remote sensing

MDPI

Article

Adaptable Convolutional Network for Hyperspectral Image Classification

Mercedes E. Paoletti [1,*] and Juan M. Haut [2]

[1] Department of Computer Architecture and Automation, Faculty of Computer Science, Complutense University of Madrid, 28040 Madrid, Spain
[2] Hyperspectral Computing Laboratory (HyperComp), Department of Computer Technology and Communications, Escuela Politecnica, University of Extremadura, 10002 Caceres, Spain; juanmariohaut@unex.es
* Correspondence: mpaolett@ucm.es; Tel.: +34-927-257-000

Abstract: Nowadays, a large number of remote sensing instruments are providing a massive amount of data within the frame of different Earth Observation missions. These instruments are characterized by the wide variety of data they can collect, as well as the impressive volume of data and the speed at which it is acquired. In this sense, hyperspectral imaging data has certain properties that make it difficult to process, such as its large spectral dimension coupled with problematic data variability. To overcome these challenges, convolutional neural networks have been proposed as classification models because of their ability to extract relevant spectral–spatial features and learn hidden patterns, along their great architectural flexibility. Their high performance relies on the convolution kernels to exploit the spatial relationships. Thus, filter design is crucial for the correct performance of models. Nevertheless, hyperspectral data may contain objects with different shapes and orientations, preventing filters from "seeing everything possible" during the decision making. To overcome this limitation, this paper proposes a novel adaptable convolution model based on deforming kernels combined with deforming convolution layers to fit their effective receptive field to the input data. The proposed adaptable convolutional network (named DKDCNet) has been evaluated over two well-known hyperspectral scenes, demonstrating that it is able to achieve better results than traditional strategies with similar computational cost for HSI classification.

Keywords: deep learning (DL); hyperspectral images (HSIs); convolutional neural networks (CNNs); receptive field (RF); deformable kernel (DK); deformable convolution (DC)

Citation: Paoletti, M.E.; Haut, J.M. Adaptable Convolutional Network for Hyperspectral Image Classification. *Remote Sens.* **2021**, *13*, 3637. https://doi.org/10.3390/rs13183637

Academic Editor: Gwanggil Jeon

Received: 13 August 2021
Accepted: 10 September 2021
Published: 11 September 2021

1. Introduction

Within the remote sensing (RS) field, major advances in spectroscopy and imaging technologies since the first approaches in 1979 have enriched the knowledge gained about the Earth's surface, significantly enhancing our capacity to explore and exploit terrestrial resources [1]. As a consequence, hyperspectral imaging (HSI) has become one of the foremost data acquisition methods within RS, as it captures the spatial distribution of large observation areas whilst differentiating the solar absorption features of the different materials contained in the target scene. Indeed, HSI facilitates the acquisition of detailed spectral information, gathering large datacubes composed of a mesh of $N_1 \times N_2$ pixels, which collect in C bands of the solar radiation reflected by materials along the visible-to-shortwave infrared (VSWIR) regions of the electromagnetic spectrum. As a result, each hyperspectral pixel contains the continuous radiance spectrum (known as spectral signature) of a particular material, in the form of peaks and valleys that capture the physico-chemical behavior of the material at different wavelengths [2]. The large amount of rich spectral information provides a detailed characterization of the land cover, as each signature is unique for each type of material. Hence, HSI data are very useful in a wide variety of fields with a high economic and social impact, such as: climate change study; crop management,

which ranges from pest and plague detection to the prevention and recovery of degraded and eroded soils; forestry; mineral identifications and resource exploitation; urban planning and land-use/land-cover (LULC) classification; military, defense and intelligence applications; risk prevention; and disaster monitoring, among many others [3–7].

Nowadays, HSI data analysis is considered a hot topic due to the great relevance of this data in RS and its consequent impact on the applications mentioned above. A multitude of methods have been implemented to efficiently process and extract the relevant spectral–spatial information contained in HSI cubes. Nevertheless, these methods must cope with several challenges, drawbacks and limitations that characterize this kind of data. For instance, there is extensive literature on unmixing methods that attempt to separate the spectral signatures contained in the scene by searching for pure or almost-pure pixels (endmembers) and measuring the spectral mixture and composition of the remaining non-pure pixels [8,9]. Indeed, spectrometers often exhibit a low spatial resolution due to instrument limitations and the restriction imposed by their high data rate, so the compactness of channels results in the limitation of pixel size. In addition, other uncontrolled phenomena may degrade and introduce non-desired artifacts within the data obtained [10,11], for instance limitations in the radiometric and geometric resolution of the sensor, atmospheric scattering, lighting disturbances (such as light obstruction by aerosols and clouds), sensor noise or imperfect imaging optics hinder the data acquisition, obtaining noisy images in the end with an important number of mixed pixels that greatly degrade the detection and recognition performance, preventing the material identification.

The resulting high data variability is compounded by the lack of labeled samples. Indeed, although it is relatively easy to collect HSI data, obtaining image-like label information is utterly arduous [12]. These challenges severely impair classification algorithms, which analyze the HSI data distribution in order to identify the land cover contained in each pixel, assigning them the corresponding label. Traditionally, HSI classification methods implemented isolated pixel analysis strategies through non-guided analysis of data divergence and similarity-based measurements (unsupervised methods) [13–15] or guided training with known ground-truth information (supervised classifiers) [16–19]. More recently, semi-supervised and hybrid strategies are gaining popularity to deal with the lack of data problem, estimating the distribution of unlabeled data from a very small pool of labeled data (semi-supervised models) [20–22]. Among these methods, the support vector machine (SVM) stands out for its efficiency in processing large data, it being relatively easy to include kernels to adjust and improve its performance in the data [23,24]. Other widely used methods are: probabilistic methods, such as multinomial logistic regression (MLR) because of its outstanding results when few training samples are provided [25,26]; ensemble methods such as decision trees (DTs) and random forests (RaF) [27,28]; and iterative classifiers, such as multilayer perceptrons (MLP) and other shallow/deep artificial neural networks (ANNs), which do not require prior information of the data distribution [29]. In fact, the great generalization capability of ANNs, their great architectural flexibility and their impressive ability to extract abstract features and relationships from raw data, i.e., without the need for handcrafted features, have made them very popular classifiers within HSI data processing, achieving a never-before-seen performance on current HSI benchmarks [30,31].

Nevertheless, learning based on isolated spectrum analysis is not stable due to the large spectral mixture of the HSI data, which is mainly introduced by the low spatial resolution. In this context, some works introduce spatial information in order to eliminate the classification uncertainty by stating that neighboring pixels must belong to the same class. This mitigates the spectral mixture, as more spectral signatures are introduced to support the classification as well as spatial information in the form of spatial patterns such as shapes, edges or textures. As a result, many classification algorithms have been implemented to process the spectral–spatial information contained in HSI data [32–34]. These algorithms may usually be divided into those that concatenate the spectra of several pixels, applying or not a band reduction/selection step in order to process them all together,

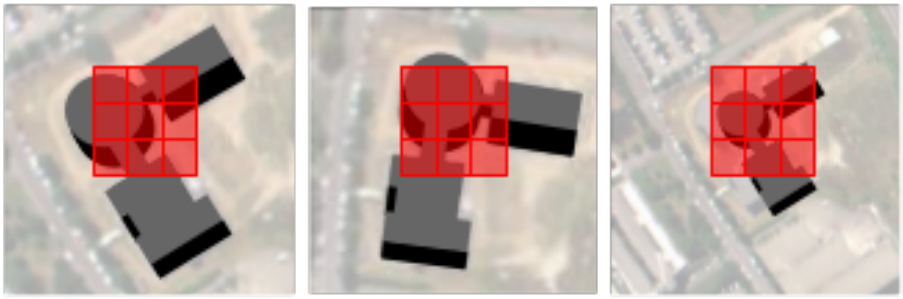
and those that extract shape and texture features, such as morphological profiles (MP) [35] or global Gabor features [36]. Despite their accurate results, many of these methods are based on a feature vector, vectorizing the original 3D data in order to process it. In this sense, the spatial and structural relationships of the data are disregarded [37]. In contrast with these methods, the convolutional neural network (CNN) is based on multidimensional kernels that process the spectral-spatial information of the HSI data without flattening the data. Indeed, the convolutional model is designed to process multi-dimensional input samples, respecting the original distribution. This is performed using kernels locally connected to small regions of the input data. These kernels are defined as arrays with one dimension more than the input data, thus indicating, on the one hand, the number of filters to be applied and, on the other hand, the size of the region where they will be applied. Hence, different kernels can be defined, such as: 1D kernels, which are applied locally to several channels of the input data; 2D kernels, which are locally applied to areas of the input maps, and 3D kernels, which are locally applied to both areas and channels of the input maps. This provides different ways to process HSI data, where 2D and 3D kernels are the most widely used because of the spectral–spatial processing they conduct [31].

