*Article*

# Tensor-based Factorization Algorithms for Pixel-wise Classification of Hyperspectral Data Using Deep Convolutional Networks

**Josué López [1,*], Deni Torres [1] and Clement Atzberger [3]**

[1] Center for Research and Advanced Studies of the National Polytechnic Institute, Telecommunications Group, Av del Bosque 1145, Zapopan 45017, Mexico; dtorres@gdl.cinvestav.mx
[3] University of Natural Resources and Life Science, Institute of Geomatics, Peter Jordan 82, Vienna 1180, Austria; clement.atzberger@boku.ac.at
* Correspondence: josue.lopez@cinvestav.mx

Version October 22, 2020 submitted to Remote Sens.

**Abstract:** Tensor-based algorithms for data compression have evolved in recent years according to the needs of several research areas. Tucker Decomposition (TKD) is one of the most popular factorization methods based on tensor algebra, but it is clear that it is not the only algorithm wich can produce a factorization for a given input data set. Besides, this decomposition does not have singular solutions, i.e., it converges to local minima. Hence, depending on the input data, tensor-based decompositions can achieve better solution to a specific input. The phenomenology of Remote Sensing (RS) Hyperspectral Images (HSI) belongs to the set of natural numbers, i.e., the set of positive integers. Hence, a non-negative tensor factorization suggest a more suitable decomposition for positive data by nature. The main purpose of this work is to prove the benefits in processing time, as well as in accuracy, of using a well-posed factorization algorithm. Specifically, this paper performs a quantitative analysis of tensor-based factorization algorithms applied to semantic segmentation of HSI using Deep Convolutional Networks (DCN).

**Keywords:** deep convolutional networks; hyperspectral imagery; tensor decomposition

## 1. Introduction

Big data compression has been one of the most active research areas in recente years []. The insertion of tensor-based algorithms for this sort of tasks drove a revolution in several areas such as image processing [].

Most of the researchs in the image processing area require data acquired by multiple sensors. Even in the simplest case, color images, data acquired by three sensors that perceive reflectance of an object are processed. Each sensor receives reflectance in different wavelength ranges, and by merging that data, a color image is produced. Thus, the images can be represented in a form of three-dimensional arrays or third-order tensors. Two dimensions represent the spatial properties and the third dimension denotes the "depth", or rather the spectral bands. Medical analysis [], minorology [], agriculture, [], radar images [] and, above all, remote sensing multi- and hyper-spectral images [] work, by nature, with third-order tensors.

Spectral imagery remarkably aids certain image processing tasks []. Recently the use of this type of data has grown exponentially in various areas []. The ability to obtain information about a target not only by its reflectance in the spatial domain, but also by response at different wavelengths, has driven a growth in accuracy and precision in tasks such as classification and segmentation [].

Initially, several unsupervised classification and segmentation algorithms [] were developed, taking advantage of the properties that spectral data produce. Subsequently, with the introduction of supervised machine learning algorithms such as SVM [], kNN [] and ANN [], it was found that, under certain conditions, there is a direct relation between the number of bands used and the performance of these algorithms []. However, with the aim of improving results, neural network models evolved into deep neural networks []. This caused that spectral image processing was not productive at all, since the processing time increased considerably. The foregoing requires having robust computer equipment to achieve competitive results in time.

Several works have opted for matrix factorization algorithm to reduce the high-dimensionality of spectral images []. More recently, with the development of tensor factorization algorithms [], it has been found that some algorithms based on tensor algebra produce certain advantages over those based on matrices []. Nevertheless, the data produced by both of them are hard to understand for supervised classification algorithms that need spatial relation between pixels to produce a wise prediction [].

In this work we propose an alternative solution to the problem described previously. To reduce processing times in supervised classification algorithms, such as deep neural networks, we propose a model that, from the benefits offered by tensor decomposition algorithms, aids compression of spectral images. This produces a lower dimensional tensor while preserving the structural and numerical nature of the original data.

### *1.1. State of art*

There are several works focused on the development of frameworks that reduce execution time of machine learning algorithms for semantic segmentation of hyperspectral data sets []. The crucial factor, which is addressed in this work, is to achieve compression of the input data to reduce the high number of computatios, but without sacrificing pixel accuracy, overall accuracy, precision and recall in the classification task.

Before the introduction of tensor decomposition algorithms, the way to use hyperspectral images as input for supervised classification algorithms was by band selection [23] and [22]. Later, matrix decomposition algorithms were used, such as PCA in [**?** ], and even non-negative matrix decomposition methods [**?** ]. In 2015 Zhang et al. [24] were pioneers in experimenting with multilinear algebra-based decompositions on hyperspectral images.

On the other hand, there was also the possibility of using multispectral images due to the small number of spectral bands, which still made efficient results in classification without dimensionality reduction achievable, as done in [11], [18], [21] and [**?** ]. However, the need to increase classification accuracy results forces researchers to use data with greater features that favor and aid the classification of various classes, which are difficult to differentiate with little spectral data. Thus, more recent researchs have decided to use hyperspectral images with tensor decompositions, which has increased the results in classification accuracy [26], [27], [29], [**?** ] and [**?** ].

Recently, Sayeh et al. [**?** ] published a work close to our research. They proposed a non-negative tensor decomposition of hyperspectral images but, different to our research, they try to preserve certain spatial-spectral features into the so called abundance maps, i.e. the projection matrices, while this work pursuits to preserve the nature of the image just compressing the main information in the positive core tensor.

Table 1 summarizes some of the most cited related papers, wich deal with the compression-classification issue.

**Table 1.** Related work in spectral imagery semantic segmentation.

| Reference | Input | Decomposition | Reduction | Classifier |
|---|---|---|---|---|
| Li, S. et al. [23] (2014) | HSI | - | Band selection | SVM |
| Zhang, L. et al. [24] (2015) | HSI | TKD | Spatial-Spectral | - |
| Wan, Q. et al. [22] (2016) | HSI | - | Band selection | SVM/kNN/CART |
| Kemker, R. et al. [11] (2017) | MSI | - | - | CNN |
| Tong L. et al. [] (2017) | HSI | NMF | Unmixing | - |
| Hamida, A. et al. [21] (2017) | MSI | - | - | CNN |
| Chien, J. et al. [] (2017) | RGB | TFNN | Spatial-Spectral | TFNN |
| Dewa, M. et al. [] (2018) | HSI | PCA | Spectral | PCA |
| Xu, Y. et al. [] (2018) | HSI | - | - | CNN |
| Li, J. et al. [28] (2019) | MSI | NTD-CNN | Spatial-spectral | - |
| An, J. et al. [27] (2019) | HSI | T-MLRD | Spatial-spectral | SVM/1NN |
| An, J. et al. [29] (2019) | HSI | TDA | Spatial-spectral | SVM/1NN |
| Lopez, J. et al. [] (2019) | MSI | TKD | Spectral | FCN |
| Sayeh, M. et al. [] (2019) | HSI | NTD | Spatial-Spectral | 3D-CNN |
| **Our framework** | **MSI/HSI** | **NTKD** | **Spectral** | **CNN** |

### *1.2. Contribution*

In the state of art we can find a variety of frameworks looking for dimensionality reduction of images of any type, that is, RGB [? ], medical [? ], SAR [? ], multispectral [? ], among others. In particular, the large number of spectral bands in hyperspectral images has focused efforts in this category [? ]. Spatial [? ], spectral [? ] and spatial-spectral [? ] compression have been achieved with matrix and tensor decompositions. It is important to highlight that, a decomposition as pre-processing for a classification algorithm, instead of favoring, could be counterproductive and greatly affect its performance. It is therefore important to make a proper selection of the type of decomposition that would produce a suitable data set for post-processing.

Unlike previous works, this work seeks to adapt the data in the best way to be the input of deep convolutional networks. Convolutional network models are designed to extract and interpret all the spatial properties of an image by moving the kernels over the input data []. Therefore, producing uncorrelated data in space and spectrum, would make harder the interpretation of the data in the convolutional network [? ]. Thus, the proposed framework maintains the positive tensor nature of the spectral images and the spatial dimensionality to preserver spatial-spectral correlation of the data while reducing the spectral dimensionality, in order to decrease computational load in the pixel-wise classification process.

