remote sensing

MDPI

Article

A Framework for Evaluating Land Use and Land Cover Classification Using Convolutional Neural Networks

Manuel Carranza-García *, Jorge García-Gutiérrez and José C. Riquelme

Division of Computer Science, University of Sevilla, ES-41012 Seville, Spain; jorgarcia@us.es (J.G.-G.); riquelme@us.es (J.C.R.)

* Correspondence: mcarranzag@us.es

Received: 27 December 2018; Accepted: 28 January 2019; Published: 30 January 2019

Abstract: Analyzing land use and land cover (LULC) using remote sensing (RS) imagery is essential for many environmental and social applications. The increase in availability of RS data has led to the development of new techniques for digital pattern classification. Very recently, deep learning (DL) models have emerged as a powerful solution to approach many machine learning (ML) problems. In particular, convolutional neural networks (CNNs) are currently the state of the art for many image classification tasks. While there exist several promising proposals on the application of CNNs to LULC classification, the validation framework proposed for the comparison of different methods could be improved with the use of a standard validation procedure for ML based on cross-validation and its subsequent statistical analysis. In this paper, we propose a general CNN, with a fixed architecture and parametrization, to achieve high accuracy on LULC classification over RS data from different sources such as radar and hyperspectral. We also present a methodology to perform a rigorous experimental comparison between our proposed DL method and other ML algorithms such as support vector machines, random forests, and k-nearest-neighbors. The analysis carried out demonstrates that the CNN outperforms the rest of techniques, achieving a high level of performance for all the datasets studied, regardless of their different characteristics.

Keywords: convolutional neural network; cross-validation; deep learning; land use classification; land cover classification; remote sensing; statistical analysis

1. Introduction

Land use and land cover (LULC) classification is a fundamental task in remote sensing (RS). The automated extraction of the earth's surface information currently plays an essential role in many RS applications such as change detection [1], environmental monitoring [2], urban expansion control [3], infrastructure planning [4], and biodiversity analysis [5]. The ability to identify the physical material of the surface or the human exploitation of land by analyzing aerial imagery has led to a significant cost and time improvement when carrying out these studies, with respect to traditional field surveys. LULC classification is the problem of assigning a class label to every pixel of an image. It often proves to be a challenging task due to the heterogeneous appearance and high intraclass variance of elements [6], the large number of spectral channels and the limited amount of labeled samples in the images [7]. However, recent technological advancements in optics and the space industry have increased the availability and quality of RS data, which vary in spatial, spectral, radiometric, and temporal resolutions [8]. The wide diversity of datasets available (e.g., aerial photographs, thermal imagery, hyperspectral imagery, radar and lidar) has opened up many lines of research and has inspired the development of new techniques for digital pattern classification.

The RS community has tried to address the problem from many different perspectives, with the purpose of seeking an efficient method for mapping LULC patterns. These studies range from conventional statistical approaches to more powerful machine learning (ML) algorithms that have enhanced the quality of the solutions for this problem. Traditional RS data classification methods include maximum-likelihood classifier (MLC) [9], distance measure [10], clustering [11] or logistic regression [12]. Over the last decade, more advanced methods such as decision trees [13], k-nearest-neighbors (kNN) [14], random forest (RF) [15], neural networks [16] and support vector machines (SVM) [17] have been used for LULC mapping. In 2016, a study on the state of the art of supervised methods for LULC classification was performed [18]. It was reported that SVM, kNN, and RF generally provide better performance than other traditional classifiers, SVM being the most efficient method. Very recently, deep learning (DL) techniques have become the state of the art in many fields such as image recognition [19]. The capacity of DL models to automatically learn informative representations of raw input data with multiple levels of abstraction has allowed for achieving high performance in many computer vision tasks [20]. Therefore, due to its rising potential, DL has lately been applied with success to LULC classification and recent studies demonstrate that it outperforms SVM approaches [21,22]. Several DL architectures have been analyzed for RS tasks: convolutional neural networks (CNNs) [23,24], stacked autoencoders (SAEs) [25], and deep belief networks (DBNs) [26]. In particular, recent surveys place CNNs as the most used model, as it generally provides a better performance [20]. The deep structure of a CNN allows for learning effective features by concatenating convolution, pooling, and fully connected layers. In addition, their capacity to encode spectral and spatial information of RS images, by using contextual information of the neighbourhood of the pixels, has led to a substantial improvement in the efficiency of the classification. Furthermore, the properties of local connectivity and parameter sharing of the CNNs vastly reduces the number of parameters that would have to be learned for multidimensional images if using a standard neural network, thus decreasing processing effort. Numerous methods based on DL have been proposed recently for LULC classification over specific RS data, especially focusing on high resolution and hyperspectral images [27,28]. Nevertheless, due to the abundance of data from distinct sources and characteristics, there is the need for developing models that are applicable for images at different scales which vary on their spatial resolution and number of spectral bands. Therefore, our aim in the present study was to build a general CNN model for LULC classification that could be used for diverse RS images such as radar or hyperspectral.

Furthermore, due to the considerable amount of studies with different proposals, it is essential to establish a standard validation procedure in order to perform a fair comparison between techniques. With the aim of drawing stronger conclusions on how novel methods improve the existing ones, the validation framework proposed in LULC studies could be improved with the use of a standard testing methodology for ML such as cross-validation and its subsequent statistical analysis. Fairly often, in remote sensing, holdout validation is carried out for comparing classifiers over multiple datasets. However, the use of repeated stratified cross-validation has been shown to be the most suitable procedure for ML problems, as it provides a reliable estimate of the capacity of generalization of different models [29]. Furthermore, the statistical validation of results is a necessity in order to establish a certain conclusion on an experimental analysis over a ML problem. Statistics allow us to verify the hypothesis of improved performance between proposals, indicating how significant our compared results are with respect to the experimentation that we have carried out [30]. Accordingly, in this paper, we propose a validation procedure to carry out an experimental comparison between the proposed DL model for LULC classification and some other ML well-known techniques such as k-nearest-neighbours, random forest, and support vector machines.

In summary, the main scientific contributions on this paper can be condensed as follows:

- A general 2D CNN, with a fixed architecture and parametrization, to achieve high accuracy in LULC classification over remote sensing imagery from different sources, concretely radar and hyperspectral images.

- A validation methodology, based on cross-validation and statistical analysis, in order to perform a rigorous experimental comparison between the performance of our proposed DL architecture and other traditional ML models.

The rest of the paper is organised as follows: Section 2 describes the materials used for the study and presents our proposed DL architecture with the experimental setup. The results obtained when applying the presented methods to the selected datasets are reported and discussed in Section 3. Finally, Section 4 summarises the main conclusions and presents a number of possibilities of further research.

2. Materials and Methods

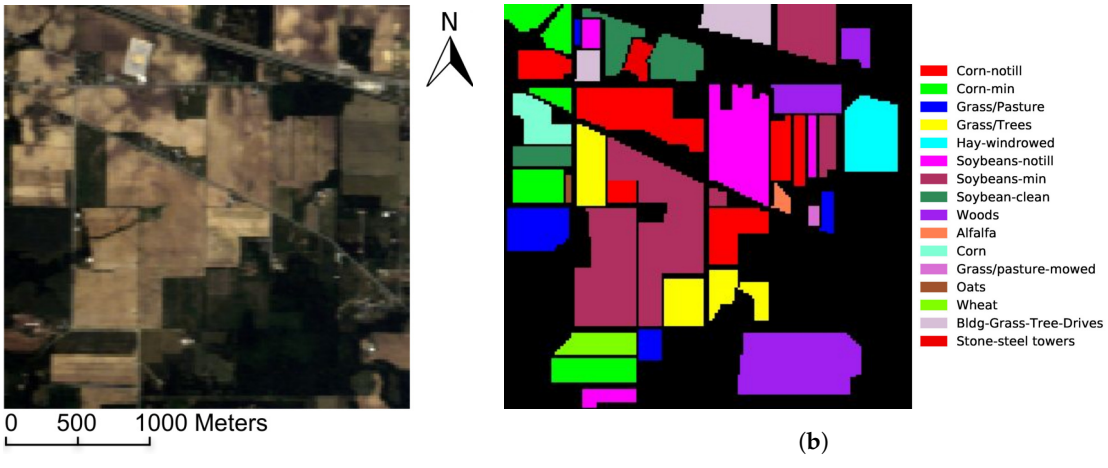
In this section, we first present the RS images selected for our study. Secondly, we describe the necessary steps performed on the original images to obtain the data that would be fed into the ML models. Thirdly, we illustrate the design of our proposed CNN architecture for LULC classification, explaining its parametrization in detail. Finally, we describe the experimental setup proposed for comparing our DL model with other techniques.

2.1. Description of Datasets

Five publicly available datasets have been selected for our study. We have used three widely known hyperspectral images (Indian Pines, Pavia University, and Salinas), and two radar images from the JPL AirSAR (San Francisco and Flevoland). They all comprise a variety of scenes in diverse environments (urban and rural) with different sizes, resolution, and number of classes, which aims to prove the generality of our model. Table 1 presents the information of each dataset, including type of image, size, number of bands, spatial resolution and number of classes to consider. Figures 1–5 show the false color images corresponding to each dataset. For the hyperspectral images, we have found the complete ground truth maps, hence they are also displayed with a colour legend for the classes. For the radar images, partial ground truths have been used, which are depicted in Section 3.3.

Table 1. Description of datasets.

Dataset	Type (Sensor)	Size	# Bands	Spatial Resolution	# Classes
Indian Pines	Hyperspectral (AVIRIS)	145 × 145	220	20 m	16
Salinas	Hyperspectral (AVIRIS)	512 × 217	204	3.7 m	16
Pavia	Hyperspectral (ROSIS)	610 × 340	103	1.3 m	9
San Francisco	Radar (AirSAR)	1168 × 2531	27 (P, L & C)	10 m	3
Flevoland	Radar (AirSAR)	1279 × 1274	27 (P, L & C)	10 m	12



(**a**) (**b**)

Figure 1. Indian Pines dataset. (**a**) false color composite image (bands 29, 19, and 9); (**b**) ground truth.