1.1. The Rigid Kernel: Limitation of CNNs for Spatial Analysis

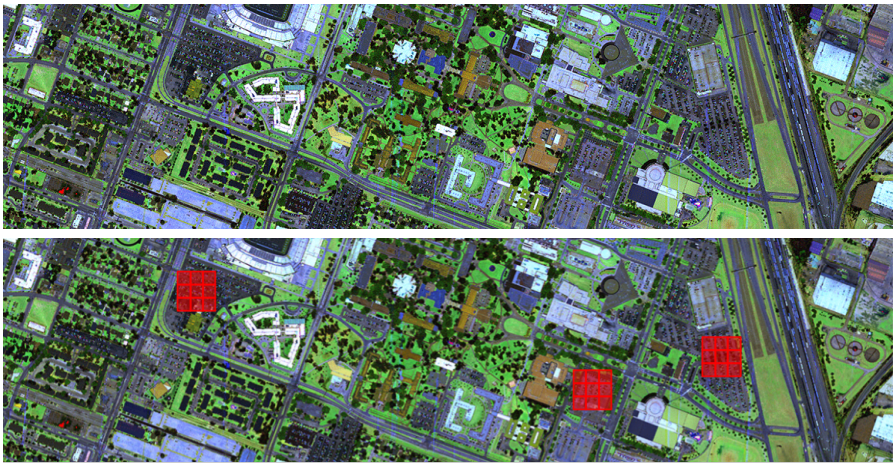
It is noteworthy that ANNs (in general) and CNNs (in particular) try to model the visual cortex by creating a deep stack of layers composed of neurons. These layers are connected by weights that modify the inputs to extract their features. As a result, each layer produces a feature representation, which is increasingly abstract and tailored to the task for which the network is designed. In this way, by learning the weights that best fit the data, they are able to extract the best data representations to enhance the classification performance [38]. Focusing on HSI data processing, after the success of the first attempts [39], a vast amount of deep learning (DL) approaches have been used on HSI processing [31]. Particularly in the case of CNNs, the flexibility of the convolution kernel provides a natural processing as it is applied as a sliding and locally-connected multidimensional-array to the input data. In addition, the backpropagation mechanism adjusts the kernel weights, creating a filter adapted to the data to automatically extract the most discriminative features. Hence, the network extracts abstract representations tailored to the classification task, obtaining very accurate results. This behaviour has positioned CNN as the current state-of-the-art in HSI analysis, and there is a vast literature of adapted and improved HSI-CNNs [2,40–49].

Nevertheless and despite its impressive results, the CNN has some drawbacks related to its training procedure and its tendency to overfit. Indeed, the CNN consumes a large amount of labeled samples in order to adjust its parameters properly and to be effective in generalizing to unseen data, i.e., the CNN requires a representative training set to cover the high data variability and its classification results tend to degrade in the absence of labeled samples. Recently, more complex and deeper models have been developed to improve the performance of the HSI classification process, introducing skip connections that reuse extracted data representations at different levels of the architecture in order to avoid overfitting and vanishing gradient problems, such as residual models (ResNets) [50,51], developing dual-path models to enhance the feature extraction [52] or introducing new architectures based on capsule networks (CapsNets) to process the spatial relationships of extracted features [53,54]. Despite these improvements, the CNN still struggles with a significant loss of information caused by the rigid structure of the convolution kernels. Inspired by the visual cortex, the multidimensional array that composes the kernel is organized as a regular square mesh, which is overlapped over the data, applied as an affine transformation, and sliding by means of a stride hyperparameter. In this regard, the structure of the kernel has a decisive impact on how the input data affect the output features, and hence the final classification. This is due to the dependencies between the generated output feature and a certain input region, known as the receptive field (RF), i.e., the size of the input region that has produced the output feature. Therefore,

the receptive field depends on the size and shape of the convolution kernel. Due to the their rigid structure, convolution kernels suffer a considerable loss of information, as the objects located in remotely sensed images do not usually have regular shapes and may appear different in size and orientation, as is observed in Figure 1. Although data augmenting and multiscale-based strategies [55,56] have traditionally been employed to overcome this limitation, their kernels exhibit the same restraint, i.e., they cannot be adaptively tailored with respect to the spatial features.



(**a**) Spatial transformations of a building (University of Pavia).



(**b**) Different parking lot areas (University of Houston).

Figure 1. Often, objects located in remotely sensed hyperspectral images do not have regular shapes and may appear with different sizes and orientations. For instance, some data augmentation methods apply spatial transformations to the input samples to increase the size of the datasets and cover the variability in the data (**a**), whilst other scenes contain the same elements with different orientations and poses; for example, in (**b**) we can observe several parking areas, with different sizes and where the cars are placed differently. However, the convolution kernel disregards these spatial variations, defining a rigid regular mesh.

Some strategies have been implemented to face the limitations of the rigid kernel. For instance, Zhu et al. [57] proposed the deformable HSI classification network (DHCNet), where deformable convolutional sampling locations are introduced by applying 2D offsets to the input features. As a result, deformable feature images are obtained, which adaptively combine the similar neighboring information of each pixel into fixed grids that will be employed by standard convolution layers. However, this proposal does not control the impact that each item of the fixed grid has on the outputs due to the intrinsic behavior of the convolution RFs, where central items exhibit a higher impact than those items in the outer area of the RF, following a Gaussian distribution. In this sense, there is a much smaller effective area (ERF) within the theoretical RF (TRF) [58], whose information flow is crucial for the model performance.

1.2. Contributions

This paper attempts to resolve the aforementioned limitation by adjusting the ERF to the HSI data through deformable kernels (DKs) [59], which adapt the kernel spaces leaving the data intact, instead of reconfiguring the data to meet geometric deformations. In this sense, not the input features (which remain untouched), but the kernel itself is modified by means of offsets to produce the deformation. Moreover, to enhance the data deformation modeling, deformable convolution (DC) [60] is combined with DK. The resulting model is an adaptable convolutional network, which adapts the application of each convolutional kernel to the features of the input data by deforming its kernel space. In this context, the main contributions of this work are:

1. A strategy based on DKs is proposed to train CNNs for HSI data classification tasks in an effective way. Moreover, CNN with DKDC (DKDCNet) has been implemented and tested over two widely-used HSI scenes.
2. An experimental section is provided to compare the performance of KDCNet with common DL-based classification models. Conducted experiments demonstrate the DKDCNet can outperform the classification performance achieved by popular DL-based classification models.

The rest of this work is organized as follows. Section 2 discusses the role of the RF, providing the details of the deformable kernel method. Section 3 describes the experimental results, evaluating the classification performance exhibited by the implemented DKDCNet over two well-known HSI datasets. Finally, Section 4 presents the obtained conclusions and future work.

2. Methodology

2.1. The Receptive Field

Traditionally, an ANN defines an architecture of interconnected neurons that communicate with each other through weighted connections. However, this fully-connected (FC) architecture presents severe limitations in image processing, involving an expensive computational burden coupled with severe overfitting due to a large number of parameters. On the contrary, CNNs have adapted their behaviour to mimic the visual cortex, where output neurons are activated by some input visual stimuli that fall within their RF, following a hierarchical organization where deeper neurons are activated by combined and more complex stimuli. In this context, the visual cortex behaviour is artificially implemented by the CNN through a deep stack of convolution layers, where each layer defines a filter bank, i.e., a set of learnable, locally-connected and shared weights that composes a linear n-dimensional kernel. Indeed, the kernel is basically a group of data-fitted filters which slide over the input data following a stride, overlapping and applying themselves to the data. Each application produces a weighted sum between the filter weights and the inputs falling within the application region. Moreover, in order to determine the existence of the features they filter (such as edges and shapes), a non-linear activation function is introduced to encode the neural response of the convolution layer to the seek features. Mathematically, this behavior is defined by Equation (1):

$$\mathbf{X}_{\mathbf{j}}^{(l)} = \mathcal{H}\left(\sum_{\mathbf{k}\in\mathcal{K}} \mathbf{X}_{\mathbf{j}+\mathbf{k}}^{(l-1)} \mathbf{W}_{\mathbf{k}}^{(l)}\right), \tag{1}$$

where the superscript l indicates the l-th layer of a CNN. Hence, $\mathbf{X}^{(l-1)} \in \mathbb{R}^{P\times P\times K^{(l-1)}}$ and $\mathbf{X}^{(l)} \in \mathbb{R}^{Q\times Q\times K^{(l)}}$ denote the input and output data, also known as input and output feature maps, respectively, where the former comprises $K^{(l-1)}$ maps of $P \times P$ features (pixels) and the latter $K^{(l)}$ maps of $Q \times Q$ features. $\mathbf{W}^{(l)} \in \mathbb{R}^{K^{(l-1)}\times M\times M\times K^{(l)}}$ denotes the convolution weights of the l-th layer (to simplify the notation, the bias is ignored), composed of $K^{(l)}$ filters of $M \times M$ weights. Coordinate $\mathbf{k}$ enumerates discrete kernel positions at the l-th layer within the support of the regular mesh $\mathcal{K} = [-M/2, M/2]^2 \cap \mathbb{Z}$ (where $\mathbb{Z}$ denotes

the set of integers), whilst $\mathbf{j}$ is indexing each output coordinate. Finally, $\mathcal{H}$ defines the activation function, which is usually implemented with ReLU. This is graphically depicted by Figure 2. Particularly, Figure 2b depicts a 3×3 kernel which is applied with stride 1 and zero padding over a 6×6 input feature map. In this context, for the output coordinate $\mathbf{j} = (3, 5, z)$, $\mathbf{k}$ will sample the input features positions from $(2, 4)$ to $(4, 6)$ (using part of the padding added to the map border) according to the grid based on $\mathcal{K} = [-1, 1]$.

