Neural Ordinary Differential Equations for Hyperspectral Image Classification

Mercedes E. Paoletti, *Student Member, IEEE*, Juan Mario Haut, *Member, IEEE*, Javier Plaza, *Senior Member, IEEE*, and Antonio Plaza, *Fellow, IEEE*

***Abstract*— Advances in deep learning (DL) have allowed for the development of more complex and powerful neural architectures. The adoption of deep convolutional-based architectures with residual learning [residual networks (ResNets)] has reached the state-of-the-art performance in hyperspectral image (HSI) classification. Traditionally, ResNets have been considered as stacks of discrete layers, where each one obtains a hidden state of the input data. This formulation must deal with very deep networks, which suffer from an important data degradation as they become deeper. Moreover, these complex models exhibit significant requirements in terms of memory due to the amount of parameters that need to be fine tuned. This leads to inadequate generalization and loss of accuracy. In order to address these issues, this article redesigns the ResNet as a continuous-time evolving model, where hidden representations (or states) are obtained with respect to time (understood as the depth of the network) through the evaluation of an ordinary differential equation (ODE), which is combined with a deep neural architecture. Our experimental results, conducted with four well-known HSI data sets, indicate that redefining deep networks as continuous systems through ODEs offers flexibility when processing and classifying these kinds of remotely sensed data, achieving significant performance even when a very few training samples are available.**

***Index Terms*— Deep learning (DL), hyperspectral images (HSIs), ordinary differential equations (ODEs), residual networks (ResNets).**

Manuscript received March 14, 2019; revised August 10, 2019; accepted October 13, 2019. Date of publication November 6, 2019; date of current version February 26, 2020. This work was supported in part by the Ministerio de Educación (Resolución de 26 de diciembre de 2014 y de 19 de noviembre de 2015, de la Secretaría de Estado de Educación, Formación Profesional y Universidades, por la que se convocan ayudas para la formación de profesorado universitario, de los subprogramas de Formación y de Movilidad incluidos en el Programa Estatal de Promoción del Talento y su Empleabilidad, and en el marco del Plan Estatal de Investigación Científica y Técnica y de Innovación 2013–2016), in part by the Junta de Extremadura (Decreto 14/2018, de 6 de febrero, por el que se establecen las bases reguladoras de las ayudas para la realización de actividades de investigación y desarrollo tecnológico, and de divulgación y de transferencia de conocimiento por los Grupos de Investigación de Extremadura) under Grant GR18060, in part by the European Union's Horizon 2020 Research and Innovation Programme under Grant 734541 (EOXPOSURE), and in part by Ministerio de Economía y Empresa (MINECO) Project under Grant TIN2015-63646-C5-5-R. *(Corresponding author: Mercedes E. Paoletti.)*

The authors are with the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, Escuela Politécnica, University of Extremadura, 10003 Cáceres, Spain (e-mail: mpaoletti@unex.es; juanmariohaut@unex.es; jplaza@unex.es; aplaza@unex.es).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TGRS.2019.2948031

I. Introduction

REMOTE sensing techniques have been widely employed for detecting, measuring, and monitoring the physical behavior and/or characteristics of large areas of the earth through the acquisition and measurement of radiation emitted or reflected by the terrestrial materials that comprise the observed surfaces, which are captured by specific sensors located on airborne or spaceborne platforms [1]. The interpretation of the obtained measurements can be beneficial to human activity [2], [3]. There is a wide range of remote sensing data, where each one exhibits different spatial and spectral properties depending on the type of employed sensor and measured radiation. Moreover, current earth observation missions are already collecting an extremely large volume of remotely sensed data from satellites and airborne systems [4]. Hyperspectral images (HSIs) are collected by passive spectrometers that measure the reflected solar radiation from the observed areas, creating huge data cubes comprised of hundreds of narrow and continuous spectral wavelengths. As a result, an HSI given by $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times n_{\text{bands}}}$ is comprised of two spatial components that determine the image's width and height ($n_1 \times n_2$) and one spectral component that indicates the number of channels or spectral bands (n_{bands}). As a result, each pixel of $\mathbf{X}$ can be interpreted as a detailed spectral signature or spectral vector $\mathbf{x}_i \in \mathbb{R}^{n_{\text{bands}}} = \{x_{i,1}, \ldots, x_{i,n_{\text{bands}}}\}$, which allows for an accurate characterization of the surface materials [5]. This has attracted the attention of many researchers who employ HSIs into a wide range of applications, including precision agriculture [6], environment and natural resources' management [7], mineralogy [8], forestry [9], disaster monitoring [10], urban planning [11], and defense applications [12], among others.

A large variety of algorithms have been developed to process and extract useful information from HSI data cubes. In this regard, HSI classification methods can greatly benefit from the rich spectral information contained in each pixel $\mathbf{x}_i$. In fact, the classification of these images aims to assign a single category (or label) to each pixel in the image. In mathematical fashion, the goal of a classifier is to approximate a mapping function of the form $f(\cdot, \theta)$, which depends on parameter θ, to map the pixels in the original HSI $\mathcal{X} \subset \mathbb{R}^{n_{\text{samples}}}$ to those labels contained in a set of categories $\mathcal{Y} \subset \mathbb{N}$, i.e., $f : \mathcal{X} \rightarrow \mathcal{Y}$. In the particular case of HSI classification, the procedure consists of mapping each pixel $\mathbf{x}_i$ in $\mathbf{X} \equiv \{\mathbf{x}_1, \ldots, \mathbf{x}_{n_{\text{samples}}}\}$ (with $n_{\text{samples}} = n_1 \cdot n_2$) to a unique numerical label of n_{classes} possible classes $y_i = \{1, \ldots, n_{\text{classes}}\}$ extracted from the set $Y \equiv \{y_1, \ldots, y_{n_{\text{samples}}}\}$, creating pairs of $\{\mathbf{x}_i, y_i\}_{i=1}^{n_{\text{samples}}}$ for each spectral pixel.

Traditional HSI classification methods are based on the analysis of the each pixel $\mathbf{x}_i$ independently, without considering spatial information, for instance, unsupervised clustering techniques, such as k-means [13], and supervised or semisupervised methods, such as the widely used support vector machine (SVM) [14] or multinomial logistic regression (MLR) [15], among others [16], [17]. Artificial neural networks (ANNs) [18], [19] have acquired great popularity due to their flexibility concerning learning modes (unsupervised, supervised, and semisupervised) and available architectures (shallow, deep, fully, or local connected). Moreover, ANNs work as universal approximators [20], [21], being able to extract representative features and to discover nonlinear relationships from the input data.

Advances in deep learning (DL) [22], [23] have allowed for the implementation of deeper and more complex ANNs, known as deep neural networks (DNNs). These networks are comprised of groups of neurons organized into a hierarchy of multiple nonlinear layers, which are stacked one by one. As a result, DNNs are comprised of one input and one output layer with several hidden layers in-between them. The original data go through the hierarchy of layers, where a different level of data representation is obtained at each layer. These representations are comprised of highly expressive features that encode complex patterns and nonlinear relationships in the data. At the end of the network, highly abstract and discriminative information is obtained, which can be employed to enhance classification tasks. In the following, we briefly review some recent DL works in the literature (focusing on those based on convolutional and residual architectures for HSI data classification), and then, we discuss some shortcomings and limitations of these works and the solutions adopted in this article.

A. Recent Trends in DL for HSI Classification

DNNs traditionally follow a biological neural model, implementing a fully connected (FC) topology where all the neurons in a layer are totally connected with all the neurons of the previous and following layers, as in the multilayer perceptron (MLP). In this way, each neuron applies a dot product between the outputs of previous neurons and the connection weights, simulating synaptic weights. The obtained result is filtered by a threshold function, also known as nonlinear activation function, which encodes the nonlinearities of the data and triggers (or not) the activation of a given neuron. In fact, the DNN approaches adopt the same strategy as traditional pixelwise classifiers. For HSI data, they take as input the spectral pixels of the HSI data cube [18]. In this regard, spectral-based DNNs are quite sensitive to variations in the spectral signatures. It should be noted that HSI data are characterized by their high intraclass variability and interclass similarity (due to perturbations and disturbances in the data collection process at the spectrometer, atmospheric conditions, and so on). Also, HSI data normally exhibit low spatial resolution, which means that a single pixel often contains multiple materials, resulting in mixed spectral signatures. These shortcomings, coupled with the curse of dimensionality and the Hughes phenomenon [24] (which establishes the need for a reasonable balance between the number of training samples and the number of spectral bands in order to ensure a reliable classification [25], [26]), are important challenges to deploy the full potential of HSI technology with traditional pixel-based DNN approaches.

A significant evolution in DL techniques was the adaptation of biological visual cortex neurons into DNN architectures, with the implementation of convolutional neural networks (CNNs) [22]. Inspired by the local receptive field of such visual cortex neurons (activated or not in the presence of certain types of visual stimuli), CNN-based models rely on the application of a sliding n-dimensional kernel on the input data of each layer. This allows for the exploitation of the visual properties of an image, learning features at certain positions of such an image and applying these features as filters to the rest of the image in order to obtain a feature-activation map [27], [28]. In this sense, the contextualization provided by the spatial components $n_1 \times n_2$ of the HSI data cube $\mathbf{X}$ can greatly reduce the variability of spectral samples by interpreting the data surrounding the pixels as belonging to the same class, which reinforces the information contained in the target pixel, reducing also the well-known "salt and pepper" noise of spectral classifiers.

CNN models exhibit excellent performance in HSI data classification through the development of a wide range of architectures from traditional spectral-based ones (CNN1D) to spatial (CNN2D) and spectral–spatial (CNN3D) models. For instance, Hu *et al.* [29] implemented a five-layer CNN1D to classify HSI data in the spectral domain, and Yue *et al.* [30] developed a CNN3D to classify HSI data taking into account spectral–spatial information. Zhao and Du [31] exploited a CNN2D model as a highly confident spatial feature extractor. Chen *et al.* [32] reviewed CNN1D, CNN2D, and CNN3D models for deep HSI feature extraction (FE) and classification. In order to enhance the classification results, several improvements have been added to the CNN architecture. For instance, Yu *et al.* [33] implemented a three-layer CNN2D model with 1×1 kernels inspired by the network-in-network (NIN) model [34] in order to overcome the presence of highly correlated bands in the HSI data cube. He *et al.* [35] combined the information contained in HSI-extracted covariances with a CNN2D model. Paoletti *et al.* [36] presented a faster end-to-end CNN3D that improved the classification accuracy, taking into account the full spectral signatures contained in HSI data.

Despite the aforementioned results, CNN models still face certain limitations related to the intrinsic characteristics of HSI data and the (high) number of parameters and the depth of the network. In particular, CNNs need a large amount of training data to properly adjust their weights [37]. They also require some variability in the data in order to extract more features [28]. Although HSI data often exhibit a wide variety of samples, very limited labeled data are often available due to their high cost, which, in the end, hampers the FE process and leads to overadjustment (overfitting) in the convolutional model's parameters.

In addition, the implementation of very deep CNN models through the stacking of successive layers has proved to be inefficient itself [38], since a significant degradation can be observed in both the forward propagation of the data and the backpropagation of the gradient signal through the layers (vanishing gradient problem) [39], [40]. To overcome these issues, the residual learning aims to facilitate data reusability through identity functions implemented by skip or residual connections. Residual networks (ResNets) [38] and other residual-based architectures (such as highway networks [41], DenseNets [42], or ResNets of Resnets (RoRs) [43]) have emerged as the current state-of-the-art in image processing [44], allowing for the development of highly complex and deep architectures, using hundreds to thousands of layers [45]. These techniques, aimed at enhancing the propagation of data through the network, have been successfully adopted in several HSI classification works [46]–[48].

However, ResNets exhibit some shortcomings in terms of architecture optimization. In fact, residual-based models for HSI classification are quite sensitive to minor architectural changes, in particular, the selection of an appropriate kernel size has a significant impact on the final classification accuracy due to the low spatial resolution of HSI data cubes [47]. In contrast, at certain levels of depth, adding more or less layers to the network does not impact the classification result significantly [48]. In turn, this obviously affects the number of parameters that must be stored and trained. The understanding of the optimal number of parameters required by a certain architecture (the number of layers, kernel sizes, and so on) is quite critical, but it is often hand-crafted and adjusted by trial and error.

B. Rethinking the ResNet Model for HSI Classification

DNNs (in general) and ResNets (in particular) have been interpreted as a discrete sequence of L stacked layers, where each one applies its transformation to the input data until reaching a final classification decision, which is performed by the last layer. This implies that the ResNet model is evaluated at fixed intervals of "time," defined by the layer depth. Also, assuming that each layer has the same number of neurons n_{neurons} (which can be interpreted as the kernel's size in the convolutional architecture), the number of trainable parameters depends directly on L, so the complexity of the network (and its memory consumption) grows linearly with the $\mathcal{O}(L)$ order, which could have an impact on the model's overfitting. Under the same assumption, the computational time of the inference stage also depends on L.

The aforementioned implications provide an idea of the importance of the model's depth. As a result, the selection of L must be carefully done. In fact, the main goal of this paper is focused on two important aspects: 1) checking the effects of the depth when $L \to \infty$ and 2) analyzing strategies to provide the network with constant and low memory cost (in terms of the number of parameters). In this context, the FE function applied by each residual unit can be interpreted as the explicit Euler discretization of a continuous-time transformation [49], [50]. Following this interpretation, the entire ResNet model can be described through an ordinary differential equation (ODE) [51], [52], whose evaluation at different times will determine the model's solution [53], [54].

