remote sensing

MDPI

Article

4D U-Nets for Multi-Temporal Remote Sensing Data Classification

Michalis Giannopoulos [1,2,*], Grigorios Tsagkatakis [1,2] and Panagiotis Tsakalides [1,2]

[1] Signal Processing Lab (SPL), Institute of Computer Science, Foundation for Research and Technology-Hellas (FORTH), 70013 Crete, Greece; greg@ics.forth.gr (G.T.); tsakalid@ics.forth.gr (P.T.)
[2] Computer Science Department, University of Crete, 70013 Crete, Greece
* Correspondence: mgiannop@ics.forth.gr

Abstract: Multispectral sensors constitute a core earth observation imaging technology generating massive high-dimensional observations acquired across multiple time instances. The collected multi-temporal remote sensed data contain rich information for Earth monitoring applications, from flood detection to crop classification. To easily classify such naturally multidimensional data, conventional low-order deep learning models unavoidably toss away valuable information residing across the available dimensions. In this work, we extend state-of-the-art convolutional network models based on the U-Net architecture to their high-dimensional analogs, which can naturally capture multi-dimensional dependencies and correlations. We introduce several model architectures, both of low as well as of high order, and we quantify the achieved classification performance vis-à-vis the latest state-of-the-art methods. The experimental analysis on observations from Landsat-8 reveals that approaches based on low-order U-Net models exhibit poor classification performance and are outperformed by our proposed high-dimensional U-Net scheme.

Keywords: remote sensing; u-nets; higher-order convolutional neural networks; multi-temporal data classification

check for updates

Citation: Giannopoulos, M.; Tsagkatakis, G.; Tsakalides, P. 4D U-Nets for Multi-Temporal Remote Sensing Data Classification. *Remote Sens.* **2022**, *14*, 634. https://doi.org/10.3390/rs14030634

Academic Editors: Michalis Savelonas and Emmanuel Vassilakis

Received: 26 November 2021
Accepted: 21 January 2022
Published: 28 January 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The field of *Remote Sensing* (RS) enables researchers to detect and monitor the physical characteristics of an area on Earth by measuring from a distance its reflected and emitted radiation. High-resolution satellite or aircraft-borne sensors are usually employed in order to observe a certain range of the electromagnetic spectrum. The collected data serve as a useful information source to various applications, such as object characterization [1], forest fire detection, and crop classification [2–4].

Designing accurate land-cover classification maps has been an item of extensive research within the RS community. The availability of publicly available *Multi-Temporal* (MT) RS data, i.e., data of specific geographical scenes acquired across multiple dates, becomes a major enabling driving force. Since various platforms contribute nowadays to the generation of MT-RS data, we need to further enhance the time-series representation of conventional *Multi-Spectral* (MS) systems in order to reveal and exploit sequential patterns useful for achieving better detection and classification performance.

Once the MT-RS data are collected, automated supervised-learning methods are usually used for their accurate classification [5]. The whole process involves two stages: feature extraction and classification. During the first stage, raw MT-RS data are transformed into "representative" features, which in turn are fed to the second stage for the respective labelling process. In MT-RS applications, features can be extracted in two ways: by deriving spectral vegetation indices directly from the data [6–8] (e.g., *Normalized Difference Vegetation Index* (NDVI) [9]) or by using the raw MT-RS data directly for classification [10,11]. Feature generation can be hand-crafted or performed automatically; in the former case, traditional *Machine Learning* (ML) methods are used, whereas, in the latter case, *Deep Learning* (DL) is

employed. The input features, manually extracted from the raw data by domain experts for the problem at hand (e.g., NDVI), are then fed to various state of the art ML classifiers: *Support Vector Machines* (SVM) [12], *k-Nearest Neighbours* (kNN) [13], *Decision Trees* (DT) [14–16], *Random Forests* (RF) [17–19], and *Artificial Neural Networks* (ANN) [20,21].

On the other hand, DL [22] models assign the feature generation step to the network itself via an end-to-end learning process. Among them, *Convolutional Neural Networks* (CNNs) have drawn significant attention over the last years as they perform extremely well in various tasks, including the classification of RGB [23], multispectral [24], and *Hyper-Spectral* (HS) [25] images and other land-cover [26] and RS applications [27].

Traditional 2D-CNNs are not directly applicable to the MT-RS land-cover classification task, as the additional-to-spatial temporal information must be collapsed. To address this problem, Kussul et al. [28] proposed a technique to tie together a 2D-spatial CNN with a 1D-spectral CNN, whereas, in [29], a 3D-CNN simultaneously employs spatial and temporal feature learning. Methods based on the so-called *Temporal CNNs* (TCNNs) [30–32] have also proved promising as they do not operate on raw MT-RS data but rather on (temporal) NDVI representations. Other approaches such as *Recurrent Neural Networks* (RNNs) [33,34] and *Long Short-Term Memory* (LSTM) networks [35,36] are, by design, suitable for time-series modelling.

As MT-RS land-cover classification is essentially a pixel-level classification problem, special attention should be placed on U-Nets, a class of CNN models introduced by Ronneberger et al. [37], for the problem of semantic segmentation (i.e., pixel-level classification). U-Net architectures originated from the the so-called *Fully Convolutional Networks* (FCNs) [38] and comprise two paths (i.e., a contracting path and an expansive path) in a U-shape manner, hence the name of the models. These paths are more or less symmetric, as the pooling layers of the contracting path are replaced with upsampling ones in the expansive path. Consequently, the large number of feature channels derived in the expansive path allows the network to propagate context information to higher resolution layers, leading to more precise segmentation results with fewer training samples.

A recent approach for the efficient combination of mono-temporal and MT data was proposed by Sefrin et al. in [39]. Therein, the authors adopt two different DL architectures: an FCN 2D U-Net baseline, which can only handle mono-temporal RS data, and an FCN 2D U-Net combined with a LSTM network in order to further exploit sequential (MT) information. More specifically, the first method proposed in [39] comprises an FCN which employs the VGG-19 model [40] (originally used for image classification tasks) as its expansive path. It employs several convolution, max-pooling, up-sampling, and normalisation blocks, just as the 2D U-net model presented in Section 3.5. It is a 5-depth architecture, and it accepts two-dimensional input data across all available spectral bands. The second method proposed in [39] is a combination of the previously FCN model with an LSTM one. In order to benefit from the sequential information of the RS data, the FCN is used as a pre-trained base model from which every temporal RS sample is passed through separately. The outputs of FCN (without the final SoftMax classification layer) are then stacked in chronological and reverse-chronological order, forming a *bi-directional* architecture. Those two output sequences are then fed to a 2D convolutional-LSTM (ConvLSTM) layer, which in turn produces an output for both sequences (of forward and backward chronological order). At the final step of the architecture, those two outputs are passed through a final SoftMax classification layer and are subsequently averaged to derive the final output. Their proposed approach is tested on Sentinel-2 RS data and demonstrates the significance of employing temporal information in land-cover classification and change detection tasks.

Retooling existing lower-order DL architectures for handling 4D MT-RS data is clearly not optimal, and it inevitably leads to loss of valuable information, as depicted in Figure 1. A MT-RS scene measured at different time instances (e.g., years) holds important evidence about the evolution of its content across time. The values of a specific spectral band can have notable differences among successive time acquisitions, which can prove useful in classifying the whole MT-RS scene. Moreover, spectral images for a single time instance carry

important information about the land-cover. Thus, the need for higher-order classification models and systems which can exploit the whole available information simultaneously becomes apparent.

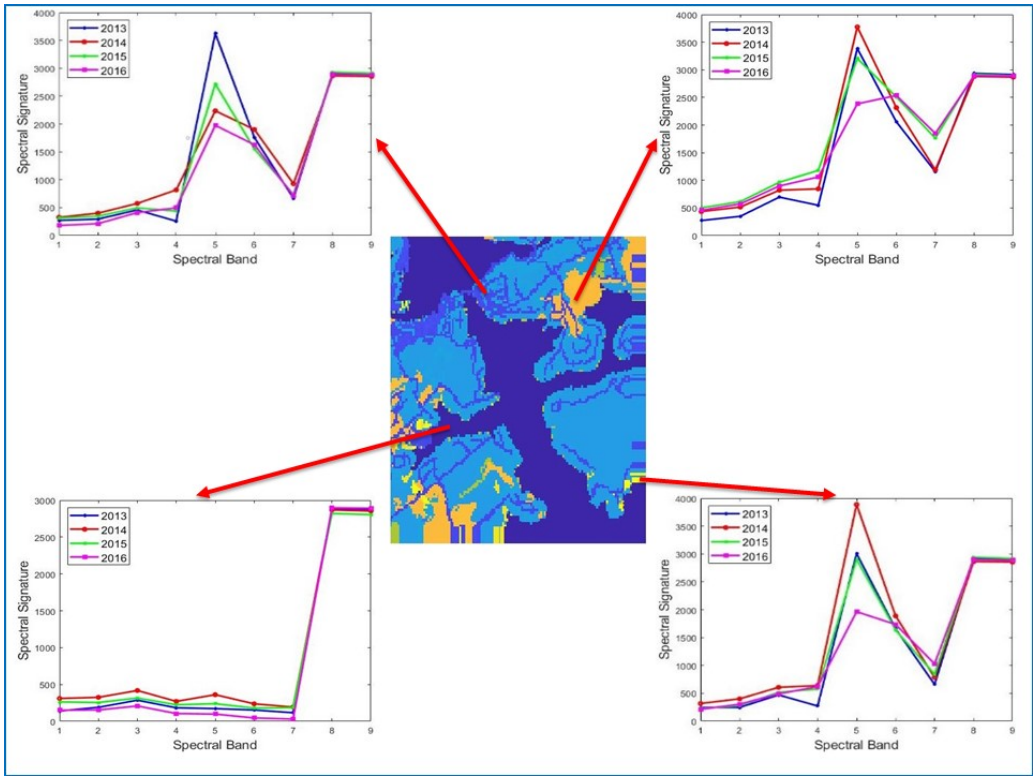


Figure 1. The motivation for higher-order classification models. MT-RS imagery obtained at different time instances comprises rich information, as it reflects the evolution of pixel values across time. At the same time, spectral signatures across different spectral bands for each time instance reveal valuable physical information that can be preserved and exploited through a specially designed 4D model architecture.

Departing from the aforementioned approaches, in this paper we propose a 4D U-Net solution to the MT-RS land-cover classification task. Instead of modelling only part of the available information, we extend the notion of 3D U-Nets to their higher-order counterparts, and we jointly learn spatio-spectro-temporal features. We employ a recently released MT-RS dataset to train and evaluate the performance of the proposed 4D U-Net model vis-à-vis its 3D and 2D counterparts, as well as state-of-the-art methods. Extensive experiments are performed in order to assess the performance of each model with respect to different architectures and several model parameters. Experimental results on real MT-RS data demonstrate the potential and superiority of the proposed higher-order feature modelling in all examined scenarios.

The key contributions of this paper can be summarized as follows:

- The design and implementation of a new 4D U-Net to address the MT-RS land-cover classification problem and its evaluation on a recently released dataset;
- The provision of a purely 4D U-Net model, extending all required 3D functionality-layers (convolution, pooling, transpose-convolution) to their higher-order counterparts;
- An effective exploitation of high-order data correlations at their nominal dimensions, without any loss of information;
- The demonstration of the 4D U-Nets' potential over their lower-order counterparts.

To the best of our knowledge, 4D U-Nets have never been employed in MT-RS imagery to derive high-resolution dense predictions, nor have they been compared to 3D U-Nets and

2D U-Nets for the problem at hand. They have been employed in the context of semantic segmentation of cardiac volumetric sequences [41], but with an architecture that does not extend all U-Net functionalities such as upsampling via kNN interpolation instead of fully learnable 4D transpose-convolution, as is the case in our proposed study.

The rest of the paper is organised as follows: In Section 2, we introduce a 4D U-Net architecture for tackling the MT-RS land-cover classification problem, alongside a detailed mathematical and conceptual description of every module and functionality in it. In Section 3, the employed dataset as well as the experimental settings are described, accompanied by the tests to evaluate the proposed 4D U-Net performance as well as the experimental conclusions that can be drawn. Moreover, Section 4 provides an interpretation of the derived results, accompanied by the strengths and limitations of the proposed method with respect to its competitors. Finally, Section 5 draws the main conclusions of the present work.

2. Materials and Methods

In this section, we initially describe the scenario under investigation for the classification task of MT-RS land-cover. Subsequently, we provide the rationale for choosing higher-order U-Nets, and we formulate the problem by applying 4D filters on the series of MT-MS datacubes. Finally, we present the complete mathematical components on how U-Nets can be used in the estimation task, we develop the proposed methodology for MT-RS land-cover classification, and we build the associated network architecture.

2.1. 4D U-Net for Multi-Temporal Remote Sensing Land-Cover Prediction Modeling

As mentioned in the introduction, CNNs constitute a particular type of DL models with excellent performance on imaging tasks. Traditional 2D CNNs convolve raw input data with learnable filters in a layer-wise manner such that a CNN can end up having several layers each one of which is assigned to detect different features of an image. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. The filters can start with very simple features, such as brightness and edges, and increase in complexity to features that uniquely define the object.

When tackling the MT-RS land-cover classification task, which is intrinsically a 4D input problem (involving 2D spatial, spectral, and temporal dimensions), 2D CNNs inevitably toss away valuable information. Even 3D models need to choose between preserving spectral or temporal information. To address this issue, we hereby introduce and employ a 4D CNN model for jointly exploiting correlations among all different dimensions. We focus our attention on U-Net models, extending 2D and 3D architectures [37,42] for the problem at hand.

As illustrated in Figure 2, in order to efficiently train our 4D U-Net model, we feed it with raw MT-RS imagery (i.e., sequences of 3D MS data), without any data preprocessing/collapsing step. Those sample images are passed through a series of high-dimensional operations (e.g., convolution, pooling, upsampling) encapsulated in the U-Net model, which become more complex as the depth of the network increases (i.e., the number of filters at the bottom of each module increases accordingly). At the same time, spatial information (i.e., the spatial size at the top of each module) is sequentially downsampled and upsampled in order to derive more representative features, while the spectral and temporal information are preserved up to the final classification module of the network. Unlike lower-order models, which have to collapse data information before training the network, our proposed architecture exploits all high-dimensional information by performing all operations in the nominal data dimensions. Experimental results on real MT-RS images demonstrate that directly learning features employing all available data dimensions, rather than selecting, averaging, or even collapsing some of them, achieves a significantly higher accuracy level, indicating the efficacy of the proposed approach.