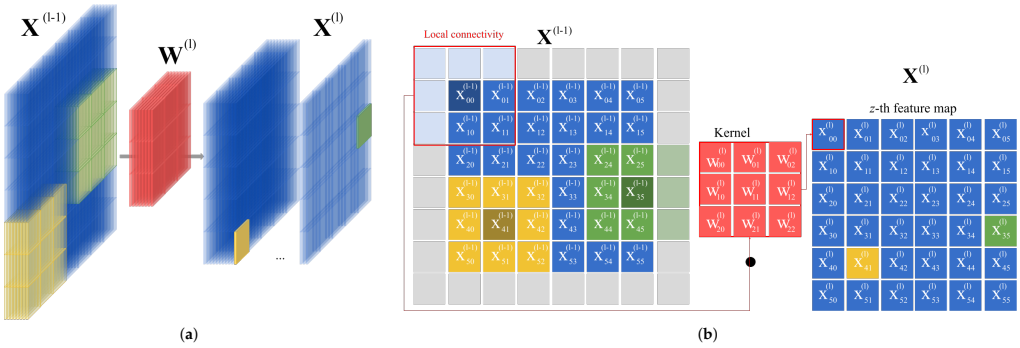


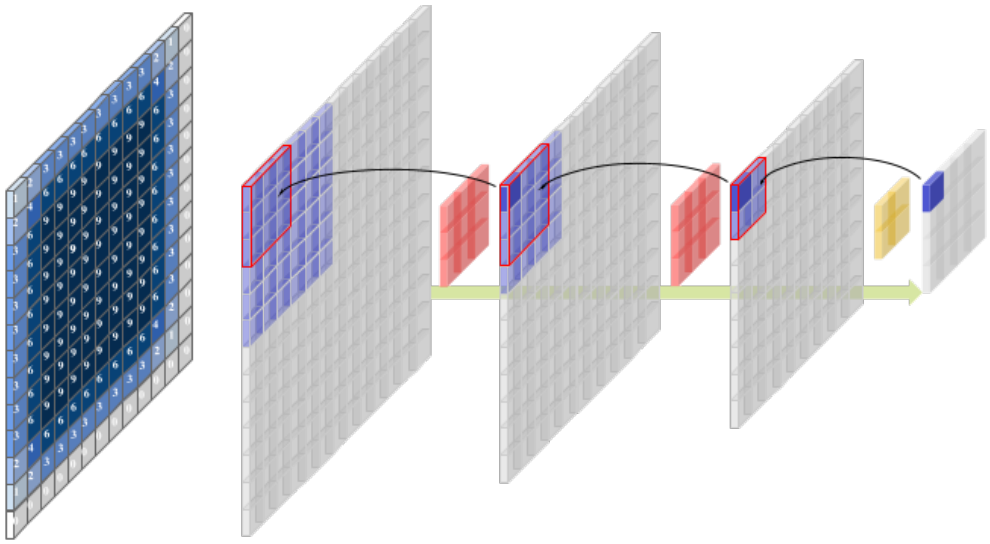
Figure 2. Graphical representation of a convolution kernel. (**a**) The multidimensional array of weights $\mathbf{W}^{(l)}$ (in red) is slid and locally applied over regions on the input feature maps $\mathbf{X}^{(l-1)}$ (highlighted in green and yellow), obtaining the corresponding output features in $\mathbf{X}^{(l)}$ as a weighted sum. (**b**) 2D detail of a 3×3 kernel with stride 1 and zero padding. Index l denotes the l-th layer, subscripts $i, j \in [0, 5]$ represent the spatial coordinates of the features, whilst z indicates the feature.

As we can observe, the output unit $\mathbf{X}_{\mathbf{i}}^{(l)}$ only depends on a small region of the input feature map that the kernel is "looking" at. This region is denoted as the RF for that unit, so any information within the input feature map but outside the RF will not affect the value of the output unit [61]. Figure 3a represents the number of times that a 3×3 kernel has been applied on a 13×13 feature map with stride 1, i.e., the overlapping of all RFs of the output features. In this sense, the central pixels are used more often than the pixels located at the edges of the map, i.e., central RF pixels are part of more calculations, playing a major role within the convolution layer as they propagate their information more times. This follows the behavior of the visual cortex, where local features are detected and processed in early visual stages (or layers) of the cortex and then progressively combined in a hierarchical fashion, creating more complex patterns [62,63]. The connotations behind this behaviour are quite intuitive: the pixels in the RF do not contribute equally to the layer output response [58]. There is an *effective receptive field* (ERF) that defines the areas of the input data that influence and modulate the output activations by tracing the hierarchy back from the considered output feature to the input image. This extends to the entire network; for instance, Figure 3b highlights the RF of a CNN output response relative to the input data. In this sense, the ERF of a CNN follows a Gaussian distribution, defining not only a region to "look at", but also exponentially concentrating the attention on the middle of the feature map. Indeed, it can be identified as a Gaussian-based soft attention map [59,64].

Following Equation (1), the computation of the output coordinate $\mathbf{j}$ within the final (L-th) layer can be generalized by Equation (2) as follows ($\mathcal{H}$ is omitted to simplify the formulation):

$$\begin{aligned} \mathbf{X}_{\mathbf{j}}^{(L)} = \sum_{\mathbf{k}_L \in \mathcal{K}} \mathbf{X}_{\mathbf{j}+\mathbf{k}_L}^{(L-1)} \mathbf{W}_{\mathbf{k}_L}^{(L)} &= \sum_{(\mathbf{k}_{L-1}, \mathbf{k}_L) \in \mathcal{K}^2} \mathbf{X}_{\mathbf{j}+\mathbf{k}_{L-1}+\mathbf{k}_L}^{(L-2)} \mathbf{W}_{\mathbf{k}_L}^{(L)} \mathbf{W}_{\mathbf{k}_{L-1}}^{(L-1)} \\ &= \sum_{(\mathbf{k}_1, \ldots, \mathbf{k}_L) \in \mathcal{K}^L} \left(\mathbf{X}_{\mathbf{j}+\sum_{l=1}^{L} \mathbf{k}_l} \prod_{l=1}^{L} \mathbf{W}_{\mathbf{k}_l}^{(l)} \right), \end{aligned} \tag{2}$$

where $\mathbf{X}$ is the original HSI input and $\mathbf{k}_l$ the discrete kernel positions related to the kernel weights $\mathbf{W}^{(l)}$ at the l-th convolution layer. In this regard, the ERF value of an output coordinate $\mathbf{j}$ that depends on input $\mathbf{i}$ can be defined as $\mathcal{R}^{(L)}(\mathbf{i}; \mathbf{j})$:



(a) **(b)**

Figure 3. Graphical representation of the receptive field. (**a**) Considers a unique input layer to which a 3×3 kernel is applied with a stride of 1. The color map indicates the number of times the kernel has been applied on each feature (i.e., the darker the color, the more kernel applications). The number of times the kernel has been applied over a feature is also included. For instance, the edge pixels are used between 0 and 3 times, while the center pixels are used 9 times. (**b**) Receptive field of an output feature obtained by a CNN composed of two 3×3 convolution layers (in red) and a 2×2 pooling layer (in yellow).

$$\mathcal{R}^{(L)}(\mathbf{i};\mathbf{j}) \equiv \partial \mathbf{X}_{\mathbf{j}}^{(L)} / \partial \mathbf{X}_{\mathbf{i}} = \sum_{(\mathbf{k}_1,\ldots,\mathbf{k}_L)\in\mathcal{K}^L} \left(\mathbb{1}\left[\mathbf{j} + \sum_{l=1}^{L} \mathbf{k}_l = \mathbf{i} \right] \prod_{l=1}^{L} \mathbf{W}_{\mathbf{k}_l}^{(l)} \right), \tag{3}$$

where $\mathbb{1}$ defines the indicator function. As we can observe, the ERF depends on the data and kernel sampling locations $\mathbf{j}$ and $\mathbf{k}$, and the kernel weight arrays $\{\mathbf{W}^{(l)}\}_{l=1}^{L}$.

In addition, if the m-th convolution kernel $\mathbf{W}^{(m)}$ is implemented by a 1×1 kernel of a single learnable parameter at position $\tilde{\mathbf{k}}_m$, i.e., $\mathbf{W}_{\tilde{\mathbf{k}}_m}^{(m)}$, the ERF can be obtained by extracting the weight of the summation and multiplying it directly to the remaining kernels, following Equation (4):

$$\mathcal{R}^{(l)}(\mathbf{i};\mathbf{j},\tilde{\mathbf{k}}_m) = \sum_{(\mathbf{k}_1,\ldots,\mathbf{k}_{m-1},\mathbf{k}_{m+1},\ldots\mathbf{k}_L)\in\mathcal{K}^{L-1}} \left(\mathbb{1}\left[\mathbf{j} + \sum_{l=\mathcal{S}} \mathbf{k}_l = \mathbf{i} \right] \prod_{l=\mathcal{S}} \mathbf{W}_{\mathbf{k}_l}^{(l)} \mathbf{W}_{\tilde{\mathbf{k}}_m}^{(m)} \right), \tag{4}$$

where $\mathcal{S} = [1, L] \setminus m$ contains all the layers of the CNN architecture except the m-th layer. Furthermore, any $M \times M$ kernel can be considered as a composition of M^2 point-wise 1×1 kernels [59], hence Equation (3) can be reformulated as:

$$\mathcal{R}^{(l)}(\mathbf{i};\mathbf{j}) = \sum_{\mathbf{k}_l\in\mathcal{K}} \mathcal{R}^{(l)}(\mathbf{i};\mathbf{j}+\mathbf{k}_l,\mathbf{k}_l). \tag{5}$$

In this regard, although the ERF can be modified in several ways, for instance by stacking more or less convolution layers, changing the kernel size M, implementing sub-sampling stages through pooling layers and striding, or including dilated convolutions and multiple data paths, the rigid shape of the convolution kernel imposes a serious limitation, as it applies a regular mesh regardless of the spatial characteristics of the elements contained in the HSI image. In this context, the convolution kernel must be properly controlled to ensure that the entire relevant region is covered, dealing with data deformations. Taking Equation (5) into account, we automatically adapt the ERF to

spatially fit HSI data, developing a deformable kernel (DK)-based architecture [59] for spectral–spatial HSI analysis.

2.2. Adaptable Deep Model with Deformable Kernels and Deformable Convolution

The strategy for adapting the ERF to geometric deformations and transformations of the data is quite simple. Indeed, the DK method takes the layer input features $\mathbf{X}^{(l-1)}$ and automatically generates a group of kernel offset $\Delta\mathbf{k}_l$. Then, taking the original kernel weights $\mathbf{W}^{(l)}$ with a certain size $M' \times M'$, the DK method performs an interpolation between them and the generated offsets $\Delta\mathbf{k}_l$, obtaining as a result the desired kernel $\tilde{\mathbf{W}}^{(l)}$ with size $M \times M$. This final kernel is convolved with the input data $\mathbf{X}^{(l-1)}$ in order to obtain the output feature maps. The procedure is detailed below.

