

# Active Learning With Convolutional Neural Networks for Hyperspectral Image Classification Using a New Bayesian Approach

Juan Mario Haut<sup>1</sup>, Student Member, IEEE, Mercedes E. Paoletti<sup>2</sup>, Student Member, IEEE, Javier Plaza<sup>1</sup>, Senior Member, IEEE, Jun Li<sup>3</sup>, Senior Member, IEEE, and Antonio Plaza<sup>1</sup>, Fellow, IEEE

**Abstract**—Hyperspectral imaging is a widely used technique in remote sensing in which an imaging spectrometer collects hundreds of images (at different wavelength channels) for the same area on the surface of the earth. In the last two decades, several methods (unsupervised, supervised, and semisupervised) have been proposed to deal with the hyperspectral image classification problem. Supervised techniques have been generally more popular, despite the fact that it is difficult to collect labeled samples in real scenarios. In particular, deep neural networks, such as convolutional neural networks (CNNs), have recently shown a great potential to yield high performance in the hyperspectral image classification. However, these techniques require sufficient labeled samples in order to perform properly and generalize well. Obtaining labeled data is expensive and time consuming, and the high dimensionality of hyperspectral data makes it difficult to design classifiers based on limited samples (for instance, CNNs overfit quickly with small training sets). Active learning (AL) can deal with this problem by training the model with a small set of labeled samples that is reinforced by the acquisition of new unlabeled samples. In this paper, we develop a new AL-guided classification model that exploits both the spectral information and the spatial-contextual information in the hyperspectral data. The proposed model makes use of recently developed Bayesian CNNs. Our newly developed technique provides robust classification results when compared with other state-of-the-art techniques for hyperspectral image classification.

Manuscript received October 19, 2017; revised April 1, 2018; accepted May 15, 2018. Date of publication June 20, 2018; date of current version October 25, 2018. This paper was supported in part by the Ministerio de Educación, 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, en el marco del Plan Estatal de Investigación Científica y Técnica y de Innovación 2013–2016, in part by the National Natural Science Foundation of China under Grant 61771496, in part by the National Key Research and Development Program of China under Grant 2017YFB0502900, and in part by the Guangdong Provincial Natural Science Foundation under Grant 2016A030313254. Funding from MINECO project TIN2015-63646-C5-5-R is gratefully acknowledged. (Corresponding author: Jun Li.)

J. M. Haut, M. E. Paoletti, J. Plaza, and A. Plaza are with the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, Polytechnic School of Cáceres, University of Extremadura, 10003 Cáceres, Spain (e-mail: juanmariohaut@unex.es; mpaoletti@unex.es; jplaza@unex.es; aplaza@unex.es).

J. Li is with the Guangdong Provincial Key Laboratory of Urbanization and Geosimulation, Center of Integrated Geographic Information Analysis, School of Geography and Planning, Sun Yat-sen University, Guangzhou 510275, China (e-mail: lijun48@mail.sysu.edu.cn).

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

Digital Object Identifier 10.1109/TGRS.2018.2838665

**Index Terms**—Active learning (AL), Bayesian-convolutional neural network (B-CNN), hyperspectral remote sensing image classification.

	NOMENCLATURE
AE	Autoencoder.
AL	Active learning.
ANN	Artificial neural network.
AVIRIS	Airborne Visible InfraRed Imaging Spectrometer.
BALD	Bayesian active learning by disagreement.
B-CNN	Bayesian-convolutional neural network.
BNN	Bayesian neural network.
CASI	Compact Airborne Spectrographic Imager.
CNN	Convolutional neural network.
CONV layer	Convolution layer.
CRNN	Convolutional recurrent neural network.
DBN	Deep belief network.
DL	Deep learning.
DNN	Deep neural network.
$D_{\text{pool}}$	Pool set.
$D_{\text{train}}$	Training set.
EnMAP	Environmental Mapping and Analysis Program.
EO-1	Earth Observing-1.
FTHSI	Fourier Transform HyperSpectral Imager.
GPU	Graphics processing unit.
HSI	Hyperspectral imaging.
HyperCam	HSI for ubiquitous computing applications.
Hymap	Hyperspectral Mapper.
HyspIRI	Hyperspectral Infrared Imager.
MAXPOOL layer	Pooling layer.
MC-dropout	Monte Carlo dropout.
ML	Machine learning.
MLP	Multilayer perceptron.
MLR	Multinomial logistic regression.
PCA	Principal component analysis.

PRISMA	Precursore IperSpettrale della Missione Applicativa.
ReLU	Rectified linear unit.
RELU layer	Nonlinearity layer with ReLU.
SAE	Stacked autoencoder.
SSAE	Stacked sparse autoencoder.
SVM	Support vector machine.
RF	Random forest.
RODIS	Reflective Optics System Imaging Spectrometer.
RNN	Recurrent neural network.
SNR	Signal-to-noise ratio.

## I. INTRODUCTION

**H**SI (or imaging spectroscopy) [1] is based on the acquisition, measurement, analysis, and interpretation of spectra captured at different wavelength channels (throughout the visible and solar-reflected infrared spectrum) over an extensive observation area on the surface of the earth. A variety of imaging spectrometers are currently available, including airborne [e.g., the AVIRIS, the CASI, the ROSIS, and the Hymap or the new hyperspectral missions based on HyperCAM technology [2]–[9]) and spaceborne (e.g., the EO-1 Hyperion or the FTHSI on MightySat II [10]–[12]). Imaging spectrometers are also available on ground-based (stationary or handheld) platforms. These instruments allow for the acquisition of the solar-reflected spectrum in a large number of narrow and contiguous spectral bands (normally several hundreds) [13], creating data cubes in which each pixel contains a detailed contiguous spectral signature that can be used to characterize the objects in the scene with great precision and detail.

Several imaging spectrometers are currently operational, providing a large volume of hyperspectral data that can be used for a wide variety of applications, such as forestry, geology, precision agriculture, hydrology, ecological monitoring, scene recognition, military applications, and disaster monitoring [14]–[17]. For instance, we highlight the following spectrometers: AVIRIS [4], which measures the solar-reflected spectrum from 0.4 to 2.5  $\mu\text{m}$  at the intervals of 0.01  $\mu\text{m}$  creating hyperspectral images with 224 bands, EO-1 Hyperion, which also collects 242 bands in the range of 0.4–2.5  $\mu\text{m}$  [16], [18], and ROSIS, which collects images with a spectral range from 0.43 to 0.96  $\mu\text{m}$  [7], among others. Also, several new satellite missions will be soon operative and ready to collect data in a very similar spectral range. For instance, the imaging spectrometer included in the NASA HypSI [19] is expected to measure the visible-to-shortwave infrared in the range 0.38–2.5  $\mu\text{m}$  or the German EnMAP [20] that is expected to collect data in the range 0.42–2.45  $\mu\text{m}$ , as well as the Italian PRISMA Program [21].

The great amount of information that these spectrometers collect is very useful in pattern recognition, which has led to the development of multiple methods for the advanced classification of hyperspectral images [22], [23]. This includes unsupervised techniques (often called *clustering* methods) [24]–[28]. However, supervised classifiers are often preferred due to their capacity to provide high classification

accuracy by considering class-specific information provided by labeled training samples [14].

In this sense, since their successful application in the field of pattern recognition in the 1990s [29], [30], ANNs have attracted the attention of a large number of researchers in the area of hyperspectral image classification [31], [32]. Their ability to learn by examples and to generalize, together with the following properties: 1) ANNs are nonparametric (i.e., they do not need prior knowledge of the statistical distribution of the classes) and 2) they offer multiple training techniques to deal with linearly nonseparable data [33], has made ANNs widely attractive for supervised classification of hyperspectral images compared with probabilistic methods.

In particular, DNNs [34], [35] have recently shown a great potential to yield high performance in image classification tasks [36]–[38]. DNNs are deep architectures (multilayer stack of simple modules) that have the capacity to learn more complex models than shallow ones [39], learning features at various levels of abstraction, i.e., the multilayer nonlinear transformations applied over DNNs architecture can adaptively extract more meaningful and discriminative features [40]. To date, four DNN models have been the mainstream DL architectures for the analysis of hyperspectral remote sensing images: DBNs, AEs, RNNs, and CNNs.

DL has emerged in part with DBN models [41], [42]. In [43], three DBNs to extract high-level features from hyperspectral data using spectral, spatial, and spectral–spatial information are introduced. In a similar way, [44] implements a DBN for feature extraction and classification, stacking spectral–spatial characteristics, while [45] investigates the hyperparameters used by the spectral and spectral–spatial DBNs in [43]. Another example is [46] in which a DBN is implemented introducing diversity promoting priors into the pretraining and fine-tuning phases in order to avoid the coadaptation of latent factors.

On the other hand, the AE has been traditionally used as an unsupervised pixel-based method to learn useful features from data and perform dimensionality reduction. In the literature, we can find deep AE architectures, also called SAEs, for hyperspectral image classification, such as the SAE proposed in [47] that performs a two-step training strategy based on pixel spectrum, with an unsupervised representation learning and a supervised fine-tuning, before a final supervised classification step conducted by a logistic regression layer. In [48], an SAE is pretrained in unsupervised fashion with spectral data, and the features are extracted by a PCA+3D Gabor wavelet filter. In [49], three SAEs are introduced to generate high-level features from hyperspectral data using spectral, spatial, and spectral–spatial information with a logistic regression method performing the final classification. Following [49], in [50], two SSAEs are proposed to extract spectral and spatial features that are stacked and embed into an SVM for classification.

SAEs and DBNs are successful DL methods for hyperspectral classification, improving their performance with the incorporation of spatial information in addition to the spectrum. However, both SAE and DBN models need to flatten the spectral–spatial features in 1-D vectors to satisfy their input

requirements, losing to a certain point the effectiveness of the spatial information [51] for characterization purposes.

Regarding RNN, it is a kind of network with loops in connections where node activations at each step depend on those of the previous step [52]. With the traditional pixel-based approach, the RNN exploits each hyperspectral pixel in the band-to-band fashion [52]. On the other hand, several CNN-based approaches (called CRNN [53]) have been implemented for hyperspectral classification. In [54], a 1-D CRNN is implemented, where spatial constraints are integrated by linear opinion pools. A similar model is used in [55], where a 1-D CRNN is trained in the semisupervised fashion with the labeled and unlabeled data (pixels) using pseudolabels. Again, RNNs and CRNNs can present the same problem as SAEs and DBNs: they need to adapt the spatial information in order to exploit it.

In this sense, we highlight CNNs [35] as a powerful tool for hyperspectral image classification [14] able to exploit both the spectral and the spatial information in an easy and natural way. CNNs successively apply convolution filters and pooling operations to the raw input data (which can be 1-D, 2-D, or 3-D), creating a hierarchy of layers whose outputs are increasingly complex feature vectors from the input data. In the literature, we can find multiple adaptations of these networks to hyperspectral analysis. Following a 1-D approach, [56] presents a five-layer 1-D CNN that receives  $n \times 1$  input vectors, where  $n$  is the number of spectral bands, to classify hyperspectral images directly in the spectral domain. On the other hand, 2-D CNNs exploit the information from neighboring pixels in order to extract spatial features, whose input data are a patch of  $d \times d$  neighboring pixels [57], normally after applying PCA to extract the spectral features [58], [59]. Also, several approaches mix both 1-D and 2-D CNNs to extract spectral-space information, respectively [60], [61]. In contrast to these methods, several 3-D CNN models have been proposed that can learn both spatial and spectral features, taking as input data 3-D patches from the original hyperspectral data, processing each pixel by means of a 3-D convolution kernel in association with its spatial neighborhood and the corresponding spectral information [37], [62]–[64]. However, the application of CNNs to hyperspectral classification presents some issues, as they require a great amount of labeled data for fine-tuning the large number of training parameters that affect their generalization power, such as the number of hidden layers and their kernel size (which involves the number of weights, their biases, and the obtained feature maps), the pooling, padding, and stride sizes, the selected optimizer and its learning rate, and the batch size. These aspects make this kind of networks quickly overfit with small training sets which may lead to poor classification accuracy in the testing phase.