With the aforementioned ideas in mind, the main contribution of this article is to redefine the traditional architecture of the ResNet model (in the context of HSI data classification) by means of a continuous-time vision using ODEs, developing a residual-based DNN with a significantly reduced number of trainable parameters (thus effectively dealing with overfitting issues) and constant and low memory cost. These are important advantages in the area of HSI classification. More specifically, this article proposes, for the first time in the literature, the implementation of a continuous-depth ResNet with a parameterized spectral–spatial ODE in order to perform HSI data classification.

The remainder of this article is organized as follows. Section II introduces our newly developed model (called hereinafter ODEnet). Section III validates the newly proposed model by providing a detailed discussion of the results obtained using four widely used HSI data sets. Finally, Section IV concludes this article with some remarks and hints at plausible future research lines.

II. Methodology

A. Residual Units as Discrete Steps of Blocks

DNN architectures are stacks of L hidden blocks [55] F_1–F_L, where each one F_l is given by the following mapping function:

$$\mathbf{X}^{(l)} = F_l(\mathbf{X}^{(l-1)}, \mathbf{W}^{(l)}, b^{(l)}) \tag{1}$$

where $\mathbf{X}^{(l-1)}$ and $\mathbf{X}^{(l)}$ are the input and output data, respectively, and $\mathbf{W}^{(l)}$ and $b^{(l)}$ are the weights and biases of the lth mapping function F_l. In order to address the classification problem $f : \mathcal{X} \to \mathcal{Y}$, the DNN model assigns a classification map $\mathbf{Y} \in \mathbb{R}^{n_{\text{samples}}}$ to the given input $\mathbf{X} \in \mathbb{R}^{n_{\text{samples}} \times n_{\text{bands}}}$ by applying L sequential operations defined by (1). In this sense, the classification function $f(\cdot, \theta)$ can be reinterpreted as the concatenation of the processing at each layer processing as follows:

$$\mathbf{Y} = f(\mathbf{X}, \theta) = \hat{F}(F_L(F_{L-1}(\cdots F_1(\mathbf{X})\cdots))) \tag{2}$$

where $\mathbf{X}$ is the original input data, $F_l(\cdot)$ is the mapping function performed by the lth network's block, and $\hat{F}(\cdot)$ is the final classification layer, while θ comprises the network's parameters [49]. In this regard, instead of considering the classification mapping as a global problem, the DNN model splits it into L mapping functions F_l, where the goal of the classification is to learn the parameters of each F_l that better minimize the convex loss function given by the following:

$$E = \frac{1}{n_{\text{samples}}} \sum_{i=1}^{n_{\text{samples}}} \parallel f(\mathbf{X}_i, \theta) - \mathbf{X}_i \parallel_2 . \tag{3}$$

If we focus on convolutional-based models, the data transformation defined by (1) is tailored in an FE stage defined by a *kernel operation* [36], which allows to easily combine the spatial–contextual information with the spectral information.

In this context, the CNN maintains the original 3-D data structure, adding a lot of flexibility to the model and a natural way to include the spectral–spatial information. Moreover, the internal structure of the CNN's layers and their operations (based on local receptive fields) have promoted it as a highly accurate feature extractor.

Two main parts can be observed in an end-to-end CNN classifier network: 1) the *FE stack*, which obtains high-level representations of the input data (also feature maps) and is usually comprised of a hierarchy of convolutional, nonlinear, and subsampling layers, among others and 2) the *FC classifier*, which actually labels the data from the previously obtained feature maps and is implemented as a standard MLP.

Focusing on the FE stack, it is usually adopted to implement an architecture of several hierarchically stacked extraction and detection stages, where the lth stage defines the lth mapping function F_l, following the notation of (1). Moreover, each F_l is usually comprised of: 1) the *convolutional layer*; 2) the *nonlinear layer*; 3) the *normalization layer*; and 4) the *pooling layer*, as (4) shows, although both the order and the type of layers may vary from one CNN architecture to another (even from one stage to another)

$$\mathbf{A}^{(l)} = (\mathbf{W}^{(l)} *_{k\times k\times q} \mathbf{X}^{(l-1)}) + b^{(l)} \tag{4a}$$

$$\hat{\mathbf{A}}^{(l)} = \frac{\mathbf{A}^{(l)} - \text{mean}[\mathbf{A}^{(l)}]}{\sqrt{\text{var}[\mathbf{A}^{(l)}] + \epsilon}} \cdot \gamma + \beta \tag{4b}$$

$$\tilde{\mathbf{A}}^{(l)} = \mathcal{H}(\hat{\mathbf{A}}^{(l)}) \tag{4c}$$

$$\mathbf{X}^{(l)} = \mathcal{P}_{k\times k}(\tilde{\mathbf{A}}^{(l)}). \tag{4d}$$

The convolutional layer performs the basic FE task of the model. The spectral–spatial convolutional layer of the F_l mapping function is comprised of a group of K filters with $\mathbf{W}^{(l)} \in \mathbb{R}^{k\times k\times q}$ weights and $b^{(l)}$ biases, being $k \times k \times q$ the local receptive field of the layer. In consequence, each layer creates a linear kernel that slides (following a stride s) and overlaps the input data, convolving ($*$) its filters on local patches of the data, as (4a) indicates. As a result, the obtained output volume is comprised of K feature maps.

After the convolutional layer, it is common to include a batch normalization layer, which imposes a Gaussian distribution on the obtained feature maps with the aim of preventing the data degradation and vanishing gradient problems (mainly due to the covariance shift that the data suffers). Equation (4b) gives the regularization expression, where ϵ is a parameter that allows a certain numerical stability and γ and β are learnable parameters.

Following the normalization layer, a nonlinear layer $\mathcal{H}(\cdot)$ defined by (4c) is introduced in order to extract the activation maps from the convolutional output volume. In fact, this layer embeds a nonlinear activation function, which encodes the detector stage of the network [56], learning the nonlinear representations and relationships inside the data. Many activation functions can be selected, such as the tanh, sigmoid, or rectified linear unit (ReLU) [57], which allows a faster training of the model due to its high computational efficiency.

Finally, the extraction and detection stage ends with the pooling layer $\mathcal{P}_{k\times k}(\cdot)$ given by (4d), which performs a downsampling strategy with the aim of reducing the spatial dimensions of the output volume by applying, for instance, a max, average, or sum operation on the spatial receptive field of dimensions $k \times k$.

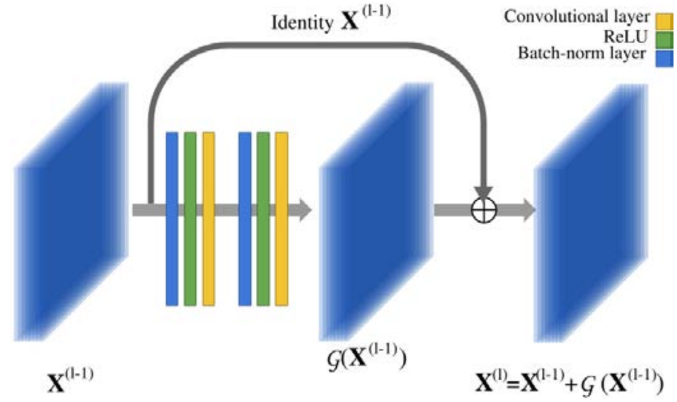


Fig. 1. Graphical representation of the lth residual unit architecture, F^l, comprised of two FE and detection stages. Each stage is comprised of normalization, nonlinear, and convolutional layers. The application of these stages gives, as a result, the output volume $\mathcal{G}(\mathbf{X}^{(l-1)})$, to which an identity mapping is added at the end of the residual unit, obtaining the final residual output volume $\mathbf{X}^{(l)} = \mathbf{X}^{(l-1)} + \mathcal{G}(\mathbf{X}^{(l-1)})$.

Based on the CNN architecture, the success of the ResNet model lies in the skip and residual connections, in which grouped operation layers (i.e., convolutional, pooling, and normalizing layers) and nonlinear activation functions comprise of the basic blocks for data mapping [47], as shown in Fig. 1. These residual units allow for the development of deeper architectures, where the inputs and outputs of each unit are connected through a residual connection, performing an additional identity mapping that allows to propagate the information from previous blocks to the rest of the network. In this context, for the lth residual unit, the FE and detection stages given by (4) can be reformulated as follows:

$$\mathbf{A}^{(l)} = \mathbf{X}^{(l-1)} + \mathcal{G}(\mathcal{W}^{(l)}, \mathbf{X}^{(l-1)}, \mathcal{B}^{(l)}) \tag{5a}$$

$$\mathbf{X}^{(l)} = \mathcal{H}(\mathbf{A}^{(l)}) \tag{5b}$$

where $\mathcal{G}(\cdot)$ comprises all the operations applied over the residual unit's input data, i.e., all the convolutions, poolings, normalizations, and activations applied over $\mathbf{X}^{(l)}$, being $\mathcal{W}^{(l)}$ and $\mathcal{B}^{(l)}$ the weights and biases of the layers involved in the residual block, respectively. Moreover, the additive residual mapping function added to $\mathcal{G}(\cdot)$ allows to recycle the features obtained at the previous level of abstraction.

Following (2), the ResNet defines each mapping function F_l through (5). In this context, the neural model can be interpreted as a discrete sequence of L hidden units or mapping functions, dividing the classification process into L steps, so that each F_l defines a hidden *state* of the process, which becomes more manageable, with simple and detailed steps that allow for a more accurate final classification. However, this implies that the quality of the model depends on its trainable parameters, and the number of trainable parameters depends directly on L. This has two main implications. On the one hand, the memory consumption grows linearly [with $\mathcal{O}(L)$ order] and there is an increment of training data that the model must assume in order to properly learn the network's parameters, avoiding the overfitting problem. On the other hand, although the residual connections alleviate the aforementioned problem, each new unit that is added to the model introduces a small error [38], [58], which may hinder the model's overall performance. These issues are particularly critical when dealing with highly variable HSI data sets.

B. Residual Units as Discrete Steps of ODEs

Our goal is to develop a residual model with constant and low memory cost through a significant reduction of the number of trainable parameters. We follow the premise of traditional optimization models: solving a lot of small steps is often better than solving fewer and more complex ones [50]. In this sense and following (2), we propose to implement an ResNet model for HSI data classification in which the forward problem is comprised of infinitesimal steps, i.e., $L \rightarrow \infty$ [54]. Each of these steps performs (5), which describes an explicit Euler discretization step of the ODE [51], [52]. Below, the mathematical relationship between ResNet models and ODEs is described in detail.

We focus on the first-order ODE expressions. Following Euler's solving method, any first-order ODE can be expressed as an initial value problem (IVP) of the form:

$$\frac{\mathrm{d}\mathbf{z}(t)}{\mathrm{d}t} = f(t, \mathbf{z}(t), \theta), \text{ with } \mathbf{z}(t_0) = \mathbf{z}_0 \tag{6}$$

where t_i is an independent variable defined in terms of *time* in an observation interval $\{0, ..., T\}$, $f(\mathbf{z}(t), t, \theta)$ is a known and continuous function with parameter θ, and $\mathbf{z}(t)$ is the unknown function that must be approximated, with initial state $\mathbf{z}_0$ at time t_0. In fact, the goal of any ODE function is to recover the closest and most accurate value $\mathbf{z}_i$ of the unknown function $\mathbf{z}(t_i)$ at each observation point t_i.

From a geometric point of view, knowing $\mathbf{z}(t_0) = \mathbf{z}_0$, an approximation of $\mathbf{z}(t_i) = \mathbf{z}_i$ in any step t_i can be performed by drawing the tangent line from previous-known points as follows:

$$\mathbf{z}_1 \approx \mathbf{z}_0 + f(t_0, \mathbf{z}_0, \theta)(t_1 - t_0) \tag{7a}$$
$$\mathbf{z}_i \approx \mathbf{z}_{i-1} + f(t_{i-1}, \mathbf{z}_{i-1}, \theta)(t_i - t_{i-1}) \tag{7b}$$
$$\mathbf{z}_T \approx \mathbf{z}_{T-1} + f(t_{T-1}, \mathbf{z}_{T-1}, \theta)(t_T - t_{T-1}). \tag{7c}$$

Generalizing the discrete steps defined above, it can be stated that any $\mathbf{z}(t_i)$ can be approximated by (7b). Assuming that the ith observation point is connected to the first one (following the relation $t_i = t_0 + \alpha \cdot i$, where α is a step-size), the Euler discretization method claims that each point t_i is related to the immediately preceding one, t_{i-1}, through the step-size α as follows: $t_i = t_{i-1} + \alpha$. Including this relationship in (7b), Euler's method gives a solution for $\mathbf{z}(t_i)$ as

$$\mathbf{z}_i = \mathbf{z}_{i-1} + f(t_{i-1}, \mathbf{z}_{i-1}, \theta) \cdot (\cancel{t_{i-1}} + \alpha - \cancel{t_{i-1}}) \tag{8a}$$
$$\mathbf{z}_i = \mathbf{z}_{i-1} + \alpha \cdot f(t_{i-1}, \mathbf{z}_{i-1}, \theta). \tag{8b}$$

At this point, it is easy to observe the relationship between the ResNet model and the first-order ODE. Focusing on (5), we can simplify it into a more condensed form

$$\mathbf{X}^{(l)} = \mathbf{X}^{(l-1)} + \mathcal{G}(\mathbf{X}^{(l-1)}, \theta_l). \tag{9}$$

The similarities between (8b) and (9) are evident. In fact, (9) defines an explicit Euler discretization step of the first-order ODE, where the step size is set to $\alpha = 1$ and the known function is implemented by the extraction and detection stages $\mathcal{G}(\cdot)$ of the residual unit, being parameterized by the weights and biases of the layers that comprise the residual unit $\theta_l = (\mathcal{W}^{(l)}, \mathcal{B}^{(l)})$. In other words, the ODE function is, in fact, a CNN.