The DK method attempts to adapt the computation of how much each input pixel contributes to the final output response directly during runtime. In this context, its purpose is to modify the kernel space by resampling the kernels weights, adapting them to the input data but leaving the data untouched. As a result, Equations (2) and (5) are re-interpreted as:

$$\mathbf{X}_{\mathbf{j}}^{(l)} = \sum_{\mathbf{k}_l \in \mathcal{K}} \mathbf{X}_{\mathbf{j}+\mathbf{k}_l}^{(l-1)} \underbrace{\mathbf{W}_{\mathbf{k}_l+\Delta\mathbf{k}_l}^{(l)}}_{\text{Interpolation through } \mathcal{B}} = \sum_{\mathbf{k}_l \in \mathcal{K}} \mathbf{X}_{\mathbf{j}+\mathbf{k}_l}^{(l-1)} \underbrace{\tilde{\mathbf{W}}_{\mathbf{k}_L}^{(l)}}_{\text{Defromable kernel}} \tag{6a}$$

$$\mathcal{R}^{(l)}(\mathbf{i};\mathbf{j}) = \sum_{\mathbf{k}_l \in \mathcal{K}} \mathcal{R}^{(l)}(\mathbf{i};\mathbf{j}+\mathbf{k}_l, \mathbf{k}_l + \Delta\mathbf{k}_l). \tag{6b}$$

Focusing on Equation (6a), the DK can be addressed as a bilinear interpolation problem within the rigid kernel grid and the desired adaptive/deformable kernel.

Denoting $\mathbf{W}^{(l)} \in \mathbb{R}^{M' \times M'}$ as the original kernel weights and $\tilde{\mathbf{W}}^{(l)} \in \mathbb{R}^{M \times M}$ as the desired deformable kernel of the l-th layer (where the M' is any *kernel scope size* that meet the restriction of the number of sampling locations M^2 determined by M), the DK conducts a sub-pixel sampling in the kernel space to handle data deformation by resampling the original weights into $\tilde{\mathbf{W}}^{(l)}$ through a group of *learned* kernel offsets $\Delta\mathbf{k}_l$. These offsets are automatically computed from the input data $\mathbf{X}^{(l-1)}$ through a generator $\mathcal{G}$; therefore, they are automatically adapted to the characteristics of the input data. It should be noted that local generators have been adopted to build the DK method. In this sense, each $\mathcal{G}^{(l)}$ has been implemented by a lightweight rigid convolution layer with the same configuration as the target kernel $\tilde{\mathbf{W}}^{(l)}$ and $2M^2$ output channels. Moreover, each offset of $\Delta\mathbf{k}_l$ is indexed by output locations $\mathbf{j}$. Knowing the original kernel weights $\mathbf{W}^{(l)}$ and the offsets $\{\Delta\mathbf{k}_l\}$, the DK samples a new set of kernel weights $\tilde{\mathbf{W}}^{(l)}$ employing a bilinear sampler $\mathcal{B}$ following Equation (7):

$$\tilde{\mathbf{W}}^{(l)} \equiv \mathbf{W}_{\mathbf{k}_l+\Delta\mathbf{k}_l}^{(l)} = \sum_{\mathbf{k}'_l \in \mathcal{K}} \mathcal{B}\left(\mathbf{k}_l + \Delta\mathbf{k}_l, \mathbf{k}'_l\right) \mathbf{W}_{\mathbf{k}'_l}^{(l)}, \tag{7}$$

where $\mathcal{B} = \max(0, 1 - |k_{l,x} + \Delta k_{l,x} - k'_{l,x}|) \max(0, 1 - |k_{l,y} + \Delta k_{l,y} - k'_{l,y}|)$. Finally, the obtained $\tilde{\mathbf{W}}^{(l)}$ is applied over the input volume following Equation (1).

In contrast to other methods, such as the DC which resamples the features to counter the effects of geometric data deformation [57,60], the application of the kernel offsets $\Delta\mathbf{k}_l$ to each kernel location $\mathbf{k}_l$ and deforms the rigid grid imposed by $\mathcal{K}$, modifying the kernel space but keeping the data untouched, which preserves the input information intact. In this sense, DK and DC are perfectly compatible methods that may be combined to enhance the feature extraction. Therefore, with the aim of completing the data deformation modeling, DK is combined with DC to implement the proposed adaptable model for HSI data classification, which have been denoted as DKDCNet. In this sense, the DC learns

a group of data offsets $\{\Delta\mathbf{j}\}$ with respect to data positions $\mathbf{j}$. DC is combined following Equation (8):

$$\mathbf{X}_{\mathbf{j}}^{(l)} = \sum_{\mathbf{k}_l \in \mathcal{K}} \mathbf{X}_{\mathbf{j}+\mathbf{k}_l+\Delta\mathbf{j}}^{(l-1)} \mathbf{W}_{\mathbf{k}_l+\Delta\mathbf{k}_l}^{(l)}, \tag{8a}$$

$$\mathcal{R}^{(l)}(\mathbf{i};\mathbf{j}) = \sum_{\mathbf{k}_l \in \mathcal{K}} \mathcal{R}^{(l)}(\mathbf{i};\mathbf{j}+\mathbf{k}_l+\Delta\mathbf{j}_l, \mathbf{k}_l+\Delta\mathbf{k}_l). \tag{8b}$$

As can be observed, the DC method introduces a learnable 2D offset $\Delta\mathbf{j}$ to the regular grid, sampling the positions of the input feature map, thereby deforming the ERF of the activation unit. Similarly to $\Delta\mathbf{k}_l$, the obtained $\Delta\mathbf{j}$ offsets are learned from the input feature maps using additional lightweight convolutional layers. As a result, the data is first deformed and then the deformable kernel is obtained by the DK method and applied to the feature maps. In this sense, the deformations applied within each DKDC block depend on the input features in a local and adaptive manner.

Figure 4 provides the graphical representation of the implemented adaptable network, detailing the DK computation. As we can observe, the developed model computes three principal components from the original HSI datacube by PCA, obtaining $25 \times 25 \times 3$ patches from each HSI pixel to encode the most relevant spectral–spatial information whilst reducing the dimensionality. These patches are sent to the adaptable DKDCNet, whose architecture is detailed in Table 1, following [57]. In particular, the implemented DKDCNet defines a convolution block with DK and DC mechanisms (denoted as *DKDC*). These layers define an offset (first row), original kernel scope (second row), and final depth-wise layer (third row), where $F = 3$ is the number of spectral bands, and N = number of classes. Maxpooling and average pooling have also been introduced. The model has been trained in about 5000 iterations, applying the ADAM optimizer with a learning rate of 1×10^{-3} and a batch size of 150.

Table 1. DKDCNet architecture.

ID	Type	Kernel/Neurons	Data Norm.	Act. Funct.	Dropout
1	CONV2D	$96 \times 1 \times 1 \times F$	Yes	ReLU	5%
2	DKDC	$18 \times 3 \times 3 \times 96$ 9×9 $96 \times 3 \times 3 \times 96$	- - Yes	- - ReLU	- - 5%
3	MaxPOOL	2×2	-	-	-
4	CONV2D	$108 \times 1 \times 1 \times 96$	Yes	ReLU	5%
5	DKDC	$18 \times 3 \times 3 \times 108$ 7×7 $108 \times 3 \times 3 \times 108$	- - Yes	- - ReLU	- - 5%
5	MaxPOOL	2×2	-	-	-
6	CONV2D	$128 \times 1 \times 1 \times 108$	Yes	ReLU	5%
7	DKDC	$18 \times 3 \times 3 \times 128$ 4×4 $128 \times 3 \times 3 \times 128$	- - Yes	- - ReLU	- - 5%
8	AvgPOOL	2×2	-	-	-
9	FC	N	Yes	Softmax	5%

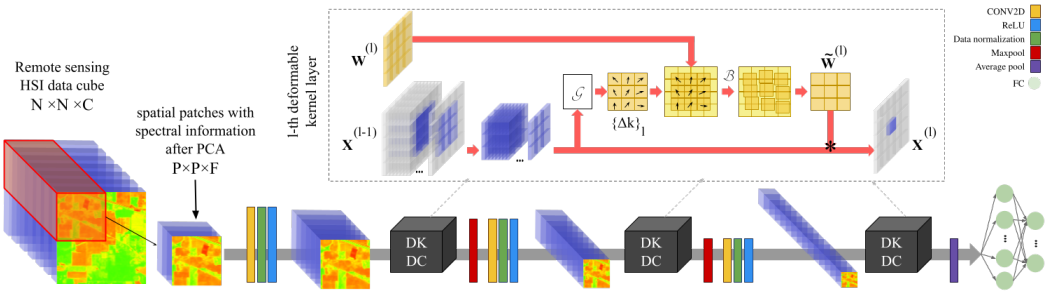


Figure 4. Graphical overview of the proposed adaptable network architecture. Standard convolutions have been implemented as point-wise CONV2D layers. Focusing on DK layers, a kernel scope $\mathbf{W}^{(l)} \in \mathbb{R}^{K^{(l-1)} \times M' \times M' \times K^{(l)}}$ is considered. Then, the generator $\mathcal{G}$ (implemented as a CONV2D) computes the kernel offsets $\{\Delta\mathbf{k}_l\}$ from input $\mathbf{X}^{(l-1)}$. The bilinear sampler $\mathcal{B}$ samples a new set of kernels $\tilde{\mathbf{W}}^{(l)}$ from $\mathbf{W}^{(l)}$ and $\{\Delta\mathbf{k}_l\}$, which is convolved over the input as a depth-wise layer, obtaining the output volume $\mathbf{X}^{(l)}$.

3. Experimental results

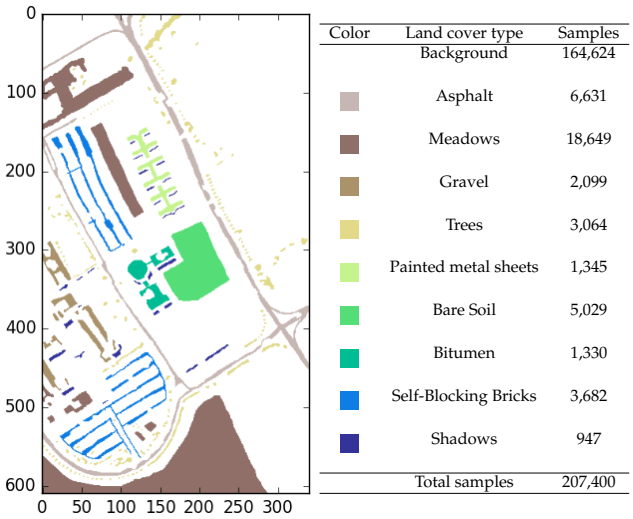
3.1. Experimental Environment

An implementation of the proposed HSI DKDCNet has been developed and tested on a hardware environment with an X Generation Intel® Core™i9-9940X processor with 19.25 M of Cache and up to 4.40 GHz (14 cores/28 way multi-task processing), installed over a Gigabyte X299 Aorus with 128 GB of DDR4 RAM. Furthermore, a graphic processing unit (GPU) NVIDIA Titan RTX GPU with 24 GB GDDR6 of video memory and 4608 cores has been used.