We can summarize the contribution of this work with the following two points:

1. The framework NTKD3-CNN proposed in this work, develops a new strategy to improve accuracy results of semantic segmentation convolutional neural networks by finding suitable tensor data, preserving spatial correlation and values in the set of the natural numbers while compressing the spectral domain and in turn decreasing computational load.
2. This work also presents an exhaustive performance analysis measuring and comparing its efficiency with the most popular metrics, i.e., as pixel accuracy (PA), also PA in function of the number of new tensor bands, precision, recall, F1, orthogonality degree of the factor matrices and the core tensor, reconstruction error of the original tensor, and execution time.

The remainder of this work is organized as follows. Section **??** introduces tensor algebra notation and basic concepts to familiarize the reader with the symbology used in this paper. Section **??** presents the problem statement of this work and the mathematical definition. In Section 4, CNN theory is described for classification and semantic segmentation. Section **??** presents the framework proposed for compression and semantic segmentation of spectral images. Experimental results are presented in Section **??**. Finally, Sections **??** and 8 present a discussion and conclusions based on the results obtained in the experiments.

## 2. Tensor-Based Factorizations

Matrix-based factorizations, such as PCA [] and SVD [] have been significant and useful tools for dimensionality reduction and other approaches. Nevertheless, they are limited by representations of data only in two modes. Most of current applications have data structures often as higher-order arrays, e.g. dimensions of space, time, and frequency. This 2-way view in matrix factorizations may be inadequate and it is natural to use tensor decomposition approaches [**?** ].

We can define a tensor as a multi-way or multidimensional array. The order of a tensor is the number of dimensions, also known as modes, i.e., an $N$-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is an $N$-dimesional array, which elements $y_{i_1,i_2,\ldots,i_n}$ are indexed by $i_n \in 1, 2, \ldots, I_n$ for $1 \leq n \leq N$.

Throughtout this paper, the mathematical notation used by Kolda et al. [17] has been adopted. Table 2 summarize this notation.

**Table 2.** Tensor algebra notation summary

| | |
|---|---|
| $\mathcal{A}$, $\mathbf{A}$, $\mathbf{a}$, $a$ | Tensor, matrix, vector and scalar respectively |
| $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ | $N$-order tensor of size $I_1 \times \cdots \times I_N$. |
| $a_{i_1 \ldots i_N}$ | An element of a tensor |
| $\mathbf{a}_{:i_2 i_3}$, $\mathbf{a}_{i_1 : i_3}$, and $\mathbf{a}_{i_1 i_2 :}$ | Column, row and tube fibers of a third order tensor |
| $\mathbf{A}_{i_1::}$, $\mathbf{A}_{:i_2:}$, $\mathbf{A}_{::i_3}$ | Horizontal, lateral and frontal slices for a third order tensor |
| $\mathbf{A}^{(n)}$, $\mathbf{a}^{(n)}$ | A matrix/vector element from a sequence of matrices/vectors |
| $\mathbf{A}_{(n)}$ | Mode-n matricization of a tensor. $\mathbf{A}_{(n)} \in \mathbb{R}^{I_n \times \prod_{m \neq n} I_m}$ |
| $\mathcal{X} = \mathbf{a}^{(\mathbf{1})} \circ \cdots \circ \mathbf{a}^{(\mathbf{N})}$ | Outer product of $N$ vectors, where $x_{i_1 i_2 \ldots i_N} = a^{(1)}_{i_1} \ldots a^{(N)}_{i_N}$ |
| $\langle \mathcal{A}, \mathcal{B} \rangle$ | Inner product of two tensors. |
| $\mathcal{B} = \mathcal{A} \times_n \mathbf{U}$ | $n$-mode product of tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ by a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ along axis $n$. |

### 2.1. Basic concepts

It is also necessary to introduce some tensor algebra operations and basic concepts used in later explanations. These notations were taken textually from [17].

#### 2.1.1. Matricization

The mode-$n$ matricization is the process of reordering the elements of a tensor into a matrix along axis $n$ and it is denoted as $\mathbf{A}_{(n)} \in \mathbb{R}^{I_n \times \prod_{m \neq n} I_m}$.

#### 2.1.2. Outer Product

The outer product of $N$ vectors $\mathcal{X} = \mathbf{a}^{(\mathbf{1})} \circ \cdots \circ \mathbf{a}^{(\mathbf{N})}$ produces a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ where $\circ$ denotes the outer product and $\mathbf{a}^{(\mathbf{n})}$ denotes a vector in a sequence of $N$ vectors and each element of the tensor is the product of the corresponding vector elements; i.e., $x_{i_1 i_2 \ldots i_N} = a^{(1)}_{i_1} \ldots a^{(N)}_{i_N}$.

#### 2.1.3. Inner Product

The inner product of two tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is the sum of the products of their entries; i.e., $\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} a_{i_1 \ldots i_N} b_{i_1 \ldots i_N}$.

#### 2.1.4. $N$-Mode Product

It means the multiplication of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ by a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ or vector $\mathbf{u} \in \mathbb{R}^{I_n}$ in mode $n$; i.e., along axis $n$. It is represented by $\mathcal{B} = \mathcal{A} \times_n \mathbf{U}$, where $\mathcal{B} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}$ [17].

#### 2.1.5. Rank-One Tensor

A tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is rank one if it can be written as the outer product of $N$ vectors; i.e., $\mathcal{X} = \mathbf{a}^{(\mathbf{1})} \circ \cdots \circ \mathbf{a}^{(\mathbf{N})}$.

#### 2.1.6. Rank-R Tensor

The rank of a tensor rank($\mathcal{X}$) is the smallest number of components in a CPD; i.e., the smallest number of rank-one tensors that generate $\mathcal{X}$ as their sum [17].

#### 2.1.7. *N*-Rank

The $n$-rank of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ denoted $\text{rank}_n(\mathcal{X})$, is the column rank of $\mathbf{X}_{(n)}$; i.e., the dimension of the vector space spanned by the mode-$n$ fibers. Hence, if $R_n \equiv \text{rank}_n(\mathcal{X})$ for $n = 1, \ldots, N$, we can say that $\mathcal{X}$ has a rank $-(R_1, \ldots, R_N)$ tensor.

### *2.2. Nonnegative Tucker Decomposition (NTKD)*

The TKD, also called the Tucker3 for the particular case of third-order tensors, or best rank ($J_1$, $J_2$, $J_3$) approximation, can be formally formulated as follows [? ]. Given a third-order data tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and three positive indices $J_1, J_2, J_3 << I_1, I_2, 1_3$, find a core tensor $\mathcal{G} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$ and three component matrices called factor or loading matrices $\mathbf{U}_1 \in \mathbb{R}^{I_1 \times J_1}$, $\mathbf{U}_2 \in \mathbb{R}^{I_2 \times J_2}$ and $\mathbf{U}_3 \in \mathbb{R}^{I_3 \times J_3}$ which perform the following approximate decomposition:

$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{U}^{(1)} \cdots \times_N \mathbf{U}^{(N)} + \mathcal{E} \tag{1}$$

where $\mathcal{E}$ denotes the approximation error. The core tensor $\mathcal{G}$ preserves the level of interaction for each factor or projection matrix $\mathbf{U}^{(n)}$. The factor matrices are commonly considered orthogonal, but in Tucker models with non-negativity constraints, that is not necessarily imposed [? ]. These matrices can be seen as the principal components in each mode [17] (see Figure 1). $J_n$ represents the number of components in the decomposition; i.e., the rank $-(R_1, \ldots, R_N)$.

We can also denote the TKD using the matricization approach and express it by

$$\mathbf{X}_{(1)} = \mathbf{U}^{(1)}\mathbf{G}_{(1)}(\mathbf{U}^{(3)} \otimes \mathbf{U}^{(2)})^T \tag{2a}$$
$$\mathbf{X}_{(2)} = \mathbf{U}^{(2)}\mathbf{G}_{(2)}(\mathbf{U}^{(3)} \otimes \mathbf{U}^{(1)})^T \tag{2b}$$
$$\mathbf{X}_{(3)} = \mathbf{U}^{(3)}\mathbf{G}_{(3)}(\mathbf{U}^{(2)} \otimes \mathbf{U}^{(1)})^T \tag{2c}$$

where $\otimes$ denotes the Kronecker product and $\mathbf{X}_{(n)}$ and $\mathbf{G}_{(n)}$ are the $n$-mode matricized versions of tensor $\mathcal{X}$ and $\mathcal{G}$ respectively.