In general, the quality of ANN-based classification methods is strongly related to the quality and number of training samples available in advance [65]. In order to effectively learn the parameters of the classifier and to create a more robust and generalist model, a sufficient number of labeled samples are often required. However, in order to make the model as efficient as possible, the training set should be kept small and focused on the pixel samples that really help to improve

the performance of the model. Moreover, the labeled samples are very difficult, expensive, and time consuming to collect in practice [66] and often only a few labeled samples are available in advance. This issue is particularly problematic in the hyperspectral image classification, since there is often an unbalance between the high dimensionality of the data and the limited number of training samples available [67], known as the curse of dimensionality or the *Hughes effect* [68]. As result, the ANN model may overfit the training data, which reduces its generalization capacity [69]. Some methods address this problem by using data augmentation techniques to generate additional training samples, performing basic transformations in the initial data set. Lu *et al.* [17] provide a method that offers robustness and flexibility in modeling scene images and report the improvements of the accuracy in scene recognition, constructing multiresolution features and modeling sparse features' selection-based manifold regularization. In contrast, AL [70] has been used to facilitate the classification of hyperspectral image data sets, by including intelligently selected unlabeled samples from the original data set, i.e., the most informative samples, to the training set. This reduces the cost of acquiring large labeled training sets and the number of needed training samples [71]–[75].

However, the combination of AL with deep architectures, such CNNs, has been more difficult. This is because the goal of AL is to create a model composed by a predictor trained on a small set of well-chosen examples that can perform as efficiently as a predictor trained on a larger number of examples randomly chosen, while being computationally tractable, but CNNs often require large amounts of training data for training and are highly prone to overfitting when they are trained with small data sets. Also, similar to MLR, AL techniques rely on probabilistic functions, which indicate the probability of a sample to belong to the different existing categories, in order to create a model uncertainty but deep architectures normally do not represent model uncertainty, obtaining as final output the predicted class label instead the probability of each class. Moreover, conventional ANNs in general (and CNNs in particular) are based on the minimization of an error function [76], typically the least squared-error function between the desired class label and the obtained one in classification, and they cannot determine the level of uncertainty of their output results. In fact, the proposed model must be able to extract an output probability matrix from input data in order to apply the AL probabilistic function (called ranking function) and to extract those samples with more uncertainty, which will provide more information to the model.

To address this issue, in this paper, we consider BNNs [77], [78], a special kind of ANN that is robust to overfitting and is able to offer uncertainty estimates and a probabilistic interpretation of DL models by inferring distributions over the models' weights, being able to learn from small data sets [79] and avoiding the tendency of conventional ANNs to make overconfident predictions in sparse data regions. In fact, we can consider BNNs as an extension of standard ANNs with posterior inference, adding a probability distribution on its weights [78], [80]. Recent works have showed that the Bayesian approach to CNNs (called hereinafter B-CNNs) can

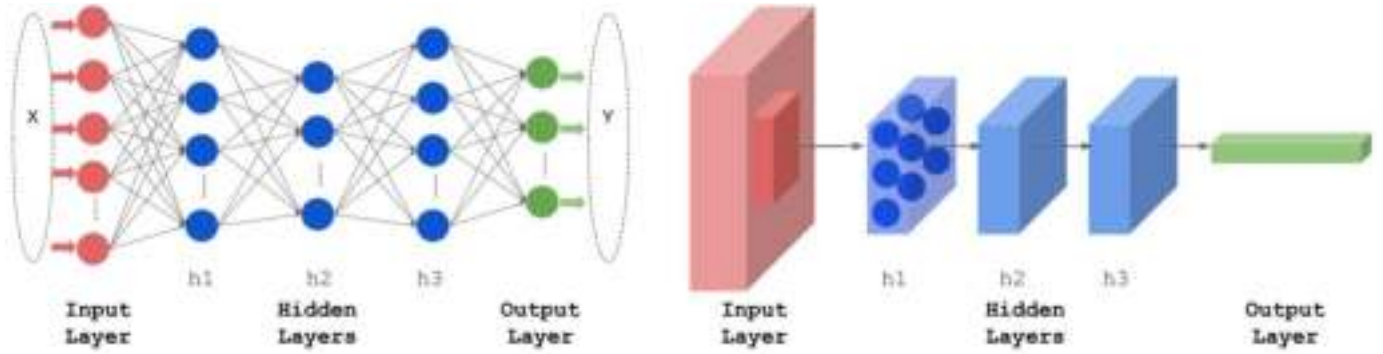


Fig. 1. Comparison between a conventional DNN (a multilayer perceptron or MLP) with three hidden fully connected layers and a CNN with three hidden CONV layers or kernels. The neurons in the CNN create 3-D blocks with sparse connectivity.

offer robustness to overfitting on small data sets and improve their generalization capacity adding dropout at every weight layer (also called convolutional layer) of the CNNs, as a Bayesian approximation of the probabilistic model defined by the Gaussian process [81]–[83], which allows to represent the model uncertainty without introducing major changes to the network architecture. By taking advantage of BNNs (that offer good uncertainty estimates and are robust to overfitting) and following the methodology of [83], we propose, for the first time in the literature, an AL model with B-CNNs for spectral–spatial classification of hyperspectral remotely sensed data. The main innovative contributions of this paper can be summarized as follows:

- 1) the development (for the first time in the literature) of a dropout-based method (called B-CNN) to extract probabilistic information from 1-D, 2-D, and 3-D CNN models with the aim of performing accurate spectral–spatial feature-based classification of hyperspectral images using limited training data with different CNN architectures;
- 2) the development of a three-step-based training phase to perform AL over the proposed B-CNN for the first time in the hyperspectral image classification literature;
- 3) the exhaustive analysis and comparison of different acquisition functions to perform AL over the implemented B-CNN model, and a detailed comparison between the implemented B-CNNs and the standard 1-D, 2-D, and 3-D CNN classifiers for hyperspectral data in addition to other traditional hyperspectral data classifiers, such as RF, MLP, SVM, and MLR.

The remainder of this paper is organized as follows. Section II provides an overview of related works and presents the newly developed classifier model. Section III validates the proposed approach using three well-known hyperspectral data sets, highlighting the advantages of the newly proposed classifier. Finally, Section IV concludes this paper with some remarks and hints at plausible future research lines.

## II. METHODOLOGY

### A. Active Learning

AL has been adopted in remote sensing as an effective strategy to reduce the cost of acquiring large labeled training sets [75], and it is based on three main aspects: 1) the

---

### Algorithm 1 AL (General Algorithm)

---

```

1: procedure AL( $\mathcal{D}_{\text{train}}^\epsilon, \mathcal{D}_{\text{pool}}^\epsilon, N$ ) ▷
    $\mathcal{D}_{\text{train}}^\epsilon = \{\mathbf{x}_i, y_i\}_{i=1}^l \in \mathbb{R}^d, \epsilon = 1 \rightarrow$  initial training set,
    $\mathcal{D}_{\text{pool}}^\epsilon = \{\mathbf{x}_i\}_{i=l+1}^{l+u} \in \mathbb{R}^d, \epsilon = 1 \rightarrow$  pool of candidates
   (pool set),  $N \rightarrow$  number of pixel samples to add at each
   iteration (until reaching a final batch of selected pixels,
    $\mathcal{D}_{\text{selected}}^\epsilon$ )
2:   repeat
3:     Train the model with the current training set  $\mathcal{D}_{\text{train}}^\epsilon$ 
4:     for  $\mathbf{x}_i \in \mathcal{D}_{\text{pool}}^\epsilon$  do
5:       Evaluate a user-defined heuristic
6:     end for
7:     Rank the candidates  $\mathbf{x}_i$  in  $\mathcal{D}_{\text{pool}}^\epsilon$  according to the
       heuristic score
8:      $\mathcal{D}_{\text{selected}}^\epsilon = \{\mathbf{x}_k\}_{k=1}^N \rightarrow$  select the  $N$  pixels with
       higher score
9:      $\mathcal{D}_{\text{selected}}^\epsilon = \{\mathbf{x}_k, y_k\}_{k=1}^N \rightarrow$  assign label to the  $N$ 
       selected pixels.
10:     $\mathcal{D}_{\text{train}}^{\epsilon+1} = \mathcal{D}_{\text{train}}^\epsilon \cup \mathcal{D}_{\text{selected}}^\epsilon \rightarrow$  add the batch
11:     $\mathcal{D}_{\text{pool}}^{\epsilon+1} = \mathcal{D}_{\text{pool}}^\epsilon - \mathcal{D}_{\text{selected}}^\epsilon \rightarrow$  remove batch from
       pool
12:     $\epsilon = \epsilon + 1 \rightarrow$  update index iteration
13:   until Classification result is acceptable
14: end procedure

```

---

availability of an initial training set; 2) the availability of a pool set; and 3) the use of an acquisition function. Let us denote by  $\mathcal{D}_{\text{train}} = [X, Y] = \{\mathbf{x}_i, y_i\}_{i=1}^l$  a training set made up of  $l$  labeled samples (where  $\mathbf{x}_i \in \mathbb{R}^d = [x_{i,1}, x_{i,2}, \dots, x_{i,3}]$  is the input data, in our case a hyperspectral pixel vector, and  $y_i = \{1, 2, \dots, C\}$  is the corresponding label, with  $C$  the number of different categories or classes) and  $\mathcal{D}_{\text{pool}} = [X] = \{\mathbf{x}_i\}_{i=l+1}^{l+u} \in \mathbb{R}^d$  the pool of candidates, i.e., a set of  $u$  unlabeled samples ( $u \gg l$ ). The AL model is generally composed by a learner (trained with a few labeled samples,  $\mathcal{D}_{\text{train}}$ ) that iteratively selects new training examples from the pool of candidates ( $\mathcal{D}_{\text{pool}}$ ) that provide maximal information about the unlabeled data set and improve the model performance [74]. Algorithm 1 provides a general approximation of how AL works. As a result of the process illustrated in Algorithm 1, the classification accuracy given by the final selected training set is expected to be higher than the one obtained by using

randomly selected labeled samples. The acquisition function, in particular the user-defined heuristic, is a crucial point in AL. Tuia *et al.* [74] make a compilation of several heuristic methods, proposing a taxonomy of AL techniques. Here, we rely on posterior probability-based AL methods. These methods use the estimation of posterior probabilities of class membership,  $p(y|\mathbf{x})$ , to rank the candidates in  $\mathcal{D}_{\text{pool}}$ . This kind of probability gives us an idea of the confidence of the class assignment, i.e., how good the classification is. However, DNNs, in general, and CNNs, in particular, normally do not calculate an uncertainty model that is needed for these AL methods. In Section II-B, we summarize how this problem is solved in [83].

### B. Bayesian-Convolutional Neural Networks

In contrast to conventional ANNs, the blocks or layers of neurons in CNNs operate such as kernels which are connected and applied over one region of the input image (also referred to as *input volume* hereinafter), i.e., layers are not fully connected to all neurons of the previous layer as in the standard multilayer perceptron or MLP (see Fig. 1). Each layer actually composes a feature extraction stage that can be of three kinds [64], [84].

- 1) *CONV Layer*: A layer where each node is in charge of computing the dot product ( $\cdot$ ) between its own weights and a predefined region of the provided input volume to which it is connected. Actually, these layers work as kernels or filters where nodes share the same weights and bias, connecting the input volume to the output volume. Let us suppose a CONV layer that receives as input volume the data cube  $X \in \mathbb{R}^{d \times d \times n}$ , where  $d$  represents the height and width and  $n$  represents the deep of the cube (also spectral bands). Each neuron in one filter that composes the CONV layer ( $k$  is the number of filters) will operate with a chunk of  $X$ , in particular, with a  $l \times l \times q$  chunk (called *filter bank*,  $W^c$ ). We can calculate the output of the neuron ( $i, j, t$ ) in the  $k$ th filter of the CONV layer as

$$z_{i,j,t} = (X \cdot W^c)_{i,j,t} = \sum_{\hat{i}=0}^{l-1} \sum_{\hat{j}=0}^{l-1} \sum_{\hat{t}=0}^{q-1} X_{(i-s+\hat{i}), (j-s+\hat{j}), (t-s+\hat{t})} \cdot w_{\hat{i}, \hat{j}, \hat{t}} + b \quad (1)$$

where  $z_{i,j,t}$  is the element ( $i, j, t$ ) in the  $k$ th feature map,  $x_{i,j,t}$  is an element of input data  $X$ ,  $w_{\hat{i}, \hat{j}, \hat{t}}$  is a weight of the cube of weights  $W$ ,  $b$  is the bias, and  $s$  is the stride of the CONV layer. In fact, we can observe each filter as a window that moves itself on  $X$  in chunks of size  $l \times l \times q$  with a displacement dictated by  $s$ . As a result, an output volume  $Z$  is obtained, which will be an array composed by  $k$  1-, 2-, or 3-D feature maps depending on the kernel's dimension.

- 2) *Nonlinearity Layer*: This layer is used to implement a nonlinear function (such as the sigmoid function or the ReLU [85]–[87]), which is then applied to each

component of the obtained feature map to learn non-linear representations:  $A = f(Z)$ .