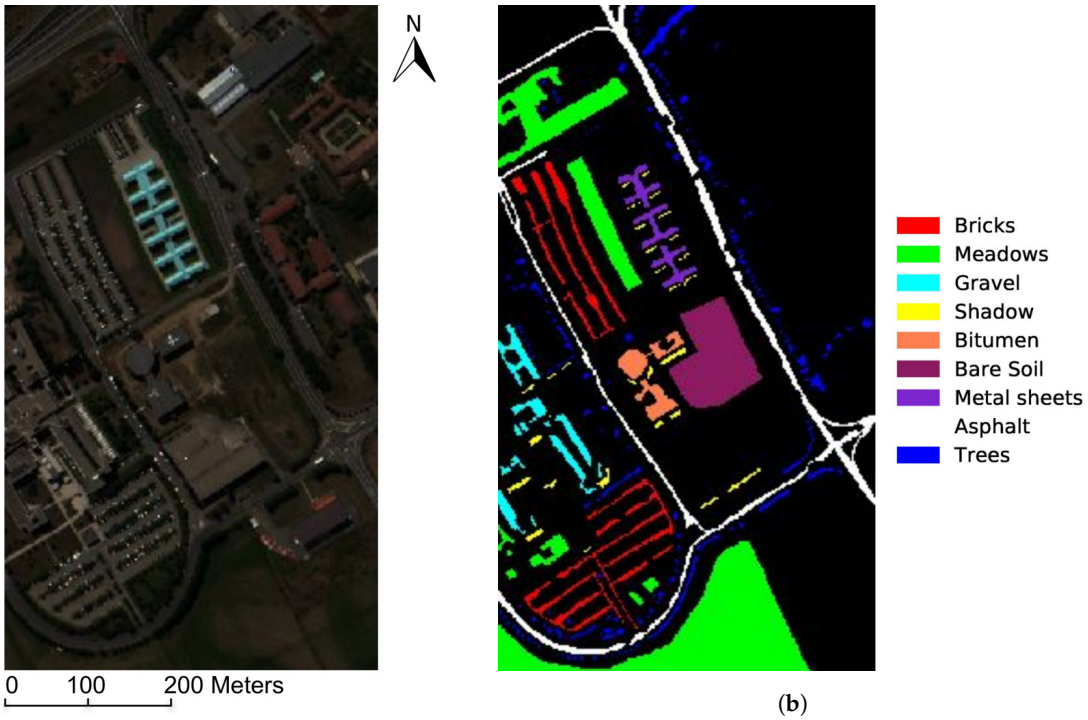


Figure 2. Pavia University dataset. (**a**) false color composite image (bands 45, 27, and 11); (**b**) ground truth.

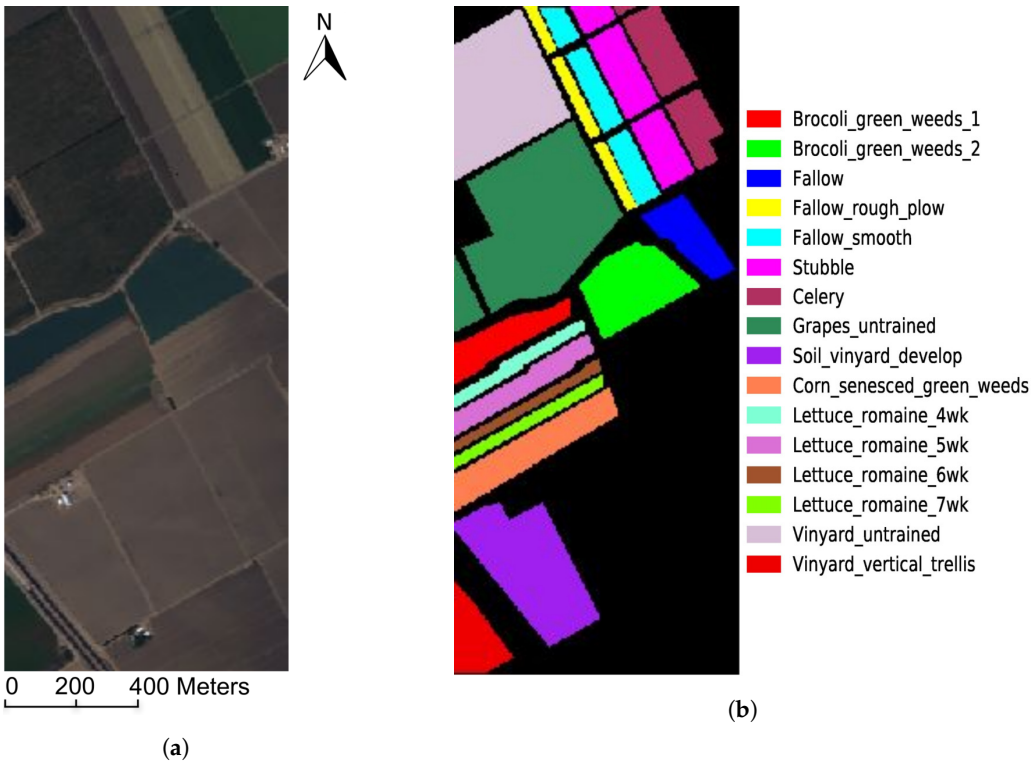


Figure 3. Salinas dataset. (**a**) false color composite image (bands 29, 19, and 9); (**b**) ground truth.

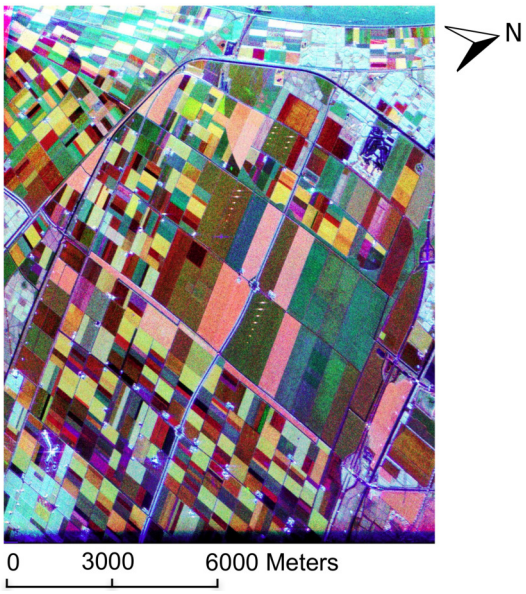


Figure 4. Flevoland dataset.

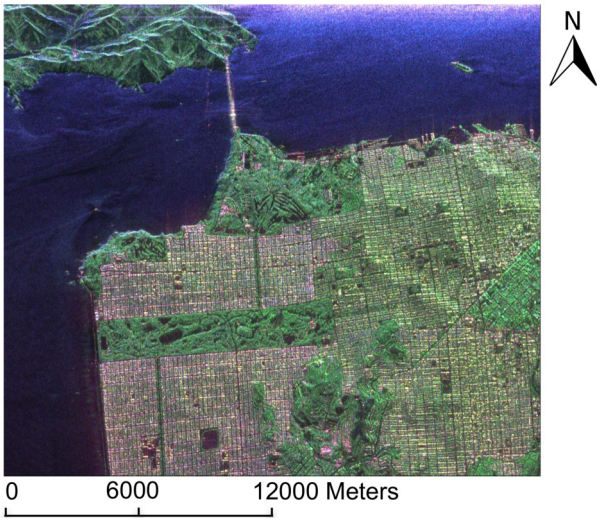


Figure 5. San Francisco dataset.

2.2. Data Generation

In this subsection, we describe the method to generate the instances that will be used for performing pixel classification with the ML models, which is illustrated in Figure 6. The patch-based approach we present here has been proposed in recent studies of LULC classification as the most suitable for a CNN model [31,32]. From the complete original images described above, whose pixel values are scaled between 0 and 1 using the Min-Max scaler, we generate as many instances as labeled pixels they have. The method consists of extracting small three-dimensional patches of fixed size ($P \times P \times B$: P being the spatial neighbourhood considered and B the number of bands) centered at each pixel, instead of working with the information of a single pixel. From these patches, we aim to classify the central pixel by extracting local spatial features from neighbouring pixels. The reason for utilizing patches is based on the observation that pixels inside a small spatial neighborhood often reflect the same underlying material [27]. The optimal size of this patch will be highly dependent on the size and resolution of the image considered. In our study, we have found that a size of 5 works well for all the datasets analyzed, after trying with different values ranging from 3 to 15. For those pixels close to the border of the image that do not have enough spatial context to extract a complete

patch, an edge padding operation is performed. Once the patches have been collected, we propose a further improvement. We carry out a simple data augmentation just based on rotation. This action will allow the models to learn different spatial distributions inside the patches, thus reducing overfitting problems. We studied the performance after applying three rotations (90 degrees) or seven rotations (45 degrees), finding that the latter option provided better results.

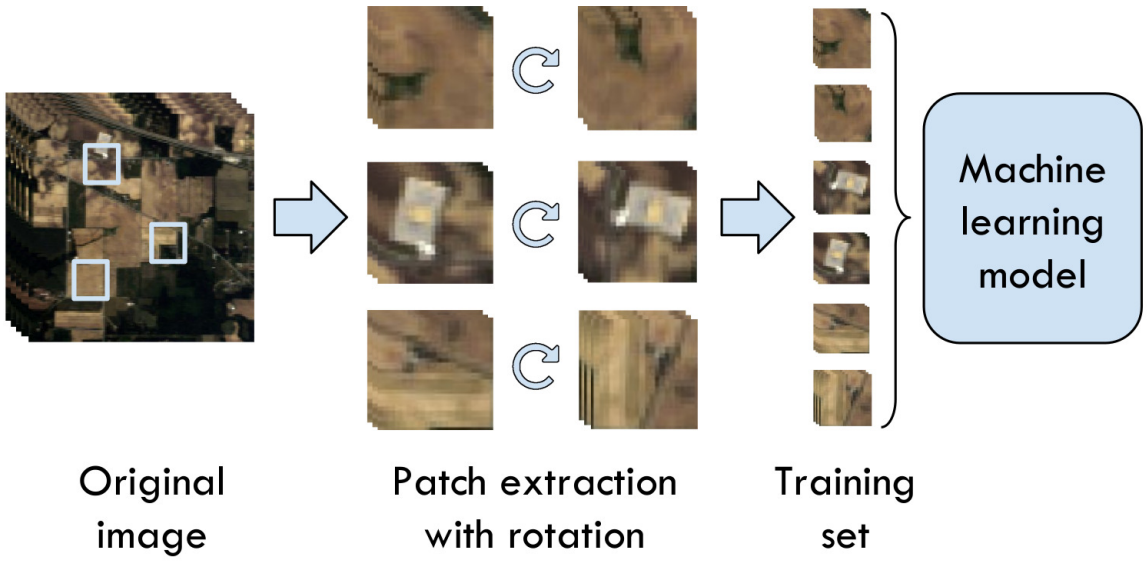


Figure 6. Patch extraction process to generate the instances that would be fed to the ML models.

2.3. 2D Convolutional Neural Network

The CNN model for LULC classification presented in this section aims to provide a general architecture to be used with RS data from different sources and characteristics such as radar or hyperspectral imagery. Since the patches obtained from the RS images (Section 2.2) have three dimensions, the objective of the designed network is to perform 2D convolution over the complete 3D input volume (Equation (1)), thus considering all spectral channels of these patches. This is the same principle used in well-known architectures such as AlexNet [33] or VGG [34], which have achieved high success in many image recognition tasks. Therefore, we have adapted these types of architectures to work with different RS images that have a larger number of spectral bands compared to conventional RGB images.