Starting from (1), the reconstruction of an approximated tensor can be given by

$$\hat{\mathcal{X}} = \mathcal{G} \times_1 \mathbf{U}^{(1)} \cdots \times_N \mathbf{U}^{(N)} \tag{3}$$

where $\hat{\mathcal{X}}$ is the reconstructed tensor.

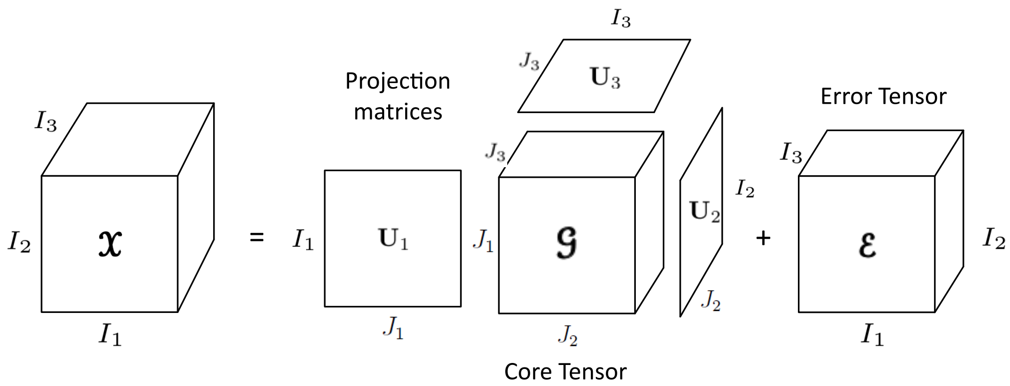


**Figure 1.** Tucker decomposition for a third-order tensor.

Then, we can acquire the core tensor $\mathcal{G}$ by the multilinear projection

$$\mathcal{G} = \mathcal{X} \times_1 \mathbf{U}^{(1)\mathrm{T}} \cdots \times_N \mathbf{U}^{(N)\mathrm{T}} \tag{4}$$

where $\mathbf{U}^{(n)\mathrm{T}}$ denotes the transpose matrix of $\mathbf{U}^{(n)}$ for $n = 1, \ldots, N$. The reconstruction error $\xi$ can be computed as

$$\xi(\hat{\mathcal{X}}) = ||\mathcal{X} - \hat{\mathcal{X}}||_F^2 \tag{5}$$

where $||\cdot||_F$ represents the Frobenius norm. To compute the best rank approximation of a tensor, it is required an iterative algorithm. HOSVD, ALS and HOOI are algorithm commonly used in TKD [? ].

HOOI initializes the factors matrices using HOSVD and assumes that othogonal matrices are known, so that the core tensor is obtained with (4). Then, it maximizes the cost function

$$\max_{\mathbf{U}^{(1)},\mathbf{U}^{(2)},\mathbf{U}^{(3)}} ||\mathcal{X} \times_1 \mathbf{U}^{(1)\mathrm{T}} \times_2 \mathbf{U}^{(2)\mathrm{T}} \cdots \times_N \mathbf{U}^{(N)\mathrm{T}}||_F^2 \tag{6}$$

with $\mathbf{U}^{(n)}$ unknown. Fixing all factor matrices but one, tensor $\mathcal{X}$ can be projected onto the $\{R_1, \ldots, R_{n-1}, R_{n+1}, \ldots, R_N\}$-dimensional space as

$$\mathcal{W}^{(-n)} = \mathcal{X} \times_1 \mathbf{U}^{(1)\mathrm{T}} \cdots \times_{n-1} \mathbf{U}^{(n-1)\mathrm{T}} \times_{n+1} \mathbf{U}^{(n+1)\mathrm{T}} \cdots \times_N \mathbf{U}^{(N)\mathrm{T}} \tag{7}$$

and the orthogonal matrices can be estimated as an orthonormal basis for the dominant subspace of the projection by applying the standard matrix SVD for mode-$n$ unfolded matrix $\mathcal{W}^{(-n)}_{(n)}$ [? ]. See Algorithm 1.

The HOOI algorithm evidently can be applied for the particular case of third-order tensor. Furthermore, it can be slightly adjusted for preserving any input dimension and develop the decomposition in a specific order. This will be explained in next chapters.

---

**Algorithm 1:** HOOI algorithm to compute a rank-$(R_1, \ldots, R_N)$ TKD for an $N$th-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$.

---

**Function** `HOOI(`$\mathcal{X}$, $J_1, \ldots, J_N$`)`**:**
- initialize $\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times J_n}$ for $n = 1, \ldots, N$ using HOSVD or random
- **repeat**
  - **for** $n = 1, \ldots, N$ **do**
    - $\mathcal{W}^{(-n)} \leftarrow \mathcal{X} \times_{-n} \{\mathbf{U}^{\mathrm{T}}\}$
    - $\left[\mathbf{U}^{(n)}, \mathbf{\Sigma}^{(\mathbf{n})}, \mathbf{V}^{(n)}\right] \leftarrow \mathrm{svds}(\mathcal{W}^{(-n)}_{(n)}, J_n, \text{'LM'})$
    - $\mathbf{U}^{(n)} \leftarrow \left[\mathbf{U}^{(n)}\right]_+$
  - **end**
- **until** *fit ceases to improve or maximum iterations exhausted;*
- $\mathcal{G} \leftarrow \mathcal{W}^{(-N)} \times_N \mathbf{U}^{(N)\mathrm{T}}$

**Output:** $\mathbf{U}^{(n)} \in \mathbb{R}_+^{I_n \times J_n}$, $\mathcal{G} \in \mathbb{R}_+^{J_1 \times J_2 \times \cdots J_N}$

---

## 3. Problem phenomenology

### 3.1. Spectral Imagery

Multi- or Hyper-spectral images are by nature multidimensional nonnegative arrays. A spectral image can be sorted and represented as a third-order tensor $\mathcal{X} \in \mathbb{R}_+^{I_1 \times I_2 \times I_3}$, where $I_!$, $I_2$ and $I_3$ represent its height, widht and spectral bands respectively. In RS image processing, spectral images are frequently used for classification of different material in a scene of interest. However, due to the low spatial resolution produced by the distance between the sensor an the target, spatial features are not sufficient to discern certain classes. That is why spectral resolution plays an important role in this type of task.

The separation into spectral bands allows perception of reflectance at different wavelengths. This helps to better characterize various materials, in order to simplify the process of discernment between classes. The effort to obtain these spectral features generates a greater amount of data, which increases the processing complexity. This is where the spectral decomposition task becomes relevant.

### *3.2. Problem Statement*

Given a Spectral Image as a third-order tensor $\mathfrak{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, and its corresponding pixelwise classification ground truth matrix $\mathbf{Y} \in \mathbb{R}^{I_1 \times I_2}$ for a specific number of classes $C$, find, through Non-negative Tensor Decomposition, the best rank-$(R_1, R_2, R_n)$ approximation and its respective core tensor $\mathfrak{G} \in \mathbb{R}^{I_1 \times I_2 \times J_3}$, where $R_1 = I_1, R_2 = I_2$ and $R_3 = J_3 << I_3$, to be the input of a pixel-wise classification Convolutional Neural Network and produce an output matrix $\hat{\mathbf{Y}}$ of predicted classes, achieving competitive performance metrics for pixel-wise classification while decreasing computational load in the classification process.