3.2. Hyperspectral Datasets

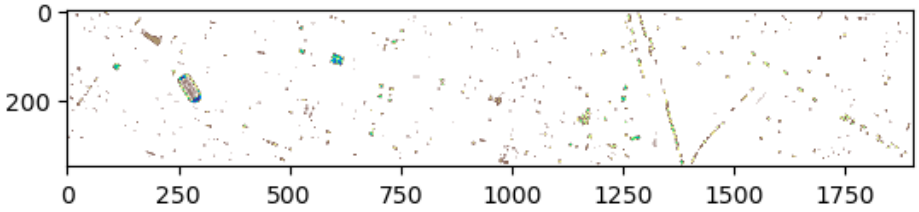
One of the greatest achievements of DL is to design and exploit the same neural architecture for the classification of different images. In this sense, the experiments developed an aim not only to evaluate the performance of the model in terms of accuracy and complexity (understood as number of parameters), but also its generalization capacity. Thus, following [57], the same architecture has been exploited for classifying two real widely-used HSI scenes with different spectral–spatial characteristics: (i) University of Pavia and (ii) University of Houston scenes. Details are provided below.

- The University of Pavia (UP, Figure 5) dataset [65] is an HSI scene gathered by the airborne reflective optics system imaging spectrometer (ROSIS-3) over the university campus of Pavia, northern Italy, in July 2002. The scene consists of 610 × 340 pixels with 1.3 m of spatial resolution, and 113 spectral bands with a spectral range coverage ranging from 430 to 860 nm. The ground truth contains a total of 42,776 labeled samples grouped into nine different land-cover classes, covering different urban elements, including asphalt, meadows, gravel, trees, metal sheet, bare soil, bitumen, brick, and shadows.
- The University of Houston (UH, Figure 6) dataset [66] is an HSI scene collected by the lightweight compact airborne spectrographic imager (CASI) over the Houston University area. It consists of 349 × 1905 pixels with 2.5 m of spatial resolution and 144 channels in the 380 nm to 1050 nm spectral region. The ground truth comprises 15,029 labeled samplescontaining 15 different classes within an urban environment too.



Color	Land cover type	Samples
	Background	164,624
	Asphalt	6,631
	Meadows	18,649
	Gravel	2,099
	Trees	3,064
	Painted metal sheets	1,345
	Bare Soil	5,029
	Bitumen	1,330
	Self-Blocking Bricks	3,682
	Shadows	947
	Total samples	207,400

Figure 5. Ground truth of University of Pavia scene.



Color	Land cover type	Samples	Color	Land cover type	Samples
	Background	649,816		Grass-healthy	1,251
	Grass-stressed	1,254		Grass-synthetic	697
	Tree	1,244		Soil	1,242
	Water	325		Residential	1,268
	Commercial	1,244		Road	1,252
	Highway	1,227		Railway	1,235
	Parking-lot1	1,233		Parking-lot2	469
	Tennis-court	428		Running-track	660
Total samples 664,845					

Figure 6. Ground truth of University of Houston scene.

3.3. Experimental Discussion

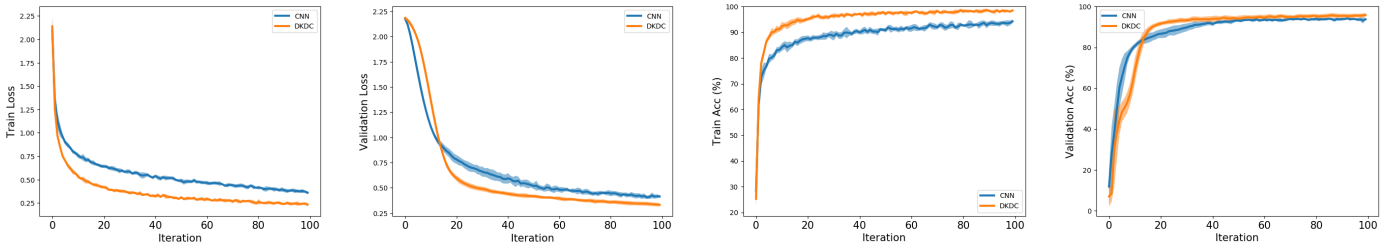
The performance of the DKDCNet for HSI data classification has been evaluated through widely used classification metrics, in particular: (i) the *overall accuracy* (OA), which measures the probability that a sample will be correctly classified by the classification model; (ii) the *average accuracy* (AA), which measures the mean of each obtained classification accuracy per class; and (iii) the statistical *Cohen's Kappa coefficient* K($\times$100), which measures the inter-rater agreement between the observed and expected agreements. Moreover, the number of parameters has been considered.

In addition, the accuracy performance of the network has been compared with standard CNN [57] and with other deformable-based models, in particular the DC-based network (DCNet) and DHCNet [57]. Finally, several well-known HSI classification methods have also been considered, such as the pyramidal ResNet (pResNet) [51], and the spectral–spatial squeeze-and-excitation residual bag-of-feature (S3EResBoF) model [67].

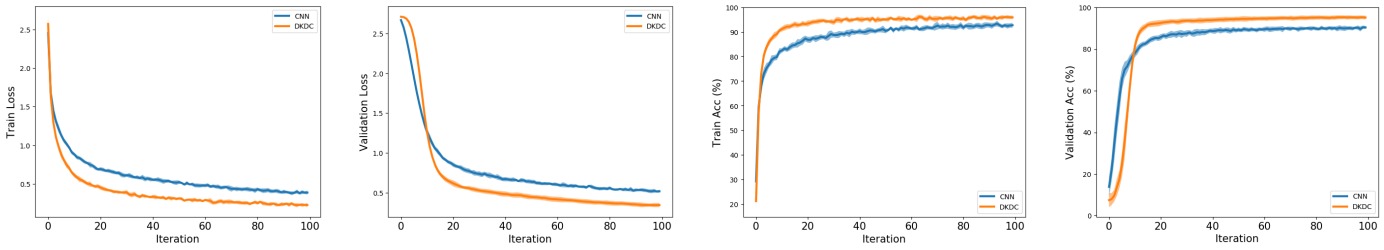
3.3.1. Experimentation on UP Dataset

The first experiment evaluates the performance of the proposed model in the UP image. In this sense, a different number of training samples have been randomly selected from the pool of labelled data; particularly, 45, 55, and 65 random samples per class have been considered to train the proposed DKDCNet and the standard CNN, DCNet, DHCNet, pResNet and S3EResBoF models.

In this context, it is quite interesting to compare the behavior of the original convolutional model and the proposed model for HSI data analysis throughout the training epochs, both in terms of training and validation loss and in terms of accuracy during the training and validation stages. In this regard, Figure 7a depicts the comparison between the proposed model and the standard CNN considering 55 randomly selected training samples. In particular, from left to right we can observe the evolution of both models regarding their training and validation loss and their training and validation accuracy along different epochs. Focusing on training and test loss, the proposed model seems to achieve a training loss quite similar to CNN during the first epochs; even its test loss is higher than CNN in the first 10 epochs. Nevertheless, the training loss drops drastically, below the CNN already in the first five epochs, whilst the validation loss drops abruptly from the first 15 epochs onward. This suggests the model parameters are not over-fitted to the training data, which allows the optimizer to converge to a more optimal solution than the CNN. Indeed, during the training stage, the obtained accuracy climbs rapidly above 90%, whilst the validation accuracy evolves slower but better than the CNN.



(**a**) University of Pavia

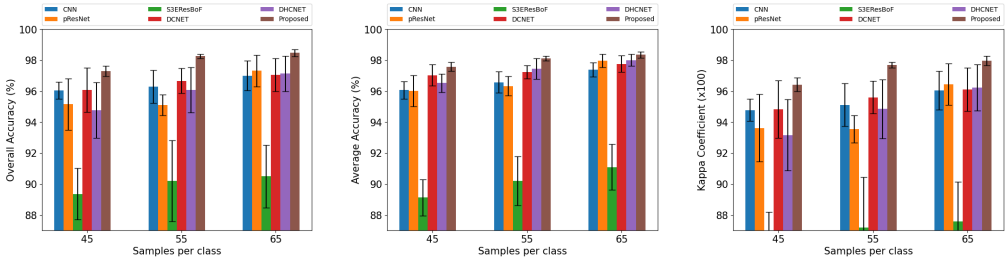


(**b**) University of Houston

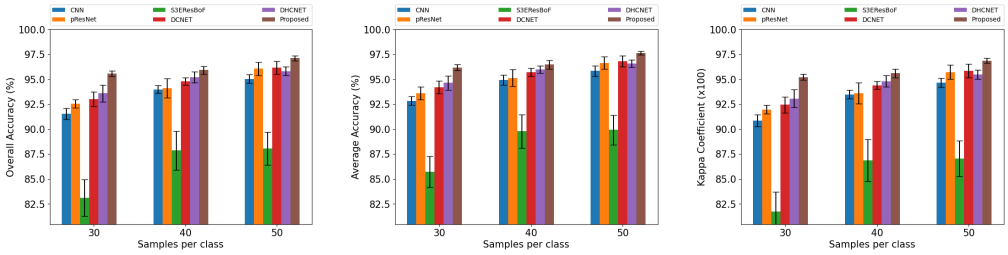
Figure 7. From left to right: training loss, validation loss, training accuracy, and validation accuracy. First row: obtained results with University of Pavia dataset. Second row: obtained results with University of Houston

Furthermore, Figure 8a provides the measurements of the OA, AA, and Kappa coefficients for each model, along with their respective standard deviations, in order to compare the accuracy achieved by each network. These data have been obtained considering 45, 55, and 65 randomly selected training samples after 10 Monte Carlo runs. As an overall conclusion, the proposed DKDCNet model outperforms the results of the other models. When evaluating in detail, the S3EResBoF model achieves the worst results with the highest deviation due to its feature processing; indeed, its Kappa coefficient does not exceed 88%, as we can observe in Table 2, whilst its OA and AA measurements involve the highest uncertainty due to their large variability. On the contrary, the CNN and the pResNet models achieve better results, where the CNN has a lower standard deviation compared to the pResNet. This is due to the higher complexity of the pResNet model, which introduces

3.22M parameters that must be appropriately learned. Indeed, behaviours of the CNN and the pResNet vary depending on the amount of training data; thus, the pResNet, which is more complex than the standard CNN, is greatly affected by the lack of samples, but outperforms the CNN when it has enough samples to properly fit its parameters. Regarding deformable models, the implemented DKDCNet clearly outperforms the DCNet and DHCNet models, exhibiting a higher OA and a very small deviation, which makes it very stable even when there are few training samples.