2D CNNs act as feature extractors that automatically learn informative representations of raw input data with multiple levels of abstraction. The proposed model combines several convolutional, nonlinearity, max-pooling, and fully connected layers to achieve this purpose. In the convolutional layers, the input data composed of n maps is convolved with n two-dimensional kernels of size $K_i \times K_j$, where i and j represent the size of each dimension. During the feed-forward process of the neural network, each kernel is convolved across the height and width of its corresponding input map, and the sum of the n convolutional responses produces a 2D feature map that is passed through a nonlinear activation function. The convolution process can be formulated as follows [35]:

$$a_{ln}^{xy} = g\left(\sum_{m}\sum_{i=0}^{I_l-1}\sum_{j=0}^{J_l-1} w_{lnm}^{ij} a_{(l-1)m}^{(x+i)(y+j)} + b_{ln}\right), \quad (1)$$

where l indicates the current layer that is considered; n is the number of feature maps in this layer; a_{im}^{xy} is the value of a neuron at position (x, y) of the nth feature map in the ith layer; g is the activation function; m indexes over the set of features maps in the $(l-1)$th layer connected to the current feature map; I_l and J_l are the height and width of the convolutional kernel; w_{lnm}^{ij} stands for the weight of position (i, j) connected to the mth feature map; and b_{ln} is the bias. Rectified Linear Units (ReLU) layers

are used to compute the nonlinear activation function $g(x) = max(0, x)$, which generally improve the convergence when training the network compared to the sigmoid function [36].

Max-pooling layers are used to progressively reduce the spatial size of the feature maps. This operation reduces the amount of parameters, hence computation in the network, and helps to control overfitting by selecting superior invariant features. In these layers, input maps are downsampled by a factor of F_x and F_y along both width and height, remaining the depth dimension unaltered. For each non-overlapping rectangular region of size (F_x, F_y) of the input, the output is given by its maximum activation value [37].

Unlike convolutional layers, fully connected layers have full connections to all activations in the previous layer. It combines all the extracted features after the convolution into a one-dimensional feature vector. The last fully connected layer of the network has one output neuron per class and performs the classification task by applying a softmax activation function. The output of this function is a probability distribution representing the likelihood of belonging to a certain class. This is formulated in Equation (2), where p is the probability distribution, i represents a certain class, and z is the C-dimensional vector of real-valued scores of the output layer of the network, C being the number of classes:

$$p_i(z) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}. \tag{2}$$

In all the layers described, the value of the weights and biases of the connections are initialized randomly using Xavier initialization, which aims to set the values within a reasonable range before starting the training procedure [38]. However, the optimal set of values has to be found in order to adjust the model to the training data, hence the whole process can be seen as an optimization problem. The network is fed with input data and the weights and biases values are updated by minimizing a loss function H. The loss function quantifies how well the network outputs, given a set of parameters, match the ground truth labels in the training set. The usual loss function applied at softmax layers is the cross-entropy (Equation (3)), where y is the one-hot encoded vector of the labels, p is the predicted probability distribution given by the softmax function (Equation (2)), and i indexes over the set of classes:

$$H(y, p) = -\sum_{i} y_i log(p_i). \tag{3}$$

For carrying out the optimization process, the mini-batch gradient descent with back-propagation algorithm is used [34]. It iteratively computes the gradient of the cost function H and updates the model's parameters. The mini-batch strategy only considers a small set of training samples for computing the gradient and it is useful for large dataset processing. The weights are updated using a mini-batch of size n as described in Equation (4), where k is the iteration index, η is the learning rate, and $x^{(i)}$ and $y^{(i)}$ represent, respectively, the training example i and its label:

$$w_{k+1} = w_k - \eta_k \nabla H(w_k; x_k^{(i:i+n)}; y_k^{(i:i+n)}). \tag{4}$$

2.3.1. CNN Architecture

The DL model we propose is a 2D CNN composed of two convolution blocks with max-pooling, and a fully connected layer that combines the extracted features after the convolution, as shown in Figure 7.

In the architecture proposed, the number of feature maps in consecutive layers is incremented to be able to properly encode the increasingly richer representations obtained: the first convolution layer has 32 filters, the second convolutional has 64 filters, and the fully connected layer has 1024 neurons. The convolutional filters have stride one (step of the sliding convolution window) and zero padding to maintain the spatial input size. The convolutional kernel size is set to three as it provides better performance in deep architectures [34]. The ReLU activation function is used after all convolution layers. The max-pooling is applied with a stride of two, in order to subsequently reduce the spatial

dimension of the patch, focusing on the central pixel. With these downsampling operations, there is only a need for two convolutional blocks since the input images are small (5×5). The final convolved features are transferred to the fully connected layer that has 1024 neurons. Finally, a layer with softmax activation computes the probability for each class and yields the predictions.

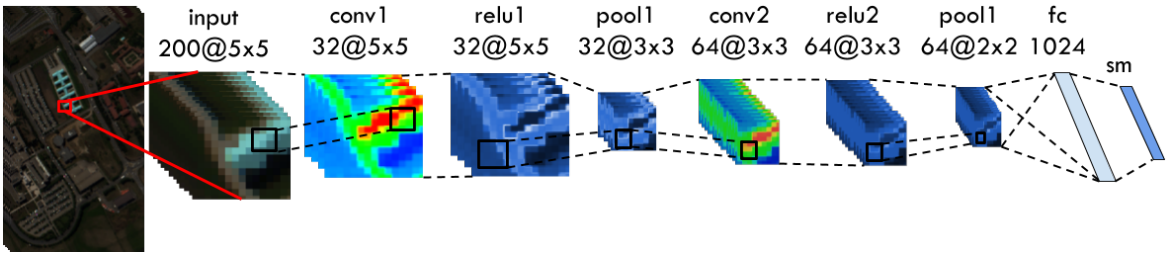


Figure 7. CNN architecture. The *conv* layers refer to convolutional layers, *pool* to max-pooling layers, *fc* to the fully connected layer, and *sm* to the softmax layer. Dropout and batch normalization are omitted to simplify the visualization.

Furthermore, dropout and batch normalization layers are introduced in the architecture, which generally improve the capacity of generalization of the network by reducing overfitting (further explained in Section 2.3.2). Batch normalization also helps to speed up the learning process of the network. It reduces the covariate shift of values of hidden units by normalizing layer inputs for each training mini-batch. This allows the network to train faster as it prevents layers from having to adapt to substantial changes in the distribution of output values of previous layers [39]. In the model proposed, dropout is applied at the fully connected layer, and batch normalization is used at the input and after convolution.

A summary of the architecture is presented in Table 2, detailing the number of layers proposed, the number of filters and the kernel sizes.

Table 2. CNN architecture. P refers to the input patch size (5 in our case), N to the number of channels of the input image, and C to the number of classes to consider.

CNN Architecture			
Layer	**Type**	**Neurons & # Maps**	**Kernel**
0	Input	$P \times P \times N$	
1	Batch normalization	$P \times P \times N$	
2	Convolutional	$P \times P \times 32$	3×3
3	ReLU	$P \times P \times 32$	
4	Batch normalization	$P \times P \times 32$	
5	Max-Pooling	$\lceil P/2 \rceil \times \lceil P/2 \rceil \times 32$	2×2
6	Convolutional	$\lceil P/2 \rceil \times \lceil P/2 \rceil \times 64$	3×3
7	ReLU	$\lceil P/2 \rceil \times \lceil P/2 \rceil \times 64$	
8	Batch normalization	$\lceil P/2 \rceil \times \lceil P/2 \rceil \times 64$	
9	Max-Pooling	$\lceil P/4 \rceil \times \lceil P/4 \rceil \times 64$	2×2
10	Fully connected	1024 neurons	
11	Dropout	1024 neurons	
12	Softmax	C neurons	

2.3.2. Regularization Techniques

Training with little input data of high dimensionality and with deep architectures having a large number of parameters frequently produces overfitting problems in CNNs. The model is adjusted excessively to the training data, without being able to generalise when unseen data is classified. Therefore, it is necessary to apply regularization techniques in the network in order to reduce this problem. In the designed CNN architecture, there are several elements that contribute to diminishing overfitting.

One of the most commonly used technique is dropout. It consists of randomly switching off certain hidden neurons in the training procedure of the network. This method keeps a neuron activated

according to some configurable probability or sets its output to zero, thus not contributing in the back-propagation algorithm. It has proven to be very effective since it prevents complex co-adaptations by training the network with different dropped neurons on every iteration [40].

Apart from improving convergence, batch normalization layers also act as regularizers [41]. They add random noise to the value of hidden units, by subtracting the mean and dividing by the standard deviation of the batches. Moreover, applying this normalization reduces the need for dropout, hence a lower dropout rate can be used and the loss of information is smaller. However, the combination of these regularization techniques is complex. According to [42], applying dropout before a batch normalization layer often leads to unstable numerical behavior in inference that introduces more erroneous predictions. Following this theory and after trying several combinations, we have found that the best results are obtained if we place a total of three batch normalization layers (one at the first input layer and the other two after each convolution operation), and only one dropout layer with a small drop ratio just before the final softmax prediction layer.

Furthermore, the introduction of nonlinearity through the ReLU activation function and the reduction of parameters provided by the max-pooling operations also helps to control overfitting. With the combination of all the described techniques and with the aid of performing data augmentation by rotating the patches, the network can handle the overfitting problem successfully. Therefore, it is able to achieve a high level of accuracy on classifying data that has not been considered for training.

2.3.3. Parameter Selection

Besides the configuration of the architecture, a CNN requires the selection of several parameters for the training process: number of training epochs (number of times the algorithm sees the entire data set), batch size (number of training instances used in one iteration), learning rate, and dropout rate. Choosing the optimal set of values for the parameters is complex and requires a preliminary study. The most appropriate configuration for performing all experiments has been obtained by a grid search (Table 3). The optimal parameter values were selected from the mean of best results of five repetitions of cross-validation using three of the images (Indian Pines, Pavia, and San Francisco). For training the network, the gradient descent algorithm with back-propagation is selected as the optimizer with the cross entropy as the cost function to minimise. The number of epochs has been fixed to 50, as it was sufficient for the network to converge to a good solution and an increment provided no further improvements. The learning rate initial value is set to 0.01 and it is decreased exponentially every epoch. Choosing a small batch size of 16 provided the best performance as it allows minority examples to play a more important role in the weight adjustment process, thus helping in reducing the imbalance problem between classes.

Table 3. Grid search and selected values for the training parameters of the CNN.