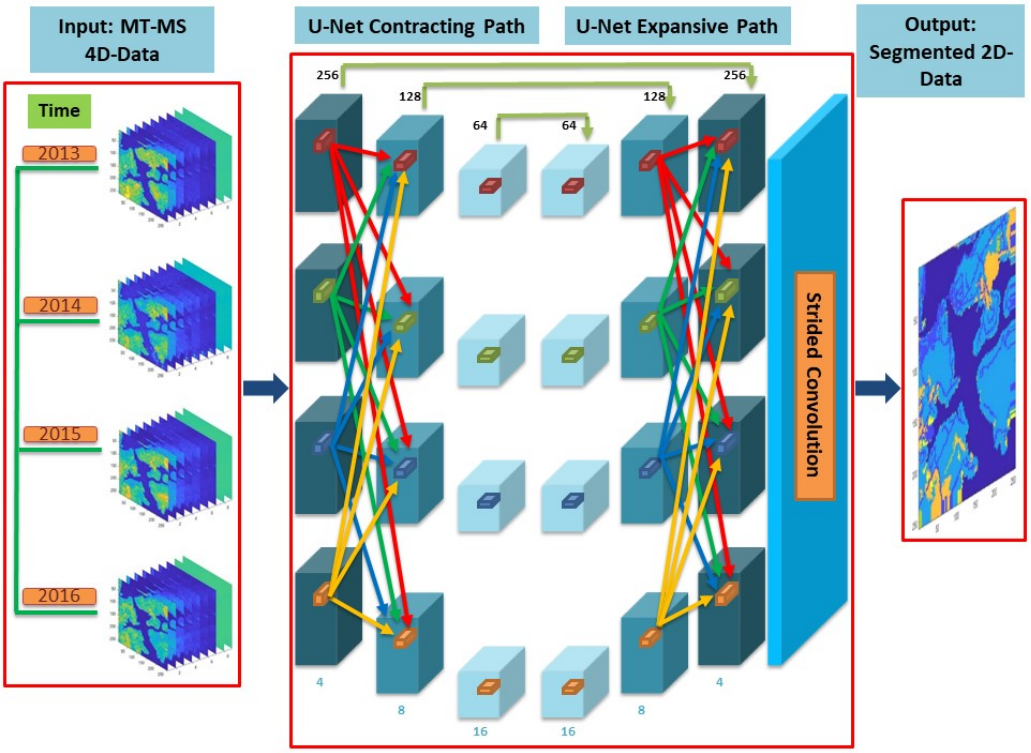


Figure 2. The proposed method pipeline for the classification task of MT-RS imagery. Raw 4D data (i.e., time series of 3D data) are fed to the designed higher-order U-Net model, which preserves all available information until the classification step, predicting each pixel value of the scene. All key-operations of the model (e.g., convolutions, transpose-convolutions) are performed in the 4D space.

2.2. 4D U-Net Modules for Multi-Temporal Multi-Spectral Feature Learning

The proposed method fits within the framework of classification-flavoured supervised learning, employing higher-order CNNs for efficient feature learning of the MT-RS image content. In order to effectively extract features from raw data, CNNs employ the notion of local receptive fields, in which each locally connected input subset of the input neuron is mapped to a single output neuron. With the intention of capturing as many representative features as possible, this process is performed in a stacked manner throughout convolutional layers. Input and output neurons are connected via convolutions by means of trainable filters, with specific filter coefficients.

In traditional 2D CNNs, the value of a convolved output neuron at position (k, l) can be expressed as follows:

$$y_{k,l} = f\left(\sum_{c}^{C_{in}} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} w_{i,j} x_{c,(k+i)(l+j)} + b_{i,j}\right), \quad (1)$$

where $f(.)$ is the activation function, $w_{i,j}$ stands for the value of the kernel connected to the current feature map at position (i, j), $x_{c,(k+i)(l+j)}$ represents the value of the input neuron at input channel c, $b_{i,j}$ is the bias of the computed feature map, C_{in} denotes the number of original channels (i.e., first layer) or the number of feature maps of the previous layer (i.e., intermediate layer), and H and W are the height and width of the kernel, respectively.

Of course, in MT-RS land-cover classification, raw input data are not 2D, so in order for (1) to be applicable, the spectral and time dimensions have to be collapsed. To partially alleviate this, Zhang et al. proposed in [29] a 3D-CNN to preserve temporal information

across all different spectral bands. In their configuration, (1) is extended to three dimensions as follows:

$$y_{k,l,m} = f\left(\sum_{c}^{C_{in}}\sum_{t=0}^{T-1}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1} w_{i,j,t}x_{c,(k+i)(l+j)(m+t)} + b_{i,j,t}\right), \tag{2}$$

where the extra parameter, T, over (1) denotes the temporal length of the kernel.

As it can be seen from (2), a 3D convolution cannot exploit the available information hidden in the spectral bands. The fact that rich information distributed across the spectral dimension goes untapped has a considerable negative impact in the classification task. To preserve all available information, we herein introduce a 4D convolutional layer/module/functionality capable of jointly exploiting spatial, spectral, and temporal information. More specifically, we extend (2) to its four-dimensional counterpart as follows:

$$y_{k,l,m,n} = f\left(\sum_{c}^{C_{in}}\sum_{s=0}^{S-1}\sum_{t=0}^{T-1}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1} w_{i,j,t,s}x_{c,(k+i)(l+j)(m+t)(n+s)} + b_{i,j,t,s}\right), \tag{3}$$

where S denotes the spectral bands of the kernel.

Unfortunately, 4D convolutional layers are not available in modern DL frameworks such as TensorFlow (https://www.tensorflow.org/) (accessed on 13 September 2021) [43], so we have implemented our own 4D convolutional layer as a custom TensorFlow layer plugin. To do so, and to exploit TensorFlow's fast libraries of implementations, we rearrange (3) as follows:

$$\begin{aligned} y_{k,l,m,n} &= f\left(\sum_{c}^{C_{in}}\sum_{s=0}^{S-1}\sum_{t=0}^{T-1}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1} w_{i,j,t,s}x_{c,(k+i)(l+j)(m+t)(n+s)} + b_{i,j,t,s}\right) \\ &= f\left(\sum_{s=0}^{S-1}\left[\sum_{c}^{C_{in}}\sum_{t=0}^{T-1}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1} w_{i,j,t,s}x_{c,(k+i)(l+j)(m+t)(n+s)}\right] + b_{i,j,t,s}\right) \end{aligned} \tag{4}$$

The rearrangement of the convolution sums shown in (4) is feasible since convolution is a linear operation and therefore the order of summation in (3) can be changed. A similar rearrangement was proposed in [41,44], and it results in summing/stacking multiple sequences of 3D convolutions (the term in brackets in (4)) along the fourth dimension. From an implementation point of view, further rearrangement of the respective for-loop takes place as in [41] (i.e., 3D input frames convolved with respective 3D filter frames) in order for our custom TensorFlow layer to implement true (instead of separable) 4D convolution.

Stacking convolutional layers in DL models allows layers close to the input to learn low-level features (e.g., lines), whereas those deeper in the network's architecture to learn high-order and more abstract features (e.g., shapes, specific objects). However, a limitation of the feature map output of convolutional layers is that they record the precise position of features in the input. This means that small movements in the position of the feature in the input data (e.g., rotation, shifting) will result in a different feature map. A common approach to address this issue, from a signal processing perspective, is to perform downsampling. The most robust and common approach for doing so is to use pooling layers, whose operation, in contrast to convolution, is specified rather than learned. Since max-pooling has been found to work better in practise for computer vision tasks than average-pooling [45], and it is commonly used in most lower-dimensional U-Net models, we accordingly employ max-pooling as the downsampling operation in the contracting path of our network. As it was the case for the 4D convolutional layers, 4D max-pooling as well is not available in TensorFlow. Consequently, we implemented it as the higher-order analogue of vanilla max-pooling layers by adopting a frame-wise logic, similar to that in (4), in order to again take advantage of fast libraries provided by TensorFlow.

More specifically, the max-pooling operation can be defined as a function $f_{\mathbf{x}}$ computing the maximum value around a neighbourhood/patch in $\mathbb{R}^{n\times n}$ for the 2D case and in $\mathbb{R}^{n\times n\times n}$ for the 3D case. Similarly, for the 4D case, we can extend operator $f_{\mathbf{x}}$ as follows:

$$\begin{aligned} \mathbf{x} &\longmapsto f_{\mathbf{x}} : \mathbb{R}^{n\times n\times n\times n} \longrightarrow \mathbb{R} \\ f_{\mathbf{x}} &\longmapsto \max(x^{\ell}) = \max(\max(x^{\ell-1})) \end{aligned} \tag{5}$$

where x^{ℓ} and $x^{\ell-1}$ are the feature maps in the ℓ-D and $(\ell-1)$-D spaces ($\ell = 4$), respectively.

Moving deeper in the contracting path of a U-Net, the model understands better "what" is present in the data but loses information about "where" it is present. In our case, the end goal is not the provision of a class label for the whole MT-RS scene, but rather the classification of every pixel in it. Consequently, there is a need for an efficient upsampling mechanism that will transform the contracting path low-resolution output to a higher-resolution in order to efficiently recover the "where" information as well.

Conceptually, we need a module in order to transition from a feature map of dimension derived from a convolution to a feature map of dimension originating from a convolution [46]. To achieve this goal, existing techniques, such as kNNs and bi-linear or cubic interpolation, do not involve any learning from the network but rather employ pre-defined manual feature engineering. Departing from these approaches, in order for our proposed 4D U-Net model to learn to upsample optimally in an *end-to-end* trainable way, we employ and extend the notion of *transposed convolution*.

As in the case of the 4D convolution and max-pooling layers, we accordingly implement the 4D transpose-convolution layer in a frame-wise logic for fast computation via the efficient implementations of the lower-order functionalities provided by Tensorflow. In traditional 2D transpose-convolution, the value of an output neuron at position (k, l) can be expressed as follows:

$$y_{k,l} = f\left(\sum_{c}^{C_{in}}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1} w_{i,j}x_{c,(i:i+k)(j:j+l)} + b_{i,j}\right), \tag{6}$$

where $f(.)$ is the activation function, $w_{i,j}$ stands for the value of the kernel connected to the current feature map at position (i, j), $x_{c,(i:i+k)(j:j+l)}$ represents the value of the input neuron at input channel c, $b_{i,j}$ is the bias of the computed feature map, C_{in} denotes the number of original channels (i.e., first layer) or the number of feature maps of the previous layer (i.e., intermediate layer), and H and W are the height and width of the kernel, respectively.

Extending (6) to the three-dimensional case, we obtain:

$$y_{k,l,m} = f\left(\sum_{c}^{C_{in}}\sum_{t=0}^{T-1}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1} w_{i,j,t}x_{c,(i:i+k)(j:j+l)(t:t+m)} + b_{i,j,t}\right) \tag{7}$$

Finally, so as to derive the equation for the four-dimensional transpose-convolution task, we further extend (7) based on the same reasoning we used in (4):

$$\begin{aligned} y_{k,l,m,n} &= f\left(\sum_{c}^{C_{in}}\sum_{s=0}^{S-1}\sum_{t=0}^{T-1}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1} w_{i,j,t,s}x_{c,(i:i+k)(j:j+l)(t:t+m)(s:s+n)} + b_{i,j,t,s}\right) \\ &= f\left(\sum_{s=0}^{S-1}\left[\sum_{c}^{C_{in}}\sum_{t=0}^{T-1}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1} w_{i,j,t,s}x_{c,(i:i+k)(j:j+l)(t:t+m)(s:s+n)}\right] + b_{i,j,t,s}\right) \end{aligned} \tag{8}$$

In conclusion, we have extended all the key-modules of conventional lower-order U-Net models to their high-dimensional counterparts. As a result, our 4D U-Net architecture can handle MT-RS data in their nominal dimensions, efficiently addressing the dense pixel-level classification task at hand.

2.3. U-Net Architectures for Multi-Temporal Remote Sensing Land-Cover Classification

The end goal of the present study is to compare the performance of U-Net models of various orders and obtain intuition about the potential merits of the proposed 4D

architecture over the respective 2D and 3D systems. We employ the vanilla 2D U-Net [37] as a benchmark-comparison model, and we construct the respective 3D and 4D analogs. There are two reasons for concentrating on "purely" U-Net architectures. First, U-Net models have been proven to be the best performers in pixel-level classification and segmentation problems, hence they are used as reference techniques in the land-cover classification field [47–51]. Second, the dimensionality of the employed dataset (i.e., raw MT-RS 4D data) makes comparisons with other, non-U-Net, models intractable for various data tiles.

Each architecture contains convolutional, max-pooling, transpose-convolutional, skip-connection, activation, and dropout layers, arranged in "stacks-of-layers" along the contracting and expansive paths of the model, as illustrated in Figure 2. More specifically, there are 3 stacks-of-layers along each path, and 1 stack at the bottom/middle of the model. Each contracting path's stack comprises 2 convolutional layers, immediately followed by ReLU activation layers. At the end of the contracting path stacks, there exists a max-pooling layer (for downsampling) alongside a dropout layer (for regularization). On the other hand, each expansive path's stack contains a transpose-convolutional layer (for upsampling), followed by skip-connection and concatenation layers (for copy-and-crop purposes), and a dropout layer. At the end of the expansive stack, there exist 2 convolutional layers immediately followed by ReLU activation layers. The middle/bottom stack also contains 2 convolutional layers immediately followed by ReLU activation layers. Finally, the prediction layer contains a convolutional layer with a number of filters equal to the number of available classes, followed by a SoftMax activation layer (which contains the probabilities of each pixel belonging to each available class).

Since our samples are 4D dimensions, i.e., two spatial, one spectral, and one temporal, (H, W, S, T), low-dimensional U-Net models have to perform a type of collapsing on every sample, in order to be used for training. For that reason, we perform averaging across the dimensions not to be used as "active information", in order to avoid a biased selection of a specific spectral band/time instance. Consequently, the 2D U-Net model operates spatially in averaged samples across all spectral bands and time instances. There exist two different alternatives for the 3D U-Net models: the 3D-Temporal U-Net (operating on spatial and temporal dimensions, averaged across all spectral bands), and the 3D-Spectral U-Net (operating on spatial and spectral dimensions, averaged across all time instances). On the other hand, our proposed 4D U-Net model operates directly on the original dimensions of the data, exploiting all available information without the need for collapsing across any specific dimension.

Since U-Nets are mostly used for segmentation purposes, the samples and labels are usually of the same size (e.g., 2D images in image segmentation, 3D volumetric-images in volumetric-image segmentation). However, in our high-dimensional U-Net models, we have to deal with the problem of different dimensionalities between samples (3D/4D) and labels (2D). To overcome this obstacle and to preserve as much information as possible, we perform all functionalities of the U-Net model at the respective high-dimensional spaces until we reach the final classification layer. Therein, we employ filters of kernel-size 1 across every dimension, with unit strides across every dimension except those to be exploited in addition to the spatial ones (i.e., $(1, 1, T)$, $(1, 1, S)$, $(1, 1, S, T)$ for the 3D-Temporal, 3D-Spectral, and 4D U-Net models, respectively). In that way, we keep the additional-to-spatial information intact until right before classification is performed, downsampling it appropriately at the utmost step in order to train the network with 2D ground-truth labels.