(**a**) University of Pavia



(**b**) University of Houston

Figure 8. From left to right, graphical representation of the OA, AA, and Kappa Coefficient (coupled with their corresponding standard deviations) reached by each classification model.

Regarding the model complexity (in terms of number of parameters), Figure 9 depicts the number of trainable parameters of each model considering the UP scene. Moreover, Table 2 provides the estimated number of parameters. The pResNet model is the most complex classifier with more than 3M parameters, followed by DHCNet, DCNet, and CNN by far. Indeed, such deformable solutions as DHCNet and DCNet exhibit a large number of parameters. On the contrary, the proposed adaptive model has significantly less parameters, exhibiting not only less over-parametrization (which helps to reduce overfitting problems), but also a lower memory consumption. In this sense, the proposed network contains light depth-wise layers that prevent the overgrowth of the parameters. This feature makes it not only a lightweight, but also a pretty efficient model for HSI data classification. In this context, the adaptation of this model to low-power environments or embedded devices for onboard processing can be highly interesting.

Table 2. Classification results and number of parameters obtained for UP scene.

		Tr = 45			Tr = 55			Tr = 65		
Method	**Params.**	**OA**	**AA**	**K(×100)**	**OA**	**AA**	**K(×100)**	**OA**	**AA**	**K(×100)**
CNN	585K	95.98(0.50)	95.95(0.51)	94.70(0.64)	96.05(1.04)	96.58(0.74)	94.80(1.36)	96.88(1.03)	97.31(0.46)	95.88(1.34)
pResNet	3.22M	95.16(1.67)	96.03(1.00)	93.63(2.17)	95.12(0.67)	96.33(0.62)	93.56(0.88)	97.33(1.03)	97.97(0.43)	96.46(1.34)
S3EResBoF	200K	89.72(1.68)	89.35(1.22)	86.55(2.13)	90.92(1.43)	90.24(1.65)	88.07(1.84)	90.28(1.96)	90.86(1.51)	87.32(2.46)
DCNet	880K	96.19(1.56)	97.07(0.75)	94.99(2.02)	96.77(0.79)	97.21(0.46)	95.74(1.03)	97.35(0.92)	97.81(0.56)	96.49(1.20)
DHCNet	1.14M	94.93(1.78)	96.67(0.59)	93.37(2.29)	96.59(1.19)	97.61(0.63)	95.50(1.55)	97.20(1.02)	98.04(0.39)	96.31(1.33)
DKDCNet	**96.3K**	**97.30**(0.34)	**97.58**(0.29)	**96.42**(0.44)	**98.27**(0.14)	**98.13**(0.16)	**97.70**(0.19)	**98.48**(0.22)	**98.36**(0.19)	**97.97**(0.30)

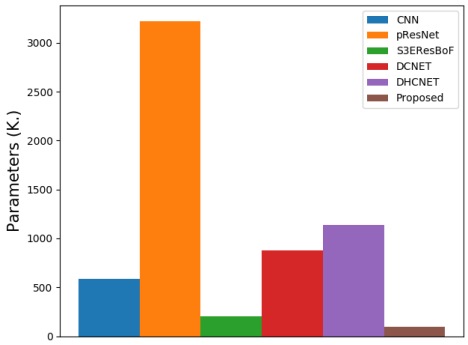


Figure 9. Trainable parameters of the models considering the image University of Pavia.

Finally, Table 2 reports in more detail the obtained classification results in terms of OA, AA, Kappa (with the respective standard deviations after 10 MonteCarlo runs), and number of trainable parameters. As already observed, the implemented HSI DKDCNet outperforms the accuracy results achieved by the other five deep classification models. Moreover, its low deviation results indicate its classification performance is quite stable. Finally, the model lightness in terms of the number of parameters makes it a potential candidate for onboard processing.

3.3.2. Experimentation on UH dataset

The second experiment evaluates the performance of the proposed model on the UH image. Similar to the first experiment with the UP scene, different numbers of training samples have been randomly selected from the pool of labeled data; particularly, 30, 40, and 50 random samples per class have been considered to train the proposed DKDCNet and the standard CNN, DCNet, DHCNet, pResNet, and S3EResBoF models.

Once more, a comparison between the standard CNN and the proposed DKDCNet has been conducted to evaluate the behaviour of the models along different epochs. Figure 7b depicts, from left to right, the evolution of both models regarding their training and validation loss and their training and validation accuracy along different epochs considering 40 randomly selected training samples. Focusing on training and test loss, the proposed model has a fairly similar evolution to that presented in the previous experiment. Indeed, the training loss drops drastically between the first 10 epochs, overcoming the CNN model, whilst the validation loss drops abruptly from the first 15 epochs onward. Furthermore, during the training stage, the obtained accuracy climbs rapidly above 90%, whilst the validation accuracy evolves slower but better than the CNN, reaching an OA of over 90%.

Furthermore, a deeper comparison is conducted between the proposed DKDCNet and the considered CNN, pResNet, S3EResBoF, DCNet, and DHCNet models. Figure 8b provides the obtained OA, AA and Kappa coefficients for each model, along with their respective standard deviations, in order to compare the accuracy achieved by each deep network. Following the previous experiment, these results have been obtained considering 30, 40, and 50 randomly selected training samples after 10 Monte Carlo runs. Once more, similar behaviors are observed in the UH scene, where the proposed method reaches the best measurements in terms of OA, AA, and Kappa. Again, the S3EResBoF model achieves the worst results with the highest deviation for each measurement. Focusing on the CNN and the pResNet models, in this case, due to the data distribution of UH scene, the pResNet obtains better performance than CNN, since its skip connections provide a path to reuse local features. On the other hand, the DKDCNet outperforms the deformable DCNet and DHCNet models.

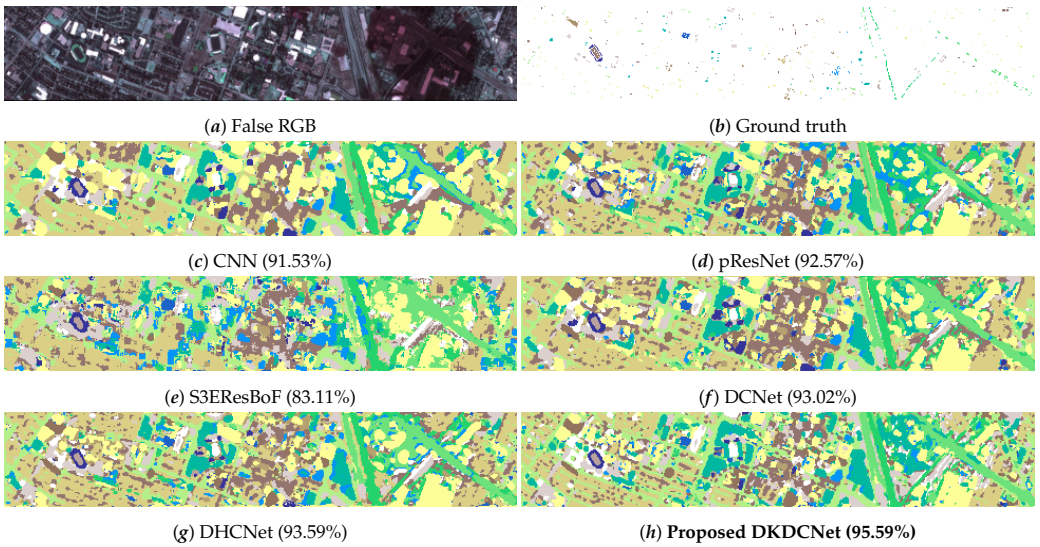
Table 2 reports not only the OA, AA, and Kappa measurement, but also the complexity of each model. Indeed, the first column of Table 3 provides the estimated number of parameters. It must be noted that the size of the model depends on the size of the input data and the number of classes, which increases or decreases the number of parameters to be trained. In this sense, models with a UH scene are slightly larger than those with UP

since they contain 16 classes instead of 9. This increase in the number of classes is hardly noticeable in very large models, such as the pResNet or the DHCNet, or even the S3EResBoF (with only 220K), so their approximation is practically the same as the parameters in UP. However, in the proposed DKDCNet model, the increase can be observed because of its reduced number of parameters.

Table 3. Classification results and number of parameters obtained for UH scene.