CNN Parameter Selection		
Parameter	**Grid Search**	**Selected Value**
Dropout rate	{0.2, 0.5}	0.2
Learning rate	{0.1, 0.01, 0.001}	0.01
Decaying learning rate	{True, False}	True
Number of epochs	{20, 50, 80, 100}	50
Batch size	{16, 32, 64, 128}	16

2.4. Experimental Setup

In this section, the proposed validation procedure for comparing the performance of different classifiers is presented. Several experiments over different RS images have been carried out to correctly assess the efficiency of the proposed CNN model. We have also evaluated the datasets over other extensively used classifiers such as k-nearest-neighbours, random forests, and support vector machines. For comparing all methods, our proposal is to perform a repeated stratified cross-validation test for each

dataset, followed by a statistical analysis combining all the results. Figure 8 illustrates the proposed experimental process, which is explained in detail in the following subsections. Finally, the models have also been tested at the Geoscience Remote Sensing Society (GRSS) DASE website competition [43]. The source code, with the implementation of the model and the experimental setup with the seeds for the cross-validation procedure, can be found at [44].

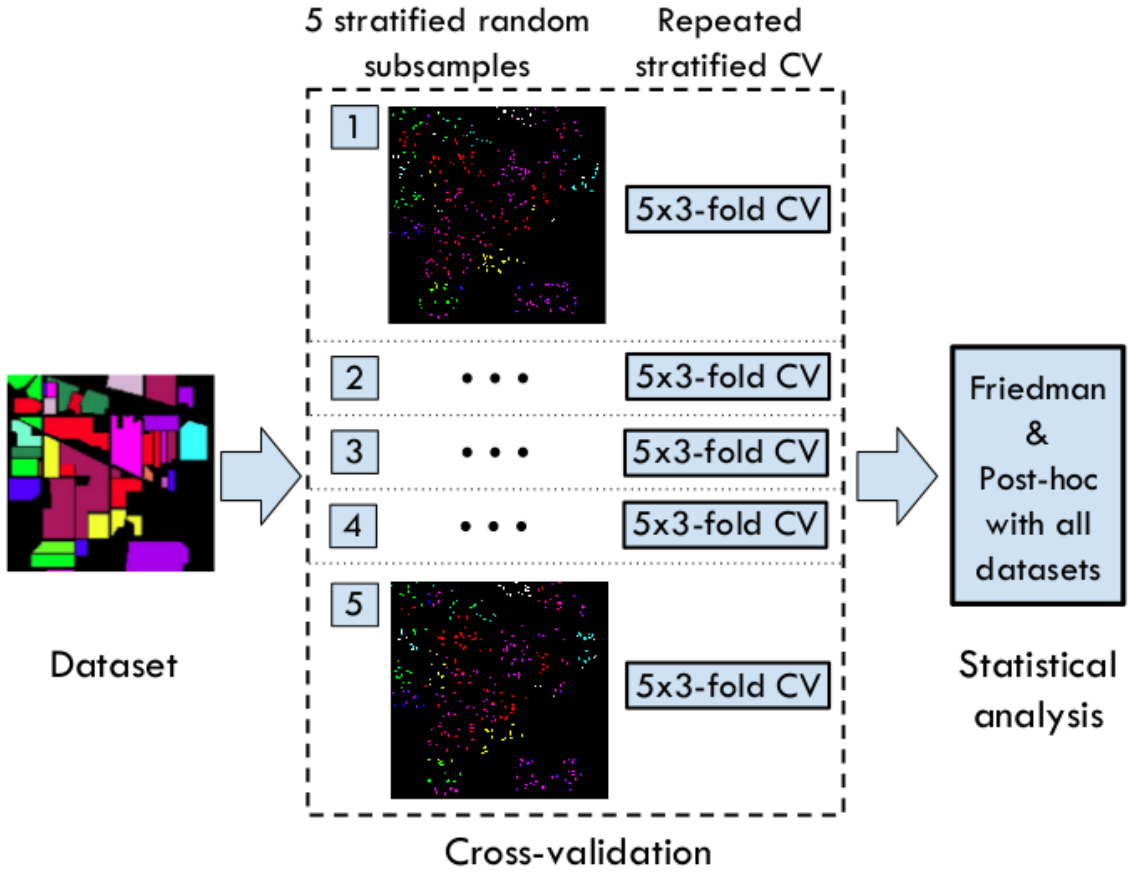


Figure 8. Experimental setup. For each dataset, we obtain five independent subsamples and perform repeated stratified cross-validation. The collected results of all datasets are analyzed with a statistical test (Friedman and Post-hoc).

2.4.1. Stratified Cross-Validation

Repeated stratified cross-validation has been recommended as the standard procedure in ML for evaluating model accuracy by several studies [29,45] because it provides a stable and reliable estimate with low bias. With the repetitions, a large number of performance estimates is obtained, thus reducing the variance of the estimation.

For our cross-validation, in order to perform a more rigorous and realistic experimentation in which the number of labeled training samples is reduced, each dataset has been separated into five random stratified subsamples. Later, all of these independent subsampled datasets have been tested over a 5 $\times$ 3-fold stratified cross validation with our DL model and the other methods considered. Then, we extract the average results of the five repetitions for each experiment, obtaining a total of 25 results (5 images $\times$ 5 subsamples) for each method. The choice of a 3-fold division is based on the fact that, since the folding is stratified and the number of samples is not very large, a greater number of folds would leave minority classes with few or no test instances, which would not provide representative results. Furthermore, related RS studies that have tested the proposed classifiers using cross-validation procedures have also adopted a three-fold division of the samples [46,47].

For all experiments, we apply the same steps for generating the training data and the same parameter choice for each method. The kNN algorithm has been tested using one, three, and five neighbours. Moreover, as dimensionality reduction is essential for kNN to avoid the Hughes phenomenon [48], we apply feature selection with the extra-trees method [49], a further randomised version of RF. For the SVM, we use the RBF kernel and the best combination of values for the parameters found on a grid search: γ (kernel coefficient) $\in \{0.01, 0.1, 0.25, 0.5, 1\}$ and C (penalty) $\in \{1, 10, 50, 100\}$. The most suitable values found were: $\gamma = 0.01$, $C = 50$. Finally, for the RF method, we fix the model to have 100 estimators and no depth limit.

2.4.2. Statistical Analysis

Since we cannot only rely on the average accuracy given by the cross-validation to compare the learning algorithms, we should perform an appropriate hypothesis statistical test for comparison [50]. Therefore, the 25 results obtained for each method (one for each subsampled dataset) are compared statistically using the Friedman test, followed by a post hoc analysis. The Friedman test is the recommended method for comparing multiple classifiers over multiple datasets in ML [30]. Being non-parametric, it is a more appropriate and safer test since it does not require any assumptions about the distribution of the samples. It ranks the algorithms for each data set separately and allows for detecting differences considering the global set of classifiers. Once the Friedman test rejects the null hypothesis, which is that all the algorithms are equivalent so their rank should be equal, a post hoc analysis should be carried out in order to perform pairwise comparison and find whether the statistical difference between the proposed technique and the others is significant.

For this purpose, we use the Holm's procedure, which works with a control algorithm and compares it with the rest of techniques. In this case, the aim is to reject the null hypothesis for each pair (no statistical significance between the control method and each of the remaining). This post hoc method intends to control the family-wise error rate when comparing multiple hypothesis, which is the probability of making one or more type I errors (rejecting a true null hypothesis). It is a step-down procedure that sequentially tests the hypotheses ordered by their significance, starting with the one with the lowest p-value. The test works as follows: if p_1 is smaller than an adjusted α value (appropriate level of confidence), the corresponding hypothesis is rejected and we can continue with p_2 and α_2. If we reject the second hypothesis, the test proceeds with the third, and so on. When a certain null hypothesis cannot be rejected, all the remaining are retained as well.

2.4.3. GRSS DASE Website Competition

Four of the datasets (Indian Pines, Pavia University, Flevoland, and San Francisco) were found to be available in the competition of the GRSS DASE Website. This site provides a train set, and allows for uploading a classification map from which it automatically computes a set of accuracy parameters with respect to undisclosed test samples. For this competition, we have used the same patch extraction procedure (Section 2.2) and we have tested the performance of all the models studied with the same parametrization as for the cross-validation experiments. In this case, a further post-processing step has been used for trying to improve performance. We propose the application of a simple modal filter with kernel size three to the complete predicted image. This would help for cleaning out the salt and pepper noise often produced when making predictions, thus producing a visually more attractive classification map.

3. Results and Discussion

In this section, we present and discuss the results obtained from the cross-validation experiments and the subsequent statistical analysis. Finally, we also show the accuracies and classification maps obtained for the GRSS DASE Website competition. For all tests, we have used a computer with an Intel Core i7-8700 CPU and a NVIDIA Titan Xp 12GB GPU.

3.1. Cross-Validation Results

In this subsection, the results of the 5 × 3-fold cross-validation experiments are reported. For each image, we have collected the mean accuracies of the five subsampled datasets to analyze the behaviour of the different methods (1NN, 3NN, 5NN, SVM, RF, and CNN). Tables 4–8 present for each dataset: the sample distribution of the pixels, the overall accuracy (OA), the average accuracy (AA), and the individual class results for all methods.

As it can be seen in Figure 9, which presents a summary of the overall accuracies, the proposed CNN outperforms the rest of techniques, achieving very high accuracies ranging from 96.78% to 99.36%. The increase in performance of the CNN is especially significant in the case of the smallest dataset, Indian Pines, obtaining an almost 10% difference with respect to the second best classifier (SVM). We can also observe that, as it was stated by the study of the literature [18], in general, SVM is the second most powerful classifier, closely followed by the RF. Both the SVM and RF achieve accuracies higher than 94% for all datasets except for Indian Pines. The poorest performance is given by the different versions of the kNN algorithm, which fail to be efficient particularly in the smallest and largest dataset (Indian Pines and Flevoland). Furthermore, during the experiments, we have observed that our CNN model was able to fit perfectly to the training set while maintaining a high accuracy for the test data. This fact reflects the effect of the regularization techniques applied in the network, which have prevented the model from overfitting.

Another factor worth considering is that the CNN has higher average accuracy for all the datasets (Tables 4–8), which implies a more stable behaviour for the classification of all classes, regardless of their distribution. It seems that the CNN configuration, especially due to the regularization techniques and the mini-batch training, allows for achieving a better performance on minority classes, thus alleviating the imbalance problem of these kinds of datasets. This fact can be seen, for example, in class number 3 of the San Francisco image (Table 7), and classes 9 and 12 of the Flevoland dataset (Table 8). This surprising performance, even when limited data on certain classes is available, appears to be an important advantage of CNNs compared to other methods [51].

Table 4. Indian Pines cross-validation results. Sample distribution in each experiment and individual class, overall, and average accuracies of all methods. Best results are highlighted in bold.