Concerning the U-Net topology, all convolutional layers are used with a small kernel size equal to 3 across every dimension and with "same"-padding for all U-Net models. We accordingly employ unit strides on all convolutions across all dimensions, whereas the number of filters is doubled as we proceed deeper in the contracting path of the U-Net as follows: The first stack-of-layers number of filters is equal to a parameter (called "starting-neurons"), which is doubled at every following stack (e.g., the second stack has 2×"starting-neurons" filters, the third stack has 4×"starting-neurons" filters, etc.). Since we intend to downsample input data samples only spatially in order to preserve as much

high-dimensional information as possible, max-pooling layers have pool-size and strides of 2 across the spatial dimensions and 1 across the spectral and temporal dimensions (for the 3D and 4D U-Net models). All max-pooling layers employ "same"-padding, whereas dropout layers have rates (i.e., fraction of input units to drop) equal to 0.5. As far as the transpose-convolutional layers are concerned, we employ small kernels of size 2 across all dimensions, with “same”-padding for all designed U-Net models. Since we perform only spatial-downsampling in the contracting path, the strides of the transpose-convolutions are accordingly set to 2 for the spatial dimensions (for the respective upsampling) and to 1 for the rest of the dimensions (for the 3D and 4D U-Net models). The number of filters in each expansive stack is reverse of the number of the respective contracting stack (e.g., the first stack has 8×”starting-neurons” filters, the second stack has 4×”starting-neurons” filters, etc.).

The aforementioned method for classifying MT-RS images is summarised in Algorithm 1. Therein, we clarify the function of each path’s block of the proposed 4D U-Net model, as well as the specific contributions of the designed higher-order layers in the whole classification pipeline.

Algorithm 1: 4D U-Net model inference.

Input: MT-RS Images: $F_I \in \mathbf{R}^{H \times W \times S \times T}$

Output: MT-RS Predictions $F_O \in \mathbf{R}^{H \times W}$

Parameter Initialization: "starting-neurons"

Contracting-path block

- Input feature maps, F_{IC}, subsampled only spatially
- Output feature maps: $F_{OC} \in \mathbf{R}^{\frac{H}{2} \times \frac{W}{2} \times S \times T}$
- $F_{OC} = g_{4D}(h_{4D}(F_{IC}))$, where g_{4D} and h_{4D} are the introduced 4D max-pooling and convolution operators, respectively

Expansive-path block

- Input feature maps, F_{IE}, upsampled only spatially
- Output feature maps: $F_{OE} \in \mathbf{R}^{2H' \times 2W' \times S \times T}$ $F_{OE} = f_{4D}(F_{IE})$, where f_{4D} is the introduced 4D transpose-convolution operator, $H' = \frac{H}{2}, W' = \frac{W}{2}$
- $F_{OE} = \Phi(F_{OC}, F_{OE})$: Concatenation with respective contracting-path’s output feature map

Prediction block

- Input feature maps, $F_{IP} \in \mathbf{R}^{H \times W \times S \times T}$
- Impose strided 4D-convolution, $strides = (1, 1, S, T)$
- Output feature maps, $F_{OP} \in \mathbf{R}^{H \times W}$

Taking into account that we perform dense pixel-level classification, each pixel can belong to one out of C mutually exclusive classes. In order to recast the network’s outputs as probabilities, categorical cross-entropy was used as the loss function, since the cross-entropy can be interpreted as the log-likelihood function of the training samples. By using this loss function, we train our models to output a probability over the C available classes for each pixel by combining the SoftMax activation ($f(s)_i$) and the cross-entropy loss (CE), as seen in (9):

$$\begin{cases} f(s)_i = \frac{e^{s_i}}{\sum_{j=1}^{C} e^{s_j}} \\ CE = -\sum_{i}^{C} t_i log(f(s)_i) \end{cases} \tag{9}$$

Given the fact that the ground-truth labels in the problem at hand will be matrices (instead of vectors as in image classification tasks) of spatial size (H, W), we convert them to one-hot encoding format by generating a third-order label-tensor of dimensions (H, W, C), in order for the multi-class classification task to take place. In this format, every slice of

the label-tensor contains the probabilities of each pixel belonging to each different class. Only the positive class C_p keeps its term in the loss described in (9), and consequently, there is only one element of the target-slice t which is not zero, with $t_i = t_p$. In that way, discarding the elements of the summation which are zero due to target labels, we can derive the categorical cross-entropy loss for our U-Net models, as described in (10):

$$CCE = -log\left(\frac{e^{s_p}}{\sum_{j=1}^{C} e^{s_j}}\right), \tag{10}$$

where s_p is the models' score for the positive class.

During the training phase of the network, the weights of every model are randomly initialised with the Glorot–Uniform initialiser [52]. Regarding the optimisation learning algorithm, the Adam scheme [53] was used with a constant learning rate of 0.0001 and exponential decay rates for the first and second moment estimates equal to $\beta_1 = 0.9$ and $\beta_2 = 0.999$, respectively.

Training was performed using a constant number of up to 100 epochs with the intention of observing how the prediction accuracy changes by increasing the number of epochs the network is trained for. Since the size of the data samples is quite memory intensive, we dynamically load the data for training each different model (2D/3D/4D) using a batch-size of 1 sample, aiming to measure fairly among the different models metrics, such as classification accuracy and computational time needed for the training process.

3. Results

In this section, we initially describe the dataset on which the upcoming experiments are performed, as well as the employed experimental setup. Afterwards, we define the evaluation metrics for the comparison of the various DL models. Subsequently, we provide detailed results of the adopted approach under different experimental scenarios, using several configurations in order to quantify the effects of all pertinent parameters in the whole MT-RS land-cover classification pipeline. Finally, we compare the proposed approach with state-of-the-art methods, and we demonstrate its improved performance over its competitors.

3.1. Dataset Description

The availability of a suitable training and test set is a critical prerequisite when evaluating DL solutions and training ML models. Fortunately, quite recently, a large MT-RS dataset was released in the context of the IEEE GRSS Data Fusion Contest ("[REF. NO.] 2021 IEEE GRSS Data Fusion Contest. Online: https://www.grss-ieee.org/community/technical-committees/2021-ieee-grss-data-fusion-contest-track-msd/, accessed on 20 December 2020) for the problem of MT-RS land-cover classification. The dataset provides 2250 different tiles (each one covering approximately a 4 km × 4 km area) over the state of Maryland, USA. The tiles are located in East Coast USA, and an exemplar map of the area under study alongside with its neighbouring states is illustrated in Figure 3. Each tile contains Landsat-8 training data (i.e., 9-band MS satellite data at a 30 m spatial resolution collected once a year through 2013–2017) as well as USGS National Land Cover Database (NLCD) ground-truth labels for years 2013 and 2016 (i.e., 15-class land cover data at a 30 m spatial resolution from the April 2019 data release). Those ground-truth labels (alongside their real values) are reported in Table 1, corresponding to 15 classes including various forest and developed categories.

In the present study, ground-truth labels are considered those of the year 2016 (the most recent ones), while training data are the respective MT-RS images through years 2013 to 2016. From the aforementioned data collection, we selected only those image tiles whose respective labels do not contain any pixels with undefined values (i.e., pixels with values equal to 0), since these pixels do not form any meaningful class out of the 15 available ones. As a result, we keep 1580 different samples (e.g., image tiles) and discard the other 670.

Bearing in mind that directly processing MT-RS images of such a large spatial size is too complex and demanding in memory during training, we adopted the following pre-processing steps for each MT-RS image: spatial padding of each MT-RS image tile until its spatial size reaches 4096×4096, followed by spatial subsampling every 16 pixels across height and width (thus reducing the spatial size from $(4096, 4096)$ to $(256, 256)$). In this way, we finally end up with a dataset comprising 1580 different samples of size $(256, 256, 9, 4)$ each.

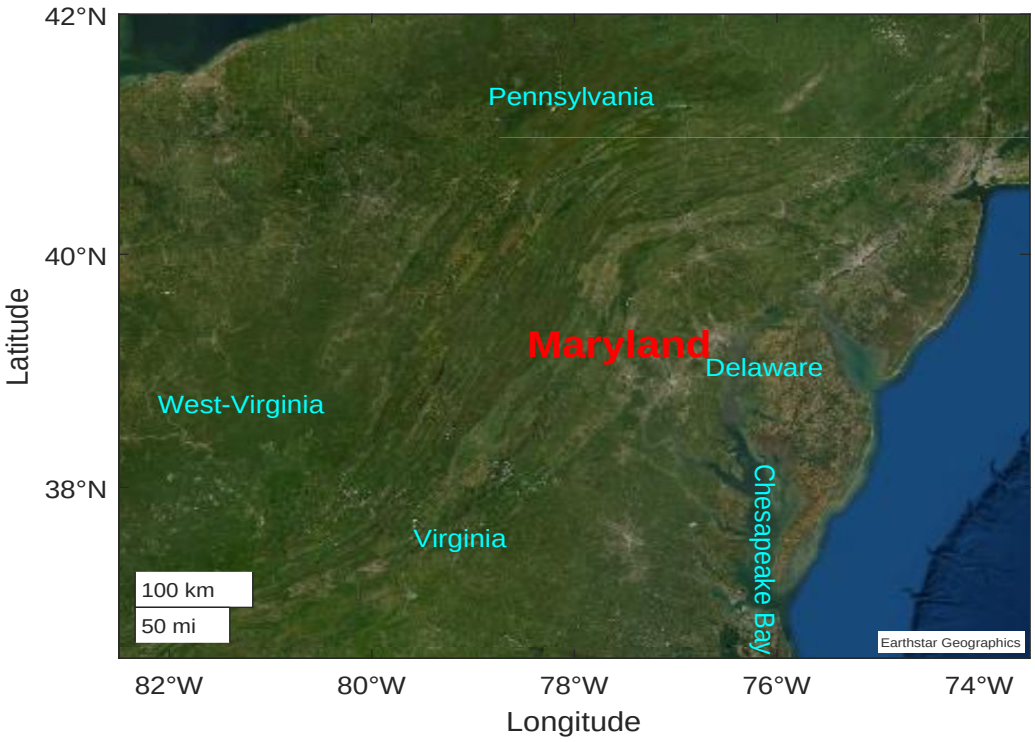


Figure 3. Visualization of the area under study, in terms of geographic location and global basemap. Maryland state, located in East Coast USA, includes several vegetation as well as forest-dominated areas.

3.2. Experimental Setup

Since the main supervised learning task of the present study focuses on classification, two non-overlapping training-test sets had to be defined a priori. Based on the reasoning explained in Section 3.1, as computational resources are not unlimited (in terms of both training time and memory), we randomly selected 100 image tiles for our experimental study. With this setup, we split the employed tiles into 80%/20% training/test sets. Moreover, 25% of the training set is used for validation purposes, ending up in a general split of 60%/20%/20% training/validation/test sets (and 60/20/20 training/validation/test samples accordingly). All splits are performed in a random way, to avoid bias in favour/against specific classes.

In order to provide an overview of the tiles used in the training/validation/test sets, in Figure 4 we provide several different NLCD label-tiles included in each set, accompanied by the respective label values previously summarized in Table 1. Moreover, in Figure 5, we visualize the respective real RGB aerial images corresponding to those of Figure 4, which were collected at a 1-meter spatial resolution on single days in 2017 as part of the National Agriculture Imagery Program (NAIP). From both Figures 4 and 5, we observe that the classification problem at hand constitutes a realistically quite difficult one, since different tiles are composed of diverse label distributions.

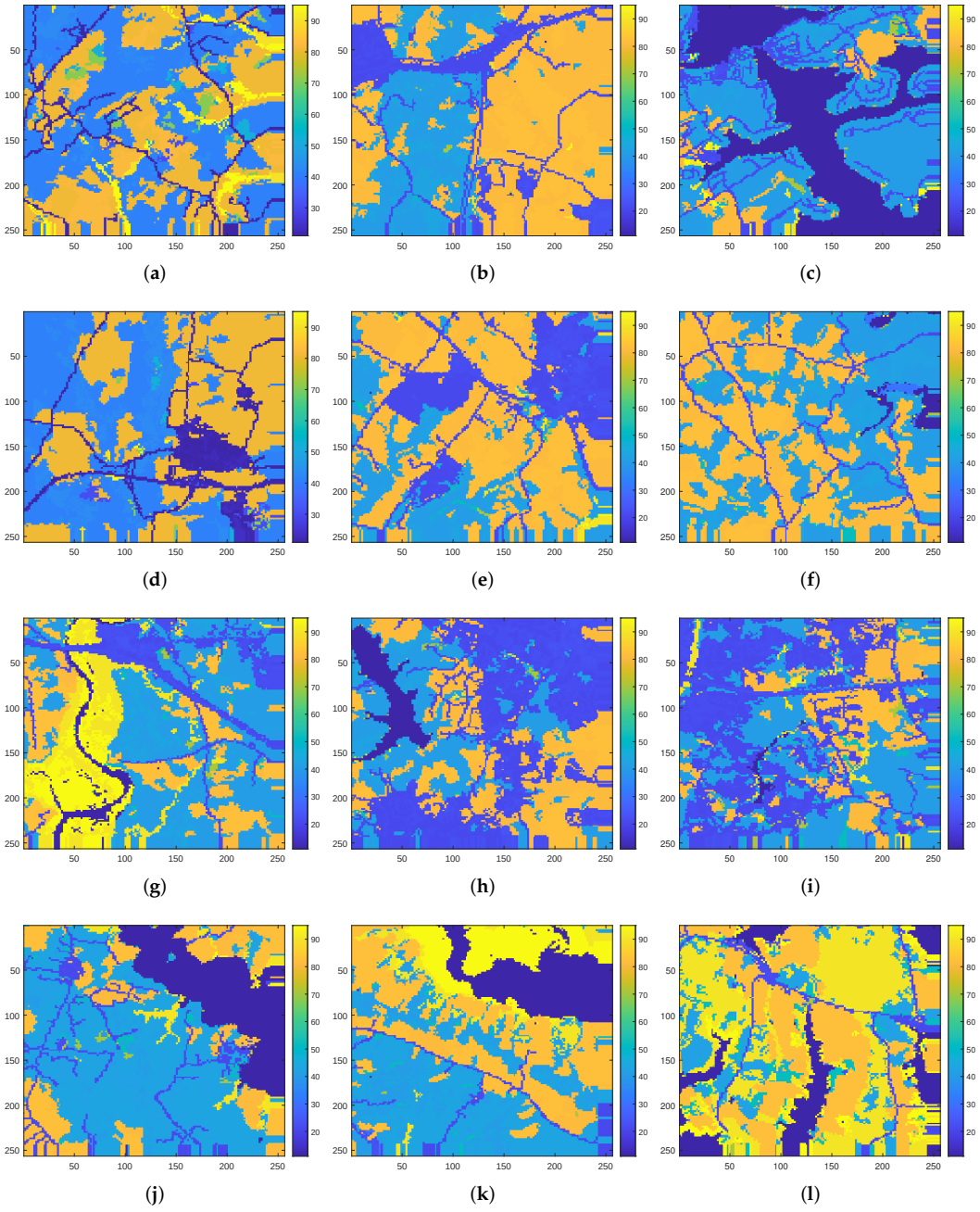


Figure 4. Overview of the label-tiles used for training (**left column**), validation (**middle column**), and test (**right column**) of the models. Different tiles are composed of different labels' distributions, indicating the difficulty of the problem at hand. (**a**) Training Set-Tile 171. (**b**) Validation Set-Tile 2388. (**c**) Test Set-Tile 224. (**d**) Training Set-Tile 380. (**e**) Validation Set-Tile 2551. (**f**) Test Set-Tile 3244. (**g**) Training Set-Tile 3376. (**h**) Validation Set-Tile 3035. (**i**) Test Set-Tile 3264. (**j**) Training Set-Tile 3663. (**k**) Validation Set-Tile 3118. (**l**) Test Set-Tile 3976.

