, Regional spatiotemporal collaborative prediction model for air quality, IEEE Access PP (2019) 1–1. doi:10.1109/ACCESS.2019.2941732.

[15] S. K. Roy, G. Krishna, S. R. Dubey, B. B. Chaudhuri, Hybridsn: Exploring 3-d–2-d cnn feature hierarchy for hyperspectral image classification, IEEE Geoscience and Remote Sensing Letters 17 (2) (2020) 277–281. doi:10.1109/lgrs.2019.2918719.
URL http://dx.doi.org/10.1109/LGRS.2019.2918719

[16] M. Ahmad, A. M. Khan, M. Mazzara, S. Distefano, M. Ali, M. S. Sarfraz, A fast and compact 3-d cnn for hyperspectral image classification, IEEE Geoscience and Remote Sensing Letters 19 (2022) 1–5. doi:10.1109/lgrs.2020.3043710.
URL http://dx.doi.org/10.1109/LGRS.2020.3043710

[17] M. Paoletti, J. Haut, J. Plaza, A. Plaza, Deep learning classifiers for hyperspectral imaging: A review, ISPRS Journal of Photogrammetry and Remote Sensing 158 (2019) 279–317. doi:https://doi.org/10.1016/j.isprsjprs.2019.09.006.
URL https://www.sciencedirect.com/science/article/pii/S0924271619302187

[18] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Computation 9 (8) (1997) 1735–1780. arXiv:https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf, doi:10.1162/neco.1997.9.8.1735.
URL https://doi.org/10.1162/neco.1997.9.8.1735

[19] K. Cho, B. Merrienboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: Encoder-decoder approaches (09 2014). doi:10.3115/v1/W14-4012.

[20] M. Paoletti, J. Haut, J. Plaza, A. Plaza, Scalable recurrent neural network for hyperspectral image classification, The Journal of Supercomputing 76 (11 2020). doi:10.1007/s11227-020-03187-0.

[21] L. Mou, P. Ghamisi, X. X. Zhu, Deep recurrent neural networks for hyperspectral image classification, IEEE Transactions on Geoscience and Remote Sensing 55 (7) (2017) 3639–3655. doi:10.1109/TGRS.2016.2636241.

[22] H. Wu, S. Prasad, Convolutional recurrent neural networks forhyperspectral data classification, Remote Sensing 9 (3) (2017). doi:10.3390/rs9030298.
URL https://www.mdpi.com/2072-4292/9/3/298

[23] F. Zhou, R. Hang, Q. Liu, X. Yuan, Integrating convolutional neural network and gated recurrent unit for hyperspectral image spectral-spatial classification, in: J.-H. Lai, C.-L. Liu, X. Chen, J. Zhou, T. Tan, N. Zheng, H. Zha (Eds.), Pattern Recognition and Computer Vision, Springer International Publishing, Cham, 2018, pp. 409–420.

[24] H. Xu, W. Yao, L. Cheng, B. Li, Multiple spectral resolution 3d convolutional neural network for hyperspectral image classification, Remote Sensing 13 (7) (2021). doi:10.3390/rs13071248.
URL https://www.mdpi.com/2072-4292/13/7/1248

[25] Z. Xu, W. Zhang, T. Zhang, J. Li, Hrcnet: High-resolution context extraction network for semantic segmentation of remote sensing images, Remote Sensing 13 (12 2020). doi:10.3390/rs13010071.

[26] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Networks 61 (2015) 85–117. doi:10.1016/j.neunet.2014.09.003.
URL http://dx.doi.org/10.1016/j.neunet.2014.09.003

[27] N. Audebert, B. Le Saux, S. Lefèvre, Deep learning for classification of hyperspectral data: A comparative review, IEEE Geoscience and Remote Sensing Magazine 7 (2) (2019) 159–173. doi:10.1109/MGRS.2019.2912563.

[28] Y. Li, H. Zhang, Q. Shen, Spectral–spatial classification of hyperspectral imagery with 3d convolutional neural network, Remote Sensing 9 (1) (2017). doi:10.3390/rs9010067.
URL https://www.mdpi.com/2072-4292/9/1/67

[29] A. Ben Hamida, A. Benoit, P. Lambert, C. Ben Amar, 3-d deep learning approach for remote sensing image classification, IEEE Transactions on Geoscience and Remote Sensing 56 (8) (2018) 4420–4434. doi:10.1109/TGRS.2018.2818945.

[30] Y. Bengio, Y. LeCun, Scaling learning algorithms toward ai (2007).

[31] Z. Lin, Y. Chen, X. Zhao, G. Wang, Spectral-spatial classification of hyperspectral image using autoencoders, in: 2013 9th International Conference on Information, Communications & Signal Processing, 2013, pp. 1–5. doi:10.1109/ICICS.2013.6782778.

[32] J. Nalepa, M. Myller, Y. Imai, K.-I. Honda, T. Takeda, M. Antoniak, Unsupervised segmentation of hyperspectral images using 3-d convolutional autoencoders, IEEE Geoscience and Remote Sensing Letters 17 (11) (2020) 1948–1952. doi:10.1109/lgrs.2019.2960945.
URL http://dx.doi.org/10.1109/LGRS.2019.2960945