- 3) *Pooling Layer*: Pooling layers are used to resume the output  $Z$  of several nodes in CONV layers using a pooling function. In addition, they also provide location invariant features. Normally, this layer executes a max operation (MAXPOOL layer) within a small region  $R$  defined by a kernel  $l \times l \times q$  over the resulting volume  $A$  after the nonlinearity layer, i.e.,  $A$  is divided into several nonoverlapping maps, whose maximum values are mapped into the final output volume  $P = \max_{i \in R} A_i$ .

Two characteristics make CNNs an ideal model for processing and classifying hyperspectral images: the sparse connectivity and shared weights. These features allow us to reduce the number of parameters to be learned by the network ensuring some degree of shift, scale, and distortion invariance. However, CNNs require huge amounts of data for regularization due to their model's complexity and to the fact that they quickly overfit with small training sets. Such an overfitting problem makes that CNNs exhibit poor predictive performance in the testing phase. To avoid this problem, we adopt BNNs due to their robustness to overfitting and to their capacity to learn from small training sets.

Specifically, we use B-CNNs that combine the features of BNNs with the classification potential of CNNs. Another advantage of B-CNNs in our context is that they provide the uncertainty model that we need to apply AL techniques. Given a training data set  $\mathcal{D}_{\text{train}} = [X, Y]$  composed by inputs  $X = \{x_1, \dots, x_l\}$  (where each  $\mathbf{x}_i \in \mathbb{R}^n = [x_{i,1}, x_{i,2}, \dots, x_{i,n}]$ ) and their corresponding outputs  $Y = \{y_1, \dots, y_l\}$  (where each  $y_i = \{1, 2, \dots, c\}$ ), the model posterior's goal is to estimate a function  $y_i = f(\mathbf{x}_i)$  as close as possible to the original function that has generated the outputs  $Y$ . The Bayesian approach proposes to put some *prior* distribution over the space of functions  $p(f)$ , so we can define a probability or *likelihood* on the output  $Y$  given the input  $X$  and a function  $f$ ,  $p(Y|X, f)$ . Therefore, the posterior distribution will be  $p(f|X, Y) = p(f|\mathcal{D}_{\text{train}})$  that captures the most likely functions given the observed data. In this way, the output  $y^*$  of a new input  $\mathbf{x}^*$  can be predicted as the marginal likelihood

$$p(y^*|\mathbf{x}^*, \mathcal{D}_{\text{train}}) = \int p(y^*|f^*)p(f^*|\mathbf{x}^*, \mathcal{D}_{\text{train}})df^*. \quad (2)$$

As (2) is normally intractable, we can approximate it adding a finite set of random variables  $\omega$  as follows:

$$p(y^*|\mathbf{x}^*, \mathcal{D}_{\text{train}}) = \int p(y^*|f^*)p(f^*|\mathbf{x}^*, \omega)p(\omega|\mathcal{D}_{\text{train}})df^*d\omega. \quad (3)$$

In a BNN with weights  $W_i$  of size  $K_i \times K_{i-1}$  for each layer  $i$ , the set of finite variables or parameters will be  $\omega = \{W_i\}_{i=1}^L$  (where  $L$  is the number of layers), and the posterior over  $\omega$  given  $X$  and  $Y$  will be  $p(\omega|\mathcal{D}_{\text{train}})$ . However, the probability distribution  $p(\omega|\mathcal{D}_{\text{train}})$  is not tractable for a BNN. To infer the model posterior in a simple way, [83] proposes the use of variational inference as an approach based on Bernoulli approximation variational distributions (and relating this to dropout training) with the aim of not increasing the number of

parameters to be trained, as in other types of approaches such as the variational inference with Gaussian [88]. The first step is to define the approximating variational distribution  $q(W_i)$  for each BNN's layer  $i$  ( $i = 1, \dots, L$ ) as

$$\begin{aligned} W_i &= M_i \cdot \text{diag}([z_{i,j}]_{j=1}^{K_i}) \\ i &= 1, \dots, L, \\ j &= 1, \dots, K_{i-1} \end{aligned}$$

where  $\text{diag}$  is the diagonal matrix with elements  $z_{i,j}$  that are Bernoulli distributed random variables with probabilities  $p_i$  and  $M_i$  are variational parameters to be optimized. Predictions follow (3). The change resides in replacing the intractable probability distribution  $p(\omega|\mathcal{D}_{\text{train}})$  by the approximate distribution  $q(\omega)$  that belongs to a tractable family, which minimizes the Kullback–Leibler divergence,  $D_{KL}(q(\omega)||p(\omega|\mathcal{D}_{\text{train}}))=0$ , a measure that returns the similarity between both the distributions

$$q(y^*|\mathbf{x}^*) = \int p(y^*|f^*)p(f^*|\mathbf{x}^*, \omega)q(\omega)df^*d. \quad (4)$$

Using Monte Carlo integration, we can approximate the integral so that we can predict the probability that the output  $y^*$  corresponds to label  $c$  as follows:

$$\begin{aligned} p(y^* = c|\mathbf{x}^*, \mathcal{D}_{\text{train}}) &= \int p(y^* = c|\mathbf{x}^*, \omega)p(\omega|\mathcal{D}_{\text{train}})d\omega \\ &\approx \int p(y^* = c|\mathbf{x}^*, \omega)q(\omega)d\omega \\ &\approx \frac{1}{T} \sum_{t=1}^T p(y^* = c|\mathbf{x}^*, \widehat{\omega}_t) \end{aligned}$$

being  $\widehat{\omega}_t \sim q(\omega)$  called MC-dropout, while  $T$  are the stochastic forward passes. This Bernoulli approximation variational inference in BNNs can be implemented by adding dropout layers after certain weight layers in a network [83]. In the B-CNN model, this is the same than adding dropout to all CONV layers as well as inner-product layers.

### C. AL Acquisition Function

The AL acquisition function  $a(\mathbf{x}, \mathcal{M})$  of a model  $\mathcal{M}$  with pool data  $\mathcal{D}_{\text{pool}}$  and inputs  $\mathbf{x} \in \mathcal{D}_{\text{pool}} \in \mathbb{R}^d$  decides which data points  $\mathbf{x}$  will be queried by an external *oracle*, which could be a human expert that performs the work of classifying the unlabeled data to be added to the training set  $\mathcal{D}_{\text{train}}$

$$S = \arg \max_{\mathbf{x} \in \mathcal{D}_{\text{pool}}} a(\mathbf{x}, \mathcal{M}).$$

Gal *et al.* [89] make a review and a comparison between different acquisition functions. This paper performs a comparison with six different acquisition methods that have been adapted to AL methodology, taking into account different measurements, such as the entropy value and distances of the samples, among the random selection of samples.

1) *Random Acquisition or Baseline*: It chooses a point  $\mathbf{x}_i$  following a uniformly random distribution from  $\mathcal{D}_{\text{pool}} \Rightarrow a(\mathbf{x}_i) = \text{unif}()$ , where  $\text{unif}()$  returns a draw from a uniform distribution over the interval  $[0, 1]$ .

2) *Mean STD* [90]: For each  $\mathbf{x}_i$ , it calculates  $\sigma(\mathbf{x}_i) = (1/C) \sum_c \sigma_c$ , where  $C$  is the number of classes,  $c$  are the classes that  $\mathbf{x}_i$  can take, and

$$\sigma_c = \sqrt{\mathbb{E}_{q(\omega)}[p(y_i = c|\mathbf{x}_i, \omega)^2] - \mathbb{E}_{q(\omega)}[p(y_i = c|\mathbf{x}_i, \omega)]^2}.$$

3) *Maximum Entropy* [91]: It chooses  $\mathbf{x}_i \in \mathcal{D}_{\text{pool}}$  with the highest classification uncertainty, i.e.,  $\mathbf{x}_i$  that maximizes the predictive entropy

$$\begin{aligned} \mathbb{H}[y_i|\mathbf{x}_i, \mathcal{D}_{\text{train}}] &:= \\ &- \sum_c p(y_i = c|\mathbf{x}_i, \mathcal{D}_{\text{train}}) \log p(y_i = c|\mathbf{x}_i, \mathcal{D}_{\text{train}}). \end{aligned}$$

4) *BALD* [92]: This method chooses  $\mathbf{x}_i \in \mathcal{D}_{\text{pool}}$  that are expected to maximize the mutual information between the predictions and the model posterior

$$\begin{aligned} \mathbb{I}[y_i, \omega|\mathbf{x}_i, \mathcal{D}_{\text{train}}] &:= \\ &:= \mathbb{H}[y_i|\mathbf{x}_i, \mathcal{D}_{\text{train}}] - \mathbb{E}_{p(\omega|\mathcal{D}_{\text{train}})}[\mathbb{H}[y_i|\mathbf{x}_i, \omega]] \end{aligned}$$

where  $\mathbb{H}[y_i|\mathbf{x}_i, \mathcal{D}_{\text{train}}]$  is the entropy [91]. The selected points exhibit high variance in the input to the softmax layer.

5) *Breaking Ties Criterion (BT-Criterion)* [93], [94]: This method focuses on the boundary region between two classes with the aim of obtaining more diversity in the composition of the training set  $\mathcal{D}_{\text{train}}$ . Sample  $\mathbf{x}^{BT}$  is selected from  $\mathcal{D}_{\text{pool}}$  by

$$\begin{aligned} \mathbf{x}^{BT} &= \arg \min_{\mathbf{x}_i \in \mathcal{D}_{\text{pool}}} \\ &\left\{ \max_{c \in C} p(y_i = c|\mathbf{x}_i, \omega) - \max_{c \in C \setminus \{c^+\}} p(y_i = c|\mathbf{x}_i, \omega) \right\} \end{aligned}$$

where  $c^+ = \arg \max_{c \in C} p(y_i = c|\mathbf{x}_i, \omega)$  is the most probable label class for sample  $\mathbf{x}_i$

6) *Mutual Information Criterion* [70], [94]: It measures the mutual dependence between samples. In fact, this function selects the sample  $\mathbf{x}^{\text{MI}}$  maximizing the MI between the obtained results and the class labels

$$\mathbf{x}^{\text{MI}} = \arg \max_{\mathbf{x}_i \in \mathcal{D}_{\text{pool}}} \mathbb{I}(\omega; y_i|\mathbf{x}_i)$$

where  $\mathbb{I}(\omega; y_i|\mathbf{x}_i) = (1/2) \log(|H^{\text{MI}}|/|H|)$  represents the MI between the obtained results and the class label  $y_i$ , with  $H$  the posterior precision matrix and  $H^{\text{MI}}$  the posterior precision matrix after including the new sample  $\mathbf{x}_i$ .

In order to compare these acquisition functions, they have been adapted to be executed with AL methodology. For example, the BALD method has been approximated with  $q(\omega)$ , as described by [89]

$$\begin{aligned} \mathbb{I}[y_i, \omega|\mathbf{x}_i, \mathcal{D}_{\text{train}}] &:= \\ &:= \mathbb{H}[y_i|\mathbf{x}_i, \mathcal{D}_{\text{train}}] - \mathbb{E}_{p(\omega|\mathcal{D}_{\text{train}})}[\mathbb{H}[y_i|\mathbf{x}_i, \omega]] \\ &= - \sum_c p(y_i = c|\mathbf{x}_i, \mathcal{D}_{\text{train}}) \log p(y_i = c|\mathbf{x}_i, \mathcal{D}_{\text{train}}) \\ &\quad + \mathbb{E}_{p(\omega|\mathcal{D}_{\text{train}})} \left[ \sum_c p(y_i = c|\mathbf{x}_i, \omega) \log p(y_i = c|\mathbf{x}_i, \omega) \right]. \end{aligned}$$

If we consider the BALD equation the identity  $p(y_i = c | \mathbf{x}_i, \mathcal{D}_{\text{train}}) = \int p(y_i = c | \mathbf{x}_i, \omega) p(\omega | \mathcal{D}_{\text{train}}) d\omega$ , we have

$$\begin{aligned} \mathbb{I}[y_i, \omega | \mathbf{x}_i, \mathcal{D}_{\text{train}}] &:= - \sum_c \int p(y_i = c | \mathbf{x}_i, \omega) p(\omega | \mathcal{D}_{\text{train}}) d\omega \\ &\cdot \log \int p(y_i = c | \mathbf{x}_i, \omega) p(\omega | \mathcal{D}_{\text{train}}) d\omega \\ &+ \mathbb{E}_{p(\omega | \mathcal{D}_{\text{train}})} \left[ \sum_c p(y_i = c | \mathbf{x}_i, \omega) \log p(y_i = c | \mathbf{x}_i, \omega) \right]. \end{aligned}$$

Now, we can apply Monte Carlo integration as follows:

$$\begin{aligned} \mathbb{I}[y_i, \omega | \mathbf{x}_i, \mathcal{D}_{\text{train}}] &:= - \sum_c \int p(y_i = c | \mathbf{x}_i, \omega) q(\omega) d\omega \\ &\cdot \log \int p(y_i = c | \mathbf{x}_i, \omega) q(\omega) d\omega \\ &+ \mathbb{E}_{q(\omega)} \left[ \sum_c p(y_i = c | \mathbf{x}_i, \omega) \log p(y_i = c | \mathbf{x}_i, \omega) \right] \\ &\approx - \sum_c \left( \frac{1}{T} \sum_{t=1}^T \hat{p}_c^t \right) \log \left( \frac{1}{T} \sum_{t=1}^T \hat{p}_c^t \right) \\ &+ \frac{1}{T} \sum_{c,t} \hat{p}_c^t \log \hat{p}_c^t. \end{aligned}$$

#### D. Proposed B-CNN Architecture for Active Learning

Finally, we present the new B-CNN architecture developed in this paper. It should be noted that the literature on CNNs applied to hyperspectral image classification shows different points of view on how the spatial and the spectral information in the original hyperspectral image can be used:

- 1) extracting only spectral information implementing a 1-D CNN architecture [14], [37], [56];
- 2) extracting only spatial information implementing a 2-D CNN architecture [57], [95]–[97];
- 3) extracting spectral–spatial information implementing a 3-D CNN architecture [37], [63].