Indian Pines Cross-Validation								
Sample Distribution			**Accuracy**					
#	**Class**	**Samples**		**1NN**	**3NN**	**5NN**	**SVM**	**RF**	**CNN**
1	Corn-notill	286		76.19	72.26	66.08	81.75	68.21	**95.78**
2	Corn-min	166		69.28	59.02	54.05	74.60	62.03	**93.32**
3	Grass/Pasture	97		93.83	91.35	88.38	92.59	88.45	**96.56**
4	Grass/Trees	146		98.33	97.36	97.20	97.94	98.79	**99.54**
5	Hay-windrowed	96		99.37	99.24	99.16	99.20	99.58	**99.92**
6	Soybeans-notill	195		86.97	79.54	77.69	81.71	67.21	**95.33**
7	Soybeans-min	491		86.83	81.21	79.79	92.40	91.40	**97.22**
8	Soybean-clean	119		64.02	42.03	32.77	72.99	46.53	**96.70**
9	Woods	253		89.23	86.46	85.50	97.00	95.99	**98.46**
10	Alfalfa	10		62.67	35.89	13.44	31.89	25.22	**94.22**
11	Corn	48		48.05	22.18	14.25	61.40	32.34	**95.35**
12	Grass/pasture-mowed	6		70.00	51.33	26.00	35.33	11.33	**94.67**
13	Oats	4		69.33	31.33	10.67	44.67	22.00	**76.02**
14	Wheat	41		98.61	97.13	95.39	98.40	97.63	**99.90**
15	Bldg-Grass-Tree-Drives	78		42.43	23.78	14.09	73.35	58.25	**94.75**
16	Stone-steel towers	19		93.27	79.05	74.16	95.14	79.81	**99.59**
	Total	2055	OA	82.18	75.24	71.81	86.80	78.98	**96.78**
			AA	78.03	65.57	58.04	76.90	65.30	**95.78**

Table 5. Pavia University cross-validation results. Sample distribution in each experiment and individual class, overall, and average accuracies of all methods. Best results are highlighted in bold.

Pavia University Cross-Validation									
Sample Distribution				Accuracy					
#	Class	Samples		1NN	3NN	5NN	SVM	RF	CNN
1	Self-Blocking Bricks	737		92.48	92.09	91.47	**97.72**	95.62	96.14
2	Meadows	3730		99.18	99.43	99.52	99.85	98.83	**99.97**
3	Gravel	420		87.62	85.18	83.52	88.67	81.71	**91.25**
4	Shadow	190		99.85	99.79	99.83	99.87	**100**	99.64
5	Bitumen	266		92.60	91.47	90.92	90.03	87.67	**93.43**
6	Bare Soil	1006		75.67	66.28	59.89	95.65	79.80	**99.14**
7	Metal sheets	269		100	100	100	100	99.66	100
8	Asphalt	1327		95.21	93.75	92.69	96.33	97.96	**98.24**
9	Trees	613		89.45	86.19	83.52	97.92	95.68	**98.56**
	Total	8558	OA	93.80	92.15	90.94	97.65	94.82	**98.45**
			AA	92.45	90.46	89.04	96.26	92.99	**97.35**

Table 6. Salinas cross-validation results. Sample distribution in each experiment and individual class, overall, and average accuracies of all methods. Best results are highlighted in bold.

Salinas Cross-Validation									
Sample Distribution				Accuracy					
#	Class	Samples		1NN	3NN	5NN	SVM	RF	CNN
1	Brocoli_green_weeds_1	402		99.26	98.94	98.75	99.84	99.99	99.99
2	Brocoli_green_weeds_2	746		99.63	99.53	99.36	99.90	**100**	99.28
3	Fallow	396		99.72	99.74	99.77	**99.86**	99.36	99.59
4	Fallow_rough_plow	279		99.76	99.66	99.58	99.76	**99.83**	99.44
5	Fallow_smooth	536		98.99	98.81	98.48	99.76	**99.42**	98.48
6	Stubble	792		100	100	100	100	100	100
7	Celery	716		99.90	99.83	99.77	99.94	**99.97**	99.95
8	Grapes_untrained	2255		82.64	83.15	83.31	92.16	91.43	**92.81**
9	Soil_vinyard_develop	1241		99.77	99.51	99.23	**99.93**	99.91	99.91
10	Corn_green_weeds	656		97.08	95.48	94.41	**98.70**	97.62	98.47
11	Lettuce_romaine_4wk	214		99.23	98.37	98.03	99.03	98.95	**99.44**
12	Lettuce_romaine_5wk	386		100	99.98	100	100	100	**100**
13	Lettuce_romaine_6wk	184		99.98	100	100	99.98	99.96	99.52
14	Lettuce_romaine_7wk	214		98.43	97.44	96.96	**99.24**	97.70	99.20
15	Vinyard_untrained	1454		81.28	81.15	80.84	80.34	77.01	**91.34**
16	Vinyard_vertical_trellis	362		98.94	98.29	97.90	99.26	98.90	**99.58**
	Total	10833	OA	93.46	93.33	93.16	95.54	94.81	**97.03**
			AA	97.16	96.87	96.65	97.98	97.50	**98.56**

Table 7. San Francisco cross-validation results. Sample distribution in each experiment and individual class, overall, and average accuracies of all methods. Best results are highlighted in bold.

San Francisco Cross-Validation									
Sample Distribution				Accuracy					
#	Class	Samples		1NN	3NN	5NN	SVM	RF	CNN
1	Ocean	3383		100	100	100	100	100	100
2	Urban	3594		91.50	94.92	96.04	94.27	**99.38**	96.97
3	Mixed trees/grass	770		35.83	28.93	25.48	71.88	43.93	**85.97**
	Total	7747	OA	89.68	90.58	90.76	94.55	94.14	**97.20**
			AA	75.78	74.62	73.84	88.72	81.10	**94.31**

Table 8. Flevoland cross-validation results. Sample distribution in each experiment and individual class, overall, and average accuracies of all methods. Best results are highlighted in bold.

Flevoland Cross-Validation									
Sample Distribution				Accuracy					
#	Class	Samples		1NN	3NN	5NN	SVM	RF	CNN
1	Rapeseed	3525		51.23	54.09	54.83	99.63	100	100
2	Potato	6571		82.96	88.02	88.06	98.47	**99.32**	99.23
3	Barley	3295		68.89	67.09	65.00	95.84	**99.52**	99.44
4	Maize	8004		96.27	96.83	97.24	98.00	**99.80**	99.62
5	Lucerne	468		12.20	6.10	6.56	96.29	86.55	**98.73**
6	Peas	482		60.23	59.45	58.77	86.81	99.99	99.99
7	Fruit	832		4.36	0.92	1.08	88.96	99.45	**99.47**
8	Beans	252		18.30	7.63	7.98	82.15	93.29	**95.02**
9	Wheat	24		50.50	27.83	17.01	6.83	13.33	**77.33**
10	Beet	112		7.77	1.04	0.21	58.11	59.23	**98.43**
11	Grass	1160		73.50	72.45	71.42	89.78	96.18	**99.20**
12	Oats	72		36.00	24.78	20.78	36.89	12.72	**74.39**
	Total	24797	OA	74.86	76.06	75.96	96.53	98.65	**99.36**
			AA	46.85	42.19	40.74	78.15	79.95	**95.06**

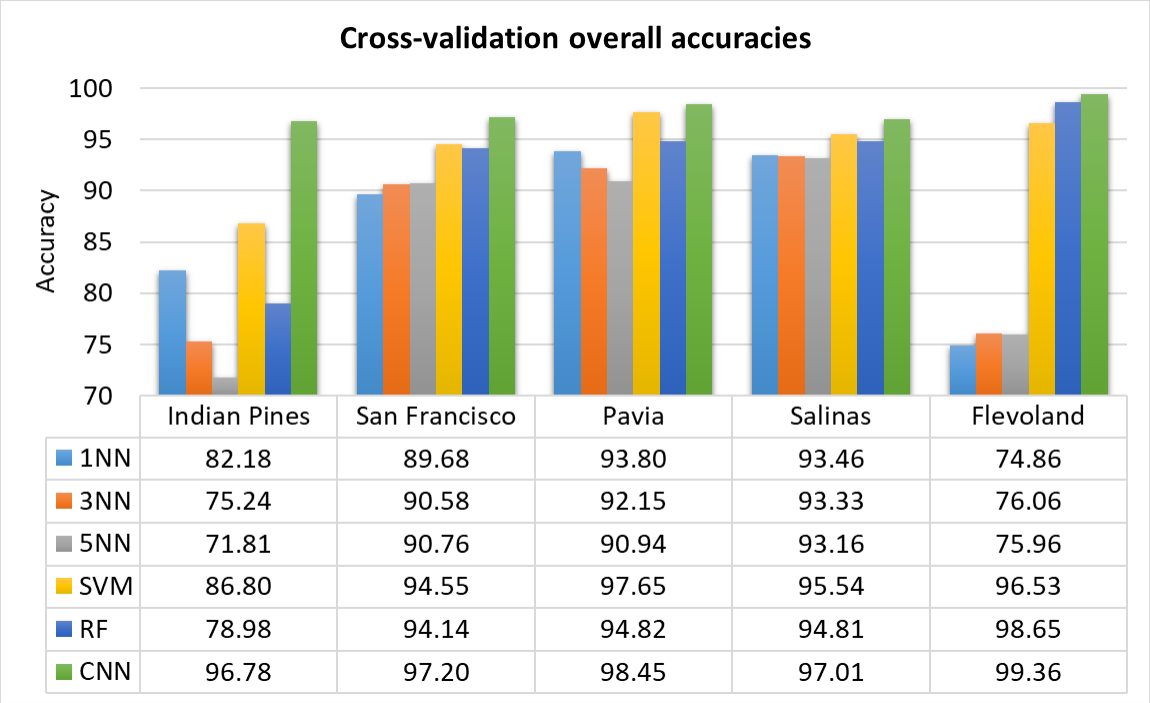


Figure 9. Cross-validation overall accuracy of all methods over each dataset, ordered by increasing size of training set.

In general, the results obtained demonstrate the capacity of CNNs to adapt to datasets with different size and characteristics, considering that we have used the same model for images of different nature (radar and hyperspectral). Furthermore, the fact that we have kept constant the architecture and parameter selection of the network proves the generality of the model, as it is able to achieve a high level of accuracy for all experiments.

3.1.1. Computation Time

The average training and test time of the cross-validation experiments with all datasets are reported in Table 9. Since the experiments using SVM and RF only used CPU for their execution,

both CPU and GPU computation time of the CNN model are presented in order to perform a fair comparison. The kNN models were not considered here due to their lower performance.

Table 9. Average computation time(s) of the cross-validation experiments for all datasets over different classifiers. The fastest methods are highlighted in bold.