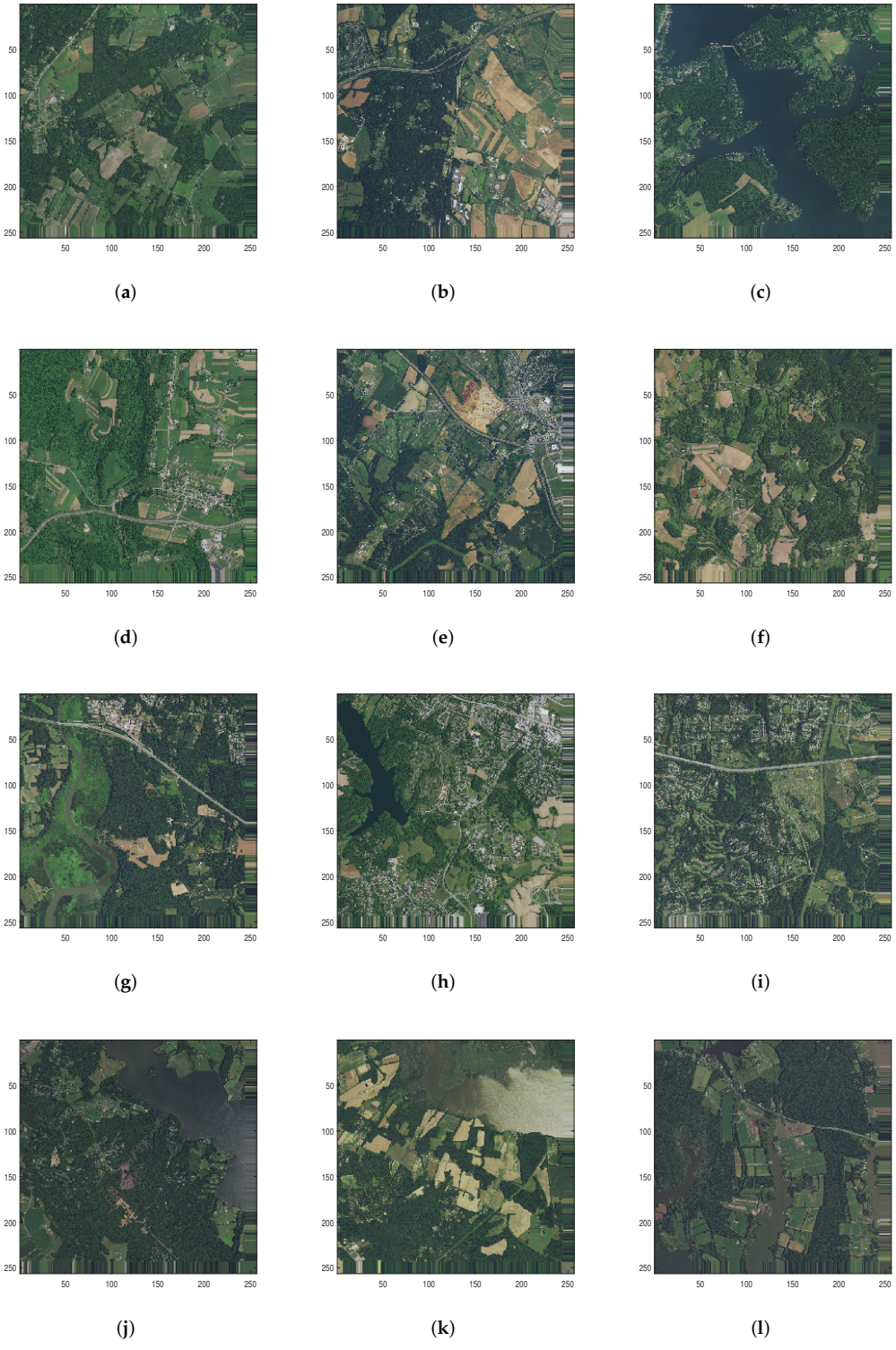


Figure 5. Overview of the real RGB version of the tiles used for training (**left column**), validation (**middle column**), and testing (**right column**) of the models. The various different objects indicate the difficulty of the classification task. (**a**) Training Set-Tile 171. (**b**) Validation Set-Tile 2388. (**c**) Test Set-Tile 224. (**d**) Training Set-Tile 380. (**e**) Validation Set-Tile 2551. (**f**) Test Set-Tile 3244. (**g**) Training Set-Tile 3376. (**h**) Validation Set-Tile 3035. (**i**) Test Set-Tile 3264. (**j**) Training Set-Tile 3663. (**k**) Validation Set-Tile 3118. (**l**) Test Set-Tile 3976.

Table 1. Ground-truth labels of the employed dataset; 15 different classes constitute a quite challenging classification task.

Class ID	Class Value	Class Name
1	11	Open Water
2	21	Developed, Open Space
3	22	Developed, Low Intensity
4	23	Developed, Medium Intensity
5	24	Developed High Intensity
6	31	Barren Land (Rock/Sand/Clay)
7	41	Deciduous Forest
8	42	Evergreen Forest
9	43	Mixed Forest
10	52	Shrub/Scrub
11	71	Grassland/Herbaceous
12	81	Pasture/Hay
13	82	Cultivated Crops
14	90	Woody Wetlands
15	95	Emergent Herbaceous Wetlands

Since our task is to classify every pixel in an MT-RS image (and not the whole image itself), it is to be expected that the dataset is going to be imbalanced. Such a case is common in pixel-level classification tasks, as a direct consequence of non-uniform label distribution among different sample MT-RS images in real-world datasets. Figure 6 depicts the normalized histogram of the different classes present in every different set (i.e., training-validation-test), clearly showing the imbalanced label distribution. We observe that some classes are quite infrequent (e.g., barren land, shrub/scrub), whereas some others dominate the dataset (e.g., deciduous forest with a relative probability of almost 25%).

Using the aforementioned dataset splits, in the upcoming experiments we train each model in the training set and validate its performance (by trying various hyper-parameter configurations) in the validation set. Subsequently, we select and report the model's performance based on the best validation accuracy it achieved during the training process (irrespective if that was obtained at the final epoch or not). These best performing models are then used in the test set, once, in order to assess the final performance of the model.

The designed U-Net architectures were developed in a Python programming platform by exploiting the TensorFlow and Keras [54] libraries. The reason behind such a choice is threefold: First, TensorFlow is an open-source ML framework, which now integrates the higher-level DL-specific Keras library and provides support and updates on most state-of-the-art DL models and algorithms. Second, TensorFlow and Keras provide a high level of customization in nearly every part of a DL model, which was highly desired in the present study since various custom layers were implemented. Finally, and most importantly, both libraries can perform calculations on a GPU, dramatically decreasing the computational time of the training process. In our experiments, we used NVIDIA's GPU model *Quadro P4000* with 8 Gb of available RAM memory.

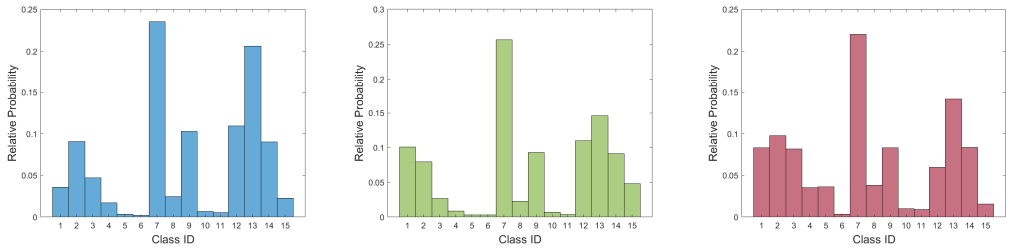


Figure 6. Labels' distribution in the employed training-validation-test sets. In every set, there exists a clear dominant class (i.e., deciduous forest), whereas some classes (e.g., barren land, shrub/scrub, grassland/herbaceous) are extremely scarce. (**a**) Labels' distribution—Training Set. (**b**) Labels' distribution—Validation Set. (**c**) Labels' distribution—Test Set.

3.3. Evaluation Metrics for Multi-Temporal Land-Cover Classification

Since the problem at hand is essentially a pixel-level multi-class classification task, we rely on several ML metrics for the evaluation of the designed models. Comparing the prediction of one pixel with its ground-truth value, the outcome can be one of four types: *True Positive* (TP), *False Positive* (FP), *False Negative* (FN), or *True Negative* (TN). *Positive* and *Negative* correspond to the class for which we compute the metric, while *True* and *False* stand for the equality between the ground-truth label and the predicted one. Based on these definitions, widely-used ML metrics are defined for each different class present in the data as follows:

$$\begin{cases} recall = \frac{TP}{TP + FN} \\ precision = \frac{TP}{TP + FP} \\ accuracy = \frac{TP + TN}{TP + FP + FN + TN} \end{cases} \tag{11}$$

Since pixel-level classification can be seen as an image segmentation task and because we are dealing with a highly imbalanced dataset, pixel-level accuracy is by no means the only suitable metric for segmentation purposes. Indeed, certain classes clearly dominate specific sample tiles, ending up in predictions which can have unclear segmentation interpretations. To overcome this issue, we additionally report the F_1-score as well as the *IoU* index. The F_1-score can be interpreted as a weighted average (harmonic mean) of the *precision* and *recall* metrics defined in (11), and it is widely used in segmentation tasks under the *Dice coefficient* name. On the other hand, IoU (also known as the *Jaccard index*) is the area of overlap between the predicted segmentation and the ground-truth, divided by the area of the union between the predicted segmentation and the ground-truth. Based on these remarks, as well as on (11), the F_1-score and the *IoU* index metrics are defined class-wise as follows:

$$\begin{cases} F_1 - score = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{TP}{TP + \frac{FP + FN}{2}} \\ IoU = \frac{|Ground - Truth \cap Prediction|}{|Ground - Truth \cup Prediction|} = \frac{TP}{TP + FP + FN} \end{cases} \tag{12}$$

Since all metrics in (11) and (12) are computed for each different class in the dataset, their overall counterparts are then calculated and reported by averaging them across all classes. This approach is called the *Macro-Averaging* of metrics, and treats all classes equally. Since our problem is a highly-imbalanced multi-class classification task, we instead compute and report the so-called *Micro-Averaged* respective metrics. In that way, we aggregate the contributions of all classes to compute the average metric, ending up with less biased estimations. We should notice at this point the fact that, from ML evaluation theory for multi-class problems [55], in the micro-averaged metrics' case, the classification accuracy metric coincides with the precision, recall, and F_1-score ones. Moreover, for each of the trained models presented below, we report the time for its training as a metric of its computational complexity.

Finally, we additionally provide several *confusion matrices*, where each row represents the observations in a true/actual class while each column represents the observations in a predicted class. To have a clearer sense of the metrics reported for each class, each cell value in all subsequent confusion matrices is normalized by the total number of observations. Furthermore, the respective class-wise recall is displayed across each confusion matrix's row, whereas the respective class-wise precision is displayed across each confusion matrix's column.

3.4. 4D U-Net Architecture Parameter Tuning

Two important hyper-parameters mostly affect the performance of the proposed 4D U-Net model: the number of stack-layers it contains (i.e., the depth of the U-Net model) and the number of filters employed for efficient MT-MS feature learning. In order to determine the best hyper-parameter configuration, we performed an ablation study considering each one of them separately as well as in combination.

The first set of experiments assesses the performance of the 4D U-Net in relation to its depth, i.e., the number of stack-layers it comprises. We trained two different 4D U-Net models with depths equal to 3 and 4, respectively, and the parameter configuration described in Section 2.3. The number of filters employed for this set of experiments was set to 4 for the first stack in the contracting path (doubled to 8 in the second stack, 16 in the third stack, and 32 in the fourth stack).

The second set of experiments quantifies the number of filters used to train the 4D U-Net model. We again trained two different models with depths equal to 3 and 4, but this time we doubled the number of filters used in the first contracting path stack from four to eight. In this way, we obtained more feature maps across every layer, aiming to a more complex model with a better performance.

Table 2 shows that as we employ more filters, the obtained metrics increase, indicating that the network's generalization capability improves. Of course, more filters automatically translate to more parameters to be learned, resulting in an increase in the time needed to train the network. We also observe that increasing the U-Net depth from three to four has a positive effect in all metrics when four filters are employed, rather than eight. All in all, we can conclude that the best U-Net model has a depth equal to 3 and achieves a classification accuracy of 61.56% in approximately 9 h.

Table 2. Classification metrics and associated training times (minutes) for the trained 4D U-Net models. Employing more filters and constructing deeper architectures leads to more complex and accurate models, requiring longer training times.