In this regard, we emphasize that our B-CNN approach can be applied to 1-D, 2-D, and 3-D CNN architectures. This paper investigates the effects of applying the proposed Bayesian network to CNN models with the aim of performing hyperspectral classification based on spectral, spatial, and spectral–spatial features. In this sense, three B-CNN models have been implemented: 1-D, 2-D B-CNN, and 3-D B-CNN.

1) *Spectral B-CNN Architecture*: This model takes advantage of only the spectral information contained in the input hyperspectral image by developing a 1-D CNN architecture and performing a traditional pixelwise-based learning. Given the hyperspectral image  $X \in \mathbb{R}^{h \times w \times n}$ , where  $h$  and  $w$  are the height and width, respectively, and  $n$  is the number of spectral bands, the 1-D B-CNN model will take as input data pixel vectors of the hyperspectral scene  $X$ ,  $\mathbf{x}_i \in \mathbb{R}^n = [x_{i,1}, x_{i,2}, \dots, x_{i,n}]$ , where  $i = 1, 2, \dots, (h \cdot w)$ . In this case, each input pixel vector  $\mathbf{x}_i$  is transformed through the net

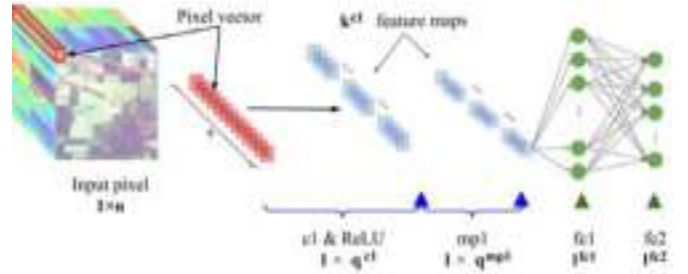


Fig. 2. Proposed spectral B-CNN (1-D B-CNN) architecture.

into feature maps, capturing the spectral information contained in  $\mathbf{x}_i$ . Equation (1) can be rewritten as

$$z_t = (\mathbf{x} \cdot W^c)_t = \sum_{\hat{i}=0}^{q-1} x_{(t,s+\hat{i})} \cdot w_{\hat{i}} + b \quad (5)$$

where  $q$  is the depth of the kernel,  $z_t$  is the  $t$ th neuron's output in the  $k$ th filter,  $x_t$  is one spectral band of the CONV layer's input  $\mathbf{x}$ ,  $W^c$  is the filter bank of the layer, characterized by  $k$  kernels of size  $1 \times q$ ,  $w_{\hat{i}}$  is a weight of vector  $W$ ,  $b$  is the bias of the layer, and  $s$  is the stride.

In order to compare the implemented 1-D B-CNN model with the 1-D CNN baseline, the model's architecture has been inspired by [56]. As we can see in Fig. 2, the proposed 1-D B-CNN model is composed by one input layer that receives the pixel vector with all its spectral bands. This input feeds one CONV layer,  $c1$ , with  $k^{c1}$  kernels of size  $1 \times q^{c1}$ , followed by the ReLU activation function and one maxpool layer,  $mp1$ , whose kernel size is  $1^{mp1}$ . The output of  $mp1$  is reshaped into a vector in order to feed two fully connected layers at the end of the network. After the maxpool and first fully connected layers, dropout is implemented in order to perform the MC-dropout. Table I shows the details of the 1-D B-CNN implementation.

2) *Spatial B-CNN Architecture*: This model takes advantage of only the spatial information contained in the input image, reducing the number of spectral bands  $n$  to 1 by applying PCA over original hyperspectral data sets. As result, a 2-D CNN architecture has been implemented, whose input is composed by patches of size  $d \times d \times 1$  extracted from the hyperspectral scene. Normally, CNNs in general (and B-CNNs in particular) receive a completely normalized image prior to classification, i.e., a 3-D input array. However, in hyperspectral images, the classes are often mixed, so we feed the pixel (vectors of  $1 \times n$ ) one by one to the B-CNN. This allows us to exploit the rich spectral information contained in the hyperspectral data in the case of the 1-D B-CNN, but we also need an additional mechanism in order to include also the spatial information in the 2-D model. In this case, we feed the network with the pixels that belong into a neighborhood window centered around each pixel under consideration. In this way, the input layer of the 2-D model accepts volumes of  $d \times d \times 1$  [64], after processing the original scene with PCA. This requires a preprocessing stage in order to create patches of  $d \times d \times 1$  for each pixel, where the desired label to be reached by the network will be the one owned by the central

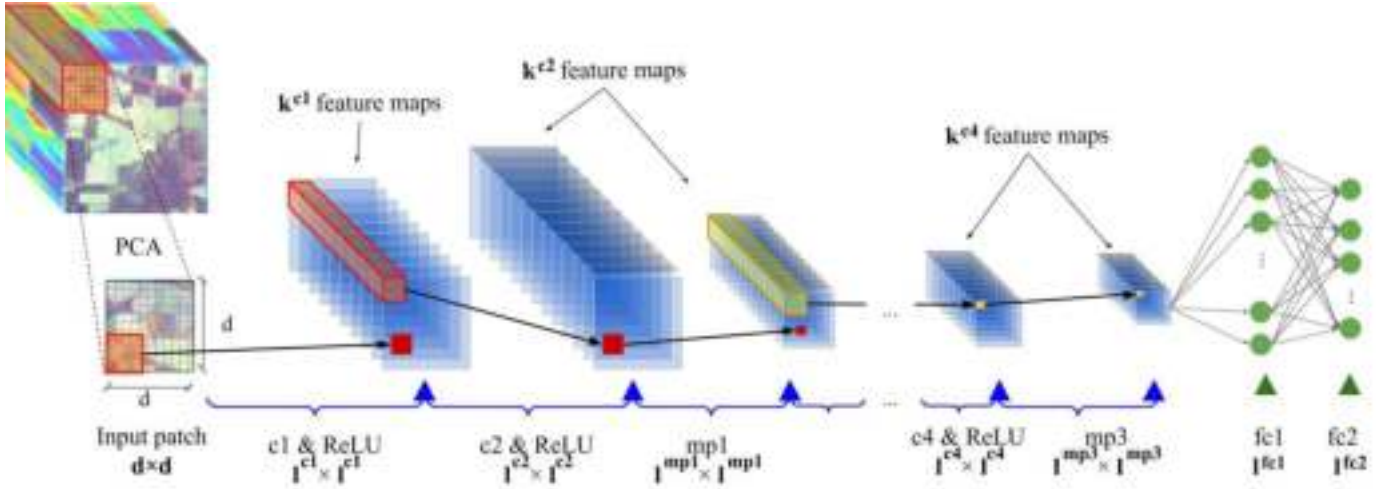


Fig. 3. Proposed spatial B-CNN (2-D B-CNN) architecture.

pixel of the patch  $[d/2 + 1, d/2 + 1, n]$ . In this case, (1) can be rewritten as

$$z_{i,j} = (X \cdot W^c)_{i,j} = \sum_{\hat{i}=0}^{l-1} \sum_{\hat{j}=0}^{l-1} x_{(i-s+\hat{i}), (j-s+\hat{j})} \cdot w_{\hat{i}, \hat{j}} + b \quad (6)$$

where  $l$  is the height and width of the kernel,  $z_{i,j}$  is the output of the neuron  $(i, j)$  in the  $k$ th filter,  $x_{i,j}$  is one pixel of the input patch  $X \in \mathbb{R}^{d \times d \times 1}$ ,  $W^c$  is the filter bank of the layer, characterized by  $k$  kernels of size  $l \times l$ ,  $w_{\hat{i}, \hat{j}}$  is a weight of matrix  $W$ , and  $b$  is the bias of the layer and  $s$  the stride.

Fig. 3 shows the implemented 2-D B-CNN model. In this case, a deeper architecture has been selected. As we can observe, the input layer receives the hyperspectral patches of size  $d \times d \times 1$ , which feeds the first CONV layer  $c1$ . After that, three pairs of CONV+maxpool layers are implemented,  $c2$  and  $mp1$ ,  $c3$  and  $mp2$ , and finally  $c4$  and  $mp3$ . The network ends with two fully connected layers  $fc1$  and  $fc2$ , where the last one performs the final classification. Dropout has been added at the end of certain layers in order to model the uncertainty of the network. Table I shows the details of the 2-D B-CNN implementation.

3) *Spectral-Spatial B-CNN Architecture*: This model takes advantage of both the spectral and the spatial information in the input hyperspectral image by developing a 3-D architecture that receives as input data patches of size  $d \times d \times n$ , where  $n$  is the number of spectral bands. As in the 2-D B-CNN model, a preprocessing stage is required in order to create the input patches for each pixel [64], where the desired label to be reached by the network will be the central pixel of the patch  $[d/2 + 1, d/2 + 1, n]$ . In this case, each CONV layer performs (1).

As we can see in Fig. 4, the proposed spectral-spatial B-CNN consists of an input layer that receives the input patches, two CONV layers,  $c1$  and  $c2$  (with ReLU as a nonlinear activation function), two maxpool layers at the end of each CONV layer,  $mp1$  and  $mp2$ , and two fully connected layers,  $fc1$  and  $fc2$ . The last one is the output layer, which obtains the desired label for the input data. After each maxpool layer, a dropout layer is inserted in order to model the probability of the proposed network that will allow to obtain

TABLE I  
CONFIGURATION OF OUR THREE B-CNN ARCHITECTURES.  
C INDICATES THE NUMBER OF CLASSES CONTAINED  
IN THE HYPERSPECTRAL DATA SET

	Convolution layers			
	Kernel $k^c \times l^c \times q^c$	Activation Function	Pooling $l^{mp} \times l^{mp}$	Dropout (%)
1D B-CNN	$20 \times 1 \times 24$	ReLU	$1 \times 5$	5%
	Fully connected layers			
	Neurons $l^{fc}$	Activation Function	Dropout (%)	
	100	ReLU	1%	
	$C$	Softmax	-	
2D B-CNN	Convolution layers			
	Kernel $k^c \times l^c \times l^c$	Activation Function	Pooling $l^{mp} \times l^{mp}$	Dropout (%)
	$50 \times 3 \times 3$	ReLU	-	25%
	$100 \times 5 \times 5$	ReLU	$2 \times 2$	25%
	$200 \times 5 \times 5$	ReLU	$2 \times 2$	25%
	$400 \times 2 \times 2$	ReLU	$1 \times 1$	25%
	Fully connected layers			
Neurons $l^{fc}$	Activation Function	Dropout (%)		
300	ReLU	50%		
$C$	Softmax	-		
3D B-CNN	Convolution layers			
	Kernel $k^c \times l^c \times l^c \times q^c$	Activation Function	Pooling $l^{mp} \times l^{mp} \times q^{mp}$	Dropout (%)
	$500 \times 3 \times 3 \times n$	ReLU	$2 \times 2 \times 1$	10%
	$100 \times 5 \times 5 \times 500$	ReLU	$2 \times 2 \times 1$	10%
	Fully connected layers			
	Neurons $l^{fc}$	Activation Function	Dropout (%)	
300	ReLU	5%		
$C$	Softmax	-		

the uncertainty estimation. Table I provides additional details about the considered spectral-spatial B-CNN architecture.