	Computation Time (s)							
	Training				**Test**			
Dataset	**SVM**	**RF**	**CNN CPU**	**CNN GPU**	**SVM**	**RF**	**CNN CPU**	**CNN GPU**
Indian Pines	90.2	36.4	84.2	**16.1**	15.1	0.11	0.05	**0.03**
Pavia	150.1	140.5	186.6	**71.3**	33.2	0.14	0.11	**0.06**
Salinas	294.1	153.4	272.3	**82.1**	96.9	0.31	0.24	**0.15**
San Francisco	24.8	**23.9**	95.2	49.6	4.8	0.12	0.06	**0.05**
Flevoland	826.4	234.1	313.6	**160.8**	89.1	0.42	0.21	**0.17**

As it can be observed, the CNN trained over GPU was the fastest model in four of the five datasets considered. The highest difference can be seen in the largest dataset, Flevoland, in which the rest of the models are considerably slower than the CNN. Although training the CNN over CPU is time-consuming, GPU training achieves an average speedup of three times faster. SVM appears to be the method taking more time for training, and RF almost doubles the time of the CNN on average. Furthermore, a clear advantage of the CNN model is that it is very quick for testing, especially when it is compared to SVM. Therefore, it can be reasoned that the designed DL model does not only achieve a higher accuracy, as shown in the previous section, but it is also the fastest method for both training and testing.

From Table 9, we can also extract the computational effort invested in performing the grid search (described in Section 2.3.3) for selecting the optimal set of parameter values of the CNN. The grid search comprised a total of 192 possible combinations, and five repetitions of cross-validation with three datasets (Indian Pines, Pavia, and San Francisco) were executed. Therefore, although obtaining an optimal configuration for the CNN has been time-consuming with around 30 h of computation, the effort for future studies would be minimal as we have been able to design a general network that can work efficiently with images from different sensors.

3.2. Statistical Analysis

After analyzing the cross-validation results, we perform a statistical analysis in order to validate the hypothesis of improved performance of the CNN. The results of the Friedman test, displayed in Table 10, allow for establishing a global ranking of the different methods. As we expected, we obtain different ranks for each method. The CNN leads the ranking, followed by the SVM and the RF. Lastly, the 1NN seems to be the best version of the kNN algorithm, but its performance is far from the rest of techniques. The p-value obtained for the Friedman test with the control method is less than 0.0001; thus, we can reject the null hypothesis as it is below the significance level 0.05.

Once we have verified that there are statistical differences between the methods, we can proceed with the post hoc analysis to perform pairwise comparison. For the Holm's post hoc procedure, we set the best algorithm (CNN) to be the control method and we compare it with the rest of techniques. The p-values and the Holm's adjusted α for each method are displayed in Table 11. From these values, we can conclude that all null hypotheses (no difference between the control algorithm and each of the remaining methods) can be rejected, since the p-values are always smaller than the α. Therefore, it can be stated that there is statistical significance between the results obtained from the CNN and the rest of algorithms. This confirms our initial hypothesis, that, with our proposed DL architecture, we have obtained the best results in our experimentation.

Table 10. Friedman test ranking.

Friedman Test Ranking	
CNN	1.000
SVM	2.240
RF	2.960
1NN	4.600
3NN	4.840
5NN	5.360

Table 11. Holm's post hoc analysis.

Post Hoc Analysis			
Method	p	z	**Holm** α
5NN	0.0000	8.2396	0.0100
3NN	0.0000	7.2569	0.0125
1NN	0.0000	6.8034	0.0167
RF	0.0002	3.7041	0.0250
SVM	0.0191	2.3434	0.0500

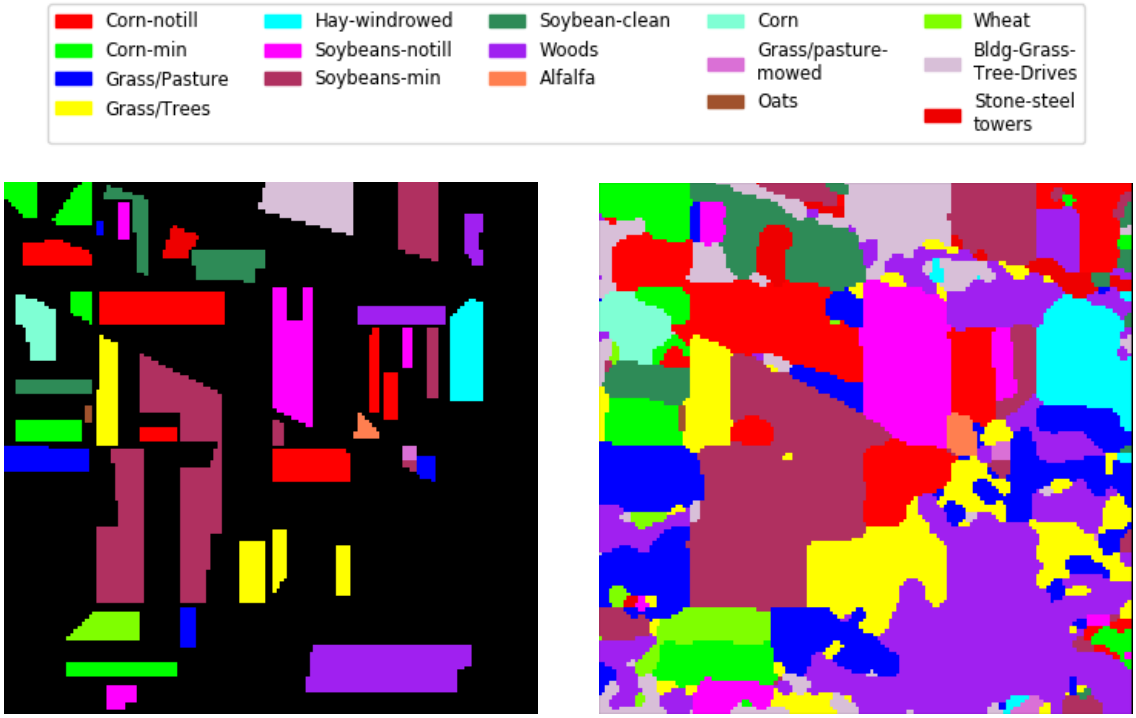
3.3. GRSS DASE Competition

The results obtained at the online competition for all datasets are shown in Table 12. As it can be observed, the CNN provides again the most competitive results, achieving high accuracies for every image: 94.64% in Indian Pines, 98.70% in San Francisco, 83.43% in Pavia, and 98.51% in Flevoland. SVM appears to be the second most robust method, while the rest of techniques (kNN and RF) show a higher variability of results, obtaining accuracies over 90% for some datasets and below 65% for others. The CNN results are also presented when the modal filter post-processing step is applied (CNN-MF), which provides an average slight improvement of almost 1% in accuracy. Figures 10–13 show for each dataset the corresponding training set and the classification map obtained using the CNN. As it can be seen, except in the case of Indian Pines, the training data provided is limited with respect to the complete images that have to be predicted. Nevertheless, the proposed network is able to produce smooth classification results for all cases, considering the large number of classes involved and the different characteristics of the images.

The classification map obtained for the Indian Pines dataset (Figure 10) accurately labels large areas of crops (corn-notill and soybeans) and other kinds of elements such as stone-steel towers. However, it has problems with a few minority classes such as corn. In the case of Pavia (Figure 11), which is the image of higher spatial resolution, the model is able to correctly map fine structures such as the areas of metal sheets, self-blocking bricks, and asphalt. Nevertheless, it can be seen that, due to the small area that each pixel covers (1 m), some distortion is introduced in certain regions where there are adjacent elements of different classes, for instance the gravel areas.

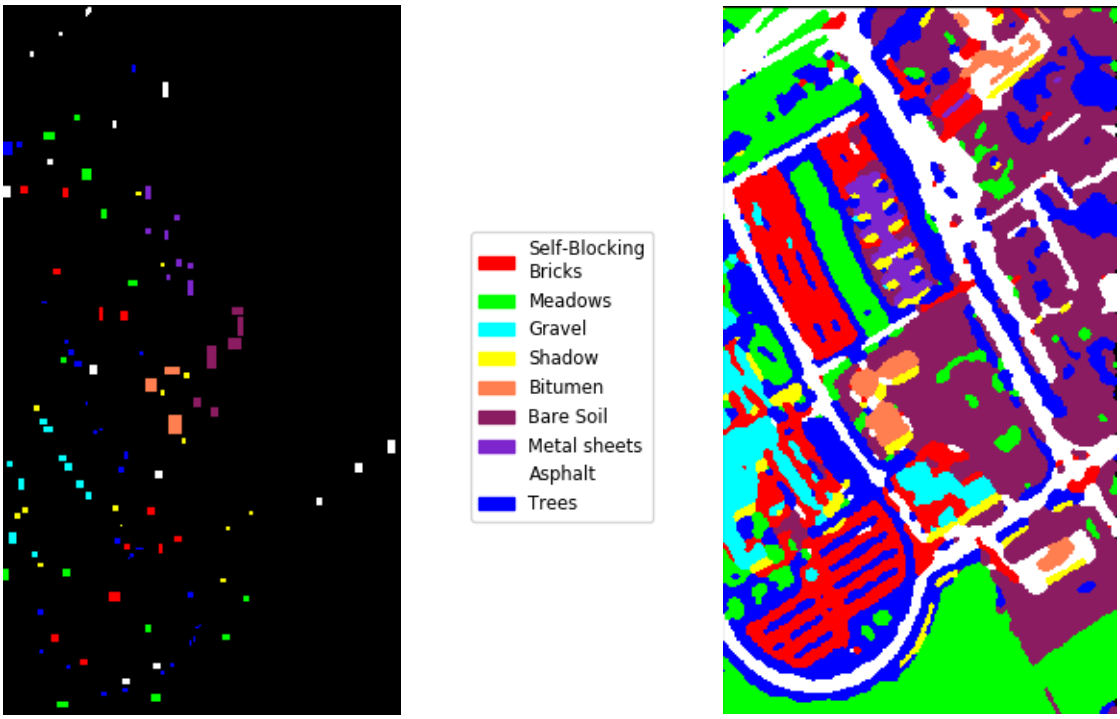
With regard to the radar images, which have a lower spatial resolution, the noise problem is reduced since the pixels cover a larger area of land. In the case of the San Francisco dataset (Figure 13), the model is able to distinguish land from water with a high level of accuracy. It also identifies the zones where vegetation is present compared to urban areas. For the Flevoland dataset (Figure 12), the results show how large fields of different crops are properly mapped, especially maize, barley and potato zones. However, the map presents problems on pixels closer to the edges of the images. This is an inherent problem of the patch-based approach considered, since the neighbouring region of border pixels has to be created artificially, thus adding additional noise.

Generally, the results obtained demonstrate the capacity of the model to adapt to images of different nature and resolutions. However, there exist some areas for improvement, such as the treatment of the edges of the images and the undesired noise introduced by the neighbourhood in patches of high resolution.