U-Net Depth	#Filters 1st Stack	Accuracy	Time	Precision	Recall
3	4	0.4192	452.59	0.4192	0.4192
	8	**0.6156**	**542.14**	**0.6156**	**0.6156**
4	4	0.4682	463.13	0.4682	0.4682
	8	0.6078	565.72	0.6078	0.6078

3.5. Comparison to Lower-Order U-Net Models

In order to compare our method and investigate its possible merits, we performed the same ablation study for similar lower-order U-Net models and tune their hyper-parameters in the spirit of Section 3.4. The lower-order U-Net models are constructed based on the approach described in detail in Section 2.3. More precisely, we employ the 2D U-Net basic architecture proposed in [37], adapted for the problem at hand, with only spatial features being learnt, whereas the spectral and temporal dimensions are averaged. We also consider two distinct 3D U-Net models, one learning spatio-temporal features (i.e., averaging the spectral dimension of the data) and one learning spatio-spectral features (i.e., averaging the temporal dimension of the data). The models are constructed with exactly the same

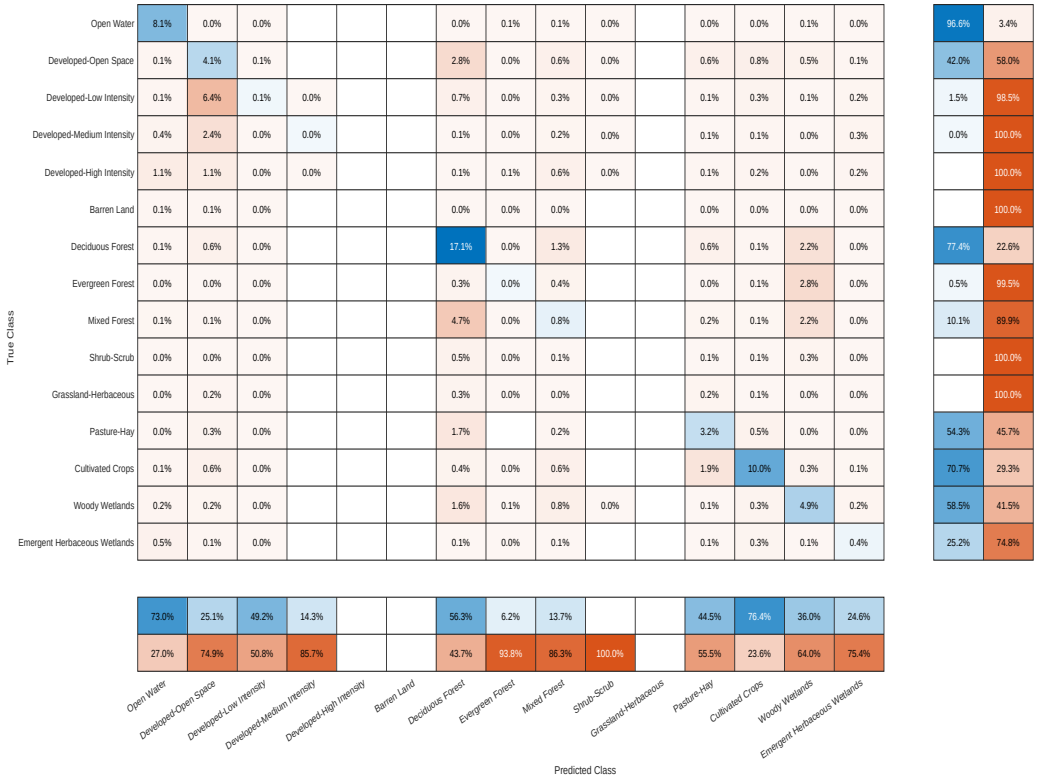
parameter values described for the 4D case (e.g., model-depth, number of filters per layer, kernel sizes according to dimension, etc.), and they are tuned similarly to the 4D U-Net model in order to converge to the best possible version.

Following the same approach as in Section 3.4, we train two distinct 2D and 3D U-Net models with depths equal to 3 and 4, each having three or four filters in the first contracting path stack. According to the results presented in Table 3, increasing the number of employed filters has a positive effect in the performance of nearly all lower-order models. We also note that increasing the U-Net depth has a significant positive effect only for the 3D U-Net-Spectral model. Of course, adopting deeper network architectures with a larger number of filters results in more complex models (many parameters), thus increasing accordingly the required time for training.

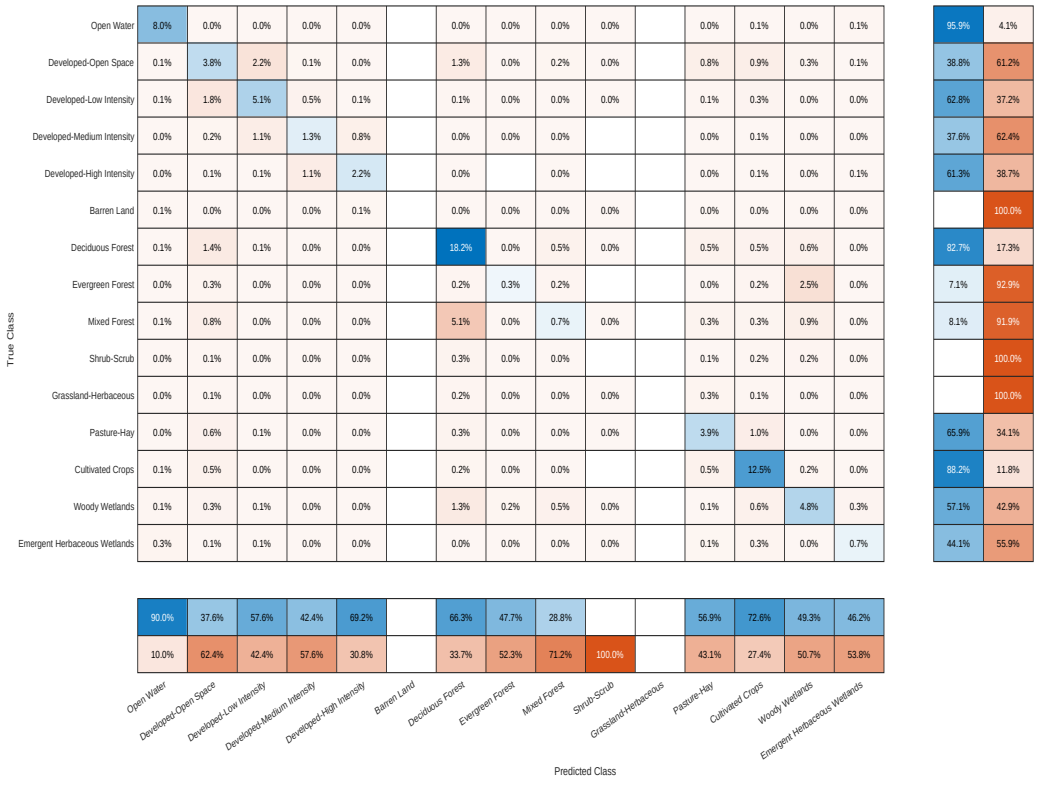
Table 3. Classification metrics and associated training times (minutes) for the trained 2D [37] and 3D U-Net models. The more filters are employed, the better the obtained classification metrics, with higher-order models outperforming the lower-order ones.

Model	U-Net Depth	#Filters 1st Stack	Accuracy	Time	Precision	Recall
2D U-Net [37]	3	4	0.2198	18.97	0.2198	0.2198
	3	8	0.2201	19.55	0.2201	0.2201
	4	4	0.2202	19.32	0.2202	0.2202
	4	8	**0.2202**	**19.68**	**0.2202**	**0.2202**
3D U-Net-T	3	4	0.2202	39.20	0.2202	0.2202
	3	8	0.2606	44.23	0.2606	0.2606
	4	4	0.2202	40.21	0.2202	0.2202
	4	8	**0.2652**	**44.76**	**0.2652**	**0.2652**
3D U-Net-S	3	4	0.2202	65.00	0.2202	0.2202
	3	8	**0.4877**	**76.40**	**0.4877**	**0.4877**
	4	4	0.2202	66.80	0.2202	0.2202
	4	8	0.4742	78.52	0.4742	0.4742

Figure 7 shows the confusion matrices for the proposed 4D U-Net model and the 3D U-Net-S that follows in performance. We observe that both models exhibit a similar performance in the prediction of the dominant classes in the dataset, namely, *deciduous forest*, *open water* and *cultivated crops*. However, for the rest of the classes, the proposed 4D U-Net model exhibits a better performance in recovering the actual labels than the 3D U-Net-S method. Among the most challenging classes, we notice the following:



(**a**)



(**b**)

Figure 7. Confusion matrices for the best 3D and 4D U-Net models. The higher dimensional U-Net model recovers better nearly every possible class. (**a**) Confusion matrix for the best 3D U-Net-S model. (**b**) Confusion matrix for the best 4D U-Net-Proposed model.

- Class *deciduous forest* is frequently mistaken for class *developed-open space* or *mixed forest* by both models, although the 4D U-Net classifies it correctly at a higher rate.
- Class *mixed forest* is mostly mistaken for class *deciduous forest* or *woody wetlands* by 3D U-Net-S and for *deciduous forest* by 4D U-Net.
- Class *developed-low intensity* is rarely predicted correctly by the 3D U-Net model, whereas classes *developed-medium intensity* and *developed-high intensity* are never predicted correctly at all. On the other hand, the proposed 4D U-Net model distinguishes these highly similar classes quite well.
- Classes *barren land*, *shrub-scrub*, and *grassland-herbaceous* are never predicted correctly by any of the methods, a result that can be attributed to their extremely rare frequency in the dataset, as it can be seen from the respective labels' distribution histograms in Figure 6.
- Predictions derived from the proposed 4D U-Net model exhibit clearly higher class-wise precision (i.e., summaries across each column of the confusion matrices) across 14 out of the 15 available classes. This is indicative of the low number of observations being attributed incorrectly to each predicted class, leading to a low *type-I error* per class.
- Predictions derived from the proposed 4D U-Net model clearly exhibit a higher class-wise recall (i.e., summaries across each row of the confusion matrices) for 8 out of the 15 available classes. This is indicative of the low number of observations being attributed incorrectly to each true class, leading to a low *type-II error* per class.

Figure 8 presents the convergence behavior of our proposed 4D U-Net architecture vis-à-vis 3D U-Net-S, which arose as the best among all lower-order U-Nets. Convergence is depicted in terms of the classification accuracy as a function of the number of epochs, for both the training and validation sets. Naturally, we used the best hyper-parameters for each U-Net model. We observe that the performance of the 4D U-Net model improves gradually, while at the same time, over-fitting issues are prevented as the validation curve tracks the training curve fairly well, indicating a strong generalisation capability. On the other hand, the best 3D U-Net-S model converges slower and to a lower classification accuracy value when compared to the proposed 4D model.

In order to interpret the obtained results from an image segmentation point of view, in Table 4, we additionally report the mean *IoU* index, as well as the F_1-score, precision, and recall metrics for the proposed 4D model and the lower-order methods. As we can see in Table 4, the proposed 4D U-Net significantly outperforms all the other U-Net models in terms of every reported metric, a strong indication of its robustness in class-imbalance situations arising quite frequently in real-world datasets.

Table 4. Mean IoU, F_1-score, precision, and recall of the best 3D and 4D U-Net models. In every metric, the proposed higher-dimensional model outperforms the second best one.

Model	IoU	F_1-Score	Precision	Recall
2D U-Net [37]	0.1237	0.2202	0.2202	0.2202
3D U-Net-T	0.1529	0.2652	0.2652	0.2652
3D U-Net-S	0.3225	0.4877	0.4877	0.4877
4D U-Net-Proposed	**0.4447**	**0.6156**	**0.6156**	**0.6156**

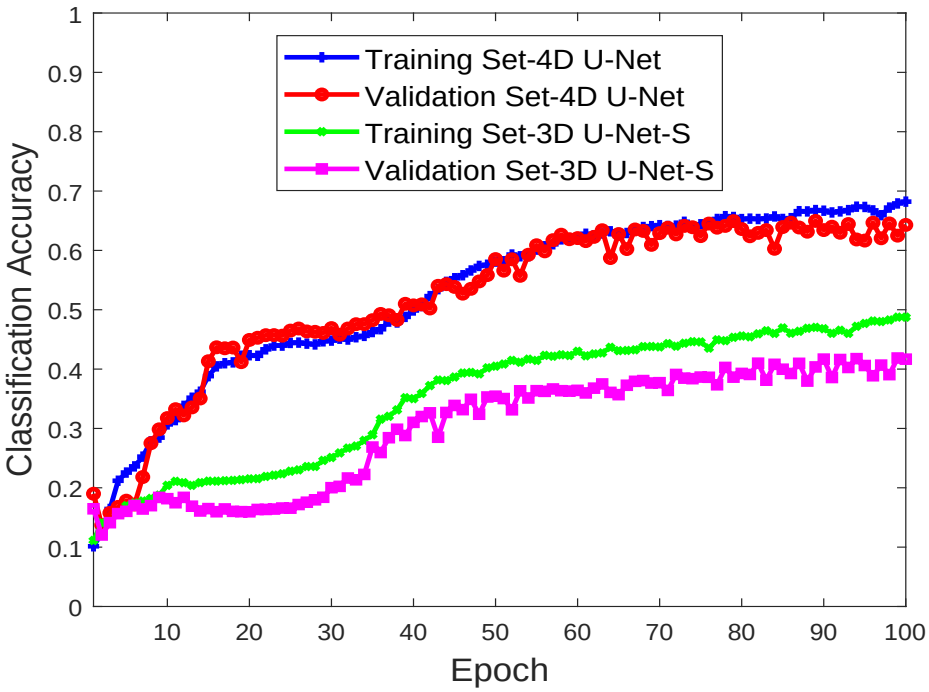


Figure 8. Classification accuracy of the 4D U-Net and 3D U-Net-S models. As the number of epochs increases, the performance of both U-Net models improves, but with a clear classification accuracy gap in favour of our proposed 4D U-Net system.

To have a comprehensive evaluation summary for all methods in the employed dataset, in Table 5, we report the obtained accuracy for each one of the test set tiles and for all the U-Net models. We observe that in 19 out of the 20 test tiles, the proposed 4D U-Net model clearly outperforms all the competing methods by considerable margins, which, in several cases, exceeds 10%, while being slightly inferior in only one tile.

To have a better visual sense of the proposed model's performance, Figure 9 depicts the actual labels of 4 different test tiles, accompanied by the predictions obtained using our proposed 4D model and the second best method, i.e., 3D U-Net-S. The classification maps shown in Figure 9 demonstrate that our 4D U-Net model captures most of the labels better, resulting in clearly segmented regions. We also notice that the 3D U-Net-S model suffers from severe "noise-effects", a result of its poor performance to correctly predict large parts of every test tile. On the other hand, the proposed 4D U-Net model captures most of the labels of the actual classification map the best, ending up with clearly segmented regions bearing a strong resemblance to the ground-truth labels.

Table 5. Classification accuracy of the best U-Net models in each of the test set tiles. In 95% of all cases, the proposed 4D U-Net model outperforms all competing ones.

Tile-ID	2D U-Net [37]	3D U-Net-T	3D U-Net-S	4D U-Net
173	0.4703	0.4521	0.5782	**0.6128**
224	0.2991	0.2782	0.6554	**0.6616**
761	0.5271	0.4849	0.5460	**0.6080**
2223	0.2736	0.3009	0.3102	**0.5417**
2877	0.4176	0.5500	0.6412	**0.7794**
2937	0.1112	0.1080	0.3550	**0.5732**
3112	0.1245	0.1206	0.2495	**0.3924**
3227	0.0325	0.0893	0.7413	**0.7907**
3244	0.3023	0.3762	0.4971	**0.6607**
3264	0.3205	0.3308	0.4366	**0.5622**
3470	0	0.0071	0.0944	**0.5722**
3522	0.0416	0.0386	0.2392	**0.6408**
3532	0.5455	0.5368	0.5546	**0.6018**
3542	**0.5896**	0.5787	0.4739	0.5768
3772	0.3154	0.3671	0.3534	**0.4095**
3976	0.0011	0.0709	0.5906	**0.6109**
4137	0.0214	0.2447	0.7474	**0.8010**
4474	0.00006	0.1033	0.6509	**0.7336**
4478	0.0053	0.1030	0.4709	**0.5358**
4960	0.0054	0.1627	0.5685	**0.6472**
Total	0.2202	0.2652	0.4877	**0.6156**

3.6. Comparison to State-of-the-Art Methods

Apart from the lower-order U-Net models presented in Section 3.5, we also compared our high-dimensional model to current state-of-the-art methods in the field. For the comparison to be fair, we chose a method proposed by Sefrin et al. in [39] (described earlier in Section 1), which operates directly on raw MT-RS data, as is the case with our proposed method. Since the FCN architecture proposed in [39] does not exploit any sequential information present in the data, we average their temporal dimension, as was the case with the 2D U-Net model designed in the present study. Concerning the FCN + LSTM approach of [39], the authors further construct two variants of the FCN + LSTM model: a *fixed-sequence* LSTM ($LSTM_F$), which employs the available MT-RS data as they are, and a *variable-sequence* LSTM ($LSTM_V$), which further adds MT-RS samples from further available sampling dates. Since in our case we employ one time-series of RS data and since the results in [39] demonstrate the superiority of the $LSTM_F$ over the $LSTM_V$, we compare our method with the $LSTM_F$ variant (denoted as FCN + LSTM from now on).