### *3.3. Mathematical Definition*

We can mathematically define the problem statement described above as an optimization problem.

$$
\begin{array}{lll}
\min\limits_{\mathfrak{G}, \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}} & ||\mathfrak{X} - \mathfrak{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}||_F^2 & \\
\text{subject to} & \mathbf{U}^{(n)} \in \mathbb{R}_+^{I_n \times J_n} \quad \text{for} \quad n = 1,2,3 \quad \text{and} \quad \mathfrak{G} \in \mathbb{R}_+^{J_1 \times J_2 \times J_3} & \\
& J_1 = I_1, J_2 = I_2 & \text{no compression in the spatial domain,} \\
& J_3 << I_3 & \text{reduced spectral domain at the core tensor,} \\
& \xi(\hat{\mathfrak{X}}) \to 0 \quad \text{and} \quad \text{PA} \geq \psi & \text{competitive pixel accuracy up to a trehold}
\end{array}
\tag{8}
$$

## 4. Deep Convolutional Neural Networks (DCNNs)

CNNs are supervised feed-forward DL-ANNs for computer vision. The idea of applying a sort of convolution of the synaptic weights of a neural network through the input data yields to a preservation of spatial features, which alleviates the hard task of classification and in turn semantic segmentation. This type of ANN works under the same linear regression model as every machine learning (ML) algorithm. Since images are three dimensional arrays, we can use tensor algebra notation to describe the input of CNNs as a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, where $I_1$, $I_2$, and $I_3$ represent height, width, and depth of the third order array respectively; i.e., the spatial and spectral domain of an image. We can write generally the linear regression model used for ANNs as

$$\hat{\mathbf{y}} = \sigma\left(\mathbf{W}\mathbf{g} + \mathbf{b}\right) \tag{9}$$

where $\hat{\mathbf{y}}$ represents the output prediction of the network; $\sigma$ denotes an activation function; $\mathbf{g}$ is the input dataset; $\mathbf{W}$ and $\mathbf{b}$ are the matrix of synaptic weights and the bias vector, respectively. These parameters are adjustable; i.e., their values are modified every iteration looking for convergence to minimize the loss in the prediction through optimization algorithms [32]. For simplicity, the bias vector can be ignored, assuming that matrix $\mathbf{W}$ will update until convergence independently of another parameter [32]. Considering that the input dataset to a CNN is a multidimensional array, we can represent (**??**) and (9) using tensor algebra notation as

$$\hat{\mathfrak{Y}} = \sigma\left(\mathfrak{W}\mathfrak{G}\right) \tag{10}$$

where $\hat{\mathcal{Y}}$ represents the prediction output tensor of the ANN (in our case, a second order tensor or matrix $\hat{\mathbf{Y}}$), $\mathcal{G}$ is the input dataset, and $\mathcal{W}$ is a $K_1 \times K_2 \times F_1$ tensor called filter or kernel with the adaptable synaptic weights. Different to conventional ANN, in CNNs, $\mathcal{W}$ is a shiftable square tensor is much smaller in height and width than the input data, i.e., $K_1 = K_2$ and $K_s << I_s$ for $s = 1, 2$; $F_1$ denotes the number of input channels; i.e., $F_1 = I_3$. For hidden layers, instead of the prediction tensor $\hat{\mathcal{Y}}$, the output is a matrix called activation map $\mathbf{M} \in \mathbb{R}^{I_1 \times I_2}$, which preserves features from the original data in each domain. Actually, it is necessary to use much kernels $\mathcal{W}^{(f_2)}$ as activation maps, with different initialization values to preserve diverse features of the image. Hence, we can also define activation maps as a tensor $\mathcal{M} \in \mathbb{R}^{I_1 \times I_2 \times F_2}$ where $F_2$ denotes the number of activation maps produced by each filter (see Figure 2). Kernels are displaced through the whole input image as a discrete convolution operation. Then, each element of the output activation map $m_{i_1 i_2 f_2}$ is computed by the summary of the Hadamard product of kernel $\mathcal{W}^{(f_2)}$ and a subtensor from the input tensor $\mathcal{G}$ centered in position $(i, j)$ and with same dimensions of $\mathcal{W}$, as follows

$$m_{i_1 i_2 f_2} = \sigma \left[ \sum_{k_1=1}^{K_1} \sum_{k_2=1}^{K_2} \sum_{f_1=1}^{F_1} w_{k_1, k_2, f_1} g_{i_1+k_1-o_1, i_2+k_2-o_2, f_1} \right] \tag{11}$$

where $m_{i_1 i_2 f_2}$ denotes the value of the output activation map $f_2$ at position $i_1, i_2$; $\sigma$ represents the activation function; and $o_1$ and $o_2$ are offsets in spatial dimensions which depend on the kernel size, and equal $\frac{K_1+1}{2}$ and $\frac{K_2+1}{2}$ respectively (see Figure 2).

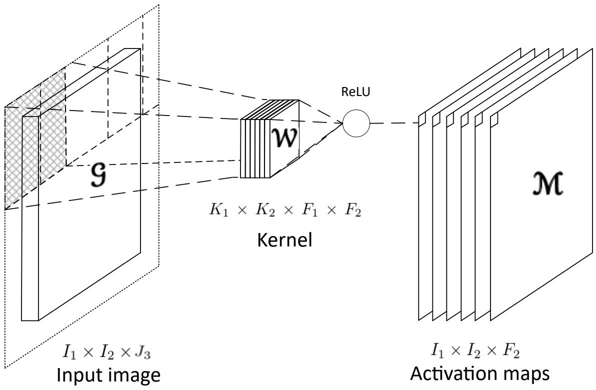


**Figure 2.** Convolutional layer with a $K_1 \times K_2 \times F_1 \times F_2$ kernel. Input channels $F_1$ must equal the spectral bands $I_3$. To preserve original dimensions at the output, zero padding is needed [18]. Output dimensions also depend on stride $S = 1$ to consider every piece of pixel information and to preserve original dimensions.

An ANN is trained by using iterative gradient-based optimizers, such as Stochastic gradient descent, Momentum, RMSprop, and Adam [32]. This drive the cost function $L(\mathcal{W})$ to a very low value by updating the synaptic weights $\mathcal{W}$. We can compute the cost function by any function that measures the difference between the training data and the prediction, such as Euclidean distance or cross-entropy [10]. Besides, the same function is used to measure the performance of the model during testing and validation. In order to avoid overfitting [32], the total cost function used to train an ANN combines one of the cost functions mentioned before, plus a regularization term.

$$J(\mathcal{W}) = L(\mathcal{W}) + R(\mathcal{W}), \tag{12}$$

where $J(\mathcal{W})$ denotes the total cost function and $R(\mathcal{W})$ represents a regularization function. Then, we can decrease $J(\mathcal{W})$ by updating the synaptic weights in the direction of the negative gradient. This is known as the method of steepest descent or gradient descent.

$$\mathcal{W}' = \mathcal{W} - \alpha \nabla_{\mathcal{W}} J(\mathcal{W}), \tag{13}$$

where $\mathcal{W}'$ represents the synaptic weights tensor in next iteration during training, $\alpha$ denotes the learning rate parameter, and $\nabla_{\mathcal{W}} J(\mathcal{W})$ the cost function gradient. Gradient descent converges when every element of the gradient is zero, or in practice, very close to zero [10].

CNNs has been successfully used in many image classification frameworks. This variation in architecture from other typical ANN models yields the network to learn spatial and spectral features, which are highly profitable for image classification. Besides, FCNs, constructed with only convolutional layers are able to classify each element of the input image; i.e., they yield pixel-wise classification, or in other words, semantic segmentation.

## 5. NNTKD for DCNNs

## 6. Experimental Results

### *6.1. Input Data*

For this work, we chose three of the most popular multi- and hyperspectral dataset for classification.

#### 6.1.1. Sentinel-2 CNNMSI

This dataset developed by Lopez et al. [? ] is composed of RS Sentinel-2 scenarios from central Europe. It has 100 scenarios for the training space and 10 scenarios for testing, all of them with $128 \times 128$ pixels with spatial resolution of $20m^2$ and 9 spectral bands in the range $490 - 2190nm$. The labels are semi-manually asigned for five classes of interest: vegetation, soil, water, clouds and shadows. Data are available in the link Sentinel-2 Dataset.

#### 6.1.2. Indian Pines

This dataset is a scene produced by AVIRIS in North-western Indiana and consists of $145 \times 145$ pixels and 224 spectral bands in the wavelength range 0.4˘2.5$\mu m$. The Indian Pines scene contains two-thirds agriculture, and one-third forest or other natural perennial vegetation. There are two major dual lane highways, a rail line, as well as some low density housing, other built structures, and smaller roads. Since the scene is taken in June some of the crops present, corn, soybeans, are in early stages of growth with less than 5% coverage. The ground truth available is designated into sixteen classes and is not all mutually exclusive. Indian Pines data are available at Indian Pines dataset.

#### 6.1.3. Salinas

This scene was collected by the AVIRIS sensor over Salinas Valley, California. It has $512 \times 217$ pixels with spatial resolution $3.7m$, and 224 spectral bands. It includes vegetables, bare soils, and vineyard fields. Salinas groundtruth contains 16 classes.