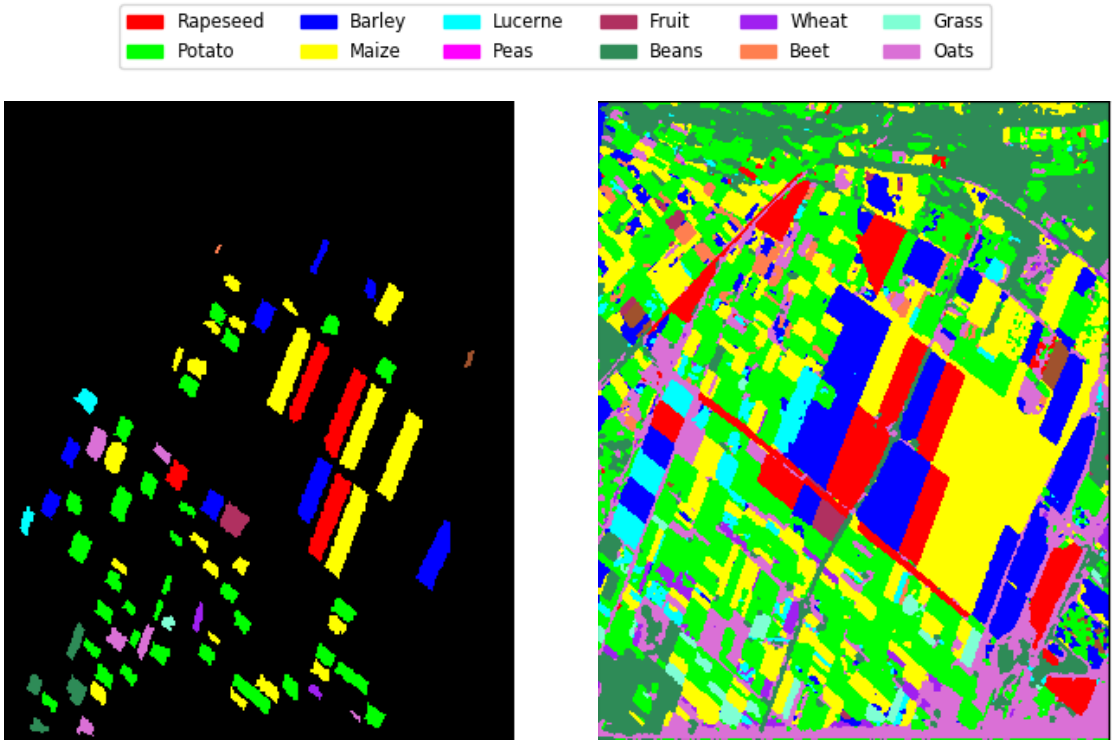
(**a**) (**b**)

Figure 10. Indian Pines GRSS competition. (**a**) training set; (**b**) classification map ($OA = 95.53\%$).



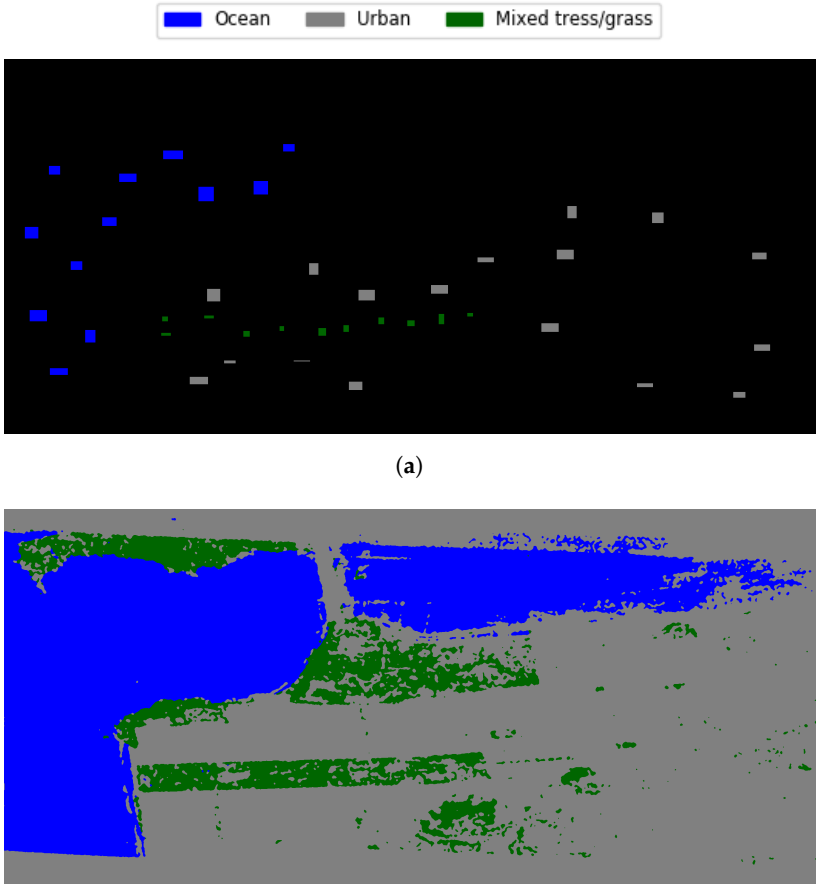
(**a**) (**b**)

Figure 11. Pavia University GRSS competition. (**a**) training set; (**b**) classification map ($OA = 84.79\%$).



(**a**) (**b**)

Figure 12. Flevoland GRSS competition. (**a**) training set; (**b**) classification map ($OA = 99.05\%$).



(**a**)

(**b**)

Figure 13. San Francisco GRSS competition. (**a**) training set; (**b**) classification map ($OA = 99.37\%$).

Table 12. Overall accuracies obtained for each method at the GRSS DASE Website competition.

Accuracies GRSS DASE Website Competition							
	1NN	**3NN**	**5NN**	**SVM**	**RF**	**CNN**	**CNN-MF**
Indian Pines	64.75	64.55	65.14	86.31	64.33	94.64	**95.53**
San Francisco	90.50	91.86	92.40	96.28	96.81	98.70	**99.37**
Pavia	63.85	62.60	62.78	79.75	65.02	83.43	**84.79**
Flevoland	78.55	78.96	77.54	95.51	96.22	98.51	**99.05**

3.3.1. Influence of Patch Size on Classification Accuracy

In order to find the optimal size of the input patch that is fed to our CNN model, we have performed experiments with different values, ranging from 3 to 9, on the datasets available at the online competition. The results obtained are reported in Table 13 and summarised graphically in Figure 14. As it can be seen, the best performance was obtained with a patch size of 5 for all datasets. The decrease in performance when using neighbourhood windows bigger than five is more significant in the two hyperspectral datasets (Indian Pines and Pavia) which also have a higher spatial resolution. In the case of the radar images (Flevoland and San Francisco), which cover larger areas, the performance is similar although slightly worse with higher patch sizes. Therefore, it can be reasoned that using the information of neighbouring pixels helps to improve accuracy. Nevertheless, an oversized region may introduce undesired noise, which seems to be an issue worth considering in LULC patch-based studies.

Table 13. Influence of input patch size of the CNN on classification accuracy with different images.

CNN Accuracy Depending on Patch Size				
	3 × 3	**5 × 5**	**7 × 7**	**9 × 9**
Indian Pines	88.23	**94.64**	90.41	89.53
Pavia	81.48	**83.43**	77.75	77.4
Flevoland	97.79	**98.51**	98.43	97.95
San Francisco	93.99	**98.70**	98.09	97.6

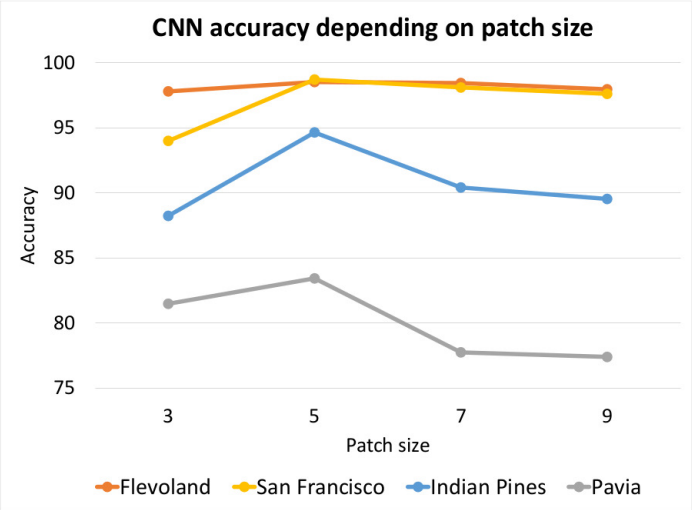


Figure 14. Influence of patch size on classification accuracy over four datasets.

4. Conclusions and Future Work

In this paper, we proposed a general deep learning framework to perform land use and land cover classification over remote sensing imagery from different sources, concretely radar and hyperspectral datasets. Furthermore, we presented a validation procedure to compare the performance between our convolutional neural network and other traditional machine learning techniques such as support

vector machines, random forests, and k-nearest-neighbours. The cross-validation experiments carried out with all methods over five different datasets and the subsequent statistical analysis of the results, demonstrated that the proposed convolutional neural network achieved an improvement in performance with respect to the rest of models considered, in both overall and per-class accuracies. These methods were also evaluated at an online competition of the GRSS, and the convolutional neural network again provided the highest accuracy. Moreover, comparison was also made with regard to the needed computation time, and the deep learning model was reported to be the fastest for both training and testing. In conclusion, our study demonstrated that deep learning is a very powerful solution for the problem of LULC classification, as it showed promising results for all the images studied, given that they are from different sources and have distinct characteristics. It also aimed to show the necessity of establishing a standard validation methodology for evaluating the quality of novel proposals from which the remote sensing community would benefit.

The study also illustrated the difficulty of designing an optimal configuration for a convolutional neural network, and provided some suggestions for the parameter settings as a result of a grid search experimentation. In addition, other aspects regarding the performance of the network were analysed such as the influence of the size of the patches extracted from the images, and the importance of preventing overfitting problems by performing data augmentation and applying regularization techniques such as dropout or batch normalization. Furthermore, one important finding made in this research was that the convolutional neural network performed exceptionally well on minority classes compared to other methods, which is essential due to the common presence of the imbalance problem between classes in remote sensing data.

Future work on this path should include a deeper study on the configuration of the architecture and parameters of the network as it a very time-consuming task. Therefore, being able to automatize this process would definitely help to progress on the field. Moreover, further research might be driven towards evaluating the efficiency of the proposed method when dealing with very large scale images, when memory and processing time start being a severe issue. Furthermore, the application of more complex post-processing techniques could be investigated in order to improve the quality of results. Finally, a more challenging future objective could be to adapt the system to work with fused data from multiple sources and sensors. For instance, more consistent and accurate classification maps could be obtained by integrating lidar information with radar and hyperspectral images.

Author Contributions: All authors made substantial contributions to conception and design of the study. M.C.-G. performed the experiments, analyzed the data, and wrote the paper. J.G.-G. and J.C.R. guided the research and reviewed the manuscript.