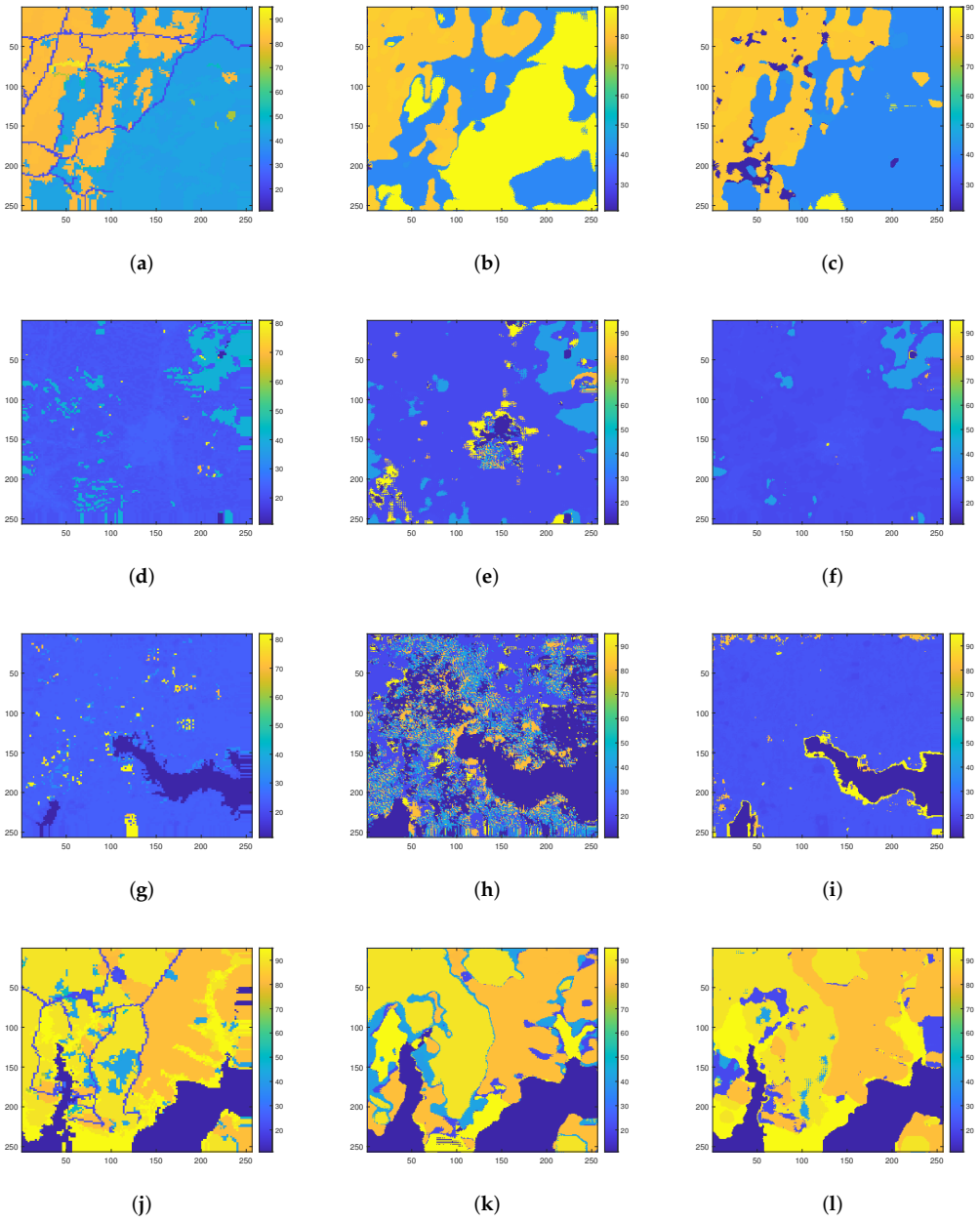


Figure 9. Actual and predicted labels corresponding to four distinct test set tiles, obtained via the best 3D and 4D (proposed) U-Net models. The proposed 4D model captures better the texture of the map as well as more of its actual labels/values. (**a**) Actual labels—Tile 2223. (**b**) Predicted labels—Tile 2223-3D U-Net-S. (**c**) Predicted labels—Tile 2223-4D U-Net. (**d**) Actual labels—Tile 2937. (**e**) Predicted labels—Tile 2937-3D U-Net-S. (**f**) Predicted labels—Tile 2937-4D U-Net. (**g**) Actual labels—Tile 3470. (**h**) Predicted labels—Tile 3470-3D U-Net-S. (**i**) Predicted labels—Tile 3470-4D U-Net. (**j**) Actual labels—Tile 4474. (**k**) Predicted labels—Tile 4474-3D U-Net-S. (**l**) Predicted labels—Tile 4474-4D U-Net.

Another aspect that should be highlighted here is the pre-processing steps proposed in [39]. More specifically, the authors propose two ways in order to address the class-imbalance problem faced in pixel-level classification tasks: Firstly, they exclude pixels which belong to extremely scarce classes by forming an *Excluded Class* which contains half of the less-frequent classes of the dataset; secondly, they further exclude shoreline-water pixels, since they vary significantly across time, based on an NDVI-threshold criterion. In the present study, though, we do not impose any pre-processing steps, in an attempt not only to

quantify the performance of the proposed method on raw MT-RS data but additionally in order to investigate its potential in real-world experimental cases where class-imbalanced datasets are highly expected. As a result, for the FCN and FCN + LSTM methods, we do not use any pre-processing steps in order to be aligned with the aforementioned results presented in Section 3.5.

Figure 10 depicts the resulting confusion matrices for both comparison models. We notice that both models outperform the lower-order U-Nets presented in Section 3.5, as they achieve a classification accuracy of up to 0.5004% (FCN) and 0.5457% (FCN + LSTM). Moreover, we observe that the FCN method predicts the dominant classes slightly better (i.e., *deciduous forest* and *cultivated crops*), while for nearly all the other classes, the FCN + LSTM recovers the actual labels more accurately. We further highlight the following:

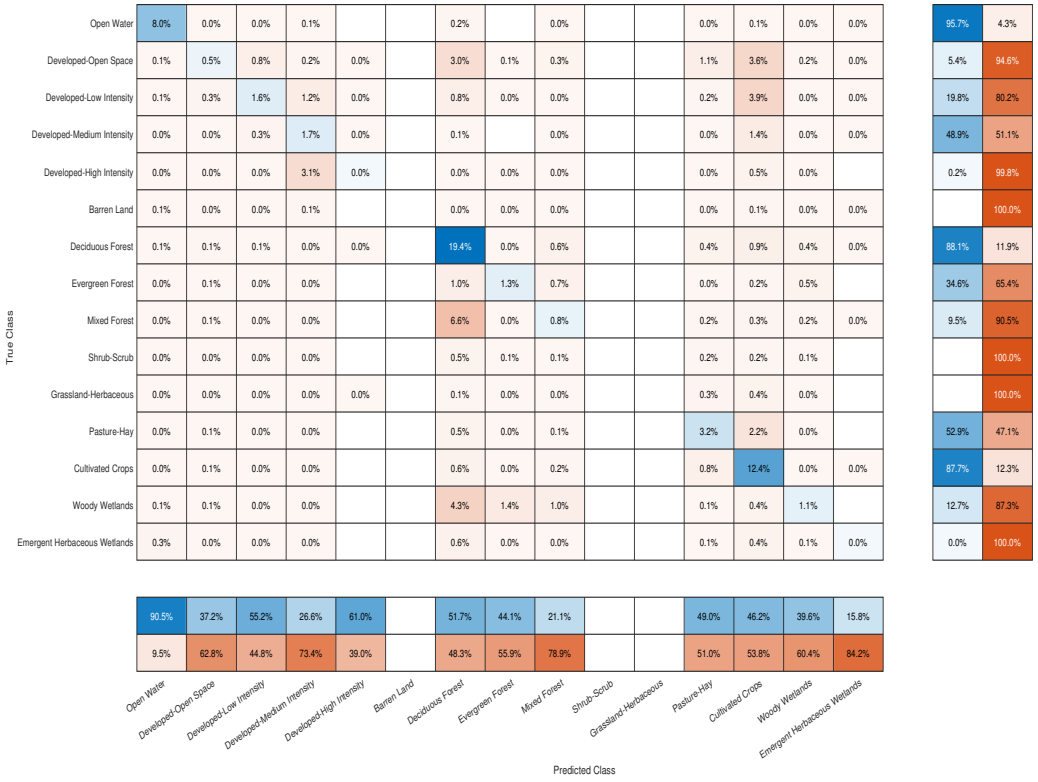
- Class *deciduous forest* is frequently mistaken for class *developed-open space* or *mixed forest* by both models, while our proposed 4D approach predicts it correctly with a higher rate.
- Predictions derived from both the FCN and FCN + LSTM methods exhibit comparable class-wise precision (i.e., summaries across each column of the confusion matrices), as the FCN model is better in 5 out of the 15 available classes, while being inferior in 5 out of 15.
- Predictions derived from the FCN + LSTM [39] clearly outperform those of the FCN [39] in terms of class-wise recall (i.e., summaries across each row of the confusion matrices), as the FCN is better in only 3 out of the 15 classes while being inferior in 8 out of 15.

In an attempt to interpret these results in terms of image segmentation metrics, in Table 6 we report the mean *IoU* index, as well as the F_1-score, precision, and recall metrics for FCN, FCN + LSTM, and 4D U-Net. We observe that the FCN + LSTM outperforms the FCN in all metrics, as expected. Nevertheless, both models are clearly outperformed by the proposed 4D U-Net in both segmentation and classification metrics, implying inferior robustness in real-world class-imbalance situations.

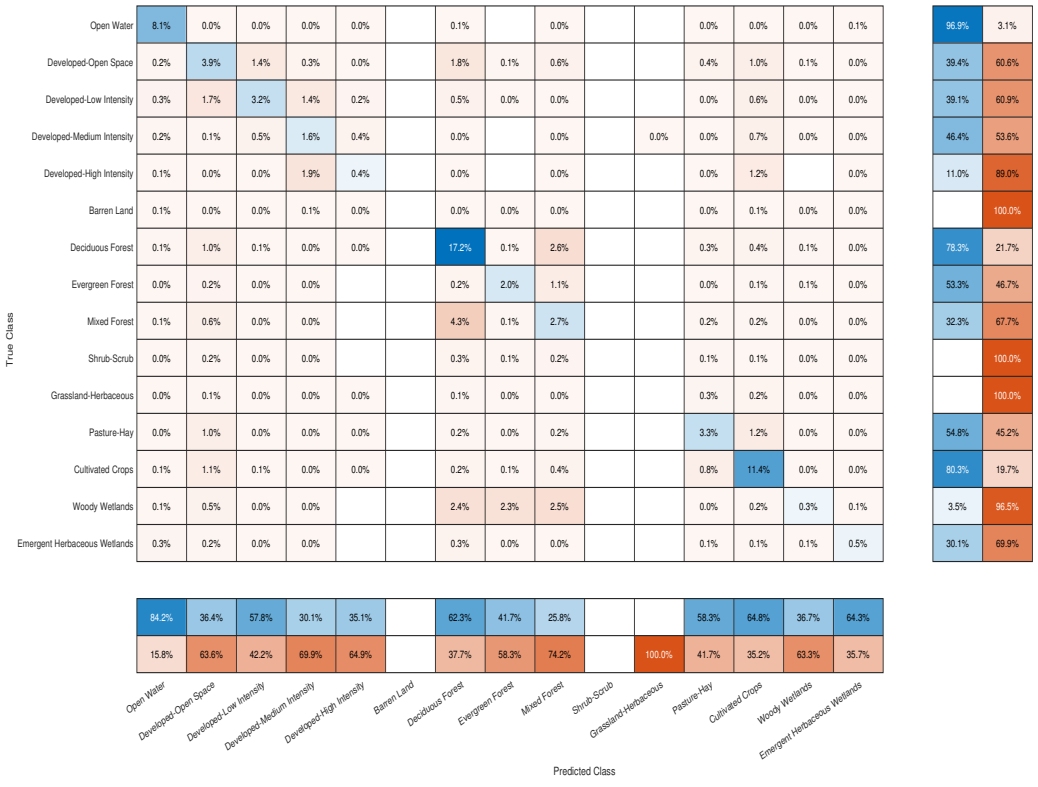
Since the FCN + LSTM outperforms the FCN, we further depict its predictions across four specific tiles in Figure 11, accompanied by their actual labels, together with the predictions obtained using our proposed 4D model. The classification maps shown in Figure 11 demonstrate that the FCN + LSTM delivers poorly segmented regions in most of the tiles. On the other hand, the proposed 4D U-Net model ends up with more clearly segmented regions, closer to the ground-truth labels.

Table 6. Mean IoU, F_1-Score, precision, and recall for FCN, FCN + LSTM, and 4D U-Net. The proposed higher-dimensional model outperforms its competitors in both metrics.

Model	IoU	F_1-Score	Precision	Recall
FCN	0.3337	0.5004	0.5004	0.5004
FCN + LSTM	0.3753	0.5457	0.5457	0.5457
4D U-Net-Proposed	**0.4447**	**0.6156**	**0.6156**	**0.6156**



(**a**)



(**b**)

Figure 10. Confusion matrices for the methods FCN and FCN + LSTM [39]. The two models are comparable in accuracy, precision and recall metrics, with the FCN + LSTM model being slightly better. (**a**) Confusion matrix for the Comparison-FCN model [39]. (**b**) Confusion matrix for the FCN + LSTM model [39].