**Table 3.** Summary of the different dataset used for experiments in this work.

| Dataset | Spatial dimensions | Bands | Classes | Samples |
|---|---|---|---|---|
| Sentinel-2 CNNMSI | $128 \times 128$ | 5 | 9 | 147,456 |
| Indian Pines | $145 \times 145$ | 224 | 16 | 4,709,600 |
| Salinas | $512 \times 217$ | 224 | 16 | 24,887,296 |

### *6.2. Metrics*

#### 6.2.1. Relative Mean Square Error (rMSE)

In order to compute the reconstruction error of the tensor $\mathcal{X}$ for the implementation of HOOI, the rMSE was used:

$$rMSE\left(\hat{\mathcal{X}}\right) = \frac{1}{Q}\sum_{q=1}^{Q}\frac{\left\|\hat{\mathcal{X}}_q - \mathcal{X}_q\right\|_F^2}{\left\|\mathcal{X}_q\right\|_F^2}, \tag{14}$$

where $\mathcal{X}_q$ represents the $q$-th CNNMSI from our dataset with $Q$ MSIs and $\hat{\mathcal{X}}_q$ its corresponding reconstruction computed by (3).

#### 6.2.2. Cohen's Kappa Coefficient

Cohen's kappa coefficient is a very robust metric used to measure reliability of multi-class and imbalanced class classification algorithms. It is computed by

$$\kappa = \frac{\rho_o - \rho_e}{1 - \rho_e} \tag{15}$$

where $\rho_o$ is the observed agrrement, and $\rho_e$ is the expected agreement. This metric will always produce values less than or equal to 1, where 1 means perfect agreement. Negative values indicate no agreement. i.e., futile classification.

#### 6.2.3. Pixel Accuracy (PA)

We used the PA metric to compute a ratio between the amount of correctly classified pixels and the total number of pixels as follows. Given a confusion matrix relating the True Positive (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN), the PA is computed by

$$PA = \frac{\sum_{c=1}^{C}\tau_{cc}}{\sum_{c=1}^{C}\sum_{d=1}^{C}\tau_{cd}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{16}$$

where $c$ is the number of class for $c = 1, \ldots, C$ and $\tau_{cc}$ is the amount of pixels of class $c$ correctly assigned to class $c$ i.e., TP and TN, and $\tau_{cd}$ is the amount of pixels of class $c$ inferred to belong to class $d$. Despite this metric is wide used, it is not a totally fair metric for imbalanced classes. In this work we used this metric with comparison purposes. However, we also used metrics more in line with multi class classification tasks.

#### 6.2.4. Precision

Another metric used in this work as a performance evaluation metric in multiclass classification is precision. This is a metric that measures the percentage of pixels from class $c$ correctly classified. It is computed with the following equation

$$\Psi_c = \frac{\tau_{cc}}{\sum_{d=1}^{C}\tau_{cd}} = \frac{TP}{TP + FP} \tag{17}$$

where $\Psi_c$ denotes the precision of class $c$, which is the number of TP divided by the TP plus FP.

#### 6.2.5. Recall or Sensitivity

Recall, or also known as sensitivity is a metric that indicates the proportion of pixels classified as class $c$ that actually belong to class $c$. It is computed with the following equation

$$\upsilon_c = \frac{\tau_{cc}}{\sum_{d=1}^{C} \tau_{dc}} = \frac{TP}{TP + FN} \tag{18}$$

where $\upsilon_c$ denotes the recall of class $c$ for $c = 1, \ldots, C$.

#### 6.2.6. F1 Score

In order to summarize precision and recall in one only metric, we use the F1 score, which is computed by

$$F1 = \frac{2\Psi\upsilon}{\Psi + \Upsilon} \tag{19}$$

This metric provides a very appropriate measure of multiclass classification for imbalanced dataset.