		Tr = 30			Tr = 40			Tr = 50		
Method	**Params.**	**OA**	**AA**	**K(×100)**	**OA**	**AA**	**K(×100)**	**OA**	**AA**	**K(×100)**
CNN	585K	91.53(0.55)	92.83(0.43)	90.84(0.60)	93.98(0.39)	94.93(0.51)	93.49(0.43)	95.04(0.43)	95.84(0.52)	94.64(0.46)
pResNet	3.22M	92.57(0.41)	93.61(0.63)	91.97(0.44)	94.09(0.96)	95.14(0.84)	93.61(1.04)	96.06(0.67)	96.64(0.63)	95.73(0.73)
S3EResBoF	200K	83.11(1.84)	85.72(1.54)	81.74(1.98)	87.86(1.96)	89.78(1.66)	86.87(2.12)	88.04(1.65)	89.92(1.50)	87.07(1.78)
DCNet	880K	93.02(0.74)	94.21(0.64)	92.45(0.80)	94.81(0.36)	95.72(0.42)	94.39(0.39)	96.18(0.63)	96.83(0.55)	95.86(0.69)
DHCNet	1.14M	93.59(0.83)	94.64(0.71)	93.07(0.90)	95.20(0.54)	95.99(0.37)	94.81(0.58)	95.83(0.43)	96.58(0.38)	95.49(0.47)
DKDCNet	**97.1K**	**95.59**(0.26)	**96.19**(0.32)	**95.23**(0.28)	**95.94**(0.39)	**96.47**(0.42)	**95.61**(0.42)	**97.12**(0.23)	**97.64**(0.20)	**96.88**(0.25)

Finally, Figure 10 provides the classification maps of the UH scene obtained by each network considering 30 randomly selected training samples. As we can observe, the HSI DKDCNet extract classification maps with high quality, even when small training sets are employed.



(*a*) False RGB (*b*) Ground truth

(*c*) CNN (91.53%) (*d*) pResNet (92.57%)

(*e*) S3EResBoF (83.11%) (*f*) DCNet (93.02%)

(*g*) DHCNet (93.59%) (*h*) **Proposed DKDCNet (95.59%)**

Figure 10. Classification maps obtained from UH scene (considering 30 samples per class). The obtained OAs are shown in brackets.

4. Conclusions and Future Lines

This paper presents a new deep convolution-based neural network for HSI data classification. The model is inspired by deformable kernels, which are combined with deformable convolutions to automatically adapt the effective receptive field of the CNN classifier to spatial deformations within remote sensing HSI data. In this regard, the adaptable classification network fits its parameters to the HSI data patches in an optimal way, not only by being able to deform the convolution (i.e., adding an offset to the feature positions), but by automatically exploiting the deformation of the kernel itself applied to each receptive field on the input feature volume. Moreover, the deformations included in the model through different offsets are learned directly from the input data in a local and adaptive way. The conducted experiments over two widely-used HSI scenes (i.e., the University of Pavia and University of Houston scenes) demonstrate the HSI DKDC model is able to overcome the current state-of-the-art HSI data classification considering different deep models, some of them also with data deformation modeling, such as DCNet and

DHCNet. In this sense, the implemented network is able to extract relevant spectral–spatial features more effectively from HSI data, using fewer parameters and reaching satisfactory accuracy results.

In future works, new deformable schemes will be studied to enhance the data deformation modeling within deep architectures, also improving the flexibility of current neural models.

Author Contributions: The authors have contributed as equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Junta de Extremadura FEDER under Grant GR18060 and Leonardo Grants for Researchers and Cultural Creators awarded by the Fundación BBVA under grant 2021/00203/00.

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

Acknowledgments: We gratefully thank the Associate Editor and the Anonymous Reviewers for their outstanding comments and suggestions, which greatly helped us to improve the technical quality and presentation of our work.

Acronyms

List of acronyms used in this paper:

Acronym	Full name
ANNs	Artificial Neural Networks
AA	Average Accuracy
CapsNets	Capsule Networks
CASI	Compact Airborne Spectrographic Imager
CNNs	Convolutional Neural Networks
CONV2D	Convolutional 2D
DC	Deforming Convolution
DK	Deforming Kernels
DL	Deep Learning
DTs	Decision Trees
EO	Earth Observation
ERF	Effective Receptive Field
FC	Fully Connected
GPU	Graphics Processing Unit
HSI	Hyperspectral Image
K(x100)	Cohen's Kappa coefficient
LULC	Land-Use/Land-Cover
MLP	Multi-Layer Perceptron
MLR	Multinomial Logistic Regression
MP	Morphological Profiles
OA	Overall Accuracy
pResNet	Pyramidal ResNet
RaF	Random Forests
RAM	Random Access Memory
RF	Receptive Field
ROSIS-3	Reflective Optics System Imaging Spectrometer
RS	Remote Sensing
RS-HSI	Remote Sensing Hyperspectral Image
S3EResBoF	Spectral-Spatial Squeeze-and-Excitation Residual Bag-of-Feature
SVMs	Support Vector Machines
TRF	Theorical Receptive Field
UH	University of Houston
UP	University of Pavia
VSWIR	Visible-to-Shortwave Infrared
ResNets	Residual Networks

References

1. Srivastava, P.K.; Malhi, R.K.M.; Pandey, P.C.; Anand, A.; Singh, P.; Pandey, M.K.; Gupta, A. Revisiting hyperspectral remote sensing: Origin, processing, applications and way forward. In *Hyperspectral Remote Sensing*; Elsevier: Amsterdam, The Netherlands, 2020; pp. 3–21.
2. Paoletti, M.; Haut, J.; Plaza, J.; Plaza, A. A new deep convolutional neural network for fast hyperspectral image classification. *ISPRS J. Photogramm. Remote Sens.* **2018**, *145*, 120–147. [CrossRef]
3. Roessner, S.; Segl, K.; Heiden, U.; Kaufmann, H. Automated differentiation of urban surfaces based on airborne hyperspectral imagery. *IEEE Trans. Geosci. Remote Sens.* **2001**, *39*, 1525–1532. [CrossRef]
4. Shobiga, R.; Selvakumar, J. Survey on properties and accuracy assessment of climate changes using hyperspectral imaging. In Proceedings of the 2015 Online International Conference on Green Engineering and Technologies (IC-GET), Coimbatore, India, 27 November 2015; pp. 1–6.
5. Adão, T.; Hruška, J.; Pádua, L.; Bessa, J.; Peres, E.; Morais, R.; Sousa, J.J. Hyperspectral imaging: A review on UAV-based sensors, data processing and applications for agriculture and forestry. *Remote Sens.* **2017**, *9*, 1110. [CrossRef]
6. Cheng, G.; Yang, C.; Yao, X.; Guo, L.; Han, J. When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative CNNs. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 2811–2821. [CrossRef]
7. Zhang, N.; Zhang, X.; Yang, G.; Zhu, C.; Huo, L.; Feng, H. Assessment of defoliation during the Dendrolimus tabulaeformis Tsai et Liu disaster outbreak using UAV-based hyperspectral images. *Remote Sens. Environ.* **2018**, *217*, 323–339. [CrossRef]
8. Tao, X.; Paoletti, M.E.; Haut, J.M.; Ren, P.; Plaza, J.; Plaza, A. Endmember Estimation with Maximum Distance Analysis. *Remote Sens.* **2021**, *13*, 713. [CrossRef]
9. Tao, X.; Paoletti, M.E.; Haut, J.M.; Hang, L.; Ren, P.; Plaza, J.; Plaza, A. Endmember Estimation from Hyperspectral Images Using Geometric Distances. *IEEE Geosci. Remote Sens. Lett.* **2021**. [CrossRef]
10. Simonetti, E.; Simonetti, D.; Preatoni, D. *Phenology-Based Land Cover Classification Using Landsat 8 Time Series*; European Commission Joint Research Center: Ispra, Italy, 2014.
11. Boonprong, S.; Cao, C.; Chen, W.; Ni, X.; Xu, M.; Acharya, B.K. The classification of noise-afflicted remotely sensed data using three machine-learning techniques: Effect of different levels and types of noise on accuracy. *ISPRS Int. J. Geo-Inf.* **2018**, *7*, 274. [CrossRef]
12. Lv, W.; Wang, X. Overview of hyperspectral image classification. *J. Sens.* **2020**, *2020*, 4817234. [CrossRef]
13. Haut, J.M.; Paoletti, M.; Plaza, J.; Plaza, A. Cloud implementation of the K-means algorithm for hyperspectral image analysis. *J. Supercomput.* **2017**, *73*, 514–529. [CrossRef]
14. Li, T.; Zhang, J.; Zhang, Y. Classification of hyperspectral image based on deep belief networks. In Proceedings of the 2014 IEEE International Conference on Image Processing (ICIP), Paris, France, 27–30 October 2014; pp. 5132–5136.
15. Gao, P.; Wang, J.; Zhang, H.; Li, Z. Boltzmann entropy-based unsupervised band selection for hyperspectral image classification. *IEEE Geosci. Remote Sens. Lett.* **2018**, *16*, 462–466. [CrossRef]
16. Qiu, Q.; Wu, X.; Liu, Z.; Tang, B.; Zhao, Y.; Wu, X.; Zhu, H.; Xin, Y. Survey of supervised classification techniques for hyperspectral images. *Sens. Rev.* **2017**, *37*, 371–382. [CrossRef]
17. Cheng, G.; Li, Z.; Han, J.; Yao, X.; Guo, L. Exploring hierarchical convolutional features for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 6712–6722. [CrossRef]
18. Wang, G.; Ren, P. Hyperspectral Image Classification with Feature-Oriented Adversarial Active Learning. *Remote Sens.* **2020**, *12*, 3879. [CrossRef]
19. Alcolea, A.; Paoletti, M.E.; Haut, J.M.; Resano, J.; Plaza, A. Inference in supervised spectral classifiers for on-board hyperspectral imaging: An overview. *Remote Sens.* **2020**, *12*, 534. [CrossRef]
20. Bruzzone, L.; Chi, M.; Marconcini, M. Semisupervised support vector machines for classification of hyperspectral remote sensing images. In *Hyperspectral Data Exploitation: Theory and Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2007; pp. 275–311.
21. Tan, K.; Li, E.; Du, Q.; Du, P. An efficient semi-supervised classification approach for hyperspectral imagery. *ISPRS J. Photogramm. Remote Sens.* **2014**, *97*, 36–45. [CrossRef]
22. Haut, J.M.; Paoletti, M.E.; Plaza, J.; Li, J.; Plaza, A. Active learning with convolutional neural networks for hyperspectral image classification using a new bayesian approach. *IEEE Trans. Geosci. Remote Sens.* **2018**, *56*, 6440–6461. [CrossRef]
23. Peng, J.; Zhou, Y.; Chen, C.P. Region-kernel-based support vector machines for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2015**, *53*, 4810–4824. [CrossRef]
24. Paoletti, M.E.; Haut, J.M.; Tao, X.; Miguel, J.P.; Plaza, A. A new GPU implementation of support vector machines for fast hyperspectral image classification. *Remote Sens.* **2020**, *12*, 1257. [CrossRef]
25. Khodadadzadeh, M.; Li, J.; Plaza, A.; Bioucas-Dias, J.M. A subspace-based multinomial logistic regression for hyperspectral image classification. *IEEE Geosci. Remote Sens. Lett.* **2014**, *11*, 2105–2109. [CrossRef]
26. Haut, J.M.; Paoletti, M.E. Cloud Implementation of Multinomial Logistic Regression for UAV Hyperspectral Images. *IEEE J. Miniaturization Air Space Syst.* **2020**, *1*, 163–171. [CrossRef]
27. Ham, J.; Chen, Y.; Crawford, M.M.; Ghosh, J. Investigation of the random forest framework for classification of hyperspectral data. *IEEE Trans. Geosci. Remote Sens.* **2005**, *43*, 492–501. [CrossRef]
28. Zhang, Y.; Cao, G.; Li, X.; Wang, B. Cascaded random forest for hyperspectral image classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2018**, *11*, 1082–1094. [CrossRef]