The parameters of the considered 1-D, 2-D, and 3-D architectures, including the number and type of layers or kernel sizes, are one of the design choices of the proposed spectral, spatial, and spectral-spatial B-CNN models. In this sense, the architectures have been selected and defined in a way that is as general as possible to adapt them to different hyperspectral images [64]. In addition, our decision to use the same architectures for different data sets further illustrates that the proposed method can achieve good classification results on very different images, extracting the samples that maximize the information gained about the model, improving



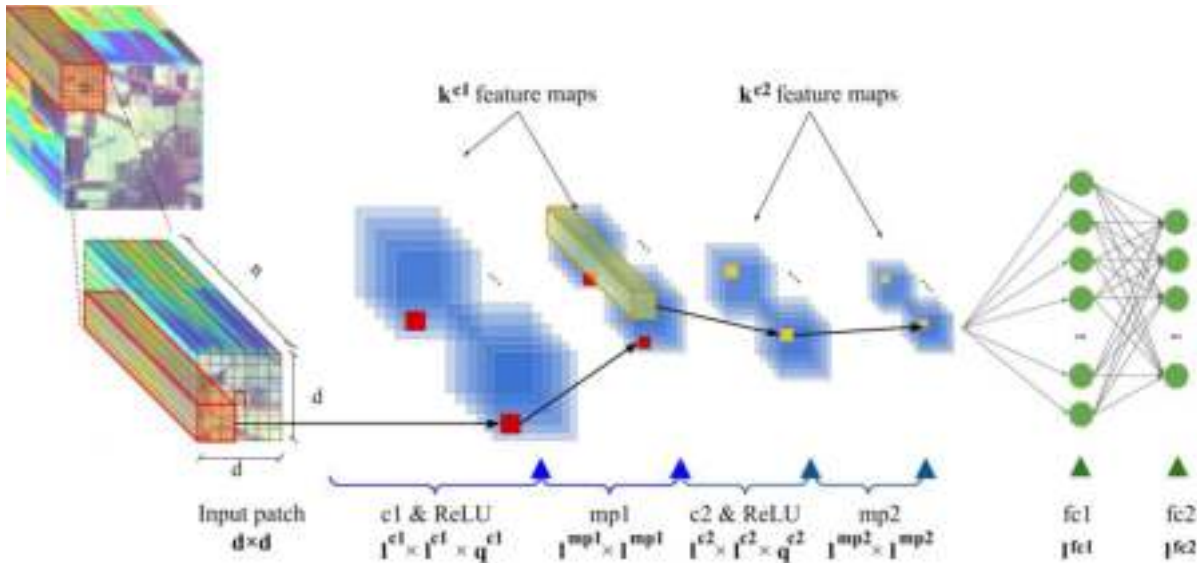


Fig. 4. Proposed spectral-spatial B-CNN (3-D B-CNN) architecture.

the training, and showing its robustness, regardless of the fact that nonoptimal or customized topologies are adopted.

Also, we must remark that the proposed 1-D, 2-D, and 3-D B-CNN models have been developed as a computation graph using the library for machine intelligence Keras with TensorFlow back end over CUDA toolkit and the library of primitives for DNNs cuDNN. This computation graph is composed by connected nodes that represents operations (also called units of computation), while connections (or edges) represent the data consumed (input connections) and produced (output connections) in the unit. These connections allow us to represent the existing dependences between different operations, making it possible to identify those operations that can be executed in parallel in an easy way.

The process of our 1-D, 2-D, and 3-D B-CNNs follows two main steps. In a first step, the hyperspectral image is first loaded and a band-mean normalized version is calculated so that the values of the image are in the range  $[0, 1]$ . Then, the hyperspectral image's ground truth is divided into two data sets: two randomly selected samples per class,  $2 \cdot C$ , will compose the initial *training set*,  $\mathcal{D}_{\text{train}}^0$ , and the remaining samples will compose the *working set*. From the *working set*, 50% of random selected pixels will compose the initial *pool set*,  $\mathcal{D}_{\text{pool}}^0$ , and the remaining 50% will be divided into testing samples (*testing set* with the 95% of samples) and validation samples (*validation test* with 5% of samples).

The next step is given by Algorithm 1. At this point, and with the aim of reducing the use of storage by the algorithm, the hyperspectral data are preprocessed in order to create the input samples that will feed the model (i.e., pixels vectors  $1 \times n$ , or patches  $d \times d \times 1$  or  $d \times d \times n$ ). In this sense, Algorithm 1 has been adapted to perform, for the first time in the literature, AL over the new B-CNN models for spectral, spatial, and spectral-spatial classification of hyperspectral data in an efficient way, both computationally and in terms of memory management. In  $\epsilon = 0$ , the training set  $\mathcal{D}_{\text{train}}^0$  and the pool set  $\mathcal{D}_{\text{pool}}^0$  are created as sets of  $1 \times n$  pixel arrays.

Then, the B-CNN models are trained by a three-step process.

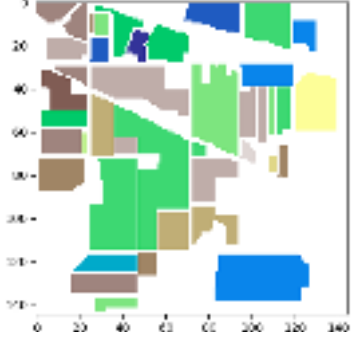

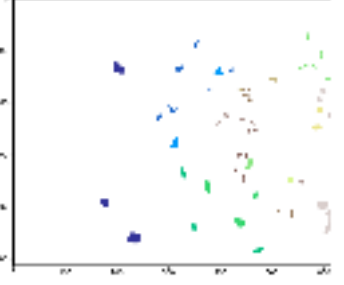
- 1) If we are working with 2-D or 3-D models, for each pixel  $x_i \in \mathcal{D}_{\text{train}}^0$ , we create a patch of size  $d \times d \times 1$  or  $d \times d \times n$ , depending on the model's dimension, centered on the pixel  $x_i$ , and assign the pixel's label,  $y_i$ , to the patch. Once we have created the patches, these are sent to the network and the model is trained with MC-dropout in order to extract the labels  $y'_i$ , optimizing the cross-entropy function

$$H_y(y') = \sum_i y_i \log(y'_i)$$

where  $y_i$  is the original label of the  $i$ th sample and  $y'_i$  is the predicted label obtained by the model.

- 2) Once a certain number of epochs have been executed and the weights and biases of the model have been adjusted,  $\mathcal{D}_{\text{pool}}^0$  is sent as the test data to the network. MC-dropout is used to capture the confidence of the model in its predictions, calculating the probability of the output  $y'_i$  for each  $x_i$  in  $\mathcal{D}_{\text{pool}}^0$ ,  $p(y'_i | x_i, \mathcal{D}_{\text{train}})$ . From an implementation point of view,  $\mathcal{D}_{\text{pool}}^0$  passes through the network  $T$  times, where  $T$  is the number of stochastic forward passes. As a result,  $T$  different outputs  $y'_i$  have been obtained for each  $x_i$  in  $\mathcal{D}_{\text{pool}}^0$ . To obtain the final probability, the average between all the outputs is calculated  $y'_i = (1/T) \sum_{t=1}^T y_i^{(t)}$ , where  $y_i^{(t)}$  is the output of the model for  $x_i$  in the  $t$ th stochastic forward pass [81].
- 3) The uncertainty over the model predictions, represented by the  $T$  predicted probabilities, is used in the AL acquisition function in order to rank the unlabeled samples in  $\mathcal{D}_{\text{pool}}^0$  according to their uncertainty. Then, those samples with higher score are selected, creating the  $\mathcal{D}_{\text{selected}}^0$  set. With this process, those samples that provide more information and diversity to the network are considered to improve the final performance. To assign the

TABLE II  
NUMBER OF SAMPLES OF THE IP, SV, AND KSC HSI DATA SETS

Indian Pines (IP)			Salinas Valley (SV)			Kennedy Space Center (KSC)		
								
Color	Land-cover type	Samples	Color	Land-cover type	Samples	Color	Land-cover type	Samples
	Background	10776		Background	56975		Background	309157
	Alfalfa	46		Broccoli-green-weeds-1	2009		Scrub	761
	Corn-notill	1428		Broccoli-green-weeds-2	3726		Willow-swamp	243
	Corn-min	830		Fallow	1976		CP-hammock	256
	Corn	237		Fallow-rough-plow	1394		Slash-pine	252
	Grass/Pasture	483		Fallow-smooth	2678		Oak/Broadleaf	161
	Grass/Trees	730		Stubble	3959		Hardwood	229
	Grass/pasture-mowed	28		Celery	3579		Swap	105
	Hay-windrowed	478		Grapes-untrained	11271		Graminoid-marsh	431
	Oats	20		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	111104		Total samples	314368

corresponding labels, each  $\mathbf{x}_i \in \mathcal{D}_{\text{selected}}^0$  is paired with its corresponding label  $y_i$ . Finally, the selected pixels in  $\mathcal{D}_{\text{selected}}^0$  are inserted into the training set as patches, each one with  $1 \times n$ ,  $d \times d \times 1$ , or  $d \times d \times n$ , depending on the model's dimension, creating the next  $\mathcal{D}_{\text{train}}^1$ . Also, the selected pixels in  $\mathcal{D}_{\text{selected}}^0$  are deleted from the pool set, creating  $\mathcal{D}_{\text{pool}}^1$ .

After validating the model, the training process is repeated successively with each  $\mathcal{D}_{\text{train}}^c$  until a satisfactory result is achieved. We note that the aforementioned procedure allows us to avoid the calculation of the corresponding patch for all the pixels of the image, reducing the computation time and memory requirements of the algorithm.

### III. EXPERIMENTS AND RESULTS

#### A. Experimental Configuration

In order to evaluate the performance of our newly developed approach, we use a hardware environment composed by a

6th Generation Intel Core i7-6700K processor with 8M of Cache and up to 4.20 GHz (four cores/eight-way multitask processing), 40 GB of DDR4 RAM with a serial speed of 2400 MHz, a GPU NVIDIA GeForce GTX 1080 with 8-GB GDDR5X of video memory and 10 Gb/s of memory frequency, a Toshiba DT01ACA HDD with 7200 r/min and 2 TB of capacity, and an ASUS Z170 pro-gaming motherboard. On the other hand, the used software environment is composed by Ubuntu 16.04.4  $\times 64$  as an operating system, CUDA 8 and cuDNN 5.1.5, and Python 2.7 as programming languages.

#### B. Hyperspectral Data Sets

In our experiments, three hyperspectral data sets have been used.

- 1) The first one is the well-known Indian Pines (IP) data set (Table II). This data set was gathered by AVIRIS [4]

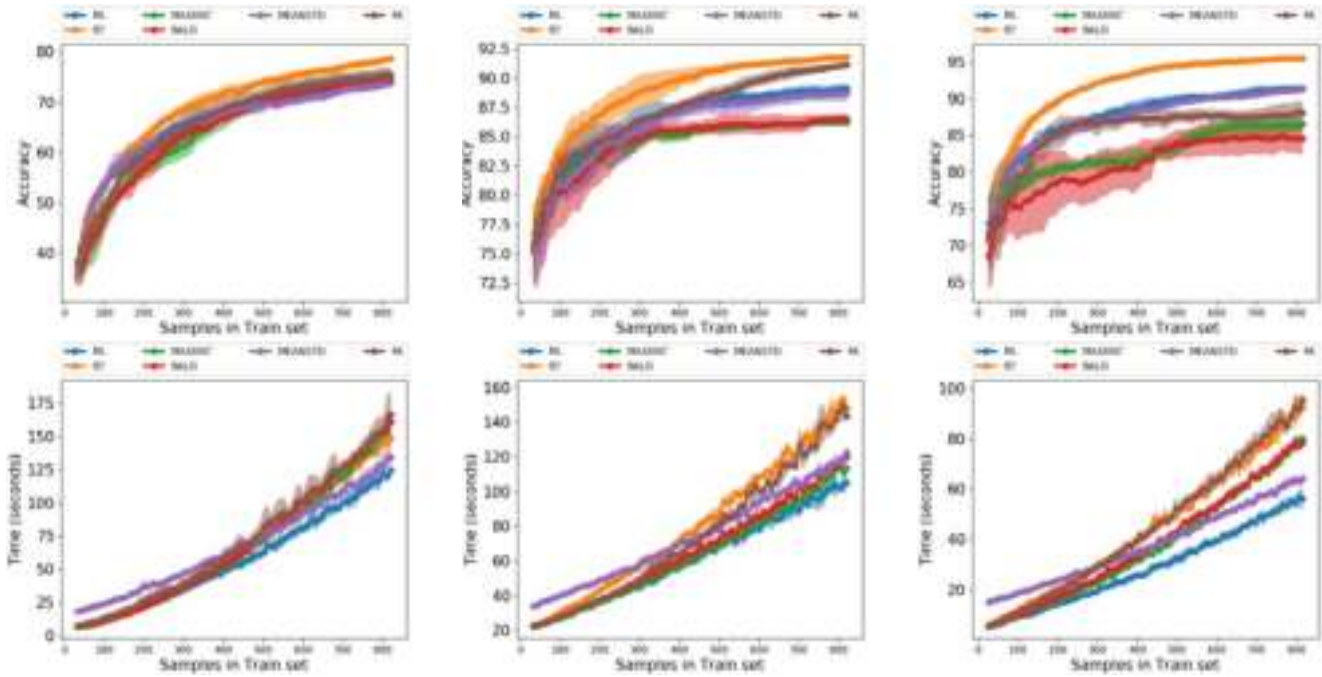


Fig. 5. AL-based performance obtained by the MLR for different acquisition functions with different sizes of  $D_{\text{train}}$ . First column: results for the IP data set. Second column: results for the SV data set. Third column: results for the KSC data set.