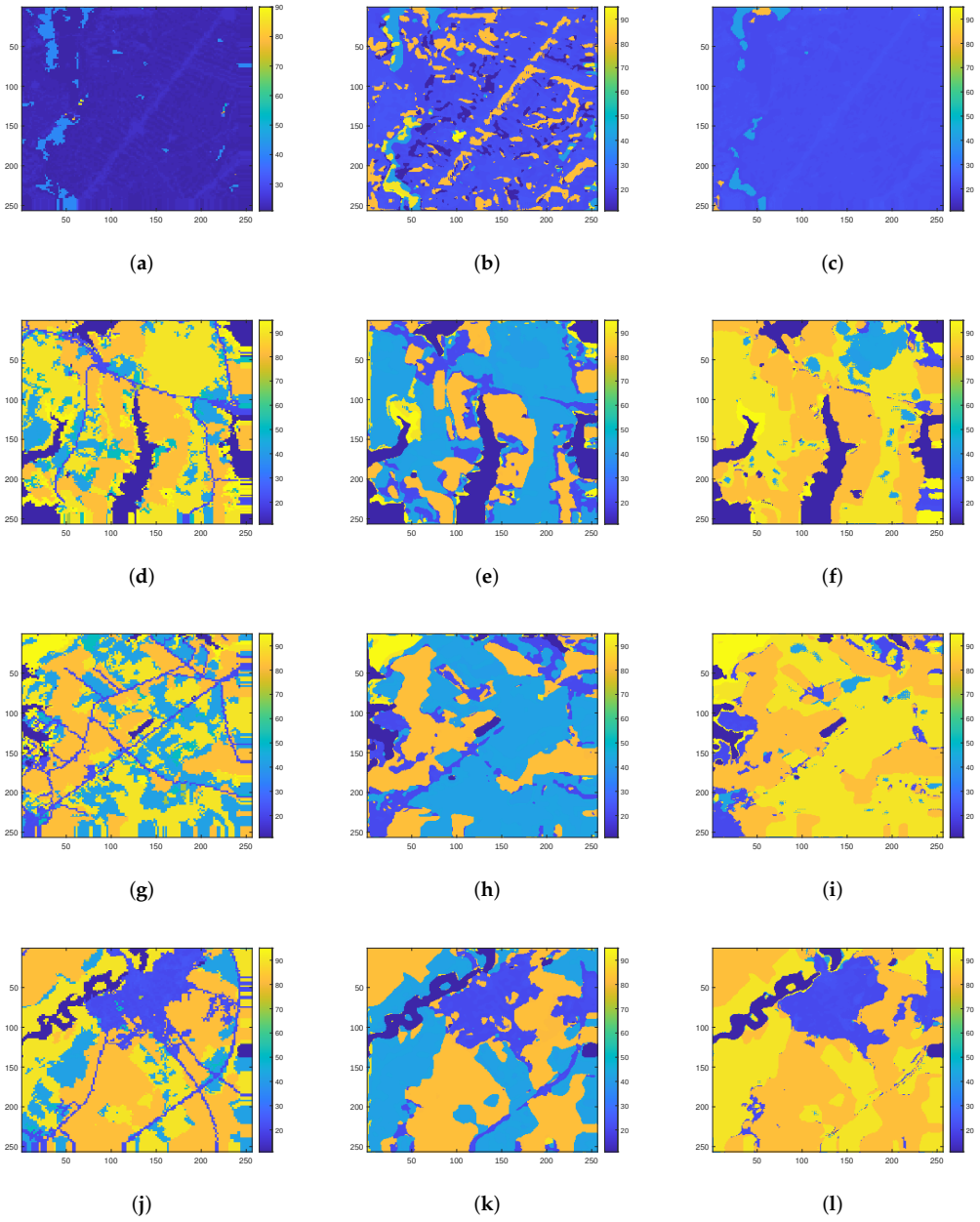


Figure 11. Actual and predicted labels corresponding to four distinct test set tiles, obtained via FCN + LSTM and 4D U-Net. The proposed 4D model captures better the texture of the map as well as more of its actual labels/values. (**a**) Actual labels—Tile 3522. (**b**) Predicted labels—Tile 3522-FCN + LSTM. (**c**) Predicted labels—Tile 3522-4D U-Net. (**d**) Actual labels—Tile 3976. (**e**) Predicted labels—Tile 3976-FCN + LSTM. (**f**) Predicted labels—Tile 3976-4D U-Net. (**g**) Actual labels—Tile 4478. (**h**) Predicted labels—Tile 4478-FCN + LSTM. (**j**) Predicted labels—Tile 4478-4D U-Net. (**j**) Actual labels—Tile 4960. (**k**) Predicted labels—Tile 4960-FCN + LSTM. (**l**) Predicted labels—Tile 4960-4D U-Net.

4. Discussion

In this section, we discuss the results derived in Section 3. We present a comprehensive discussion concerning the performance of the models as a function of the size of the available training data in an attempt to quantify each model's needs for efficient pixel-level classification. We highlight the strengths and limitations of each employed model in order for their comparison to be clear and complete.

4.1. Effect of Training Set Size on Classification Accuracy

In order to assess the performance of the proposed 4D U-Net model with respect to its needs in terms of available training data, we doubled the number of samples used for training the models (from 60 to 120 MT-RS images) while the validation and test samples remained the same (i.e., 20 MT-RS images each). We performed the same ablation studies for all lower dimensional U-Net variants, as well as the FCN and FCN + LSTM methods. We summarise the obtained results in Table 7. We observe that increasing the training samples has a minor effect on all U-Net models, with the highest performance improvement being observed for 3D U-Net-S. On the other hand, the FCN and FCN + LSTM enjoy a higher accuracy when trained with more samples. The 4D U-Net does not seem to improve when trained with more samples but still outperforms the second best method by approximately 3%, indicating the robustness of the proposed approach.

Table 7. Classification accuracy and respective training time (minutes) w.r.t. the number of training samples for the best U-Net and comparison models. Employing more samples slightly improves the performance of the lower-order models and the comparison ones but not enough to surpass the proposed 4D architecture.

Model	#Training Samples	Accuracy	Time
2D U-Net [37]	60	0.2202	19.68
	120	0.2256	34.52
3D U-Net-T	60	0.2652	44.76
	120	0.2742	86.91
3D U-Net-S	60	0.4877	76.40
	120	0.5768	154.48
FCN [39]	60	0.5004	26.54
	120	0.5753	47.2813
FCN + LSTM [39]	60	0.5457	64.06
	120	0.5810	121.77
4D U-Net-Proposed	60	**0.6156**	**542.14**
	120	**0.6094**	**1129.05**

4.2. Proposed Method Failure Cases

Even if the proposed method performs quite well under the aforementioned experimental scenarios, it is deemed appropriate to report those cases in which it seems to have a hard time.

More specifically, observing more closely the respective confusion matrix in Figure 7b, there exist some classes which our proposed method predicts with very low accuracy rate (e.g., developed-medium intensity, developed-high intensity), whereas others cannot be predicted correctly at all (e.g., barren land, shrub-scrub, grassland-herbaceous). In order to explain the reasoning of such failures, in Figure 12, we visualise two test-set's tiles in which our proposed model did not perform so well (i.e., tile 3112 with 39.24%, and tile 3772 with 40.95%), along with the respective predictions and histograms. Interpreting those predictions, we observe that classes with similar characteristics (i.e., developed-open space, developed-low intensity, developed-medium intensity, and developed-high intensity) seem to confuse the proposed model, a fact that can be attributed exactly to the slight differentiation among them. Moreover, the fact that some classes cannot be predicted correctly at all (e.g., barren land) can be attributed to their extremely low frequency in the whole dataset.

On the other hand, classes whose characteristics can change throughout the years (e.g., deciduous forest, cultivated crops) are predicted correctly by the proposed method with high accuracy rates. The reasoning behind this is based on the fact that our 4D U-Net model better captures the temporal variability of those classes across the spectral bands of the data, ending up with analogous increased performance.

4.3. Models' Quality Evaluation

A final comparison between all employed methods is shown in Figure 13, which depicts the classification accuracy obtained in the test set by the four considered U-Net models, together with the FCN and FCN + LSTM methods. Among the U-Net variants, our proposed 4D model achieves the best classification accuracy (61.56%) among all, followed by the 3D U-Net-S model (48.77%), the 3D U-Net-T model (26.52%), and finally the 2D U-Net model (22.02%). Moreover, both the FCN and FCN + LSTM outperform the lower-order U-Nets as they achieve a classification accuracy of 50.04% and 54.57%, respectively. Still, they remain at least 7% below the proposed 4D U-Net model. All in all, the designed higher-order architecture clearly outperforms all its competitors in every experimental scenario examined.

Of course, this significant performance improvement does not come for free, as shown by the model training times shown in Table 8. Training a 4D U-Net takes approximately 7 times longer than the second competing lower-order method (3D U-Net-S) and approximately 8.5 times longer than the second best competing method (FCN + LSTM). The number of parameters employed by each model are also shown in Table 8. Notice that the proposed 4D architecture contains approximately 25 times fewer parameters than the FCN + LSTM, its second best competitor. Of course the operations in higher-dimensional spaces are more time consuming. However, a smaller number of parameters to be learned implies that optimized (instead of customized) higher-order operations could drastically diminish the training-time gap between the architectures. In addition, this weakness of the proposed method could be alleviated if better hardware (e.g., GPUs) were employed. All in all, the results presented throughout the present study clearly demonstrate that the proposed higher-order U-Net architecture outperforms the lower-order counterparts as well as state-of-the-art methods in every examined case, at the cost of longer training time.

Table 8. Number of parameters and respective training times (minutes) for the best employed models. The higher-dimensional processing requires more computational time, even if this is not translated into accordingly more parameters to be learned.

Model	#Parameters	Time
2D U-Net [37]	80,699	19.32
3D U-Net-T	936,631	44.76
3D U-Net-S	350,583	76.40
FCN [39]	29,067,455	26.54
FCN + LSTM [39]	**29,099,975**	64.06
4D U-Net-Proposed	1,181,087	**542.14**

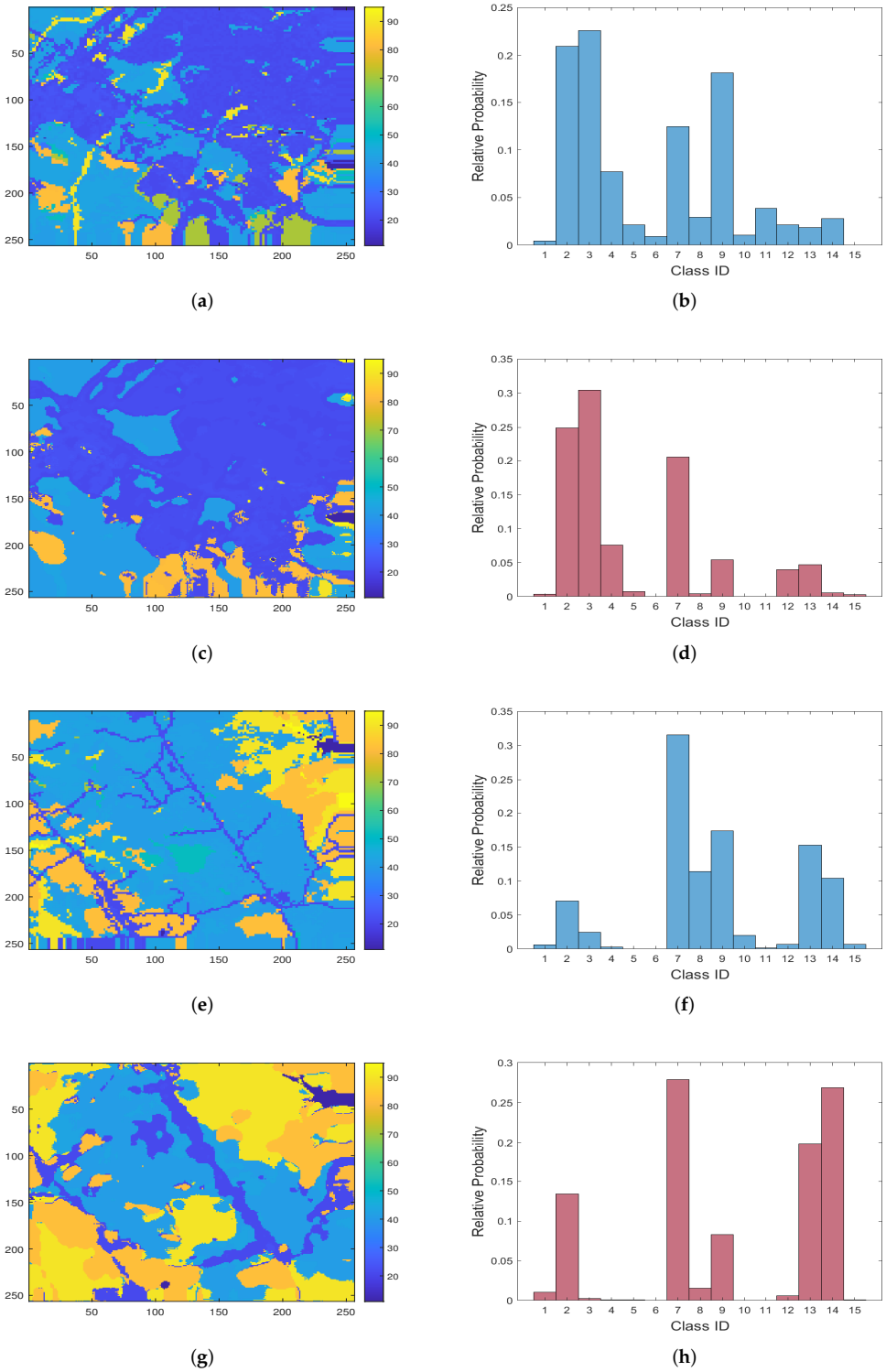


Figure 12. Actual, predicted labels, and respective histograms corresponding to two distinct test set tiles obtained via the proposed 4D U-Net model. Extremely improbable classes as well as slightly different ones lead to poor prediction accuracy. (**a**) Actual labels—Tile 3112. (**b**) Actual labels—Tile 3112-Histogram. (**c**) Predicted labels—Tile 3112-4D U-Net. (**d**) Predicted labels—Tile 3112-4D U-Net-Histogram. (**e**) Actual labels—Tile 3772. (**f**) Actual labels—Tile 3772-Histogram. (**g**) Predicted labels—Tile 3772-4D U-Net. (**h**) Predicted labels—Tile 3772-4D U-Net-Histogram.