Following this intuition, we can replace the discrete block-by-block performance of a ResNet model by a continuous-time ODE function. In particular, we assume a residual model with $L \rightarrow \infty$ equal residual units. In this sense, each mapping function F_l has to perform the same extraction and detection stages in $\mathcal{G}(\cdot)$, so each unit has the same number of parameters θ and works in the same feature space $F_1, \ldots, F_L \in \mathbb{R}^{\hat{n}_1 \times \hat{n}_2 \times \hat{n}_3}$, where $\hat{n}_1 \times \hat{n}_2 \times \hat{n}_3$ are the spatial–spectral dimensions of the feature maps.

In this way, the successive transformations given by (2), $F_1, \ldots, F_L$, can be interpreted as the continuous mapping function $F(t)$ evaluated at different times (with a relationship between layers and time). So, at the ith observation time, we can obtain $F(t_i) = \mathbf{X}_i$. As a result, the residual model can be reformulated as the ODE in (10b), which gives the discretization step of Euler's method and the expression of the first-order ODE

$$\mathbf{X}_i = \mathbf{X}_{i-1} + \mathcal{G}(t_{i-1}, \mathbf{X}_{i-1}, \theta) \tag{10a}$$

where

$$\frac{\mathrm{d}F(t)}{\mathrm{d}t} = \mathcal{G}(t, F(t), \theta), \text{ with } F(t_0) = \mathbf{X}_0. \tag{10b}$$

As it can be observed, the ODE is implemented by the neural network defined by $\mathcal{G}(\cdot)$ and parameterized by θ.

C. Proposed ODEnet for HSI Classification

We propose, for the first time in the literature, to reinterpret the ResNet model (for HSI data classification) as a continuous transformation given by the first-order ODE described in (10b). Fig. 2 gives a general overview of the proposed ODEnet, which receives as input the HSI data cube with dimensions $\mathbf{X} \in \mathbb{R}^{d \times d \times n_{\text{bands}}}$. In fact, the model is fed with hyperspectral patches cropped from the original HSI cube, comprised of $d \times d$ pixels and n_{bands} spectral bands, where the label corresponds to the central pixel of the patch. Also, in order to take advantage of border pixels, a mechanism for mirroring the image edges has been implemented [36].

The proposed network architecture is divided into the FE layers and the final classification layers. Focusing on the FE layers, they are grouped into three categories: 1) FE head; 2) FE body; and 3) FE tail. The FE head performs a downsampling of the data, reducing noise, and cleaning the information contained in the input. It is comprised of a convolutional layer F_1 and a residual unit F_2. F_1 prepares the input data, extracting the initial features from the HSI cube, which are fundamental to the performance of the rest of the layers. During the training process, these features will become more and more robust and discriminative, being decisive for the final classification. F_2 has been implemented following the *preactivation* architecture

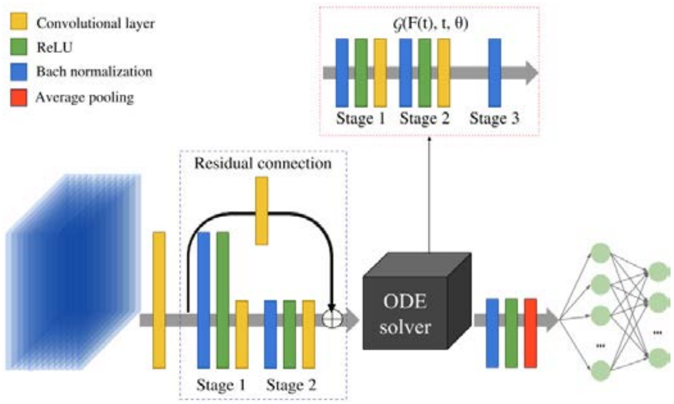


Fig. 2. Architecture of the proposed ODEnet for HSI data classification. The feature extractor part is comprised of three well-differentiated parts: 1) a preprocessing step that filters the spatial–spectral noise and extracts low-level feature representations; 2) the ODE solver that evaluates the function $\mathcal{G}(\cdot)$ defined by a neural network; and 3) the ODE solver's output, after being refined, is employed to perform the final classification, implemented by two FC layers.

proposed in [45], performing data downsampling, and it is comprised of two FE and detection stages with normalization, nonlinear, and convolutional layers, adding a convolutional layer on the skip connection to maintain the data shape.

The obtained features are sent to the FE body, which is implemented by a continuous-time ResNet. In this context, the ODE implemented by (10b) has been parameterized by a CNN model. As Fig. 2 shows, this model follows the *preactivation* architecture [45] and has three stages, where each one is comprised of normalization, nonlinear, and convolutional layers (stages 1 and 2), and a normalization layer (stage 3). This ODE is solved from some initial time t_0 to some ending time t_T, creating an integration time interval $[0, T]$. Furthermore, during each forward pass, the traditional discrete-layer execution of the model is eventually replaced by $\hat{L}$ evaluations of (10b), performed by a black-box solver in the interval $[0, T]$, which receives as the initial condition $\mathbf{X}_0$ the output of F_2, the known function $\mathcal{G}(\cdot)$ and its parameters θ, in addition to the integration time interval, and a tolerance threshold of the estimated error, tol

$$F(t_T) = \mathbf{X}_T = \text{ODEsolver}(\mathbf{X}_0, \mathcal{G}, \theta, [t_0, t_T], \text{tol}). \quad (11)$$

Equation (11) can be performed by any off-the-shelf ODE solver. There is a great variety of methods for this purpose, grouped in different categories depending on their internal characteristics and working modes [59], being some of the methods framed within the most well-known Runge–Kutta family, which are as follows.

1) *Forward Euler:* This is the most popular numerical explicit method for solving the first-order ODEs. It is also the simplest method to implement, where the new states are obtained through previously known ones by the intersection of tangent lines, as (8) shows. Given the first-order ODE of (6) and using α as the step size, the approximation error of Euler's discretization method will be proportional to $\mathcal{O}(\alpha^2)$.
2) *Explicit Midpoint Method also Known as the Modified Euler method:* Given (6), the evaluations are made at $\alpha/2$, so this method determines the value $\mathbf{z}(t_i) = \mathbf{z}_i$ as the following approximation:

$$\mathbf{z}_i = \mathbf{z}_{i-1} + \alpha \cdot f\left(t_{i-1} + \frac{\alpha}{2}, \mathbf{z}_{i-1} + \frac{\alpha}{2} \cdot k_1\right) \quad (13a)$$
$$k_1 = f(t_{i-1}, \mathbf{z}_{i-1}) \quad (13b)$$

This method reduces the estimation error when Euler's step size is too high and the tangent needs to be elongated to find the intersection point.
3) *Fourth-Order Runge–Kutta (RK4) Method:* This is the most widely used method of the Runge–Kutta family. Inspired by the midpoint method, the basic idea is that, given two equidistant points $t_i = t_{i-1} + \alpha$, the function $\mathbf{z}(t_i) = \mathbf{z}_i$ can be approximated as the sum of the previously known value and the weighted average of s slopes [60]

$$\mathbf{z}_i = \mathbf{z}_{i-1} + \sum_{n=1}^{s} b_n, k_n \quad (14a)$$
$$k_1 = \alpha f(t_{i-1}, \mathbf{z}_{i-1}) \quad (14b)$$
$$k_n = \alpha f\left(t_{i-1} + c_n\alpha, \mathbf{z}_{i-1} + \sum_{\hat{n}=1}^{n-1} a_{n,\hat{n}} k_{\hat{n}}\right) \quad (14c)$$

where $a_{n,\hat{n}}$, b_n, and c_n are weighted coefficients. In this sense, given (6), the RK4 method determines the value at t_i as an approximation of the previously known $\mathbf{z}_{i-1}$ and the weighted average of four increments: $(k_1+2k_2+2k_3+k_4)/6$, which are calculated on certain points of the slope defined by $f(\mathbf{z}(t), t, \theta)$, in particular, the starting, ending, and midpoints [61]

$$\mathbf{z}_i = \mathbf{z}_{i-1} \cdot \frac{1}{6}(k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4) \quad (15a)$$
$$k_1 = \alpha f(t_{i-1}, \mathbf{z}_{i-1}) \quad (15b)$$
$$k_2 = \alpha f\left(t_{i-1} + \frac{\alpha}{2}, \mathbf{z}_{i-1} + \frac{k_1}{2}\right) \quad (15c)$$
$$k_3 = \alpha f\left(t_{i-1} + \frac{\alpha}{2}, \mathbf{z}_{i-1} + \frac{k_2}{2}\right) \quad (15d)$$
$$k_4 = \alpha f\left(t_{i-1} + \alpha, \mathbf{z}_{i-1} + k_3\right). \quad (15e)$$

Following (15), the approximation error is proportional to $\mathcal{O}(\alpha^4)$, being more precise than the two previous methods.
4) *Dormand-Prince Method (DOPRI5):* This is an explicit and adaptive Runge–Kutta method to calculate the fourth- and fifth-order solutions. In fact, following (14), it calculates seven slopes: k_1–k_7, which are employed to calculate two approximations of $\mathbf{z}(t_i) = \mathbf{z}_i$ by two different linear combinations. Equation (12a), as shown at the bottom of the next page, gives the first approximation, with $\mathcal{O}(\alpha^4)$ order, while (12b) gives the second approximation, with $\mathcal{O}(\alpha^5)$ order. An interesting aspect of the DOPRI5 solver is its ability to adapt the step size α to keep the estimated error $|\hat{\mathbf{z}}_i - \mathbf{z}_i|$ below a predefined threshold. The updating of the optimal step size α_{opt} is obtained as

$$s = \left(\frac{\text{tol} \cdot \alpha}{2|\hat{\mathbf{z}}_i - \mathbf{z}_i|}\right)^{\frac{1}{5}} \quad (16a)$$
$$\alpha_{\text{opt}} = s \cdot \alpha \quad (16b)$$

TABLE I
PROPOSED NETWORK TOPOLOGY

Feature extraction network								
	Module ID	Sub-module	Norm.	Activation	Kernel	Stride	Padding	Pooling
	F_1	-	-	-	$64 \times 3 \times 3 \times n_3$	1	-	-
FE-head	F_2	Stage 1	Yes (64)	ReLU	$64 \times 3 \times 3 \times 64$	2	Yes	-
		Stage 2	Yes (64)	ReLU	$64 \times 3 \times 3 \times 64$	1	-	-
		Skip connection	-	-	$64 \times 1 \times 1 \times 64$	2	-	-
FE-body	$\mathcal{G}(\cdot)$	Stage 1	Yes (64)	ReLU	$64 \times 3 \times 3 \times 64$	1	Yes	-
		Stage 2	Yes (64)	ReLU	$64 \times 3 \times 3 \times 64$	1	Yes	-
		Stage 3	Yes (64)	-	-	-	-	-
FE-tail	F_3	Stage 1	Yes (64)	ReLU	-	-	-	Average (1×1)

Classification network			
	Layer	Neurons	Activation
MLP	F_4	64	ReLU
	F_5	$n_{classes}$	Softmax

where tol defines the tolerance level, which provides robustness and reliability to the model.

In addition to obtaining the corresponding state $\mathbf{X}_T = F(t_T)$ at t_T (forward-propagation), the ODEsolver should optimize the network's parameters associated with the differential equation $\mathcal{G}(t, F(t), \theta)$ by backpropagating the internal error signal $E_{\text{ode}}(\cdot)$ defined by the following expression:

$$E_{\text{ode}}(F(t_T)) = E\left(F(t_0) + \int_{t_0}^{t_T} \mathcal{G}(t, F(t), \theta) dt\right). \tag{17}$$

This optimization can be implemented by two methods: 1) traditional integration through a *Runge–Kutta integrator*, for instance, or 2) employing the *adjoint method* [54], [62]. The first one directly integrates the operations of the forward pass and still presents an important memory requirement in the sense that, for $\hat{L}$ evaluations, the memory cost grows to the order of $\mathcal{O}(\hat{L})$. However, the adjoint method allows to optimize the parameters of $\mathcal{G}(\cdot)$ while significantly reducing their management, keeping constant the memory cost in the order $\mathcal{O}(1)$ [54].

Finally, the FE-layers end with the FE tail, which receives $\mathbf{X}_T$, the estimated output of the ODEsolver at evaluation time t_T, and performs a final processing. This entails an FE and detection stages, denoted as F_3, which comprises normalization, nonlinear, and average pooling layers. The obtained feature maps are then reshaped and sent to the classifier, which has been implemented as an MLP with two FC layers: F_4 and F_5, where the last one produces the final classification.