in 1992 over a set of agricultural fields with regular geometry and with a multiple crops and irregular patches of forest in Northwest Indiana. The IP scene has  $145 \times 145$  pixels with 224 spectral bands in the range from 400 to 2500 nm, with 10-nm spectral resolution, 20-m spatial resolution, and 16-bit radiometric resolution. After an initial analysis, 4 zero bands and another 20 bands with lower SNR because of atmospheric absorption have been removed, retaining only 200 spectral channels. Moreover, about half of the pixels in the hyperspectral image (10 249 of 21 025) contain ground-truth information, which comes in the form of a single label assignment having a total of 16 ground-truth classes.

- 2) The second hyperspectral data set used in experiments was also collected by the AVIRIS instrument, in this case over Salinas Valley (SV), California (Table II). The covered area has  $512 \times 217$  samples and the spatial resolution is 3.7 m/pixel; 204 out of the 224 bands are kept after 20 water absorption bands are removed. The ground truth is composed of 54 129 pixels and 16 land-cover classes, including vegetables, bare soils, and vineyard fields.
- 3) The third data set used in experiments is the Kennedy Space Center (KSC) (Table II), also collected by the AVIRIS instrument over Florida in 1996. Once noisy bands have been removed, the resulting image contains 176 bands with a  $512 \times 614$  size, ranging from 400 to 2500 nm, and with 20-m spatial resolution. A total of 5122 pixels labeled in 13 classes, representing different land cover types, are considered for classification purposes.

### C. Performance Evaluation

In order to test the proposed method, five different experiments have been carried out. In the first, second, third, and fourth experiments, the AL acquisition functions presented in Section II-C are tested considering an MLR classifier and the proposed 1-D, 2-D, and 3-D B-CNN models, respectively, with the aim of comparing the performance of each function over different classifiers, based on statistical or neural models, and using spectral, spatial, and spectral-spatial information. The fifth experiment makes a comparison between the AL methods (MLR: 1-D, 2-D, and 3-D B-CNN models adapted to AL) and the original ones (baseline MLR: 1-D, 2-D, and 3-D CNN models) with the aim of studying the impact of  $D_{\text{train}}$  on the performance of both AL and traditional methods.

Also, we remark that each experiment uses the three considered hyperspectral data sets, running each model (with each acquisition function) over each scene 5 times, creating batches of 100 pixels, and working with the limited-memory Broyden-Fletcher-Goldfarb-Shanno optimizer [98], [99] in the case of the MLR, with  $L_2$  as penalty and tolerance value fixed to  $1e-18$ , being 1000 the number of maximum iterations, and with the Adam optimizer [100] for the 1-D, 2-D, and 3-D B-CNN models, with a learning rate of 0.001 and 100 epochs. Spatial and spectral-spatial patches have been created using a size of  $d = 23$  for spatial patches and  $d = 19$  for spectral-spatial patches with the aim of extracting enough spatial information from neighboring pixels. We have empirically observed that the value of  $d$  should be large enough to characterize the spatial-contextual information around each pixel [64]. In this regard,  $d = 23$  and  $d = 19$  provide an appropriate compromise for the considered images (the selection of other close values of  $d$  did not have a significant





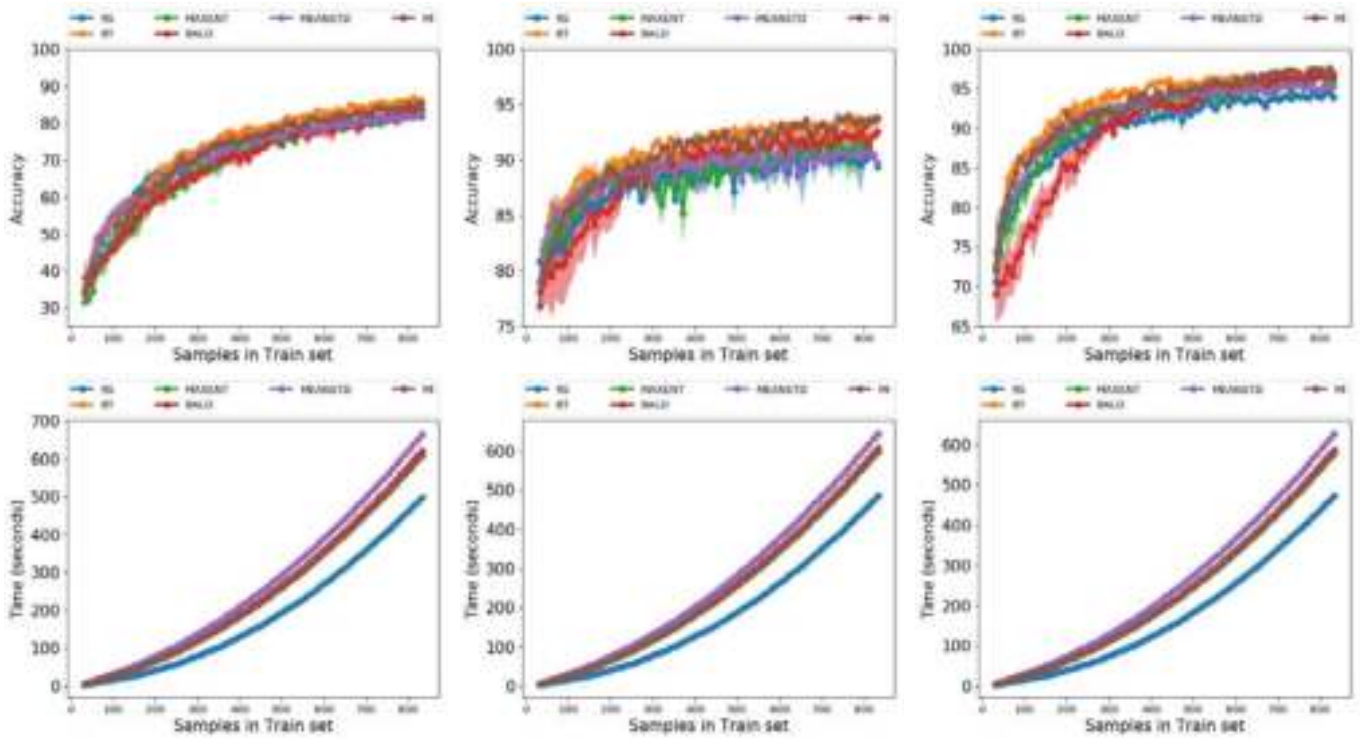


Fig. 6. AL-based performance obtained by the 1-D B-CNN for different acquisition functions with different sizes of  $D_{\text{train}}$ . First column: results for the IP data set. Second column: results for the SV data set. Third column: results for the KSC data set.

distance-based BT-criterion reaches the highest OA value, 86.14% with 8.12% of the ground truth (i.e., 832 training samples), followed by the MI function, while the random function achieves the lower results with 81.83% of accuracy. As in the AL-MLR case, BALD and max-entropy reach very similar results with BALD exhibiting better generalization performance than the other tested acquisition functions. In Fig. 6, we can observe the performance of the proposed model. In the IP plot, the acquisition functions are very close one to each other at different  $D_{\text{train}}$  sizes in terms of both OA values and execution times. In this case, the mean STD is the slowest method and the random function is the fastest one, while BT-criterion, MI, BALD, and max-entropy provide a very similar performance.

For the SV data set, the MI method reaches the best accuracy result, 97.27% with 1.54% of the ground truth (i.e., 832 training samples), demonstrating a high generalization power, which is overcome only by the BALD function. Observing Fig. 6, we can see that the BT-criterion has good OA between 100 and 300 training samples, being outperformed by MI with some variability. Again, the execution times are very similar for different functions, being mean STD the slowest one and random the fastest one, while BT-criterion, MI, BALD, and max-entropy are quite similar.

The results for the KSC are similar to those with the SV scene, with distance-based methods reaching the highest OA values, being the MI the best one: 93.57% OA with 15.85% of the ground truth. Also, the execution times are very similar with regard to those obtained for the SV data set.

With these data sets, we can also observe how 1-D B-CNN is able to scale logarithmically, rather than linearly, as in the case of the MLR. Also, we can see how the

max-entropy and BALD functions suffer when a few samples are used. In this case, the model is not able to obtain good uncertainty values for these acquisition functions due to a poor dropout. To address this issue, we can add more uncertainty and variability to the model increasing the dropout values at the CONV and fully connected layers.

3) *Experiment 3 (Performance of Different Acquisition Functions With the 2-D B-CNN)*: The third experiment performs a comparison of different acquisition function with the proposed 2-D B-CNN model, whose architecture is described in Table I. The parameter  $T$  has been set to 300, and the initialization of  $D_{\text{train}}$ ,  $D_{\text{pool}}$ , test and validation sets are the same as in our experiments with the 1-D B-CNN and MLR.

Table V shows the obtained results over the three considered hyperspectral data sets at iteration  $\epsilon = 80$ . Focusing on the IP data set, we can observe that distance-based methods are able to reach classification results over 99% accuracy, followed by entropy-based methods, with an OA around 98%. Fig. 7 shows the performance of the proposed spatial model with different sizes of  $D_{\text{train}}$ . MI and BT-criterion provide similar OA values from 600 to 800 training samples, while BALD remains close to max-entropy. On the other hand, although the random function is the fastest one, it reaches the lowest OA value, being BALD the slowest one, while BT-criterion, MI, and max-entropy exhibit similar execution times.