## 7. Discussion and Comparison

## 8. Conclusions

### *8.1. Figures, Tables and Schemes*

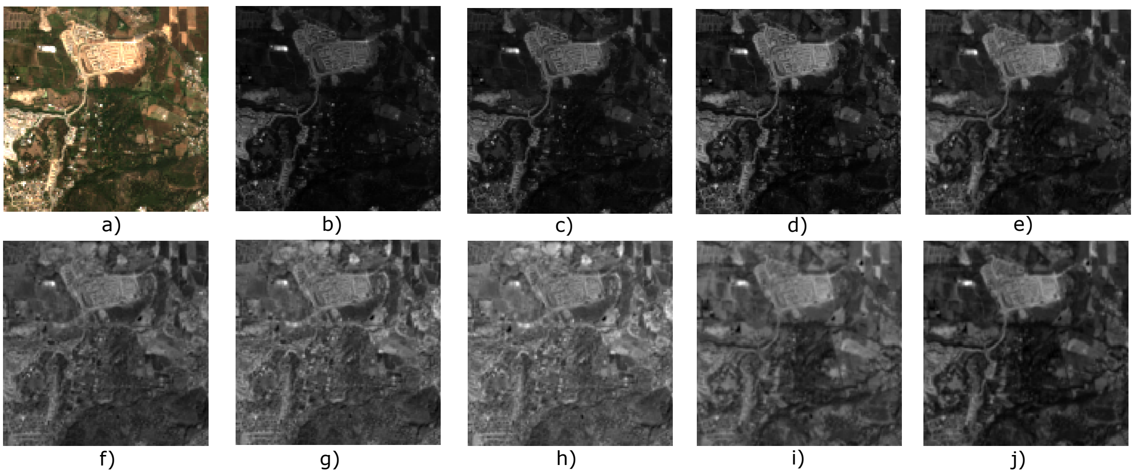


**Figure 3.** Original Sentinel-2 spectral bands, a) True color image, b) to j) 1st to 9th spectral band respectively.

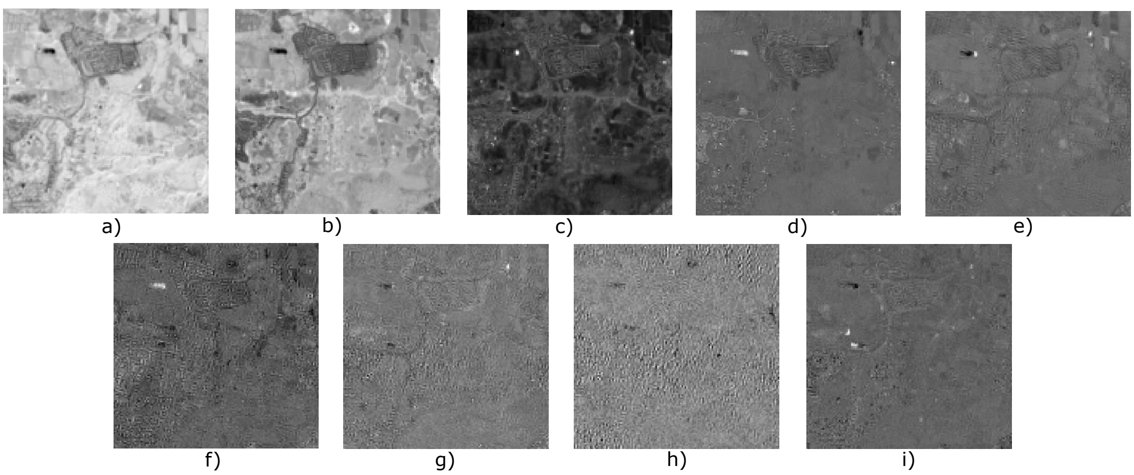


**Figure 4.** Tensor bands from the Tucker Decomposition, a) to i) 1st to 9th band respectively.

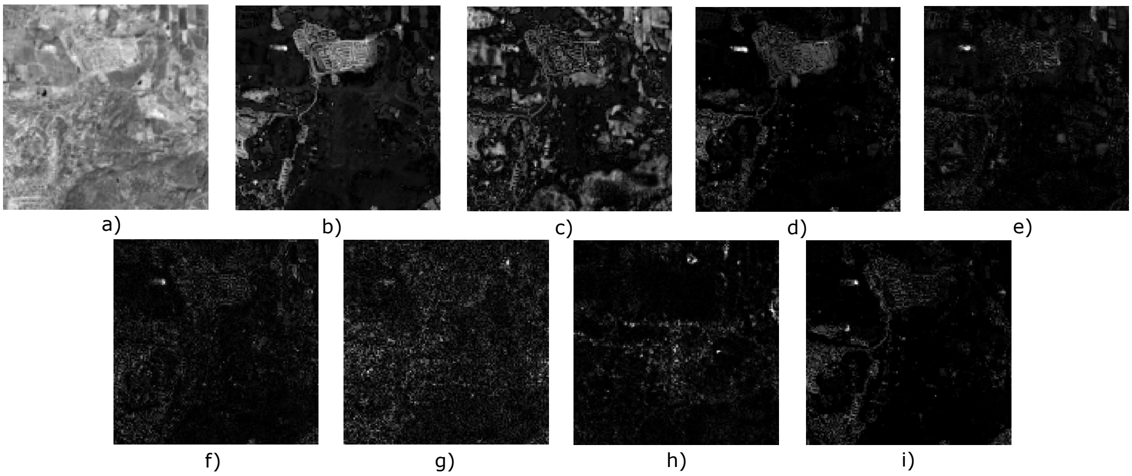


**Figure 5.** Tensor bands from the Non-negatice Tucker Decomposition, a) to i) 1st to 9th band respectively.

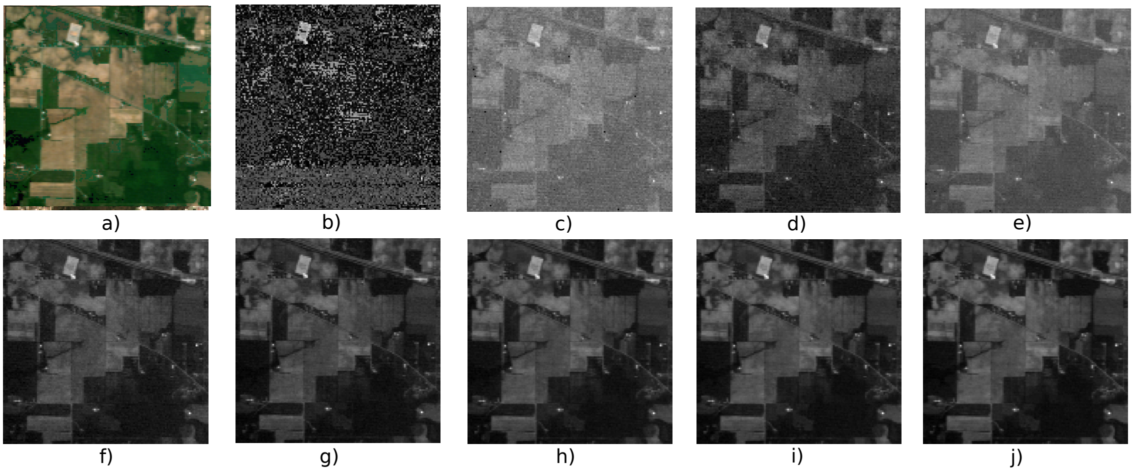


**Figure 6.** Original Sentinel-2 spectral bands.

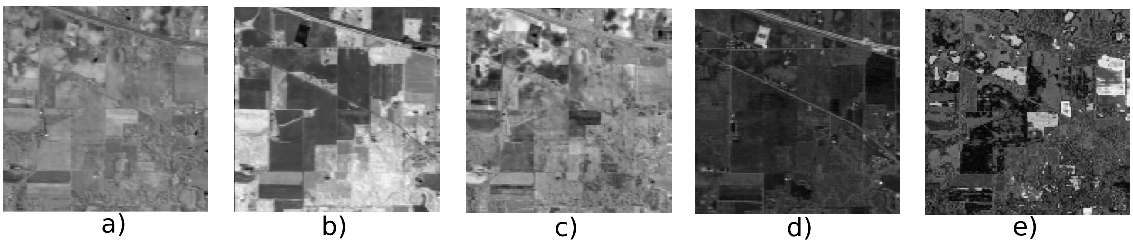


**Figure 7.** Tucker Decomposition Tensor bands.

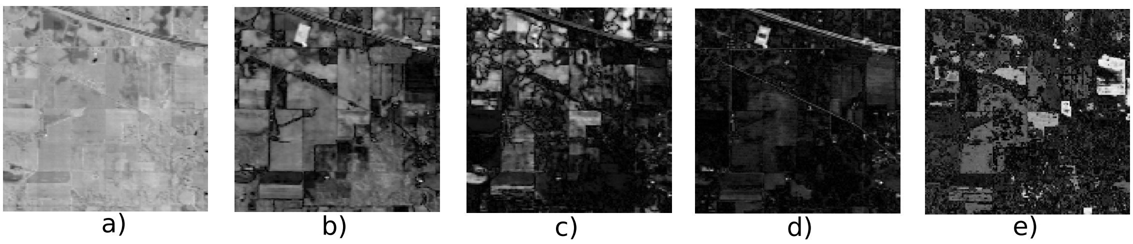


**Figure 8.** Nonnegative Tucker Decomposition Tensor bands.

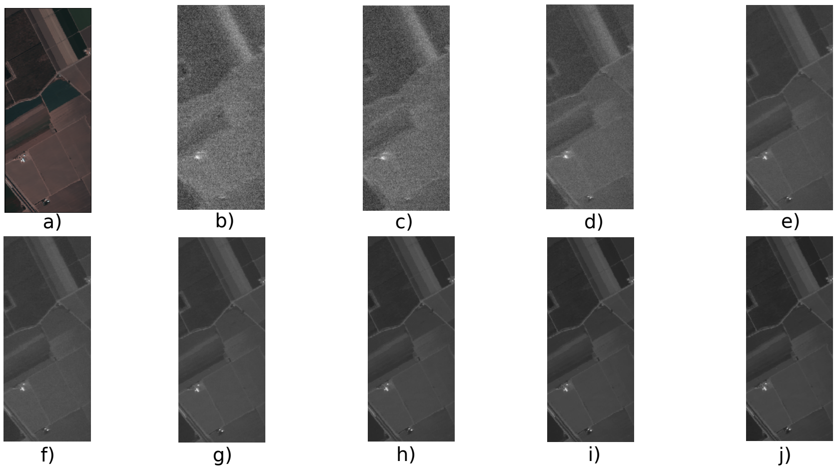


**Figure 9.** Original Sentinel-2 spectral bands.

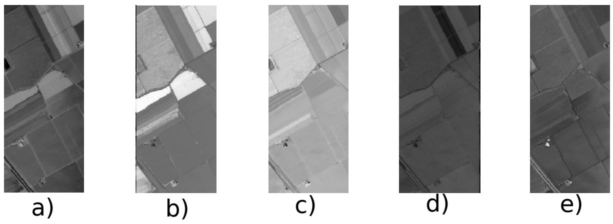


**Figure 10.** Tucker Decomposition Tensor bands.

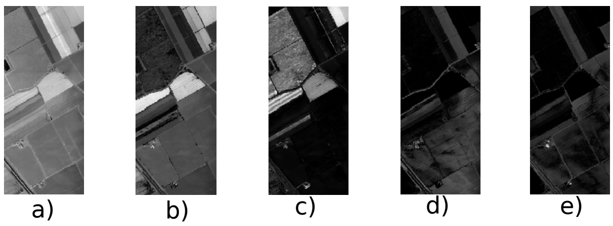