Funding: This research was funded by the Spanish Ministry of Economy and Competitiveness under the projects TIN2014-55894-C2-1-R and TIN2017-88209-C2-2-R

Acknowledgments: We are grateful to NVIDIA for their GPU Grant Program that has provided us with the latest technology for performing our experiments.

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

Abbreviations

The following abbreviations are used in this manuscript:

LULC	Land Use and Land Cover
RS	Remote Sensing
DL	Deep Learning
ML	Machine Learning
CNN	Convolutional Neural Network
SVM	Support Vector Machine
RF	Random Forest
kNN	k-Nearest Neighbours

References

1. Hussain, M.; Chen, D.; Cheng, A.; Wei, H.; Stanley, D. Change detection from remotely sensed images: From pixel-based to object-based approaches. *ISPRS J. Photogramm. Remote Sens.* **2013**, *80*, 91–106. [CrossRef]
2. Dandois, J.; Ellis, E. High spatial resolution three-dimensional mapping of vegetation spectral dynamics using computer vision. *Remote Sens. Environ.* **2013**, *136*, 259–276. [CrossRef]
3. He, C.; Liu, Z.; Tian, J.; Ma, Q. Urban expansion dynamics and natural habitat loss in China: A multiscale landscape perspective. *Glob. Chang. Biol.* **2014**, *20*, 2886–2902. [CrossRef] [PubMed]
4. Xie, J.; Chen, H.; Liao, Z.; Gu, X.; Zhu, D.; Zhang, J. An integrated assessment of urban flooding mitigation strategies for robust decision making. *Environ. Model. Softw.* **2017**, *95*, 143–155. [CrossRef]
5. Zolkos, S.; Goetz, S.; Dubayah, R. A meta-analysis of terrestrial aboveground biomass estimation using lidar remote sensing. *Remote Sens. Environ.* **2013**, *128*, 289–298. [CrossRef]
6. Paisitkriangkrai, S.; Sherrah, J.; Janney, P.; van den Hengel, A. Semantic Labeling of Aerial and Satellite Imagery. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2016**, *9*, 2868–2881. [CrossRef]
7. Chen, Y.; Lin, Z.; Zhao, X.; Wang, G.; Gu, Y. Deep learning-based classification of hyperspectral data. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2014**, *7*, 2094–2107. [CrossRef]
8. Jensen, J.R. *Introductory Digital Image Processing: A Remote Sensing Perspective*, 4th ed.; Prentice Hall Press: Upper Saddle River, NJ, USA, 2015.
9. Rogan, J.; Franklin, J.; Roberts, D.A. A comparison of methods for monitoring multitemporal vegetation change using Thematic Mapper imagery. *Remote Sens. Environ.* **2002**, *80*, 143–156. [CrossRef]
10. Du, Q.; Chang, C.I. A linear constrained distance-based discriminant analysis for hyperspectral image classification. *Pattern Recognit.* **2001**, *34*, 361–373. [CrossRef]
11. Kal-Yi, H. A synergistic automatic clustering technique (SYNERACT) for multispectral image Analysis. *Photogramm. Eng. Remote Sens.* **2002**, *68*, 33–40.
12. Etter, A.; McAlpine, C.; Wilson, K.; Phinn, S.; Possingham, H. Regional patterns of agricultural land use and deforestation in Colombia. *Agric. Ecosyst. Environ.* **2006**, *114*, 369–386. [CrossRef]
13. Xu, M.; Watanachaturaporn, P.; Varshney, P.K.; Arora, M.K. Decision tree regression for soft classification of remote sensing data. *Remote Sens. Environ.* **2005**, *97*, 322–336. [CrossRef]
14. Samaniego, L.; Bardossy, A.; Schulz, K. Supervised Classification of Remotely Sensed Imagery Using a Modified k-NN Technique. *IEEE Trans. Geosci. Remote Sens.* **2008**, *46*, 2112–2125. [CrossRef]
15. Gislason, P.; Benediktsson, J.; Sveinsson, J. Random forests for land cover classification. *Pattern Recognit. Lett.* **2006**, *27*, 294–300. [CrossRef]
16. Mas, J.; Flores, J. The application of artificial neural networks to the analysis of remotely sensed data. *Int. J. Remote Sens.* **2008**, *29*, 617–663. [CrossRef]
17. Melgani, F.; Bruzzone, L. Classification of hyperspectral remote sensing images with support vector machines. *IEEE Trans. Geosci. Remote Sens.* **2004**, *42*, 1778–1790. [CrossRef]
18. Khatami, R.; Mountrakis, G.; Stehman, S.V. A meta-analysis of remote sensing research on supervised pixel-based land-cover image classification processes: General guidelines for practitioners and future research. *Remote Sens. Environ.* **2016**, *177*, 89–100. [CrossRef]
19. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:1512.03385.
20. Li, Y.; Zhang, H.; Xue, X.; Jiang, Y.; Shen, Q. Deep learning for remote sensing image classification: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1264. [CrossRef]
21. Kussul, N.; Lavreniuk, M.; Skakun, S.; Shelestov, A. Deep Learning Classification of Land Cover and Crop Types Using Remote Sensing Data. *IEEE Geosci. Remote Sens. Lett.* **2017**, *14*, 778–782. [CrossRef]
22. Ding, J.; Chen, B.; Liu, H.; Huang, M. Convolutional Neural Network With Data Augmentation for SAR Target Recognition. *IEEE Geosci. Remote Sens. Lett.* **2016**, *13*, 364–368. [CrossRef]
23. Romero, A.; Gatta, C.; Camps-Valls, G. Unsupervised deep feature extraction for remote sensing image classification. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 1349–1362. [CrossRef]
24. Castelluccio, M.; Poggi, G.; Sansone, C.; Verdoliva, L. Land Use Classification in Remote Sensing Images by Convolutional Neural Networks. *arXiv* **2015**, arXiv:1508.00092.
25. Li, W.; Fu, H.; Yu, L.; Gong, P.; Feng, D.; Li, C.; Clinton, N. Stacked Autoencoder-based deep learning for remote-sensing image classification: A case study of African land-cover mapping. *Int. J. Remote Sens.* **2016**, *37*, 5632–5646. [CrossRef]

26. Chen, Y.; Zhao, X.; Jia, X. Spectral–Spatial Classification of Hyperspectral Data Based on Deep Belief Network. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 2381–2392. [CrossRef]
27. Li, Y.; Zhang, H.; Shen, Q. Spectral–Spatial Classification of Hyperspectral Imagery with 3D Convolutional Neural Network. *Remote Sens.* **2017**, *9*, 67. [CrossRef]
28. Hou, F.; Lei, W.; Li, H.; Xi, J. FMRSS Net: Fast Matrix Representation-Based Spectral-Spatial Feature Learning Convolutional Neural Network for Hyperspectral Image Classification. *Math. Probl. Eng.* **2018**, *2018*, 1–11. [CrossRef]
29. Kohavi, R. A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. In Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95), Montreal, QC, Canada, 20–25 August 1995; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1995; Volume 2, pp. 1137–1143.
30. García, S.; Herrera, F. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *J. Mach. Learn. Res.* **2008**, *9*, 2677–2694.
31. Mahdianpari, M.; Salehi, B.; Rezaee, M.; Mohammadimanesh, F.; Zhang, Y. Very Deep Convolutional Neural Networks for Complex Land Cover Mapping Using Multispectral Remote Sensing Imagery. *Remote Sens.* **2018**, *10*, 1119. [CrossRef]
32. Sharma, A.; Liu, X.; Yang, X.; Shi, D. A patch-based convolutional neural network for remote sensing image classification. *Neural Netw.* **2017**, *95*, 19–28. [CrossRef]
33. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems, Volume 1 (NIPS'12)*; Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q., Eds.; Curran Associates Inc.: Red Hook, NY, USA, 2012; pp. 1097–1105.
34. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
35. Chen, Y.; Jiang, H.; Li, C.; Jia, X.; Ghamisi, P. Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 6232–6251. [CrossRef]
36. Nair, V.; Hinton, G. Rectified linear units improve Restricted Boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML 2010), Haifa, Israel, 21 June 2010; Fürnkranz, J., Joachims, T., Eds.; pp. 807–814.
37. Boureau, Y.L.; Ponce, J.; Lecun, Y. A Theoretical Analysis of Feature Pooling in Visual Recognition. In Proceedings of the 27th International Conference On Machine Learning (ICML 2010), Haifa, Israel, 21 June 2010; Fürnkranz, J., Joachims, T., Eds.; pp. 111–118.
38. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*; Teh, Y.W., Titterington, M., Eds.; PMLR: Sardinia, Italy, 2010; Volume 9, pp. 249–256.
39. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv* **2015**, arXiv:1502.03167.
40. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv* **2012**, arXiv:1207.0580.
41. Luo, P.; Wang, X.; Shao, W.; Peng, Z. Towards Understanding Regularization in Batch Normalization. *arXiv* **2018**, arXiv:1809.00846.
42. Li, X.; Chen, S.; Hu, X.; Yang, J. Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift. *arXiv* **2018**, arXiv:1801.05134.
43. GRSS DASE Website Competition. Available online: http://dase.grss-ieee.org (accessed on 10 December 2018).
44. 2D Convolutional Neural Networks for Land Use and Land Cover Classification of Radar and Hyperspectral Images. Available online: https://github.com/carranza96/cnn-landcover (accessed on 28 January 2019).
45. Kim, J.H. Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Comput. Stat. Data Anal.* **2009**, *53*, 3735–3745. [CrossRef]
46. Hurni, K.; Schneider, A.; Heinimann, A.; Nong, D.H.; Fox, J. Mapping the Expansion of Boom Crops in Mainland Southeast Asia Using Dense Time Stacks of Landsat Data. *Remote Sens.* **2017**, *9*, 320. [CrossRef]

47. Graves, S.J.; Asner, G.P.; Martin, R.E.; Anderson, C.B.; Colgan, M.S.; Kalantari, L.; Bohlman, S.A. Tree Species Abundance Predictions in a Tropical Agricultural Landscape with a Supervised Classification Model and Imbalanced Data. *Remote Sens.* **2016**, *8*, 161. [CrossRef]
48. Hughes, G. On the mean accuracy of statistical pattern recognizers. *IEEE Trans. Inf. Theory* **1968**, *14*, 55–63. [CrossRef]
49. Geurts, P.; Ernst, D.; Wehenkel, L. Extremely randomized trees. *Mach. Learn.* **2006**, *63*, 3–42. [CrossRef]
50. Demšar, J. Statistical Comparisons of Classifiers over Multiple Data Sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
51. Gao, Q.; Lim, S.; Jia, X. Hyperspectral Image Classification Using Convolutional Neural Networks and Multiple Feature Learning. *Remote Sens.* **2018**, *10*, 299. [CrossRef]