Table I gives the topology details of the proposed ODEnet. Moreover, our ODEnet model has been trained by the stochastic gradient descend (SGD) optimizer to minimize the classification loss given by (3), with input patches of 11×11, using 160 epochs and 0.1 as the learning rate, taking into account a momentum of 0.9 and learning rate decay, and a batch size of 128, while the ODEsolver is implemented via the DOPRI5 solver with a tolerance fixed to $\text{tol} = 1e - 3$ and an integration time interval of [0, 1], which directly controls the number of evaluations $\hat{L}$ of the model by obtaining the optimal step size α.

$$\mathbf{z}_i = \mathbf{z}_{i-1} + \frac{35k_1}{384} + \cancel{0k_2} + \frac{500k_3}{1113}\frac{125k_4}{192} - \frac{2187k_5}{6784} + \frac{11k_6}{84} + \cancel{0k_7} \tag{12a}$$

$$\hat{\mathbf{z}}_i = \mathbf{z}_{i-1} + \frac{5179k_1}{57600} + \cancel{0k_2} + \frac{7571k_3}{16695}\frac{393k_4}{640} - \frac{92097k_5}{339200} + \frac{187k_6}{2100} + \frac{k_7}{40} \tag{12b}$$

$$k_1 = \alpha f(t_{i-1}, \mathbf{z}_{i-1}) \tag{12c}$$

$$k_2 = \alpha f\left(t_{i-1} + \frac{\alpha}{5}, \mathbf{z}_{i-1} + \frac{k_1}{5}\right) \tag{12d}$$

$$k_3 = \alpha f\left(t_{i-1} + \frac{3\alpha}{10}, \mathbf{z}_{i-1} + \frac{3k_1}{40} + \frac{9k_2}{40}\right) \tag{12e}$$

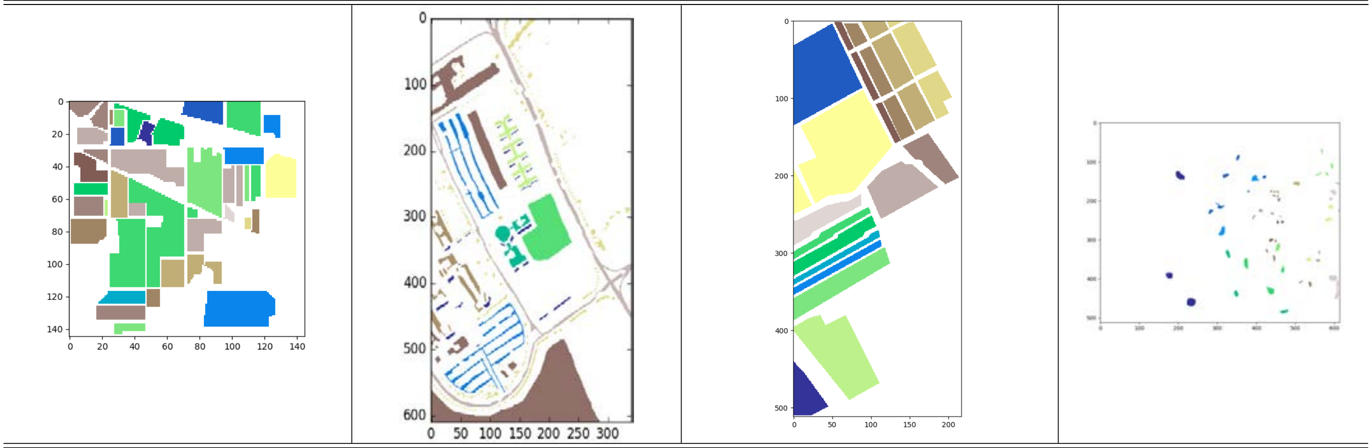
$$k_4 = \alpha f\left(t_{i-1} + \frac{4\alpha}{5}, \mathbf{z}_{i-1} + \frac{44k_1}{45} - \frac{56k_2}{15} + \frac{32k_3}{9}\right) \tag{12f}$$

$$k_5 = \alpha f\left(t_{i-1} + \frac{8\alpha}{9}, \mathbf{z}_{i-1} + \frac{19372k_1}{6561} - \frac{25360k_2}{2187} + \frac{64448k_3}{6561} - \frac{212k_4}{729}\right) \tag{12g}$$

$$k_6 = \alpha f\left(t_{i-1} + \alpha, \mathbf{z}_{i-1} + \frac{9017k_1}{3168} - \frac{355k_2}{33} - \frac{46732k_3}{5247} + \frac{49k_4}{176} - \frac{5103k_5}{18656}\right) \tag{12h}$$

$$k_7 = \alpha f\left(t_{i-1} + \alpha, \mathbf{z}_{i-1} + \frac{35k_1}{384} + \cancel{0k_2} + \frac{500k_3}{1113}\frac{125k_4}{192} - \frac{2187k_5}{6784} + \frac{11k_6}{84}\right) \tag{12i}$$

INDIAN PINES (IP)			UNIVERSITY OF PAVIA (UP)			SALINAS VALLEY (SV)			KENNEDY SPACE CENTER (KSC)		



Color	Land-cover type	Samples	Color	Land-cover type	Samples	Color	Land-cover type	Samples	Color	Land-cover type	Samples
	Background	10776		Background	164624		Background	56975		Background	309157
	Alfalfa	46		Asphalt	6631		Brocoli-green-weeds-1	2009		Scrub	761
	Corn-notill	1428		Meadows	18649		Brocoli-green-weeds-2	3726		Willow-swamp	243
	Corn-min	830		Gravel	2099		Fallow	1976		CP-hammock	256
	Corn	237		Trees	3064		Fallow-rough-plow	1394		Slash-pine	252
	Grass/Pasture	483		Painted metal sheets	1345		Fallow-smooth	2678		Oak/Broadleaf	161
	Grass/Trees	730		Bare Soil	5029		Stubble	3959		Hardwood	229
	Grass/pasture-mowed	28		Bitumen	1330		Celery	3579		Swap	105
	Hay-windrowed	478		Self-Blocking Bricks	3682		Grapes-untrained	11271		Graminoid-marsh	431
	Oats	20		Shadows	947		Soil-vinyard-develop	6203		Spartina-marsh	520
	Soybeans-notill	972					Corn-senesced-green-weeds	3278		Cattail-marsh	404
	Soybeans-min	2455					Lettuce-romaine-4wk	1068		Salt-marsh	419
	Soybean-clean	593					Lettuce-romaine-5wk	1927		Mud-flats	503
	Wheat	205					Lettuce-romaine-6wk	916		Water	927
	Woods	1265					Lettuce-romaine-7wk	1070			
	Bldg-Grass-Tree-Drives	386					Vinyard-untrained	7268			
	Stone-steel towers	93					Vinyard-vertical-trellis	1807			
	Total samples	21025		Total samples	207400		Total samples	111104		Total samples	314368

Fig. 3. Number of available labeled samples in the IP, UP, SV, and KSC HSI data sets.

III. Experimental Results

A. Experimental Environment

In order to study the performance of the proposed ODEnet for HSI classification, an implementation has been developed and tested on a hardware environment with a sixth-generation Intel Core i7-6700 K processor with 8M of Cache and up to 4.20 GHz (four cores/eight-way multitask processing), installed over an ASUS Z170 pro-gaming motherboard. The available memory is 40 GB of DDR4 RAM with serial speed of 2400 MHz and a Toshiba DT01ACA HDD with 7200 RPM and 2 TB of storage capacity. Also, a graphic processing unit (GPU) NVIDIA GeForce GTX 1080 with 8-GB GDDR5X of video memory and 10 Gb/s of memory frequency is available. In order to provide an efficient implementation, the proposed model has been parallelized over the GPU using CUDA 9.0 and cuDNN 7.1.1 language over the Pytorch framework, with Ubuntu 18.04.1 $\times$ 64 as the operating system.

B. Hyperspectral Data Sets

Fig. 3 presents the four real HSI data sets that have been considered in our experiments: Indian Pines (IP), Salinas Valley (SV), and Kennedy Space Center (KSC) scenes, acquired by the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) sensor [5], and the University of Pavia (UP) scene, captured by the Reflective Optics System Imaging Spectrometer (ROSIS) sensor [63]. A detailed description of these images is provided in the following.

1) The IP scene comprises an area with different agricultural fields in Northwestern Indiana, USA, imaged during a flying campaign of the AVIRIS sensor in 1992. The scene contains 145 $\times$ 145 samples, where each one comprises 20 m, and the spectral information consists of 200 bands in the wavelength range from 0.4 to 2.5 μm, after removing 24 noisy and corrupted bands. As it can be observed in Fig. 3, the ground truth of the IP scene contains a total of 16 different classes.
2) The UP image was acquired in 2001 by the ROSIS sensor over the UP, Northern Italy, capturing an urban area of 610 $\times$ 340 pixels, where each one comprises 1.3 m, and with spectral (103 bands, after elimination of noisy and corrupted bands) in the wavelength range from 0.43 to 0.86 μm. The number of different classes contained in the UP scene is nine.
3) The SV image was captured during a flying campaign of the AVIRIS sensor in 1998 over the agricultural area described as SV in CA, USA. The data comprise 512 $\times$ 217 pixels with the spatial resolution of 3.7 m/pixel and 200 spectral bands in the range from 0.4 to 2.5 μm (200 bands, after elimination of the noisiest bands). The available ground truth for the SV scene contains 16 classes.

4) Finally, the KSC scene was also gathered by the AVIRIS instrument in 1996 over the KSC in FL, USA. In this scene, 512×614 pixels were obtained with the spatial resolution of 20 m/pixel. The data comprises 176 spectral bands in the range from 0.4 to 2.5 μm after the removal of noisy bands. The available ground truth for this scene comprises 13 different classes.

C. Experimental Setting

To evaluate the classification performance of the proposed ODEnet for HSI classification, three widely used quantitative metrics have been considered: the overall accuracy (OA), average accuracy (AA), and Kappa coefficient. Moreover, the number of model's parameters and execution times has also been measured to determine the volume of data to be trained and the computational cost. In this regard, with the aim of providing a complete and detailed experimentation, several experiments have been carried out, which are as follows.

1) Our first experiment evaluates the performance of the proposed ODEnet by implementing it with different ODEsolvers, in particular, forward Euler (EULER), explicit midpoint (MIDPOINT), RK4, and DOPRI5. For this experiment, the IP data set has been considered, selecting randomly 10% of the available labeled samples for training and using the remaining 90% of the samples for testing, setting the tolerance threshold to tol $= 1e{-}3$. Each experiment has been executed ten times, and the average and standard deviations have been reported.
2) Our second experiment focuses on the DOPRI5 solver due to its ability to adapt the step size α, adapting, in turn, the number of evaluations $\hat{L}$ contained in the defined integration time interval $[t_0, t_T]$ to the complexity of the function, as opposed to the EULER, MIDPOINT, and RK4 methods that set a fixed size for α, making the same number of evaluations in each step. In this regard, our second experiment analyzes the behavior of the DOPRI5 solver with different tolerance thresholds, in particular: tol $= \{1e-1,\ 1e-2,\ 1e-3\ 1e-4,\ 1e-5\}$. For this purpose, the OA values, the number of evaluations during the forward and backward steps, and the training execution times have been measured. Again, in this experiment, we randomly select 10% of the available labeled samples of the IP data set for training and use the remaining 90% for testing. Each experiment is executed ten times and the average and standard deviations are reported.
3) Once the model's behavior has been evaluated with different solvers and tolerance levels, our third experiment performs several comparisons between the proposed ODEnet and the traditional ResNet model for spectral–spatial HSI data classification. In this context, this experiment compares the robustness of the models, analyzing their performance based on the amount of available training data, the number of parameters used by each model, and the evolution of the accuracy in each epoch. For a fair comparison, the ResNet has been implemented in the same way as the ODEnet, using the topology in Table I, and changing the ODEsolver by six residual units comprised of exactly the same stages as the proposed ODEnet's FE body, but adding the corresponding residual connections. Moreover, the proposed ODEnet has been implemented with the DOPRI5 solver, employing Runge–Kutta integration and adjoint methods and a tolerance threshold of $1e-3$. These models have been tested with all the available scenes. For the IP and KSC scenes, we have randomly selected 5%, 10%, and 15% of the available labeled samples for training and used the remaining samples for testing. The fact that we consider larger training percentages for these two images is due to the low spatial resolution and highly mixed nature of these scenes, which exhibit high intraclass variability. In turn, for the UP and SV scenes (which exhibit much larger spatial resolution), we have randomly selected 1%, 5%, and 10% of the available labeled samples for training, using the remaining samples for testing. In all the cases, we have executed each experiment ten times and the average and standard deviations are reported.
4) The fourth experiment compares the behavior of the proposed ODEnet models and the ResNet depending on different network configurations, in particular, the spatial windows' size of the network's input data and the depth of the convolutional filters. In this sense, the proposed models have been implemented with DOPRI5 during the forward pass, while employing both Runge–Kutta integrator and the adjoint method during the backward step. For each experiment, the 10% of IP and KSC and the 5% of UP and SV data sets have been considered to perform the training of the models.
   Regarding the first experiment, it compares the performance of the neural models when different amounts of spatial information confirm the network's input data. In this context, different window sizes have been considered, in particular input patches of 5×5, 7×7, 9×9, 11×11, 13×13, and 15×15 pixels have been tested. Separately, the second experiment compares the networks' behavior when the number of convolutional filters grows. In this regard, convolutional layers have been implemented with 8, 16, 32, 64, and 128 filters.
5) Our last experiment conducts a comparison of the proposed ODEnet with other widely used HSI classifiers. In this context, eight different classification methods have been selected to conduct the experimental validation. Specifically, three pixelwise classifiers (MLR, SVM with radial basis function kernel, and MLP), one deep spatial classifier (CNN2D), and three spectral–spatial deep architectures (CNN3D, ResNet, and the proposed ODEnet) have been considered. In this experiment, we have randomly selected 15% of the available labeled data from the IP and KSC scenes and used the remaining 85% of the labeled data for testing. Considering the higher spatial resolution of the UP and SV scenes, we have randomly selected 10% of the available labeled samples for these scenes and used the remaining 90% for testing. As in the previous experiments, we repeated

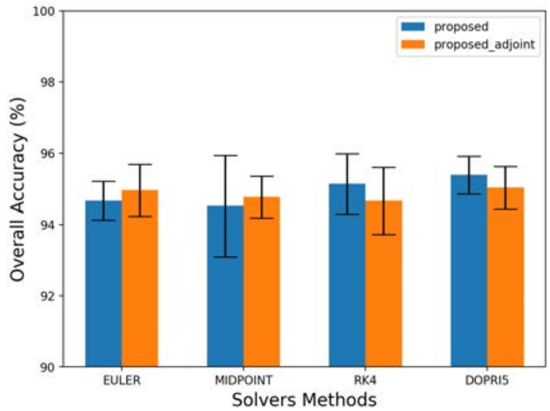


Fig. 4. OA values (and corresponding standard deviations) obtained by the proposed method (implemented with four different solvers with Runge–Kutta integration and adjoint methods) for the IP scene.

each experiment ten times and report the average and standard deviations. Moreover, for the spatial (CNN2D) and the spectral–spatial (CNN3D, ResNet, and ODEnet) methods, the original HSI scene has been cropped into patches of 11×11. In the case of the CNN2D, principal component analysis (PCA) has been used to reduce the number of spectral bands to a single principal component. All the hyperparameters of the considered methods have been optimally fixed to obtain the best possible performance for each method.