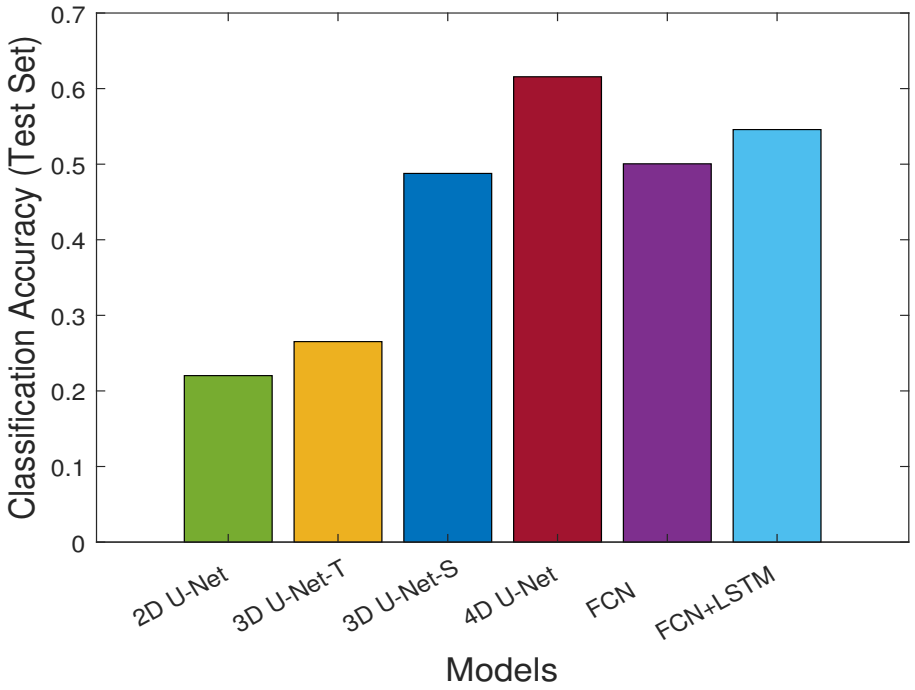


Figure 13. Classification accuracy for the best employed models. The performance improvement between the proposed 4D U-Net model and the second best lower-order 3D U-Net-S reaches 12.79%. Moreover, the proposed 4D U-Net model outperforms both FCN and FCN + LSTM by at least 6.99%.

5. Conclusions

In this work, we proposed a 4D U-Net architecture for the problem of classifying MT-RS data. The approach is based on generalizing conventional U-Net models to their high-dimensional analogues, by customizing and modifying every functionality of them in each of their paths. Furthermore, we designed, tested, and evaluated with real-world data various network configurations in order to determine the best one for the problem at hand. Based on our experimental findings on a recently released MT-RS dataset, we demonstrated that clearly improved classification accuracy, with respect to lower-order and state-of-the-art methods, is feasible if feature learning is performed in the nominal domain of the data.

Author Contributions: Conceptualization, M.G., G.T. and P.T.; methodology, M.G., G.T. and P.T.; formal analysis, M.G., G.T. and P.T.; writing—original draft preparation, M.G. and G.T.; writing—review and editing, P.T.; visualization, M.G. and G.T.; supervision, P.T.; project administration, P.T.; funding acquisition, P.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Hellenic Foundation for Research and Innovation (HFRI) and the General Secretariat for Research and Innovation (GSRI) through the HFRI Faculty under Grant 1725 V4-ICARUS and in part by the CALCHAS project (contract no. 842560) within the H2020 Framework Program of the European Commission.

Acknowledgments: The authors would like to thank the IEEE GRSS Image Analysis and Data Fusion Technical Committee and Microsoft for organizing the Data Fusion Contest.

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

Abbreviations

The following abbreviations are used in this manuscript:

RS	Remote Sensing
MT	Multi-Temporal
MS	Multi-Spectral
NDVI	Normalized Difference Vegetation Index
ML	Machine Learning
DL	Deep Learning
SVM	Support Vector Machines
kNN	k-Nearest Neighbours
DT	Decision Tree
RF	Random Forest
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
HS	Hyper-Spectral
TCNN	Temporal Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
FCN	Fully Convolutional Network
NLCD	National Land Cover Database
NAIP	National Agriculture Imagery Program
TP	True Positive
FP	False Positive
FN	False Negative
TN	True Negative

References

1. Bioucas-Dias, J.M.; Plaza, A.; Camps-Valls, G.; Scheunders, P.; Nasrabadi, N.; Chanussot, J. Hyperspectral remote sensing data analysis and future challenges. *IEEE Geosci. Remote Sens. Mag.* **2013**, *1*, 6–36. [CrossRef]
2. Wardlow, B.D.; Egbert, S.L.; Kastens, J.H. Analysis of time-series MODIS 250 m vegetation index data for crop classification in the US Central Great Plains. *Remote Sens. Environ.* **2007**, *108*, 290–310. [CrossRef]
3. Mathur, A.; Foody, G.M. Crop classification by support vector machine with intelligently selected training data for an operational application. *Int. J. Remote Sens.* **2008**, *29*, 2227–2240. [CrossRef]
4. Belgiu, M.; Csillik, O. Sentinel-2 cropland mapping using pixel-based and object-based time-weighted dynamic time warping analysis. *Remote Sens. Environ.* **2018**, *204*, 509–523. [CrossRef]
5. Gómez, C.; White, J.C.; Wulder, M.A. Optical remotely sensed time series data for land cover classification: A review. *ISPRS J. Photogramm. Remote Sens.* **2016**, *116*, 55–72. [CrossRef]
6. Xiao, X.; Boles, S.; Liu, J.; Zhuang, D.; Frolking, S.; Li, C.; Salas, W.; Moore III, B. Mapping paddy rice agriculture in southern China using multi-temporal MODIS images. *Remote Sens. Environ.* **2005**, *95*, 480–492. [CrossRef]
7. Conrad, C.; Colditz, R.R.; Dech, S.; Klein, D.; Vlek, P.L. Temporal segmentation of MODIS time series for improving crop classification in Central Asian irrigation systems. *Int. J. Remote Sens.* **2011**, *32*, 8763–8778. [CrossRef]
8. Sexton, J.O.; Urban, D.L.; Donohue, M.J.; Song, C. Long-term land cover dynamics by multi-temporal classification across the Landsat-5 record. *Remote Sens. Environ.* **2013**, *128*, 246–258. [CrossRef]
9. Xiong, J.; Thenkabail, P.S.; Gumma, M.K.; Teluguntla, P.; Poehnelt, J.; Congalton, R.G.; Yadav, K.; Thau, D. Automated cropland mapping of continental Africa using Google Earth Engine cloud computing. *ISPRS J. Photogramm. Remote Sens.* **2017**, *126*, 225–244.
10. Zhu, X.; Liu, D. Accurate mapping of forest types using dense seasonal Landsat time-series. *ISPRS J. Photogramm. Remote Sens.* **2014**, *96*, 1–11. [CrossRef]
11. Guerschman, J.P.; Paruelo, J.; Bella, C.D.; Giallorenzi, M.; Pacin, F. Land cover classification in the Argentine Pampas using multi-temporal Landsat TM data. *Int. J. Remote Sens.* **2003**, *24*, 3381–3402. [CrossRef]
12. Löw, F.; Michel, U.; Dech, S.; Conrad, C. Impact of feature selection on the accuracy and spatial uncertainty of per-field crop classification using support vector machines. *ISPRS J. Photogramm. Remote Sens.* **2013**, *85*, 102–119. [CrossRef]
13. Blanzieri, E.; Melgani, F. Nearest neighbor classification of remote sensing images with the maximal margin principle. *IEEE Trans. Geosci. Remote Sens.* **2008**, *46*, 1804–1811. [CrossRef]
14. Simonneaux, V.; Duchemin, B.; Helson, D.; Er-Raki, S.; Olioso, A.; Chehbouni, A. The use of high-resolution image time series for crop classification and evapotranspiration estimate over an irrigated area in central Morocco. *Int. J. Remote Sens.* **2008**, *29*, 95–116. [CrossRef]

15. Esch, T.; Metz, A.; Marconcini, M.; Keil, M. Combined use of multi-seasonal high and medium resolution satellite imagery for parcel-related mapping of cropland and grassland. *Int. J. Appl. Earth Obs. Geoinf.* **2014**, *28*, 230–237. [CrossRef]
16. Gebhardt, S.; Wehrmann, T.; Ruiz, M.A.M.; Maeda, P.; Bishop, J.; Schramm, M.; Kopeinig, R.; Cartus, O.; Kellndorfer, J.; Ressl, R.; et al. MAD-MEX: Automatic wall-to-wall land cover monitoring for the Mexican REDD-MRV program using all Landsat data. *Remote Sens.* **2014**, *6*, 3923–3943. [CrossRef]
17. Clark, M.L.; Aide, T.M.; Riner, G. Land change for all municipalities in Latin America and the Caribbean assessed from 250-m MODIS imagery (2001–2010). *Remote Sens. Environ.* **2012**, *126*, 84–103. [CrossRef]
18. Inglada, J.; Vincent, A.; Arias, M.; Tardy, B.; Morin, D.; Rodes, I. Operational high resolution land cover map production at the country scale using satellite image time series. *Remote Sens.* **2017**, *9*, 95. [CrossRef]
19. Tatsumi, K.; Yamashiki, Y.; Torres, M.A.C.; Taipe, C.L.R. Crop classification of upland fields using Random forest of time-series Landsat 7 ETM+ data. *Comput. Electron. Agric.* **2015**, *115*, 171–179. [CrossRef]
20. Furby, S.; Caccetta, P.; Wu, X.; Chia, J. Continental scale land cover change monitoring in Australia using Landsat imagery. In Proceedings of the Earth Conference—Studying Modeling and Sense Making of Planet Earth, Lesvos, Greece, 1–6 June 2008.
21. Murthy, C.; Raju, P.; Badrinath, K. Classification of wheat crop with multi-temporal images: Performance of maximum likelihood and artificial neural networks. *Int. J. Remote Sens.* **2003**, *24*, 4871–4890. [CrossRef]
22. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
23. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
24. Giannopoulos, M.; Aidini, A.; Pentari, A.; Fotiadou, K.; Tsakalides, P. Classification of Compressed Remote Sensing Multispectral Images via Convolutional Neural Networks. *J. Imaging* **2020**, *6*, 24. [CrossRef] [PubMed]
25. 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]
26. Castelluccio, M.; Poggi, G.; Sansone, C.; Verdoliva, L. Land use classification in remote sensing images by convolutional neural networks. *arXiv* **2015**, arXiv:1508.00092.
27. Tsagkatakis, G.; Aidini, A.; Fotiadou, K.; Giannopoulos, M.; Pentari, A.; Tsakalides, P. Survey of deep-learning approaches for remote sensing observation enhancement. *Sensors* **2019**, *19*, 3929. [CrossRef]
28. 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]
29. Ji, S.; Zhang, C.; Xu, A.; Shi, Y.; Duan, Y. 3D convolutional neural networks for crop classification with multi-temporal remote sensing images. *Remote Sens.* **2018**, *10*, 75. [CrossRef]
30. Pelletier, C.; Webb, G.I.; Petitjean, F. Temporal convolutional neural network for the classification of satellite image time series. *Remote Sens.* **2019**, *11*, 523. [CrossRef]
31. Di Mauro, N.; Vergari, A.; Basile, T.M.A.; Ventola, F.G.; Esposito, F. *End-to-End Learning of Deep Spatio-Temporal Representations for Satellite Image Time Series Classification*; DC@ PKDD/ECML: Bilbao, Spain, 2017.
32. Zhong, L.; Hu, L.; Zhou, H. Deep learning based multi-temporal crop classification. *Remote Sens. Environ.* **2019**, *221*, 430–443. [CrossRef]
33. Garnot, V.S.F.; Landrieu, L.; Giordano, S.; Chehata, N. Time-Space tradeoff in deep learning models for crop classification on satellite multi-spectral image time series. In Proceedings of the IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium, Yokohama, Japan, 28 July–2 August 2019; IEEE: New York, NY, USA, 2019; pp. 6247–6250.
34. Rußwurm, M.; Körner, M. Multi-temporal land cover classification with sequential recurrent encoders. *ISPRS Int. J. -Geo-Inf.* **2018**, *7*, 129. [CrossRef]
35. Bozo, M.; Aptoula, E.; Çataltepe, Z. A Discriminative Long Short Term Memory Network with Metric Learning Applied to Multispectral Time Series Classification. *J. Imaging* **2020**, *6*, 68. [CrossRef] [PubMed]
36. Rußwurm, M.; Körner, M. Convolutional LSTMs for cloud-robust segmentation of remote sensing imagery. *arXiv* **2018**, arXiv:1811.02471.
37. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 234–241.
38. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Hong Kong, China, 13–16 January 2015; pp. 3431–3440.
39. Sefrin, O.; Riese, F.M.; Keller, S. Deep Learning for Land Cover Change Detection. *Remote Sens.* **2021**, *13*, 78. [CrossRef]
40. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
41. Myronenko, A.; Yang, D.; Buch, V.; Xu, D.; Ihsani, A.; Doyle, S.; Michalski, M.; Tenenholtz, N.; Roth, H. 4D CNN for semantic segmentation of cardiac volumetric sequences. In Proceedings of the International Workshop on Statistical Atlases and Computational Models of the Heart, Lima, Peru, 4 October 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 72–80.
42. Çiçek, Ö.; Abdulkadir, A.; Lienkamp, S.S.; Brox, T.; Ronneberger, O. 3D U-Net: Learning dense volumetric segmentation from sparse annotation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Athens, Greece, 17–21 October 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 424–432.

43. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
44. Zhang, S.; Guo, S.; Huang, W.; Scott, M.R.; Wang, L. V4D: 4D Convolutional neural networks for video-level representation learning. *arXiv* **2020**, arXiv:2002.07442.
45. Chollet, F. *Deep Learning with Python*; Manning: New York, NY, USA, 2018; Volume 361.
46. Dumoulin, V.; Visin, F. A guide to convolution arithmetic for deep learning. *arXiv* **2016**, arXiv:1603.07285.
47. Giang, T.L.; Dang, K.B.; Le, Q.T.; Nguyen, V.G.; Tong, S.S.; Pham, V.M. U-Net convolutional networks for mining land cover classification based on high-resolution UAV imagery. *IEEE Access* **2020**, *8*, 186257–186273. [CrossRef]
48. Zhang, P.; Ke, Y.; Zhang, Z.; Wang, M.; Li, P.; Zhang, S. Urban land use and land cover classification using novel deep learning models based on high spatial resolution satellite imagery. *Sensors* **2018**, *18*, 3717. [CrossRef]
49. Li, R.; Zheng, S.; Duan, C.; Zhang, C. Land cover classification from remote sensing images based on multi-scale fully convolutional network. *arXiv* **2020**, arXiv:2008.00168.
50. Stoian, A.; Poulain, V.; Inglada, J.; Poughon, V.; Derksen, D. Land cover maps production with high resolution satellite image time series and convolutional neural networks: Adaptations and limits for operational systems. *Remote Sens.* **2019**, *11*, 1986. [CrossRef]
51. Rakhlin, A.; Davydow, A.; Nikolenko, S. Land cover classification from satellite imagery with u-net and lovász-softmax loss. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Salt Lake City, UT, USA, 18–22 June 2018; pp. 262–266.
52. 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. JMLR Workshop and Conference Proceedings, Haifa, Israel, 25 June 2010; pp. 249–256.
53. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
54. Chollet, F. Keras. Available online: https://keras.io (accessed on 13 September 2021).
55. Grandini, M.; Bagli, E.; Visani, G. Metrics for multi-class classification: An overview. *arXiv* **2020**, arXiv:2008.05756.