The SV data set provides very similar results, being the BT-criterion and MI the acquisition functions with better OA, followed quite closely by max-entropy. Again, in Fig. 7, we can observe how the BT-criterion and MI present very similar results, while max-entropy and BALD stay close one to each other, being mean STD and random the methods with the lowest OA results. The execution times are rather similar to the

TABLE VII  
CLASSIFICATION RESULTS OBTAINED BY AL-BASED PRESENTED METHODOLOGIES IN COMPARISON WITH THOSE OBTAINED WITH TRADITIONAL HYPERSPECTRAL DATA CLASSIFIERS AFTER 80 ITERATIONS AND 10 ACQUISITIONS PER ITERATION

Class	Indian Pines											
	RF	MLP	SVM	MLR	1D-CNN	2D-CNN	3D-CNN	AL-MLR	1D B-CNN	2D B-CNN	3D B-CNN	
0	25.43 (17.39)	56.52 (13.40)	57.17 (19.20)	23.48 (9.47)	55.8 (6.72)	93.48 (6.52)	97.83 (2.17)	31.74 (19.18)	71.74 (15.17)	100 (0.00)	98.91 (1.09)	
1	62.67 (4.59)	76.27 (2.04)	76.90 (2.61)	74.22 (2.45)	82.05 (1.63)	94.40 (0.63)	96.29 (1.33)	78.14 (2.12)	82.12 (4.13)	100 (0.00)	99.51 (0.21)	
2	47.75 (4.95)	62.36 (4.00)	64.41 (5.23)	52.94 (1.61)	61.77 (4.61)	92.17 (0.84)	99.82 (0.06)	61.25 (4.22)	72.97 (3.73)	99.82 (0.18)	99.76 (0.24)	
3	34.09 (11.38)	57.85 (5.36)	65.82 (11.92)	41.94 (8.88)	63.01 (8.78)	97.47 (0.69)	99.16 (0.84)	47.17 (9.11)	83.12 (3.1)	99.58 (0.42)	100 (0.00)	
4	81.04 (4.48)	83.79 (4.17)	89.01 (2.62)	74.95 (6.43)	90.82 (1.67)	90.79 (1.14)	96.17 (1.76)	85.09 (2.58)	94.13 (1.98)	98.55 (1.45)	100 (0.00)	
5	94.88 (2.75)	94.47 (1.32)	95.70 (1.14)	93.78 (1.12)	98.31 (0.74)	95.27 (0.48)	99.59 (0.41)	94.79 (1.64)	98.26 (0.26)	99.79 (0.21)	99.73 (0.27)	
6	31.07 (18.28)	72.14 (15.55)	81.07 (15.65)	60.71 (9.85)	77.38 (8.42)	96.43 (3.57)	91.07 (8.93)	65 (17.11)	82.14 (10.51)	100 (0.00)	100 (0.00)	
7	96.82 (3.23)	96.92 (1.22)	97.80 (1.40)	97.07 (1.08)	98.68 (0.71)	100 (0.00)	100 (0.00)	96.99 (1.06)	97.42 (1.73)	100 (0.00)	100 (0.00)	
8	28.00 (11.00)	66.00 (17.44)	70.00 (16.73)	41.00 (17.72)	66.67 (10.27)	100 (0.00)	85 (15)	63 (4)	86.67 (10.27)	100 (0.00)	100 (0.00)	
9	66.01 (6.94)	74.50 (2.25)	73.30 (4.96)	64.77 (4.35)	84.43 (3.88)	89.87 (1.08)	96.6 (1.34)	68.79 (4.12)	86.01 (3.99)	99.54 (0.05)	99.69 (0.31)	
10	85.56 (4.35)	81.35 (1.81)	83.00 (2.79)	73.49 (1.93)	76.78 (1.82)	97.13 (0.61)	98.9 (0.16)	79.23 (2.2)	83.54 (6.24)	99.98 (0.02)	99.98 (0.02)	
11	44.23 (7.74)	63.27 (6.48)	76.51 (5.96)	54.03 (2.28)	78.98 (4.03)	95.03 (0.76)	96.12 (0.51)	65.87 (3.26)	86.9 (3.1)	99.33 (0.67)	99.92 (0.08)	
12	92.20 (3.69)	97.56 (2.08)	96.83 (1.76)	98.54 (0.31)	98.21 (1.15)	99.76 (0.24)	99.51 (0.49)	98.73 (0.66)	94.47 (4.53)	100 (0.00)	100 (0.00)	
13	96.03 (0.92)	94.06 (1.60)	94.22 (1.86)	92.14 (1.18)	94.6 (2.28)	99.29 (0.71)	99.8 (0.04)	93.88 (1.92)	95.34 (2.97)	99.01 (0.99)	99.64 (0.36)	
14	41.50 (7.21)	63.34 (5.28)	54.27 (7.11)	65.85 (6.89)	53.89 (1.68)	98.45 (0.51)	95.73 (4.27)	66.99 (4.98)	65.72 (7.32)	100 (0.00)	100 (0.00)	
15	83.55 (3.85)	86.24 (7.48)	90.00 (3.85)	85.16 (1.85)	90.32 (2.63)	94.09 (4.84)	96.77 (0.00)	79.57 (9.4)	92.47 (2.32)	1.61 (99.46)	97.31 (1.61)	
OA	74.06 (0.70)	79.73 (0.79)	81.33 (0.55)	74.16 (0.91)	81.83 (0.78)	95.58 (0.41)	98.14 (0.00)	79.79 (0.53)	86.14 (0.47)	99.68 (0.18)	99.78 (0.07)	
AA	63.18 (1.59)	76.66 (2.03)	79.13 (2.32)	68.38 (1.64)	79.48 (0.03)	95.85 (0.19)	96.77 (1.68)	73.52 (2.12)	85.81 (0.07)	99.69 (0.17)	99.65 (0.2)	
K	70.00 (0.89)	76.80 (0.92)	78.64 (0.62)	74.16 (1.08)	81.83 (0.91)	95.58 (0.47)	98.14 (0.01)	78.79 (0.58)	86.14 (0.5)	99.68 (0.2)	98.78 (0.08)	
Class	Salinas Valley											
	RF	MLP	SVM	MLR	1D-CNN	2D-CNN	3D-CNN	AL-MLR	1D B-CNN	2D B-CNN	3D B-CNN	
0	97.71 (1.94)	98.16 (0.96)	97.59 (1.31)	98.26 (0.61)	99 (0.42)	99.85 (0.15)	99.98 (0.02)	99.52 (0.21)	99.8 (0.22)	99.55 (0.45)	100 (0)	
1	99.83 (0.07)	99.48 (0.40)	99.35 (0.45)	99.78 (0.07)	99.95 (0)	94.15 (1.45)	100 (0)	99.79 (0.09)	99.97 (0.02)	99.72 (0.28)	100 (0)	
2	93.74 (3.59)	96.89 (1.76)	96.88 (2.08)	94.94 (1.82)	97.79 (0.56)	99.62 (0.03)	100 (0)	98.93 (0.58)	99.68 (0.21)	100 (0)	100 (0)	
3	97.06 (3.00)	99.44 (0.31)	98.98 (0.61)	99.24 (0.38)	98.76 (0.96)	99.86 (0.14)	97.49 (2.22)	99.4 (0.3)	99.71 (0.1)	99.89 (0.11)	99.61 (0.18)	
4	96.25 (0.99)	97.50 (1.15)	97.87 (0.72)	97.36 (1.21)	96.98 (1.18)	99.79 (0.06)	99.07 (0.93)	99.29 (0.31)	98.51 (1.51)	100 (0)	99.91 (0.06)	
5	98.73 (0.99)	99.52 (0.22)	99.43 (0.40)	99.57 (0.18)	99.8 (0.13)	99.73 (0.21)	99.55 (0.45)	99.67 (0.22)	99.97 (0)	100 (0)	100 (0)	
6	99.09 (0.41)	99.27 (0.33)	99.44 (0.21)	99.66 (0.16)	99.68 (0.09)	99.09 (0.15)	99.9 (0.1)	99.74 (0.08)	99.97 (0)	100 (0)	100 (0)	
7	81.85 (2.60)	81.16 (5.33)	87.53 (1.78)	81.89 (3.01)	83.43 (3.15)	92.31 (1.01)	91.98 (6.45)	86.43 (0.66)	86.61 (5.95)	99.97 (0.03)	99.88 (0.12)	
8	98.90 (0.44)	99.34 (0.43)	99.39 (0.52)	99.86 (0.07)	99.26 (0.43)	99.84 (0.06)	100 (0)	99.87 (0.07)	99.96 (0.01)	100 (0)	100 (0)	
9	85.53 (1.96)	89.33 (2.19)	91.13 (1.74)	88.5 (2.12)	93.49 (2.15)	96.19 (2.81)	99.44 (0.5)	95.33 (1.14)	98.59 (0.48)	99.82 (0.18)	100 (0)	
10	88.16 (4.53)	90.02 (3.76)	93.93 (1.83)	91.95 (3.05)	94.48 (1.99)	96.82 (0.84)	100 (0)	95.3 (0.91)	98.94 (0.94)	100 (0)	100 (0)	
11	97.19 (1.37)	97.21 (2.40)	99.14 (0.56)	99.03 (0.73)	99.97 (0.05)	99.82 (0.18)	99.87 (0.03)	99.47 (0.11)	99.48 (0.26)	99.92 (0.08)	99.97 (0.03)	
12	97.79 (0.74)	97.66 (1.32)	97.39 (2.38)	94.39 (8.09)	98.25 (0.62)	98.42 (1.15)	99.51 (0.49)	98.41 (0.95)	99.49 (0.19)	100 (0)	99.95 (0.05)	
13	90.88 (3.21)	91.38 (2.33)	91.92 (3.07)	92.26 (1.34)	91.03 (1.75)	96.82 (0)	94.07 (5.84)	96.06 (0.69)	99.19 (0.23)	99.63 (0.09)	100 (0)	
14	59.21 (4.36)	64.87 (8.76)	64.20 (2.91)	60.89 (3.55)	66.41 (7.54)	84.74 (1.35)	94.5 (4.6)	65.39 (1.02)	74.95 (9.27)	99.64 (0.3)	99.66 (0.34)	
15	92.92 (2.26)	96.36 (1.20)	96.70 (1.84)	95.29 (2.48)	98.34 (0.57)	85.78 (0.39)	99.11 (0.89)	98.48 (0.58)	99.56 (0.24)	99.81 (0.19)	100 (0)	
OA	88.22 (0.29)	89.57 (0.41)	91.07 (0.37)	89.2 (0.3)	90.85 (0.77)	94.95 (0.07)	97.25 (0.9)	91.8 (0.08)	93.57 (0.29)	99.88 (0.08)	99.91 (0.08)	
AA	92.18 (0.28)	93.60 (0.56)	94.43 (0.38)	93.3 (0.59)	94.79 (0.64)	96.43 (0.23)	98.4 (0.66)	95.69 (0.11)	97.15 (0.33)	99.87 (0.07)	99.94 (0.05)	
K	86.86 (0.33)	88.38 (0.47)	90.03 (0.41)	89.2 (0.33)	90.85 (0.87)	94.95 (0.08)	97.25 (0.99)	91.8 (0.09)	93.57 (0.32)	99.88 (0.09)	99.91 (0.09)	
Class	Kennedy Space Center											
	RF	MLP	SVM	MLR	1D-CNN	2D-CNN	3D-CNN	AL-MLR	1D B-CNN	2D B-CNN	3D B-CNN	
0	94.95 (1.39)	96.35 (0.79)	95.32 (1.44)	95.9 (0.87)	97.33 (0.16)	95.93 (0.53)	98.49 (0.72)	97.98 (0.68)	98.69 (0.77)	99.8 (0.2)	100 (0.00)	
1	87.94 (1.68)	89.63 (4.04)	94.49 (3.20)	88.81 (1.75)	93.42 (1.16)	87.65 (0.82)	100 (0.00)	89.55 (2.04)	95.61 (3.03)	99.38 (0.62)	100 (0.00)	
2	89.49 (2.29)	91.52 (2.46)	91.88 (1.47)	87.97 (4.75)	86.85 (8.15)	86.72 (0.00)	94.53 (2.34)	93.2 (0.72)	91.67 (7.13)	99.8 (0.2)	100 (0.00)	
3	75.60 (2.71)	75.32 (6.34)	78.25 (4.55)	67.7 (11.4)	83.86 (9.71)	89.29 (0.4)	95.63 (3.97)	88.41 (1.53)	92.2 (1.9)	99.4 (0.6)	100 (0.00)	
4	59.25 (6.92)	66.58 (7.63)	75.03 (4.73)	62.11 (8.24)	70.6 (6.92)	97.83 (2.17)	99.69 (0.31)	77.76 (3.88)	90.89 (1.28)	100 (0.00)	100 (0.00)	
5	58.12 (6.80)	69.74 (4.59)	80.39 (5.88)	71.35 (4.48)	83.11 (2.09)	94.1 (3.28)	96.94 (0.87)	83.32 (4)	89.96 (5.57)	100 (0.00)	100 (0.00)	
6	85.90 (4.74)	87.81 (5.32)	88.19 (5.04)	84.19 (5.51)	92.38 (6.07)	78.57 (0.48)	100 (0.00)	95.24 (0.85)	87.3 (11.33)	99.5 (0.00)	100 (0.00)	
7	87.24 (2.12)	93.76 (1.81)	94.99 (3.25)	90.35 (1.26)	95.13 (1)	89.91 (1.04)	99.65 (0.35)	97.54 (0.78)	98.14 (2.3)	99.54 (0.00)	100 (0.00)	
8	93.65 (3.03)	97.58 (0.89)	97.58 (0.93)	96.81 (0.63)	98.65 (0.16)	96.54 (0.38)	99.81 (0.19)	98.38 (0.58)	99.62 (0.42)	100 (0.00)	100 (0.00)	
9	89.60 (2.53)	97.45 (1.72)	98.42 (0.72)	94.9 (2.79)	97.69 (1.69)	96.91 (1.86)	98.64 (1.36)	97.23 (0.79)	99.17 (0.51)	99.88 (0.12)	100 (0.00)	
10	97.42 (0.97)	98.07 (1.23)	98.00 (0.99)	96.95 (0.71)	98.65 (0.92)	98.21 (0.12)	100 (0.00)	98.23 (0.63)	99.05 (0.78)	100 (0.00)	100 (0.00)	
11	90.97 (1.98)	94.63 (1.16)	95.84 (1.40)	92.41 (1.19)	96.69 (1.53)	94.73 (3.28)	100 (0.00)	94.99 (0.66)	97.55 (3.47)	99.62 (0.2)	100 (0.00)	
12	99.69 (0.13)	100.00 (0.00)	100.00 (0.00)	100 (0.00)	99.5 (0.64)	99.68 (0.32)	99.95 (0.05)	99.74 (0.13)	100.00 (0.00)	100 (0.00)	100 (0.00)	
OA	89.99 (0.28)	93.14 (0.49)	94.40 (0.50)	91.49 (0.43)	94.84 (0.21)	94.77 (0.47)	98.99 (0.47)	95.56 (0.28)	97.27 (0.49)	99.8 (0.01)	100 (0.00)	
AA	85.37 (0.72)	89.11 (0.74)	91.41 (0.92)	86.88 (0.6)	91.83 (0.29)	92.77 (0.41)	98.72 (0.55)	93.2 (0.3)	95.37 (1.06)	99.73 (0.06)	100 (0.00)	
K	88.85 (0.31)	92.35 (0.55)	93.76 (0.56)	91.49 (0.47)	94.84 (0.24)	94.77 (0.52)	98.99 (0.52)	95.56 (0.31)	97.27 (0.55)	99.8 (0.01)	100 (0.00)	

IP data set, being the processing of SV slowest than IP. Again, BALD is the slowest method and random is the the fastest one, being BT-criterion, max-entropy, and MI very similar.