D. Experiment 1: Testing Different ODEsolvers

The performance of the proposed ODEnet depends on two main aspects: 1) the solver that performs the forward evaluation and 2) the backpropagation method that implements the reverse-mode differentiation. In this experiment, the fixed-α solvers: EULER, MIDPOINT, and RK4, and the adaptive solver: DOPRI5 have been compared using the IP data set, testing each one with Runge–Kutta integration (simply referred to ODEnet hereinafter) and the adjoint method (ODEnetAdj hereinafter).

Fig. 4 gives the obtained OA results and the standard deviations for each considered model. As a general comment, it should be noted that all methods achieve an OA greater than 94% with small differences between them. Specifically, the difference between the implementation of each solver with Runge–Kutta integration and adjoint method is very small, achieving very similar results.

If we compare the fixed-α solvers (EULER, MIDPOINT, and RK4) with the adaptive DOPRI5 solver, it can be observed that DOPRI5 reaches the best OA values for both back-propagation methods, Runge–Kutta integration, and adjoint, exceeding 95% OA with very low standard deviation, due to its capability of adapting the evaluations to the problem's complexity. Furthermore, MIDPOINT and RK4 exhibit the worse OA scores when implemented using Runge–Kutta integration and adjoint methods, respectively. In particular, the MID-POINT method implemented with Runge–Kutta integration exhibits the highest standard deviation, because the adopted approximation strategy performed by calculating the midpoint of the slope is not the most appropriate for complex data such as HSI scenes.

E. Experiment 2: Testing Different Tolerance Thresholds for DOPRI5 Solver

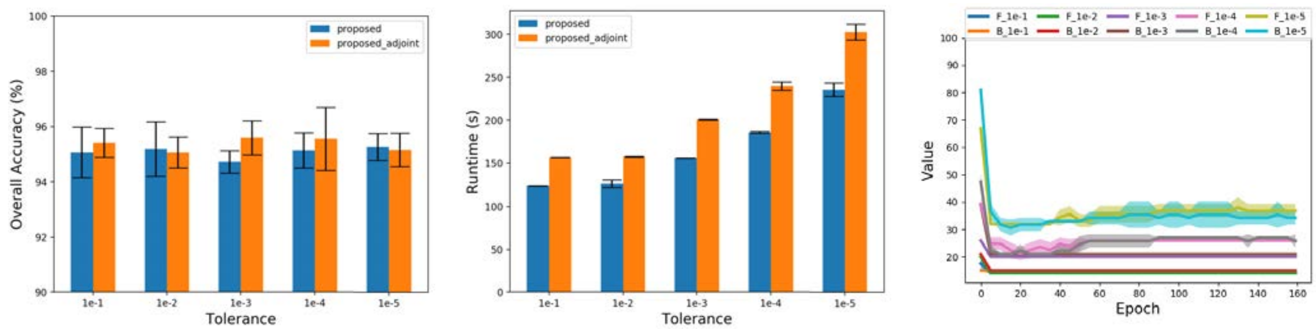
The DOPRI5 solver is able to adapt the step size α that controls the number of evaluation points ($\hat{L}$) carried out inside the integration time interval $[t_0, t_T]$, providing a flexible mechanism to adapt the ODE resolution to the complexity of the considered HSI data. In this sense, five different values for the tolerance threshold have been considered: $\{1e-1,\ 1e-2,\ 1e-3\ 1e-4,\ 1e-5\}$.

Fig. 5 shows the obtained results, comparing the obtained OA values [see Fig. 5(a)], the training runtimes [see Fig. 5(b)], and the number of evaluations performed during the forward and backward steps (for each tolerance value) [see Fig. 5(c)]. If we focus on Fig. 5(a), it can be observed that the tolerance threshold does not have a relevant impact on the OA values in the sense that the differences are very small and the slight variations are mainly due to the random procedure used for the selection of training samples.

However, if we focus on Fig. 5(b), it can be clearly observed that, for lower tolerances, the execution times gradually increase, being the implementations with DOPRI5 and adjoint method the slowest ones. This is due to the number of evaluations $\hat{L}$ that need to be carried out both in the forward evaluations and the backward propagation. To further investigate this issue, Fig. 5(c) focuses on the DOPRI5 solver implementation with the adjoint method. In general, the number of forward and backward evaluations, in this case, is high in the early epochs with the aim of adjusting them to minimize the approximation error, descending abruptly until the number becomes stable in subsequent epochs. In addition, for lower tolerances, it can be observed that the number of evaluations is higher than the tolerance values of $1e-1$ and $1e-2$, where the difference is minimal. With the aforementioned observations in mind, we consider a tolerance of $1e-3$ as a good choice, in the sense that it provides a good balance between performance and training times, together with a sufficiently high number of evaluations.

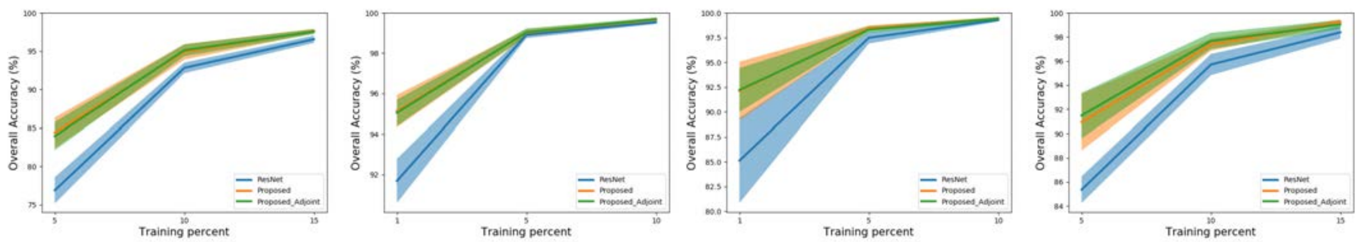
F. Experiment 3: Comparing ODEnet With ResNet

In this experiment, we illustrate the benefit of implementing a ResNet-inspired model as a continuous function defined by an ODE. Fig. 6 shows the OA evolution of the proposed ODEnet when different amounts of training samples are available. In general, the proposed method, implemented either with DOPRI5 and Runge–Kutta integrator (ODEnet) or with the adjoint method (ODEnetAdj), exhibits the best OA results for all the considered HSI scenes, regardless of the training percentage employed. The differences between our ODEnet/ODEnetAdj models and the ResNet become particularly evident when a very few training samples are available with the proposed models exhibiting the most robust results. Again, the observable differences between the Runge–Kutta integrator and the adjoint method are quite small, being the



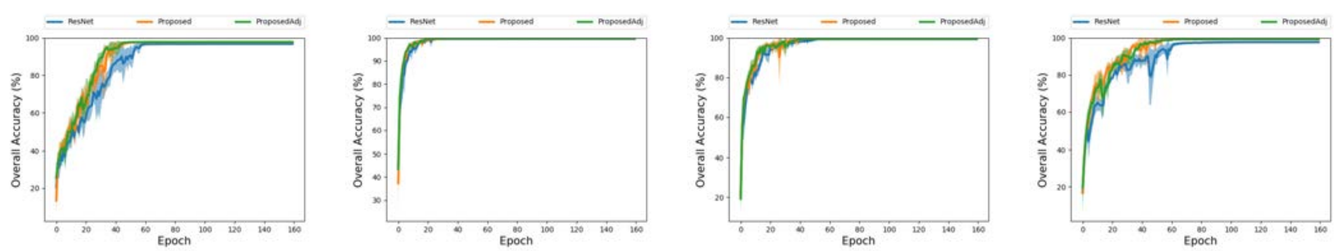
(a) (b) (c)

Fig. 5. Performance of ODEnet (on the IP scene) with the DOPRI5 solver, using Runge–Kutta integration and adjoint methods, considering different tolerance values. Specifically, we analyze the impact on (a) OA, (b) training runtimes, and (c) number of evaluations per epoch during the forward and backward steps of the DOPRI5 solver implemented with the adjoint method.



(a) (b) (c) (d)

Fig. 6. Evolution of the OA reached by ResNet (blue), the proposed ODEnet with the DOPRI5 solver and Runge–Kutta integration (orange), and the proposed ODEnet with the DOPRI5 solver and adjoint method (green), considering different amounts of training data. We report the results obtained for (a) IP, (b) UP, (c) SV, and (d) KSC scenes.



(a) (b) (c) (d)

Fig. 7. Evolution of the OA reached by ResNet (blue), the proposed ODEnet with the DOPRI5 solver and Runge–Kutta integration (orange), and the proposed ODEnet with the DOPRI5 solver and adjoint method (green) at different epochs. We report the results obtained for (a) IP, (b) UP, (c) SV, and (d) KSC scenes.

adjoint method better for KSC and SV scenes with low training percentages.

The aforementioned results clearly illustrate the impact that the overfitting of learnable parameters has on the ResNet model, which needs more training data to achieve the same performance as our ODEnet models. Moreover, Fig. 7 shows that this overfitting problem happens at early epochs of the classifiers. Specifically, it can be observed in this figure how the OA obtained by ODEnet increases faster than that achieved by ResNet in the earliest epochs, in particular when complex scenes (such as IP and KSC) are classified.

These observed benefits confirm the following introspections: the ability of the DOPRI5 solver to adapt the model's learning to the complexity of the problem and the significant reduction that can be achieved in terms of the required number of parameters. The latter important benefit is quantitatively measured in Table II, where the number of required model parameters is displayed for each HSI data set. Specifically, the proposed ODEnet and ODEnetAdj models are able to overcome the performance of the traditional ResNet model by

TABLE II

NUMBER OF TRAINABLE PARAMETERS FOR THE STANDARD RESNET MODEL AND THE PROPOSED METHOD IMPLEMENTED WITH DOPRI5 SOLVER AND RUNGE–KUTTA INTEGRATION (ODENET) AND WITH THE ADJOINT METHOD (ODENETADJ)

Dataset	ResNet	ODEnet	ODEnetAdj	Reduction
IP	638416	268624	268624	2.38
UP	582089	212297	212297	2.74
SV	640720	270928	270928	2.36
KSC	624397	254605	254605	2.45

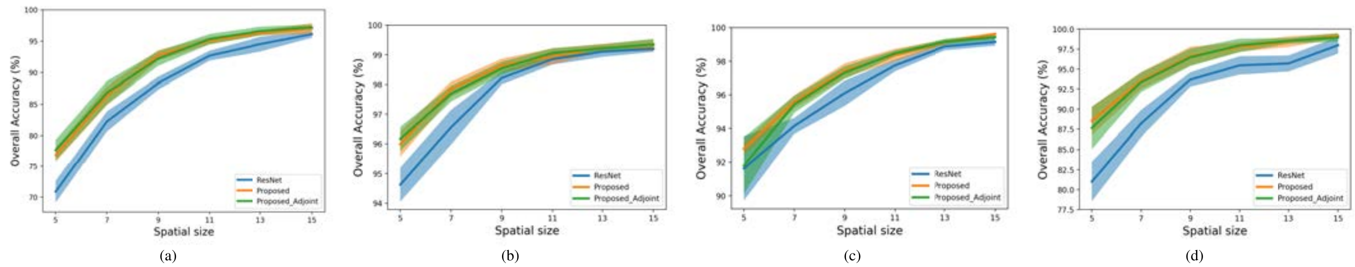


Fig. 8. Evolution of the OA reached by ResNet (blue), the proposed ODEnet the with DOPRI5 solver and Runge–Kutta integration (orange), and the proposed ODEnet with the DOPRI5 solver and adjoint method (green), considering different spatial window sizes. We report the results obtained for (a) IP, (b) UP, (c) SV, and (d) KSC scenes.

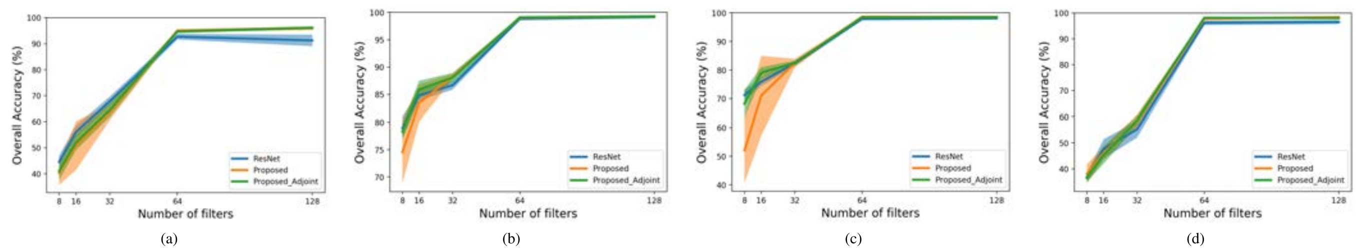


Fig. 9. Evolution of the OA reached by ResNet (blue), the proposed ODEnet with the DOPRI5 solver and Runge–Kutta integration (orange), and the proposed ODEnet with the DOPRI5 solver and adjoint method (green), considering different numbers of filters in each block. We report the results obtained for (a) IP, (b) UP, (c) SV, and (d) KSC scenes.