**Figure 11.** Nonnegative Tucker Decomposition Tensor bands.

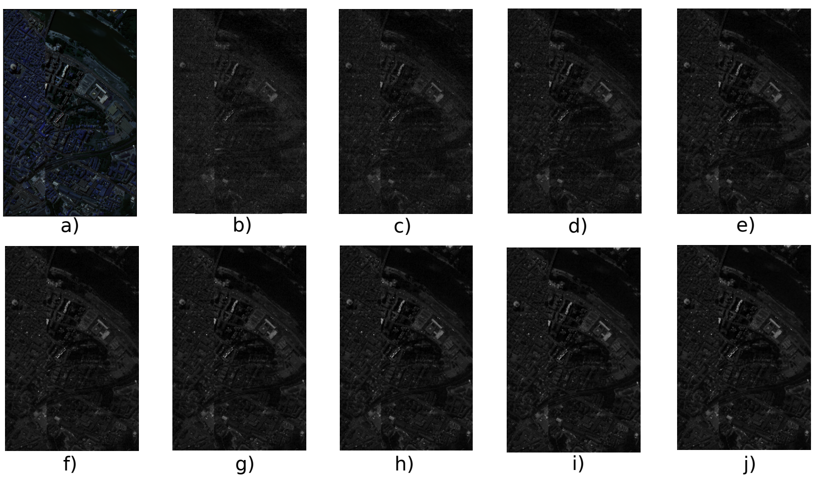


**Figure 12.** Original Sentinel-2 spectral bands.

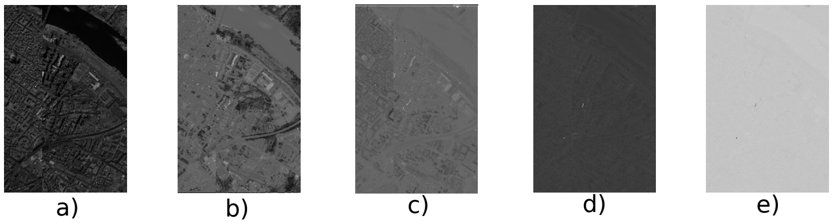


**Figure 13.** Tucker Decomposition Tensor bands.


**Figure 14.** Nonnegative Tucker Decomposition Tensor bands.

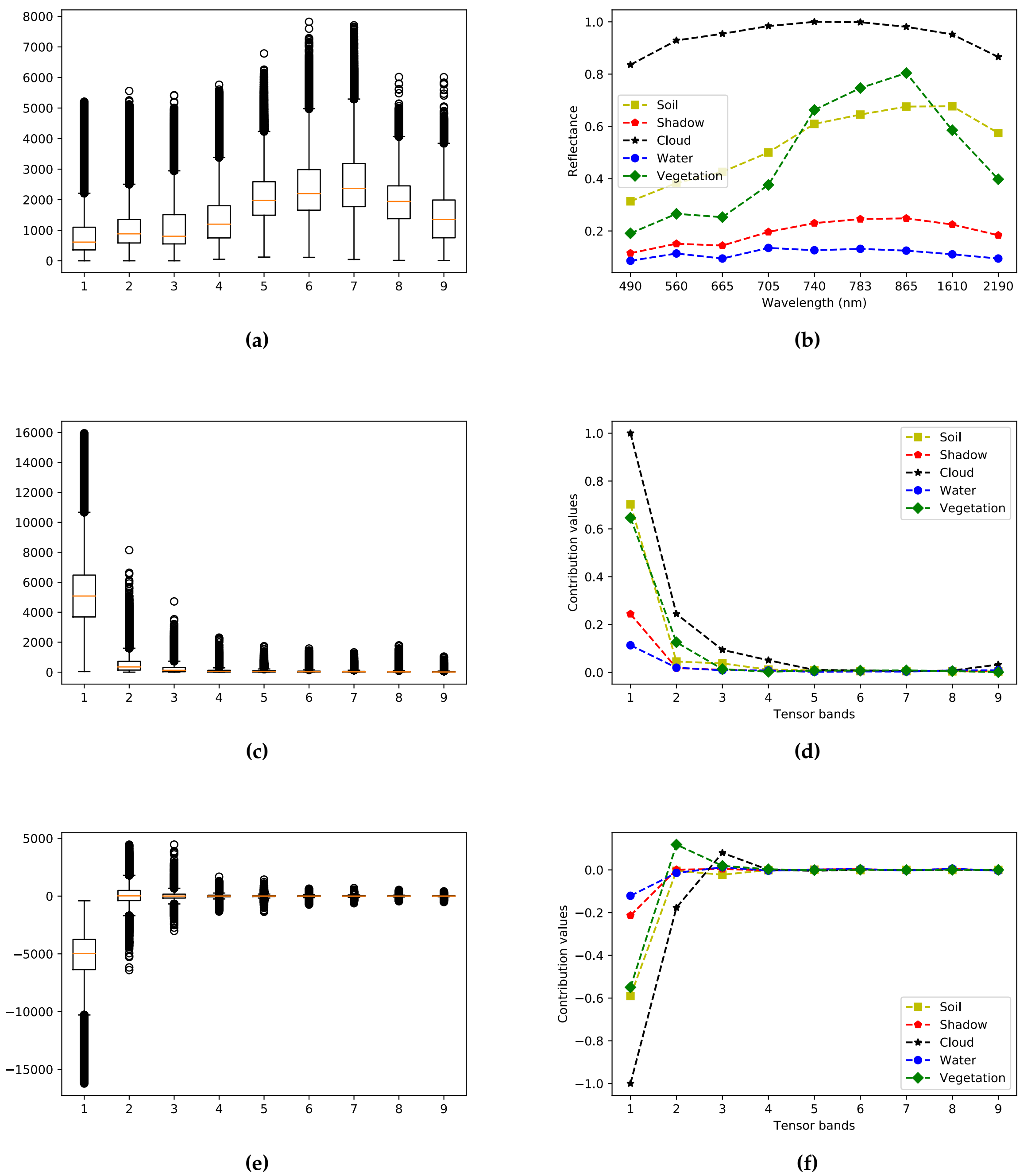


**Figure 15.** Box and whiskers plot and spectral signatures for a) and b) original spectral bands, c) and d) tensor bands of the NTKD and e) and f) tensor bands of the TKD

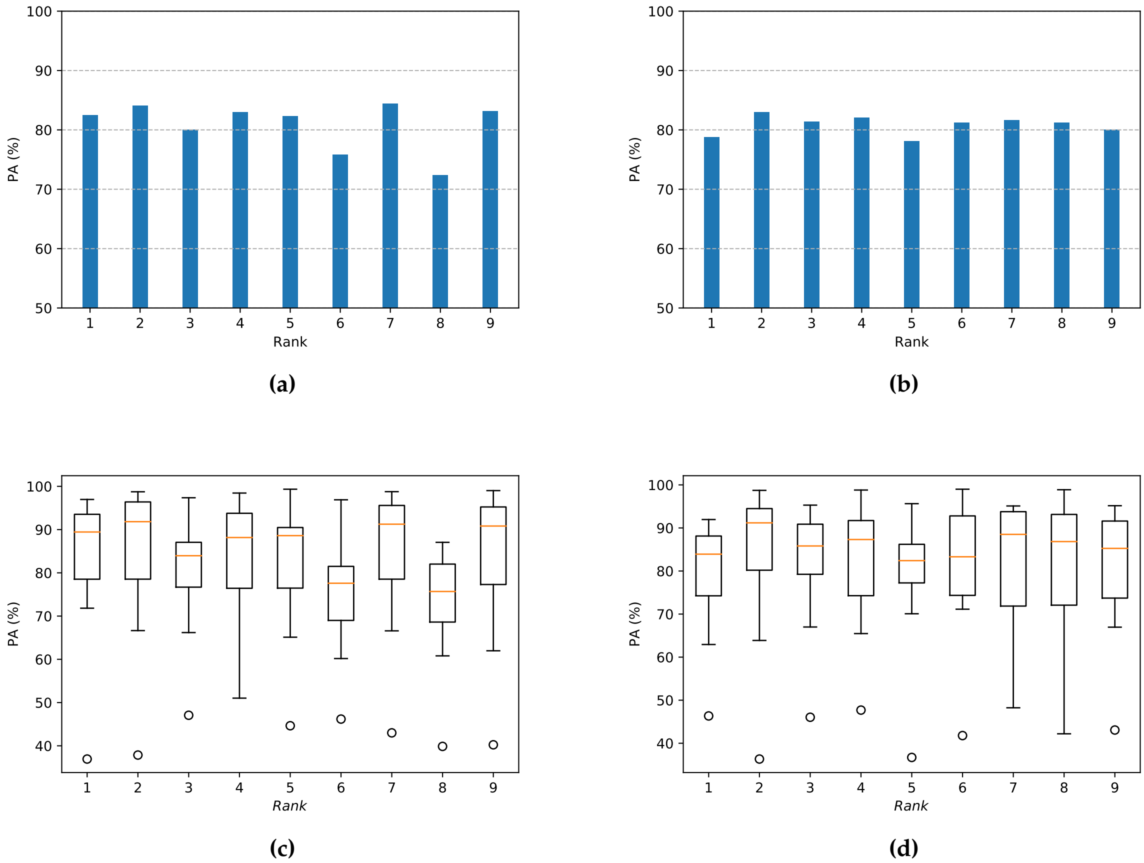


**Figure 16.** Pixel accuracy vs Rank results a) Comparative bar plot NTKD and TKD, b) Comparative bar plot NTKD and TKD c) Box and whiskers plot for NTKD, and d) Box and whiskers plot for TKD

**Author Contributions:** Conceptualization, J.L.; formal analysis, D.T.; investigation, J.L.; methodology, J.L., D.T., and C.A.; resources, C.A.; software, J.L.; supervision, D.T. and C.A.; validation, D.T. and C.A.; writing—original draft, J.L. and D.T.

**Funding:** This work was supported by the National Council of Science and Technology CONACYT of Mexico under grant XXXXXXX.