29. Santos, A.B.; de Albuquerque Araújo, A.; Menotti, D. Combining multiple classification methods for hyperspectral data interpretation. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2013**, *6*, 1450–1459. [CrossRef]
30. Audebert, N.; Le Saux, B.; Lefèvre, S. Deep learning for classification of hyperspectral data: A comparative review. *IEEE Geosci. Remote Sens. Mag.* **2019**, *7*, 159–173. [CrossRef]
31. Paoletti, M.; Haut, J.; Plaza, J.; Plaza, A. Deep learning classifiers for hyperspectral imaging: A review. *ISPRS J. Photogramm. Remote Sens.* **2019**, *158*, 279–317. [CrossRef]
32. Chutia, D.; Bhattacharyya, D.; Sarma, K.K.; Kalita, R.; Sudhakar, S. Hyperspectral remote sensing classifications: A perspective survey. *Trans. GIS* **2016**, *20*, 463–490. [CrossRef]
33. He, L.; Li, J.; Liu, C.; Li, S. Recent advances on spectral–spatial hyperspectral image classification: An overview and new guidelines. *IEEE Trans. Geosci. Remote Sens.* **2017**, *56*, 1579–1597. [CrossRef]
34. Gu, Y.; Chanussot, J.; Jia, X.; Benediktsson, J.A. Multiple kernel learning for hyperspectral image classification: A review. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 6547–6565. [CrossRef]
35. Fauvel, M.; Benediktsson, J.A.; Chanussot, J.; Sveinsson, J.R. Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles. *IEEE Trans. Geosci. Remote Sens.* **2008**, *46*, 3804–3814. [CrossRef]
36. Shen, L.; Zhu, Z.; Jia, S.; Zhu, J.; Sun, Y. Discriminative Gabor feature selection for hyperspectral image classification. *IEEE Geosci. Remote Sens. Lett.* **2012**, *10*, 29–33. [CrossRef]
37. Mirzapour, F.; Ghassemian, H. Improving hyperspectral image classification by combining spectral, texture, and shape features. *Int. J. Remote Sens.* **2015**, *36*, 1070–1096. [CrossRef]
38. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
39. 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]
40. Yu, S.; Jia, S.; Xu, C. Convolutional neural networks for hyperspectral image classification. *Neurocomputing* **2017**, *219*, 88–98. [CrossRef]
41. Liang, H.; Li, Q. Hyperspectral imagery classification using sparse representations of convolutional neural network features. *Remote Sens.* **2016**, *8*, 99. [CrossRef]
42. Li, Y.; Zhang, H.; Shen, Q. Spectral–spatial classification of hyperspectral imagery with 3D convolutional neural network. *Remote Sens.* **2017**, *9*, 67. [CrossRef]
43. Ding, C.; Li, Y.; Xia, Y.; Wei, W.; Zhang, L.; Zhang, Y. Convolutional neural networks based hyperspectral image classification method with adaptive kernels. *Remote Sens.* **2017**, *9*, 618. [CrossRef]
44. Wang, W.; Dou, S.; Jiang, Z.; Sun, L. A fast dense spectral–spatial convolution network framework for hyperspectral images classification. *Remote Sens.* **2018**, *10*, 1068. [CrossRef]
45. Haut, J.M.; Bernabé, S.; Paoletti, M.E.; Fernandez-Beltran, R.; Plaza, A.; Plaza, J. Low–high-power consumption architectures for deep-learning models applied to hyperspectral image classification. *IEEE Geosci. Remote Sens. Lett.* **2018**, *16*, 776–780. [CrossRef]
46. Li, Y.; Xie, W.; Li, H. Hyperspectral image reconstruction by deep convolutional neural network for classification. *Pattern Recognit.* **2017**, *63*, 371–383. [CrossRef]
47. Gao, Q.; Lim, S.; Jia, X. Hyperspectral image classification using convolutional neural networks and multiple feature learning. *Remote Sens.* **2018**, *10*, 299. [CrossRef]
48. Paoletti, M.E.; Haut, J.M.; Plaza, J.; Plaza, A. Deep&dense convolutional neural network for hyperspectral image classification. *Remote Sens.* **2018**, *10*, 1454.
49. Fang, B.; Li, Y.; Zhang, H.; Chan, J.C.W. Hyperspectral images classification based on dense convolutional networks with spectral-wise attention mechanism. *Remote Sens.* **2019**, *11*, 159. [CrossRef]
50. Zhong, Z.; Li, J.; Luo, Z.; Chapman, M. Spectral–spatial residual network for hyperspectral image classification: A 3-D deep learning framework. *IEEE Trans. Geosci. Remote Sens.* **2017**, *56*, 847–858. [CrossRef]
51. Paoletti, M.E.; Haut, J.M.; Fernandez-Beltran, R.; Plaza, J.; Plaza, A.J.; Pla, F. Deep pyramidal residual networks for spectral–spatial hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2018**, *57*, 740–754. [CrossRef]
52. Kang, X.; Zhuo, B.; Duan, P. Dual-path network-based hyperspectral image classification. *IEEE Geosci. Remote Sens. Lett.* **2018**, *16*, 447–451. [CrossRef]
53. Paoletti, M.E.; Haut, J.M.; Fernandez-Beltran, R.; Plaza, J.; Plaza, A.; Li, J.; Pla, F. Capsule networks for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2018**, *57*, 2145–2160. [CrossRef]
54. Ding, X.; Li, Y.; Yang, J.; Li, H.; Liu, L.; Liu, Y.; Zhang, C. An adaptive capsule network for hyperspectral remote sensing classification. *Remote Sens.* **2021**, *13*, 2445. [CrossRef]
55. Haut, J.M.; Paoletti, M.E.; Plaza, J.; Plaza, A.; Li, J. Hyperspectral image classification using random occlusion data augmentation. *IEEE Geosci. Remote Sens. Lett.* **2019**, *16*, 1751–1755. [CrossRef]
56. Zhang, C.; Li, G.; Du, S. Multi-Scale Dense Networks for Hyperspectral Remote Sensing Image Classification. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 9201–9222. [CrossRef]
57. Zhu, J.; Fang, L.; Ghamisi, P. Deformable convolutional neural networks for hyperspectral image classification. *IEEE Geosci. Remote Sens. Lett.* **2018**, *15*, 1254–1258. [CrossRef]

58. Luo, W.; Li, Y.; Urtasun, R.; Zemel, R. Understanding the effective receptive field in deep convolutional neural networks. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 4898–4906.
59. Gao, H.; Zhu, X.; Lin, S.; Dai, J. Deformable Kernels: Adapting Effective Receptive Fields for Object Deformation. *arXiv* **2019**, arXiv:1910.02940.
60. Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; Wei, Y. Deformable convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 764–773.
61. Le, H.; Borji, A. What are the receptive, effective receptive, and projective fields of neurons in convolutional neural networks? *arXiv* **2017**, arXiv:1705.07049.
62. Hubel, D.H.; Wiesel, T.N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J. Physiol.* **1962**, *160*, 106–154. [CrossRef]
63. Lehky, S.R.; Sejnowski, T.J. Network model of shape-from-shading: Neural function arises from both receptive and projective fields. *Nature* **1988**, *333*, 452–454. [CrossRef] [PubMed]
64. Araujo, A.; Norris, W.; Sim, J. Computing receptive fields of convolutional neural networks. *Distill* **2019**, *4*, e21. [CrossRef]
65. Kunkel, B.; Blechinger, F.; Lutz, R.; Doerffer, R.; van der Piepen, H.; Schroder, M. ROSIS (Reflective Optics System Imaging Spectrometer)—A candidate instrument for polar platform missions. In *SPIE 0868 Optoelectronic Technologies for Remote Sensing from Space*; Seeley, J., Bowyer, S., Eds.; International Society for Optics and Photonics: Bellingham, WA, USA, 1988; p. 8. [CrossRef]
66. Xu, X.; Li, J.; Plaza, A. Fusion of hyperspectral and LiDAR data using morphological component analysis. In Proceedings of the 2016 IEEE International Geoscience and Remote Sensing Symposium, Beijing, China, 10–15 July 2016; pp. 3575–3578. [CrossRef]
67. Roy, S.K.; Chatterjee, S.; Bhattacharyya, S.; Chaudhuri, B.B.; Platoš, J. Lightweight spectral-spatial squeeze-and-excitation residual bag-of-features learning for hyperspectral classification. *IEEE Trans. Geosci. Remote Sens.* **2020**, *58*, 5277–5290. [CrossRef]