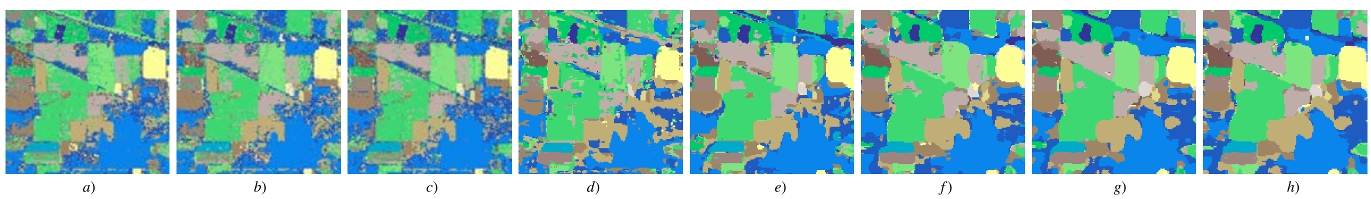


Fig. 10. Classification maps obtained for the IP scene by different classifiers (see Table III). Note that the overall classification accuracies are shown in brackets and the best result is highlighted in bold font. (a) MLR (78.19%). (b) SVM (83.63%). (c) MLP (84.03%). (d) CNN2D (87.16%). (e) CNN3D (95.45%). (f) ResNet (96.55%). (g) ODEnet (97.61%). (h) ODEnetAdj (97.55%).

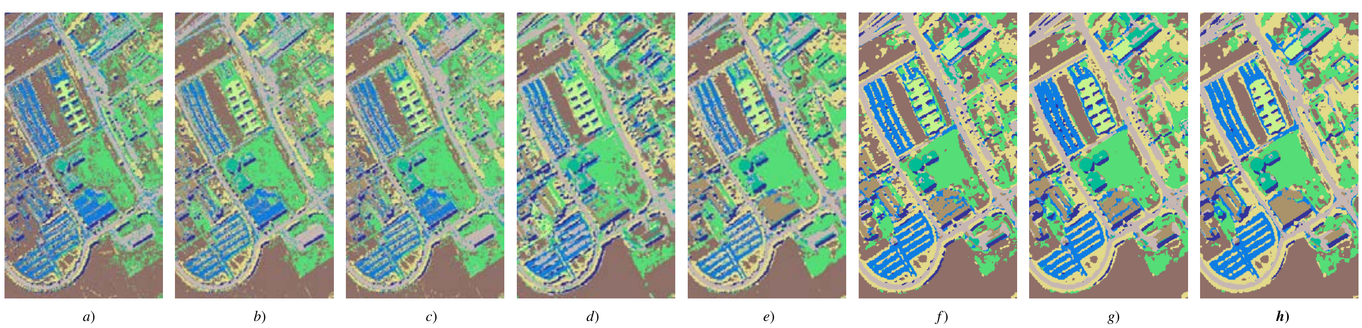


Fig. 11. Classification maps obtained for the UP scene by different classifiers (see Table IV). Note that the overall classification accuracies are shown in brackets and the best result is highlighted in bold font. (a) MLR (89.89%). (b) SVM (94.40%). (c) MLP (94.39%). (d) CNN2D (96.02%). (e) CNN3D (99.02%). (f) ResNet (99.54%). (g) ODEnet (99.67%). (h) ODEnetAdj (99.69%).

using less than half of its training parameters, avoiding quite effectively the overfitting problem.

G. Experiment 4: Testing Different Network Configurations

In this experiment, we report the results obtained by the proposed ODEnet considering different configurations of the model, in particular, the initial amount of information employed by the ODEnet, ODEnetAdj, and ResNet by testing different spatial sizes of the models' input data patch, and the number of features extracted and processed by the convolutional layers.

On the one hand, Fig. 8 shows the obtained results in terms of OA considering input patches comprised of 5×5, 7×7, 9×9, 11×11, 13×13, and 15×15 pixels. As we can observe, the proposed models exhibit very similar behaviors, being able to outperform the accuracy reached by the ResNet in every

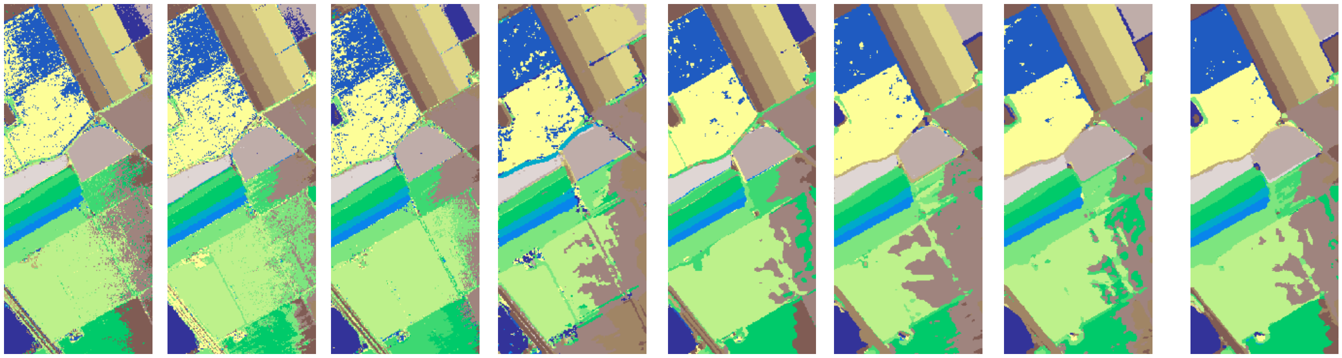


Fig. 12. Classification maps obtained for the SV scene by different classifiers (see Table V). Note that the overall classification accuracies are shown in brackets and the best result is highlighted in bold font. (a) MLR (92.37%). (b) SVM (93.65%). (c) MLP (93.15%). (d) CNN2D (95.27%). (e) CNN3D (98.45%). (f) ResNet (99.28%). (g) ODEnet (99.42%). (h) ODEnetAdj (99.41%).

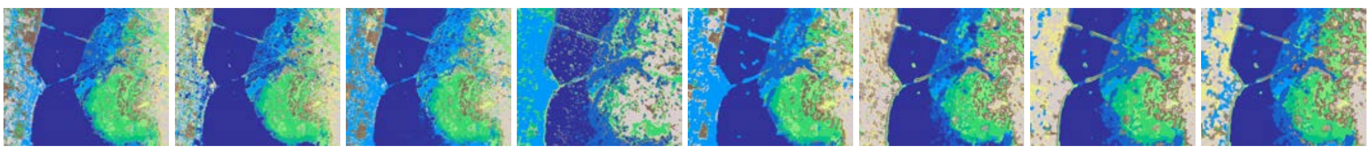


Fig. 13. Classification maps obtained for the KSC scene by different classifiers (see Table VI). Note that the overall classification accuracies are shown in brackets and the best result is highlighted in bold font. (a) MLR (92.74%). (b) SVM (92.92%). (c) MLP (90.22%). (d) CNN2D (66.04%). (e) CNN3D (98.10%). (f) ResNet (98.39%). (g) ODEnet (99.24%). (h) ODEnetAdj (99.03%).

TABLE III

CLASSIFICATION RESULTS OBTAINED BY DIFFERENT METHODS FOR THE IP DATA SET, USING 15% OF THE AVAILABLE LABELED DATA FOR TRAINING AND 11 $\times$ 11 INPUT SPATIAL PATCH SIZE

Class	MLR	SVM	MLP	CNN2D	CNN3D	ResNet	ODEnet	ODEnetAdj
0	74.59±0.48	81.36±0.42	81.23±0.96	85.61±1.09	95.34±0.72	96.07±0.55	**97.28**±0.34	97.21±0.38
1	33.33±10.51	56.92±12.96	65.38±13.96	75.38±17.7	80.77±15.44	86.41±9.87	93.08±5.13	**93.59**±5.29
2	76.36±1.9	81.12±1.09	78.92±2.09	84.63±2.56	94.68±1.84	94.15±1.02	96.02±1.43	**96.08**±0.97
3	57.3±2.15	74.03±2.01	68.14±3.81	77.63±3.52	95.32±1.44	94.4±2.79	**97.15**±1.44	97.09±1.69
4	43.03±6.83	61.29±4.81	73.78±4.65	84.13±5.09	91.84±3.68	**96.77**±2.48	96.62±2.61	96.32±2.72
5	86.73±3.94	89.71±3.83	88.46±2.8	83.29±5.47	96.15±2.07	97.24±1.21	96.63±1.7	**97.63**±2.86
6	96.23±0.9	97.0±1.13	95.13±1.66	90.47±2.48	99.11±0.47	97.71±1.08	**99.18**±0.38	98.76±0.59
7	54.78±7.33	77.39±10.07	82.61±7.78	84.78±16.53	80.87±17.95	80.87±12.17	96.52±5.43	**96.96**±4.37
8	98.45±0.96	97.73±1.87	98.69±1.13	97.17±2.27	99.73±0.58	99.43±1.09	**99.9**±0.16	99.9±0.16
9	18.24±13.27	50.59±11.22	64.71±13.67	**85.88**±11.53	83.53±17.41	67.65±18.27	82.94±10.0	77.65±9.41
10	65.3±2.2	76.42±1.97	78.98±3.67	78.75±3.78	94.49±2.55	95.33±1.79	**97.53**±1.35	96.38±1.46
11	79.87±1.71	84.21±1.65	83.25±2.19	90.48±1.53	96.87±1.19	97.77±0.72	98.23±0.47	**98.26**±0.43
12	61.23±2.31	77.64±2.14	79.58±4.16	76.25±3.54	89.48±4.18	94.01±1.81	94.11±2.46	**95.32**±1.49
13	98.51±0.64	98.33±1.13	98.1±1.06	98.33±1.13	**99.89**±0.23	99.54±0.5	99.25±0.85	99.6±0.45
14	94.99±1.31	94.51±1.71	95.79±0.92	97.4±0.95	98.91±0.47	99.09±0.87	99.23±1.07	**99.29**±0.7
15	63.2±3.62	62.93±4.7	65.88±3.48	89.7±3.71	91.68±4.25	97.13±2.58	**97.65**±1.45	96.55±2.62
16	86.08±2.59	87.47±4.22	94.05±4.57	93.42±5.39	**98.48**±2.18	93.92±5.18	96.84±3.93	95.7±3.81
OA (%)	77.87±0.42	83.68±0.38	83.57±0.85	87.43±0.95	95.92±0.63	96.55±0.48	**97.61**±0.3	97.55±0.33
AA (%)	69.6±0.59	79.21±1.49	81.97±1.76	86.73±1.91	93.24±1.61	93.21±1.45	**96.3**±0.87	95.94±1.09
Kappa (x100)	74.59±0.48	81.36±0.42	81.23±0.96	85.61±1.09	95.34±0.72	96.07±0.55	**97.28**±0.34	97.21±0.38
Runtime (s)	5.99±1.03	**0.28**±0.01	99.49±5.46	66.55±4.16	172.89±15.73	81.63±0.13	192.6±0.19	257.55±2.62

scene, in particular when the spatial windows are very small. Moreover, the improvement in the OA's values increases as the spatial windows' size increases. However, while the difference, in terms of accuracy, between small spatial windows is very pronounced (for instance, between windows of 5×5 and 9×9 pixels, there are approximately ten percentage points of improvement in IP and KSC and four percentage points in UP and SV), between bigger windows, the difference is noticeably smaller (for instance, between windows of 11×11 and 15×15). In this sense, as the amount of information to be processed increases with the dimensions of the input data patch, increasing also both memory requirements and computation times, we consider paths of 11×11 pixels as an optimal input data size, with a good ratio between performance and computing time.