**Acknowledgments:**

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| ANN | Artificial Neural Network |
| CNN | Convolutional neural network |
| CPD | Canonical Polyadic Decomposition |
| DL | Deep Learning |
| FCN | Fully Convolutional Network |
| HOOI | Higher-Order Orthogonal Iteration |
| HOSVD | Higher-Order Singular Value Decomposition |

## References

1. Tempfli, K.; Huurneman, G.; Bakker, W.; Janssen, L.; Feringa, W.; Gieske, A.; Grabmaier, K.; Hecker, C.; Horn, J.; Kerle, N.; et al. *Principles of Remote Sensing: An Introductory Textbook*, 4th ed.; ITC: Geneva, Switzerland, 2009.

2. He, Z.; Hu, J.; Wang, Y. Low-rank tensor learning for classification of hyperspectral image with limited labeled sample. *IEEE Signal Process.* **2017**, *145*, 12–25.
3. Richards, A.; Xiuping, J.J. Band selection in sentinel-2 satellite for agriculture applications. In *Remote Sensing Digital Image Analysis*, 4th ed.; Springer-Verlag: Berlin, Germany, 2006.
4. Zhang, T.; Su, J.; Liu, C.; Chen, W.; Liu, H.; Liu, G. Band selection in sentinel-2 satellite for agriculture applications. In Proceedings of the 23rd International Conference on Automation & Computing, University of Huddersfield, Huddersfield, UK, 7–8 September 2017.
5. Xie, Y.; Zhao, X.; Li, L.; Wang, H. Calculating NDVI for Landsat7-ETM data after atmospheric correction using 6S model: A case study in Zhangye city, China. In Proceedings of the 18th International Conference on Geoinformatics, Beijing, China, 18–20 June 2010.
6. Gao, B. NDWI—A normalized difference water index for remote sensing of vegetation liquid water from space. *Remote Sens. Environ.* **1996**, *58*, 1–6.
7. Ham, J.; Chen, Y.; Crawford, M.; Ghosh, J. Investigation of the random forest framework for classification of hyperspectral data. *IEEE Trans. Geosci. Remote Sens.* **2005**, *43*, 492–501.
8. Hearst, Marti A. Support Vector Machines. *IEEE Intell. Syst. J.* **1998**, *13*, 18–28.
9. Huang, X.; Zhang, L. An SVM Ensemble Approach Combining Spectral, Structural, and Semantic Features for the Classification of High-Resolution Remotely Sensed Imagery. *IEEE Trans. Geosci. Remote Sens.* **2013**, *51*, 257–272.
10. Delalieux, S.; Somers, B.; Haest, B.; Spanhove, T.; Vanden Borre, J.; Mucher, S. Heathland conservation status mapping through integration of hyperspectral mixture analysis and decision tree classifiers. *Remote Sens. Environ.* **2012**, *126*, 222—231.
11. Kemker, R.; Salvaggio, C.; Kanan, C. Algorithms for semantic segmentation of multispectral remote sensing imagery using deep learning. *ISPRS J. Photogramm. Remote Sens.* **2018**, *145*, 60–77.
12. Pirotti, F.; Sunar, F.; Piragnolo, M. Benchmark of machine learning methods for classification of a sentinel-2 image. In Proceedings of the XXIII ISPRS Congress, Prague, Czech Republic, 12–19 July 2016.
13. Mateo-García, G.; Gómez-Chova, L.; Camps-Valls, G. Convolutional neural networks for multispectral image cloud masking. In Proceedings of the IGARSS, Fort Worth, TX, USA, 23–28 July 2017.
14. Guo, X.; Huang, X.; Zhang, L.; Zhang, L.; Plaza, A.; Benediktsson, J. A. Support Tensor Machines for Classification of Hyperspectral Remote Sensing Imagery. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 3248–3264.
15. Cichocki, A.; Mandic, D.; De Lathauwer, L.; Zhou, G.; Zhao, Q.; Caiafa, C.; Phan, H. Tensor Decompositions for Signal Processing Applications: From two-way to multiway component analysis. *IEEE Signal Process. Mag.* **2015**, *32*, 145–163.
16. Jolliffe, I.T. *Principal Component Analysis*, 2nd ed.; Springer Verlag: New York, NY, USA, 2002.
17. Kolda, T.; Bader, B. Tensor Decompositions and Applications. *SIAM Rev.* **2009**, *51*, 455–500.
18. Lopez, J.; Santos, S.; Torres, D.; Atzberger, C. Convolutional Neural Networks for Semantic Segmentation of Multispectral Remote Sensing Images. In Proceedings of the LATINCOM, Guadalajara, Mexico, 14–16 November 2018.
19. European Space Agency. Available online: https://sentinel.esa.int/web/sentinel/missions/sentinel-2 (accessed on 15 July 2019).
20. Kemker, R.; Kanan, C. Deep Neural Networks for Semantic Segmentation of Multispectral Remote Sensing Imagery. *arXiv* **2017**, arXiv:abs/1703.06452.
21. Hamida, A.; Benoît, A.; Lambert, P.; Klein, L.; Amar, C.; Audebert, N.; Lefèvre, S. Deep learning for semantic segmentation of remote sensing images with rich spectral content. In Proceedings of the IGARSS, Fort Worth, TX, USA, 23–28 July 2017.
22. Wang, Q.; Lin, J.; Yuan, Y. Salient Band Selection for Hyperspectral Image Classification via Manifold Ranking. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *27*, 1279–1289.
23. Li, S.; Qiu, J.; Yang, X.; Liu, H.; Wan, D.; Zhu. Y. A novel approach to hyperspectral band selection based on spectral shape similarity analysis and fast branch and bound search. *Eng. Appl. Artif. Intell.* **2014**, *27*, 241–250.
24. Zhang. L.; Zhang, L.; Tao, D.; Huang, X.; Du, B. Compression of hyperspectral remote sensing images by tensor approach. *Neurocomputing* **2015**, *147*, 358–363.
25. Astrid, M.; Lee, Seung-Ik. CP-decomposition with Tensor Power Method for Convolutional Neural Networks compression. In Proceedings of the BigComp, Jeju, Korea, 13–16 February 2017.
26. Chien, J.; Bao, Y. Tensor-factorized neural networks. *IEEE Trans. Neural Networks Learn. Syst.* **2018**, *29*, 1998–2011.

27. An, J.; Lei, J.; Song, Y.; Zhang, X.; Guo J. Tensor Based Multiscale Low Rank Decomposition for Hyperspectral Images Dimensionality Reductio. *Remote Sens.* **2019**, *11*, 1485.
28. Li, J.; Liu, Z. Multispectral Transforms Using Convolution Neural Networks for Remote Sensing Multispectral Image Compression. *Remote Sens.* **2019**, *11*, 759.
29. An, J.; Song, Y.; Guo, Y.; Ma, X.; Zhang, X. Tensor Discriminant Analysis via Compact Feature Representation for Hyperspectral Images Dimensionality Reduction. *Remote Sens.* **2019**, *11*, 1822.
30. Absil, P.-A.; Mahony, R.; Sepulchre, R. *Optimization Algorithms on Matrix Manifolds*, 1st ed.; Princeton University Press: Princeton, NJ, USA, 2007.
31. De Lathauwer, L.; De Moor, B.; Vandewalle, J. On the best rank-1 and rank-(R 1 ,R 2 ,···,R N ) approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.* **2000**, *21*, 1324–1342.
32. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*, 1st ed.; MIT Press, 2016.
33. Sheehan, B. N.; Saad, Y. Higher Order Orthogonal Iteration of Tensors (HOOI) and its Relation to PCA and GLRAM. In Proceedings of the 7th SIAM International Conference on Data Mining, Minneapolis, MN, USA, 26–28 April 2007.
34. Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495.
35. De Lathauwer, L.; De Moor, B.; Vandewalle, J. A Multilinear Singular Value Decomposition. *SIAM J. Matrix Anal. Appl.* **2000**, *21*, 1253–1278.
36. Rodes, I.; Inglada, J.; Hagolle, O.; Dejoux, J.; Dedieu, G. Sampling strategies for unsupervised classification of multitemporal high resolution optical images over very large areas. In Proceedings of the 2012 IEEE International Geoscience and Remote Sensing Symposium, Munich, Germany, 22–27 July 2012.

**Sample Availability:** Samples of the compounds ...... are available from the authors.