On the other hand, Fig. 9 shows the obtained results in terms of OA too, considering input patches of 11×11 and convolutional layers with 8, 16, 32, 64, and 128 filters. As we can observe for each data set, the OA increases its value as more filters are added. In particular, the best

TABLE IV

CLASSIFICATION RESULTS OBTAINED BY DIFFERENT METHODS FOR THE UP DATA SET, USING 10% OF THE AVAILABLE LABELED DATA FOR TRAINING AND 11 × 11 INPUT SPATIAL PATCH SIZE

Class	MLR	SVM	MLP	CNN2D	CNN3D	ResNet	ODEnet	ODEnetAdj
0	86.42±0.2	92.56±0.14	92.42±0.25	94.72±0.29	99.08±0.13	99.39±0.11	99.56±0.1	**99.59**±0.11
1	92.31±0.62	94.46±0.58	94.98±0.72	95.12±0.99	99.19±0.38	99.44±0.25	**99.64**±0.17	99.59±0.3
2	96.07±0.36	98.31±0.18	97.82±0.3	98.21±0.33	99.89±0.06	99.87±0.09	**99.93**±0.06	99.89±0.05
3	74.0±1.86	79.37±1.84	80.32±2.92	90.33±1.45	96.88±0.79	98.3±1.18	98.89±0.52	**99.08**±0.65
4	88.49±0.96	94.55±0.58	93.59±1.65	97.6±0.54	99.18±0.39	**99.23**±0.38	99.22±0.38	99.13±0.44
5	99.31±0.27	99.28±0.18	99.54±0.15	99.16±0.49	**100.0**±0.0	99.98±0.05	99.98±0.05	99.98±0.05
6	77.42±0.84	88.86±1.06	89.93±1.62	92.01±0.96	**99.88**±0.14	99.62±0.37	99.82±0.18	99.87±0.22
7	57.03±3.04	85.41±2.02	85.98±1.94	83.54±3.16	93.85±2.17	98.08±0.74	98.48±1.2	**98.77**±1.04
8	86.77±0.76	90.67±1.11	89.43±1.83	96.63±0.77	98.84±0.24	99.37±0.38	99.4±0.36	**99.65**±0.24
9	99.77±0.1	**99.91**±0.1	99.89±0.1	98.92±0.87	99.82±0.18	98.98±1.1	99.33±0.75	99.72±0.24
OA (%)	89.84±0.16	94.41±0.1	94.3±0.19	96.02±0.22	99.31±0.1	99.54±0.08	99.67±0.08	**99.69**±0.08
AA (%)	85.69±0.33	92.31±0.24	92.39±0.37	94.61±0.47	98.62±0.26	99.21±0.19	99.41±0.22	**99.52**±0.1
Kappa (x100)	86.42±0.2	92.56±0.14	92.42±0.25	94.72±0.29	99.08±0.13	99.39±0.11	99.56±0.1	**99.59**±0.11
Runtime (s)	9.11±1.57	0.4±0.01	439.55±22.77	306.69±19.04	409.88±65.98	160.17±0.43	508.68±1.71	671.88±3.54

TABLE V

CLASSIFICATION RESULTS OBTAINED BY DIFFERENT METHODS FOR THE SV DATA SET, USING 10% OF THE AVAILABLE LABELED DATA FOR TRAINING AND 11 × 11 INPUT SPATIAL PATCH SIZE

Class	MLR	SVM	MLP	CNN2D	CNN3D	ResNet	ODEnet	ODEnetAdj
0	91.55±0.11	92.94±0.16	92.37±0.17	94.72±0.49	98.23±0.21	99.2±0.18	**99.35**±0.16	99.35±0.19
1	99.49±0.24	99.57±0.28	99.52±0.37	97.2±2.3	**100.0**±0.0	99.84±0.24	99.91±0.1	99.91±0.1
2	99.92±0.06	99.79±0.13	99.89±0.11	98.96±1.04	**99.98**±0.04	99.91±0.13	99.86±0.21	99.88±0.19
3	99.45±0.25	99.58±0.16	99.15±0.51	96.66±3.15	**99.94**±0.07	99.85±0.14	99.91±0.17	99.8±0.25
4	99.25±0.21	99.3±0.38	99.31±0.34	99.79±0.39	99.67±0.28	99.44±0.62	99.68±0.43	**99.83**±0.17
5	99.07±0.23	98.69±0.47	99.13±0.26	98.75±1.29	99.24±0.38	**99.86**±0.16	99.83±0.25	99.85±0.16
6	99.94±0.05	99.85±0.09	99.86±0.14	99.35±0.46	100.0±0.01	**100.0**±0.0	100.0±0.0	100.0±0.01
7	99.68±0.17	99.71±0.16	99.62±0.25	99.51±0.36	99.74±0.28	99.88±0.11	**99.97**±0.05	99.92±0.11
8	87.57±0.6	88.38±0.63	87.1±1.06	92.47±1.43	96.54±0.5	98.42±0.39	**98.82**±0.38	98.62±0.41
9	99.8±0.15	99.72±0.13	99.87±0.17	99.13±0.44	99.96±0.08	**99.98**±0.02	99.96±0.07	99.96±0.04
10	95.41±0.88	96.16±0.71	96.34±0.81	96.66±1.03	99.19±0.34	99.59±0.25	**99.75**±0.32	99.48±0.64
11	97.54±0.91	97.8±0.81	98.3±1.65	97.64±1.14	99.56±0.23	**99.84**±0.31	99.69±0.51	99.4±0.48
12	99.68±0.2	99.74±0.21	99.73±0.16	97.76±1.41	**100.0**±0.0	99.97±0.07	99.99±0.02	99.99±0.02
13	99.14±0.4	98.68±0.68	98.39±1.37	98.68±0.77	99.65±0.79	99.58±0.6	99.62±0.54	**99.82**±0.29
14	96.87±1.07	96.23±1.87	96.81±1.41	97.65±1.18	99.17±0.64	99.69±0.41	**99.89**±0.28	99.85±0.19
15	67.47±0.57	75.4±1.08	73.2±2.13	85.05±2.24	94.79±1.13	97.91±0.67	98.08±0.47	**98.49**±0.73
16	98.64±0.42	98.76±0.37	98.79±0.43	93.37±1.94	99.51±0.35	99.58±0.45	**99.79**±0.18	99.76±0.19
OA (%)	92.42±0.1	93.66±0.14	93.15±0.15	95.26±0.44	98.41±0.19	99.28±0.16	**99.42**±0.14	99.41±0.18
AA (%)	96.18±0.08	96.71±0.09	96.56±0.18	96.79±0.24	99.18±0.1	99.58±0.12	**99.67**±0.09	99.66±0.09
Kappa (x100)	91.55±0.11	92.94±0.16	92.37±0.17	94.72±0.49	98.23±0.21	99.2±0.18	**99.35**±0.16	99.35±0.19
Runtime (s)	50.49±0.22	1.3±0.02	689.03±29.61	457.91±12.95	826.38±58.19	239.64±0.18	730.73±36.55	972.9±36.21

OA is reached with 64 filters, remaining quite similar with 128 filters. Actually, the OA is improved very slightly with 128 filters; however, the computational cost of this network's configuration is considerably higher than with 64 filters. For this reason, we consider 64 to be the optimum number of filters for each convolutional layer.

H. Experiment 5: Testing Different HSI Classifiers

Our final experiment compares our proposed ODEnet models with some widely used classifiers available in the HSI classification literature. Fig. 10 (IP), Fig. 11 (UP), Fig. 12 (SV), and Fig. 13 (KSC) show the classification maps obtained by each considered method, while Table III (IP), Table IV (UP), Table V (SV), and Table VI (KSC) give the individual class accuracies and the global OA, AA, and Kappa values obtained by each classifier with the corresponding standard deviations, respectively, including also the obtained runtimes of each experiment.

As a general comment, the improvement introduced by spatial and spectral–spatial models over pixelwise classifiers is remarkable. For instance, CNN2D introduces around 2% points of improvement in OA when compared to the most accurate spectral model, i.e., the SVM (for UP and SV) and the MLP (for IP), with an exception in the KSC scene, in which the spatial information appears to be not enough discriminatory to carry out an accurate classification, as we can observe in Table VI and the corresponding classification maps in Fig. 13. The limitations of pixelwise and spatial-based classifiers can be easily overcome by spectral–spatial classifiers, where the combination of spectral and spatial–contextual information is able to significantly reduce the uncertainty and data variability of HSI pixels, as it can be observed on complex data sets, such as IP (see Table III) and, particularly, KSC (see Table VI). This results in better classification maps, where the "salt and pepper" classification noise is practically removed. However, it is interesting to focus on the classification maps produced by the spatial–spectral CNN3D classifier

TABLE VI

CLASSIFICATION RESULTS OBTAINED BY DIFFERENT METHODS FOR THE KSC DATA SET, USING 15% OF THE AVAILABLE LABELED DATA FOR TRAINING AND 11 × 11 INPUT SPATIAL PATCH SIZE

Class	MLR	SVM	MLP	CNN2D	CNN3D	ResNet	ODEnet	ODEnetAdj
0	91.57±0.76	92.52±0.57	88.87±0.55	61.8±1.35	98.22±0.27	98.2±0.59	**99.16**±0.24	98.93±0.42
1	95.0±0.96	95.57±1.31	96.53±0.98	96.35±1.93	**100.0**±0.0	99.47±0.82	100.0±0.0	99.83±0.26
2	93.01±2.24	91.55±3.06	84.76±2.08	37.23±6.03	95.78±2.86	96.7±2.74	98.93±1.55	**99.22**±0.66
3	88.99±1.73	88.89±3.52	88.06±3.62	21.15±8.09	97.37±1.6	96.31±3.35	**99.12**±1.33	96.59±3.74
4	71.17±5.63	74.91±4.73	60.84±5.86	21.26±7.13	86.4±3.18	88.79±2.1	**95.09**±1.91	94.49±4.34
5	71.1±6.49	73.75±4.89	59.56±5.35	57.87±5.07	90.0±4.45	91.99±8.4	90.51±5.11	**92.06**±5.44
6	70.77±7.17	74.79±4.34	58.09±2.73	40.26±14.57	96.6±2.14	98.2±1.09	**98.92**±1.41	98.81±1.56
7	81.57±5.8	86.18±5.1	84.04±4.71	30.79±29.78	99.21±1.33	97.87±2.22	**99.89**±0.34	99.33±1.68
8	91.72±1.84	92.76±2.49	88.5±0.87	35.41±4.06	98.93±1.22	98.91±1.05	99.73±0.35	**99.86**±0.18
9	96.54±1.18	96.92±1.54	96.04±1.43	71.0±4.87	**99.91**±0.21	99.32±0.93	99.59±0.38	99.68±0.61
10	96.27±1.52	97.32±1.91	93.62±1.91	52.94±5.82	99.85±0.44	99.62±0.39	**99.91**±0.19	99.56±1.04
11	97.58±0.92	96.99±2.22	96.99±1.24	95.28±2.07	**99.97**±0.08	99.89±0.34	99.92±0.25	99.55±0.69
12	94.54±3.19	96.21±1.31	92.22±1.16	61.45±6.42	99.65±0.42	99.02±1.17	**99.93**±0.15	99.51±0.74
13	**100.0**±0.0	100.0±0.0	99.72±0.34	91.39±3.37	100.0±0.0	100.0±0.0	100.0±0.0	100.0±0.0
OA (%)	92.43±0.68	93.29±0.51	90.02±0.5	66.03±1.23	98.4±0.24	98.39±0.53	**99.24**±0.21	99.03±0.37
AA (%)	88.33±1.0	89.68±0.87	84.54±0.87	54.8±2.42	97.21±0.48	97.39±0.93	**98.58**±0.49	98.35±0.71
Kappa (x100)	91.57±0.76	92.52±0.57	88.87±0.55	61.8±1.35	98.22±0.27	98.2±0.59	**99.16**±0.24	98.93±0.42
Runtime (s)	2.89±0.24	0.05±0.01	87.98±3.74	66.91±4.63	73.17±1.36	57.47±0.08	114.57±0.17	154.55±7.2

(for instance in Figs. 11 and 13), where multiple patches have been wrongly labeled, obtaining visually noisy classification maps. This can be observed in the lower leftmost corner of the KSC scene (see Fig. 13), where the vast majority of pixels have been missclassified as Salt-marsh. These deficiencies are highlighted by the differences between the OA and AA values, where the AA is several percentual points lower, indicating the existence of an overfitting problem (see Tables III and VI).

Adding residual learning via ResNet can improve the accuracy results, reducing the gap between the OA and AA on some HSI data sets, such as UP (see Table IV), SV (see Table V), and KSC (see Table VI), and improving the visual appearance of the corresponding classification maps. However, this gap between the OA and AA scores cannot be reduced by ResNet in the IP scene (in fact, for this scene, the gap becomes larger). In turn, the proposed ODEnet models are able to reach very similar OA, AA, and Kappa values in all the cases, exhibiting very good consistency in terms of model performance with higher robustness on the obtained results. As we can observe, the proposed method is able to reach the best accuracy scores in all the considered data sets, visually clean classification maps, where the number of missclassified patches is drastically reduced.

If we now focus on the execution times reported in Table III (IP), Table IV (UP), Table V (SV), and Table VI (KSC), it can be observed that pixelwise methods are faster than spatial and spectral–spatial ones, being SVM the fastest classifier. The computational cost of the proposed ODEnet model is higher when compared to the other deep models (CNN2D, CNN3D, and ResNet), mainly due to the great optimization performed by the frameworks in which these classifiers have been implemented. In this regard, it is necessary to conduct an effort to optimize the code of the ODEsolver in order to provide a more efficient version, although (as shown in our second experiment) the use of higher tolerance values allows for a significant reduction of computation times.

IV. CONCLUSION AND FUTURE WORK

This article proposes, for the first time in the literature, a redefinition of the traditional discrete-layer ResNet model as a continuous-time evolving model through the implementation of an ODE parameterized by a neural network with the aim of improving the classification of remotely sensed HSI data by producing better and more robust feature representations.

The obtained experimental results, conducted using four widely used HSI data sets, demonstrate the significant benefits and improvements introduced by the proposed method, which are able to reach consistently higher accuracy values in comparison with the traditional ResNet model, at the same time it significantly reduces the number of parameters that need to be used and fine-tuned, providing a highly efficient mechanism to address the problems of overfitting and data degradation in very deep networks. Moreover, the integration of adaptive solvers, such as DOPRI5, offers great flexibility when processing and classifying complex HSI scenes, allowing the model to obtain highly refined features for classification purposes.

Encouraged by the good results obtained in terms of model's accuracy, in the future, we will develop an optimized and parallelized implementation of the proposed ODEnet, exploring other solver algorithms in order to reduce the computational complexity.

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor and the three anonymous reviewers for their outstanding comments and suggestions, which greatly helped us to improve the technical quality and presentation of this article.

REFERENCES

[1] J. A. Richards, *Remote Sensing Digital Image Analysis*, vol. 3. Berlin, Germany: Springer, 1999.

[2] X.-L. Chen, H.-M. Zhao, P.-X. Li, and Z.-Y. Yin, "Remote sensing image-based analysis of the relationship between urban heat island and land use/cover changes," *Remote Sens. Environ.*, vol. 104, no. 2, pp. 133–146, 2006.

[3] D. A. Mariano *et al.*, "Use of remote sensing indicators to assess effects of drought and human-induced land degradation on ecosystem health in Northeastern Brazil," *Remote Sens. Environ.*, vol. 213, pp. 129–143, Aug. 2018.

[4] T. Lillesand, R. W. Kiefer, and J. Chipman, *Remote Sensing and Image Interpretation*. Hoboken, NJ, USA: Wiley, 2014.

[5] R. O. Green *et al.*, "Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (AVIRIS)," *Remote Sens. Environ.*, vol. 65, no. 3, pp. 227–248, Sep. 1998.

[6] D. J. Mulla, "Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps," *Biosyst. Eng.*, vol. 114, no. 4, pp. 358–371, Apr. 2013.

[7] M. Govender, K. Chetty, and H. Bulcock, "A review of hyperspectral remote sensing and its application in vegetation and water resource studies," *Water SA*, vol. 33, no. 2, pp. 145–151, 2007.

[8] B. Hörig, F. Kühn, F. Oschütz, and F. Lehmann, "HyMap hyperspectral remote sensing to detect hydrocarbons," *Int. J. Remote Sens.*, vol. 22, no. 8, pp. 1413–1422, 2001.

[9] P. M. Treitz and P. J. Howarth, "Hyperspectral remote sensing for estimating biophysical parameters of forest ecosystems," *Progr. Phys. Geography*, vol. 23, no. 3, pp. 359–390, 1999.

[10] V. Sander *et al.*, "Hyperspectral remote sensing of fire: State-of-the-art and future perspectives," *Remote Sens. Environ.*, vol. 216, pp. 105–121, Oct. 2018.

[11] M. Herold, D. A. Roberts, M. E. Gardner, and P. E. Dennison, "Spectrometry for urban area remote sensing—Development and analysis of a spectral library from 350 to 2400 nm," *Remote Sens. Environ.*, vol. 91, nos. 3–4, pp. 304–319, 2004.

[12] P. W. Yuen and M. Richardson, "An introduction to hyperspectral imaging and its application for security, surveillance and target acquisition," *Imag. Sci. J.*, vol. 58, no. 5, pp. 241–253, Oct. 2010.

[13] J. Haut, M. Paoletti, J. Plaza, and A. Plaza, "Cloud implementation of the K-means algorithm for hyperspectral image analysis," *J. Supercomput.*, vol. 73, no. 1, pp. 514–529, 2017.

[14] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 8, pp. 1778–1790, Aug. 2004.

[15] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Semisupervised hyperspectral image classification using soft sparse multinomial logistic regression," *IEEE Geosci. Remote Sens. Lett.*, vol. 10, no. 2, pp. 318–322, Mar. 2013.

[16] P. Ghamisi *et al.*, "Advances in hyperspectral image and signal processing: A comprehensive overview of the state of the art," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 4, pp. 37–78, Dec. 2017.

[17] P. Ghamisi, J. Plaza, Y. Chen, J. Li, and A. J. Plaza, "Advanced spectral classifiers for hyperspectral images: A review," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 1, pp. 8–32, Mar. 2017.

[18] P. K. Goel, S. O. Prasher, R. M. Patel, J.-A. Landry, R. Bonnell, and A. A. Viau, "Classification of hyperspectral data by decision trees and artificial neural networks to identify weed stress and nitrogen status of corn," *Comput. Electron. Agricult.*, vol. 39, no. 2, pp. 67–93, 2003.

[19] F. Ratle, G. Camps-Valls, and J. Weston, "Semisupervised neural networks for efficient hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 5, pp. 2271–2282, May 2010.

[20] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989.

[21] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, 1991.

[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[23] L. He, J. Li, C. Liu, and S. Li, "Recent advances on spectral–spatial hyperspectral image classification: An overview and new guidelines," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 3, pp. 1579–1597, Mar. 2018.

[24] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE Trans. Inf. Theory*, vol. IT-14, no. 1, pp. 55–63, Jan. 1968.

[25] S. Kaewpijit, J. L. Moigne, and T. El-Ghazawi, "Automatic reduction of hyperspectral imagery using wavelet spectral analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 41, no. 4, pp. 863–871, Apr. 2003.

[26] J. M. Haut, M. E. Paoletti, J. Plaza, and A. Plaza, "Fast dimensionality reduction and classification of hyperspectral images with extreme learning machines," *J. Real-Time Image Process.*, vol. 15, no. 3, pp. 439–462, 2018.

[27] D. J. Field, "Wavelets, vision and the statistics of natural scenes," *Phil. Trans. Roy. Soc. London A, Math., Phys. Eng. Sci.*, vol. 357, no. 1760, pp. 2527–2542, 1999.

[28] M. E. Paoletti *et al.*, "Capsule networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 4, pp. 2145–2160, Apr. 2019.

[29] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *J. Sensors*, vol. 2015, Jan. 2015, Art. no. 258619.

[30] J. Yue, W. Zhao, S. Mao, and H. Liu, "Spectral–spatial classification of hyperspectral images using deep convolutional neural networks," *Remote Sens. Lett.*, vol. 6, no. 6, pp. 468–477, Jun. 2015.

[31] W. Zhao and S. Du, "Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 8, pp. 4544–4554, Aug. 2016.

[32] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 6232–6251, Oct. 2016.

[33] S. Yu, S. Jia, and C. Xu, "Convolutional neural networks for hyperspectral image classification," *Neurocomputing*, vol. 219, pp. 88–98, Jan. 2017.

[34] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*. [Online]. Available: https://arxiv.org/abs/1312.4400

[35] N. He *et al.*, "Feature extraction with multiscale covariance maps for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 755–769, Feb. 2019.

[36] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "A new deep convolutional neural network for fast hyperspectral image classification," *ISPRS J. Photogram. Remote Sens.*, vol. 145, pp. 120–147, Nov. 2018.

[37] J. M. Haut, M. E. Paoletti, J. Plaza, J. Li, and A. Plaza, "Active learning with convolutional neural networks for hyperspectral image classification using a new Bayesian approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 11, pp. 6440–6461, Nov. 2018.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[39] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[40] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2377–2385.

[41] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," 2015, *arXiv:1505.00387*. [Online]. Available: https://arxiv.org/abs/1505.00387

[42] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 4700–4708.

[43] K. Zhang, M. Sun, T. X. Han, X. Yuan, L. Guo, and T. Liu, "Residual networks of residual networks: Multilevel residual networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 6, pp. 1303–1314, Jun. 2018.

[44] J. M. Haut, R. Fernandez-Beltran, M. E. Paoletti, J. Plaza, A. Plaza, and F. Pla, "A new deep generative network for unsupervised remote sensing single-image super-resolution," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 11, pp. 6792–6810, Nov. 2018.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 630–645.

[46] Z. Zhong, J. Li, Z. Luo, and M. Chapman, "Spectral–spatial residual network for hyperspectral image classification: A 3-D deep learning framework," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 2, pp. 847–858, Feb. 2018.

[47] M. E. Paoletti, J. M. Haut, R. Fernandez-Beltran, J. Plaza, A. J. Plaza, and F. Pla, "Deep pyramidal residual networks for spectral–spatial hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 740–754, Feb. 2019.

[48] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "Deep&dense convolutional neural network for hyperspectral image classification," *Remote Sens.*, vol. 10, no. 9, p. 1454, 2018.

[49] M. Thorpe and Y. van Gennip, "Deep limits of residual neural networks," 2018, *arXiv:1810.11741*. [Online]. Available: https://arxiv.org/abs/1810.11741

[50] L. Ruthotto and E. Haber, "Deep neural networks motivated by partial differential equations," 2018, *arXiv:1804.04272*. [Online]. Available: https://arxiv.org/abs/1804.04272

[51] E. Weinan, "A proposal on machine learning via dynamical systems," *Commun. Math. Statist.*, vol. 5, no. 1, pp. 1–11, 2017.

[52] E. Haber, L. Ruthotto, E. Holtham, and S.-H. Jun, "Learning across scales—Multiscale methods for convolution neural networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, Apr. 2018, pp. 3142–3148.

[53] Y. Lu, A. Zhong, Q. Li, and B. Dong, "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations," 2017, *arXiv:1710.10121*. [Online]. Available: https://arxiv.org/abs/1710.10121

[54] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6572–6583.

[55] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "An investigation on self-normalized deep neural networks for hyperspectral image classification," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2018, pp. 3607–3610.

[56] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.

[57] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[58] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, 2018.

[59] J. D. Lambert, *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. Hoboken, NJ, USA: Wiley, 1991.

[60] J. R. Dormand and P. J. Prince, "A family of embedded Runge-Kutta formulae," *J. Comput. Appl. Math.*, vol. 6, no. 1, pp. 19–26, 1980.

[61] K. Atkinson, *An Introduction to Numerical Analysis*. Hoboken, NJ, USA: Wiley, 2008.

[62] J. T. Betts, "Survey of numerical methods for trajectory optimization," *J. Guid. Control Dyn.*, vol. 21, no. 2, pp. 193–207, Mar./Apr. 1998.

[63] B. Kunkel, F. Blechinger, R. Lutz, R. Doerffer, H. van der Piepen, and M. Schroder, "ROSIS (Reflective Optics System Imaging Spectrometer)—A candidate instrument for polar platform missions," *Proc. SPIE*, vol. 0868, pp. 134–141, Apr. 1988.

Mercedes E. Paoletti (S'17) received the B.Sc. and M.Sc. degrees in computer engineering from the University of Extremadura, Cáceres, Spain, in 2014 and 2016, respectively, where she is currently pursuing the Ph.D. degree with the University Teacher Training Programme from the Spanish Ministry of Education.

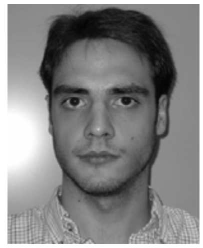
She is currently a member of the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura. Her research interests include remote sensing and analysis of very high spectral resolution with the current focus on deep learning and high performance computing.

Ms. Paoletti was a recipient of the 2019 Outstanding Paper Award Recognition in the WHISPERS 2019 Congress. She has been a Reviewer of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING and the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS.

Juan Mario Haut (S'17–M'19) received the B.Sc. and M.Sc. degrees in computer engineering from the University of Extremadura, Cáceres, Spain, in 2011 and 2014, respectively, and the Ph.D. degree in information technology with the University Teacher Training Programme from the Spanish Ministry of Education, University of Extremadura in 2019.

He is a member of the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura. His research interests include remote sensing and analysis of very high spectral resolution with the current focus on machine (deep) learning and cloud computing.

Dr. Haut was a recipient of the Best Reviewers Recognition Award of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS in 2019 and the Outstanding Paper Award in the Whispers 2019 Congress. He has been a Reviewer of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING, and the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS.



Javier Plaza (M'09–SM'15) received the M.Sc. and Ph.D. degrees in computer engineering from the University of Extremadura, Cáceres, Spain, in 2004 and 2008, respectively.

He is a member of the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura. He has authored more than 150 publications, including over 50 JCR journal articles, ten book chapters, and 90 peer-reviewed conference proceeding articles. His main research interests include hyperspectral data processing and parallel computing of remote sensing data.

Dr. Plaza was a recipient of the Outstanding Ph.D. Dissertation Award at the University of Extremadura in 2008, the Best Column Award of the *IEEE Signal Processing Magazine* in 2015, the most highly cited paper (2005–2010) in the *Journal of Parallel and Distributed Computing*, and the best paper awards at the IEEE International Conference on Space Technology and the IEEE Symposium on Signal Processing and Information Technology. He has guest edited four special issues on hyperspectral remote sensing for different journals. He is an Associate Editor of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS and the IEEE REMOTE SENSING CODE LIBRARY.

Antonio Plaza (M'05–SM'07–F'15) received the M.Sc. and Ph.D. degrees in computer engineering from the University of Extremadura, Cáceres, Spain, in 1999 and 2002, respectively

He is currently the Head of the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura. He has authored more than 600 publications, including over 200 JCR journal articles (over 160 in IEEE journals), 23 book chapters, and around 300 peer-reviewed conference proceeding articles. His research interests include hyperspectral data processing and parallel computing of remote sensing data.

Dr. Plaza is a fellow of the IEEE for the contributions to hyperspectral data processing and parallel computing of earth observation data. He was a member of the Editorial Board of the IEEE GEOSCIENCE AND REMOTE SENSING NEWSLETTER from 2011 to 2012 and the *IEEE Geoscience and Remote Sensing Magazine* in 2013. He was a recipient of the recognition of Best Reviewers of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS in 2009 and the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING in 2010, for which he served as an Associate Editor from 2007 to 2012. He was a recipient of the Best Column Award of the *IEEE Signal Processing Magazine* in 2015, the 2013 Best Paper Award of the JSTARS Journal, the most highly cited article (2005–2010) in the *Journal of Parallel and Distributed Computing*, and the best paper awards at the IEEE International Conference on Space Technology and the IEEE Symposium on Signal Processing and Information Technology. He has guest-edited ten special issues on hyperspectral remote sensing for different journals. He is also an Associate Editor of IEEE ACCESS (receiving a recognition as an outstanding Associate Editor of the journal in 2017). He was also a member of the Steering Committee of the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING (JSTARS). He served as the Director of Education Activities for the IEEE Geoscience and Remote Sensing Society (GRSS) from 2011 to 2012, and as the President of the Spanish Chapter of the IEEE GRSS from 2012 to 2016. He has reviewed more than 500 manuscripts for over 50 different journals. He served as the Editor-in-Chief for the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING from 2013 to 2017.