Finally, for the KSC data set, the best OA is reached with the BT-criterion as an acquisition function with distance-based methods and entropy-based functions providing the highest overall values. Looking at Fig. 7, we can see in this case how BALD performs even worst than the random function until 600 training samples are reached. This is because the network has not achieved a sufficiently adjusted accuracy in the training phase, resulting from the great variability in the dropout. On the other hand, the execution times are very similar to those achieved in the experiments with the SV data set.

4) *Experiment 4 (Performance of Different Acquisition Functions With the 3-D B-CNN)*: Our fourth experiment

implements the proposed spectral-spatial BCNN classifier using the six considered acquisition functions. Table VI and Fig. 8 show the obtained results.

With the IP data set, the best OA is reached by the BALD acquisition function, while max-entropy exhibits the best generalization power. Entropy-based methods are closely followed by distance-based methods, being random and mean STD the functions with the lowest OA. In Fig. 8, we can observe how the BT-criterion stands out with 100 training samples, achieving good results with a very few samples, while BALD stands out with 600 training samples. However, BALD is the slowest method, being the random function the fastest one, while max-entropy, MI, and BT-criterion exhibit similar execution times.

Focusing on SV, the best OA is reached by max-entropy, followed by MI and BALD. As we can see in Fig. 8,

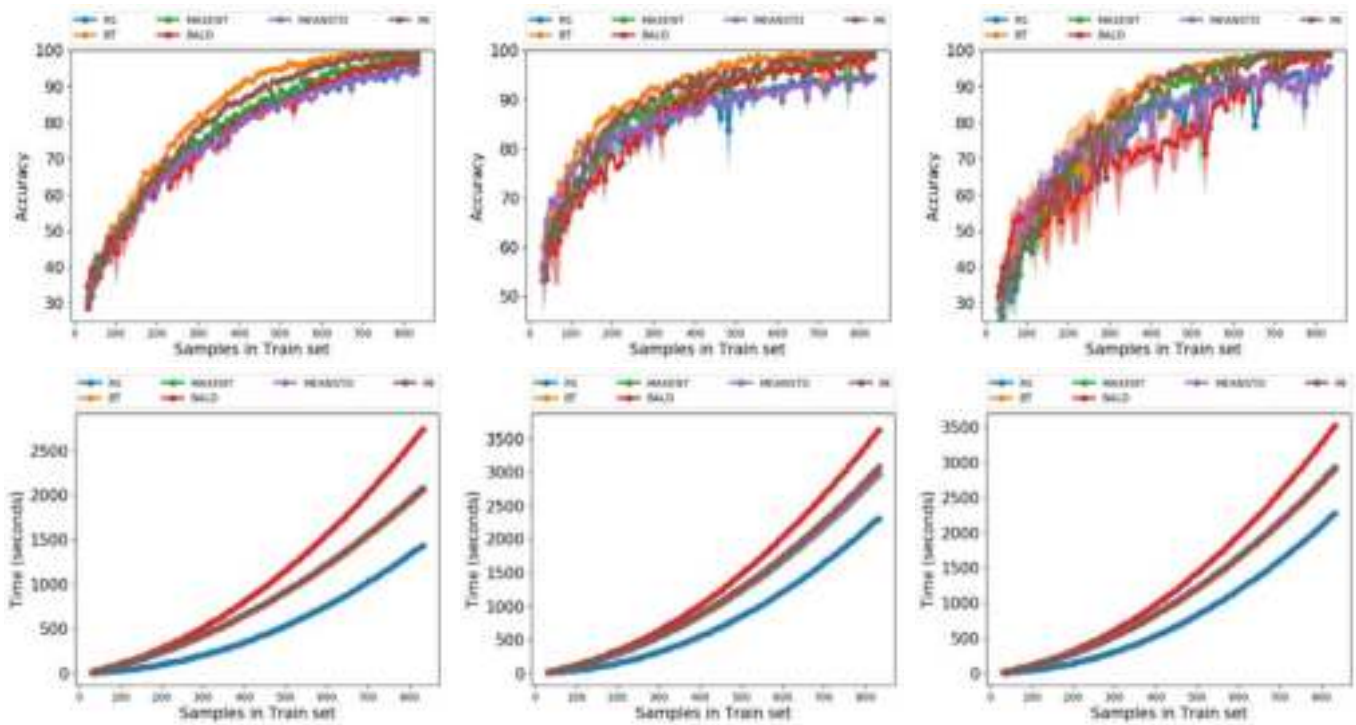


Fig. 7. AL-based performance obtained by the 2-D B-CNN for different acquisition functions with different sizes of  $D_{\text{train}}$ . First column: results for the IP data set. Second column: results for the SV data set. Third column: results for the KSC data set.

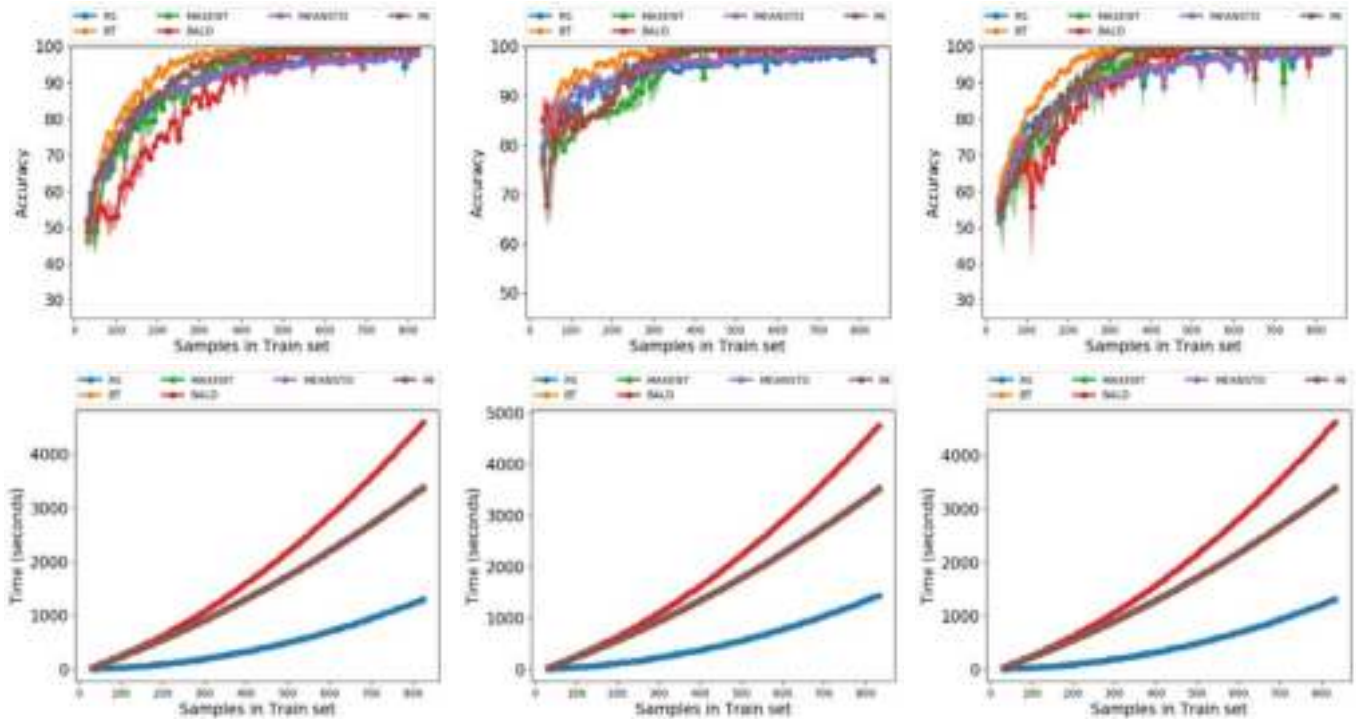


Fig. 8. AL-based performance obtained by the 3-D B-CNN for different acquisition functions with different sizes of  $D_{\text{train}}$ . First column: results for the IP data set. Second column: results for the SV data set. Third column: results for the KSC data set.

BALD stands out with 100 training samples, until it is reached by MI and max-entropy. Again, BALD is the slowest method and random and random the fastest, while MI, max-entropy, and mean STD share the same computation time.

In the case of the KSC scene, all acquisition functions provide excellent classification performance, being random and mean STD the functions with the lowest OA (98.99% and 98.56%, respectively). In Fig. 8, we can



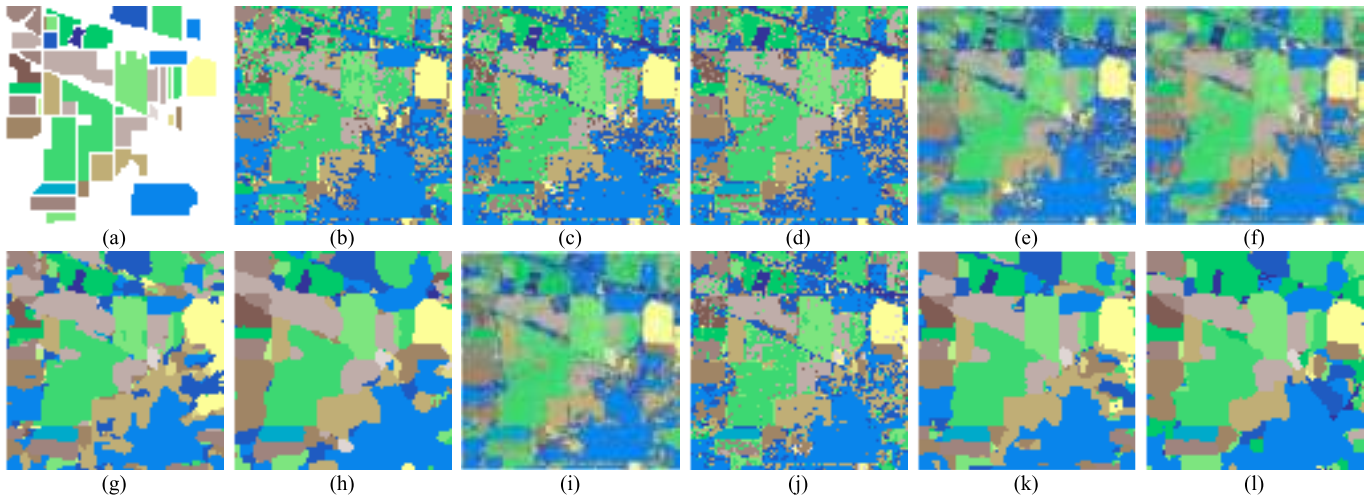


Fig. 9. Classification maps for the IP data set. (a) Ground-truth classification map. (b) RF (74.06%). (c) MLP (79.73%). (d) SVM (81.33%). (e) MLR (74.16%). (f) 1-D CNN (81.83%). (g) 2-D CNN (95.58%). (h) 3-D CNN (98.14%). (i) AL-MLR (79.79%). (j) 1-D B-CNN (86.14%). (k) 2-D B-CNN (99.68%). (l) 3-D B-CNN (99.78%). (b)–(l) Classification maps corresponding to Table VII. Note that the overall classification accuracies are shown in brackets.

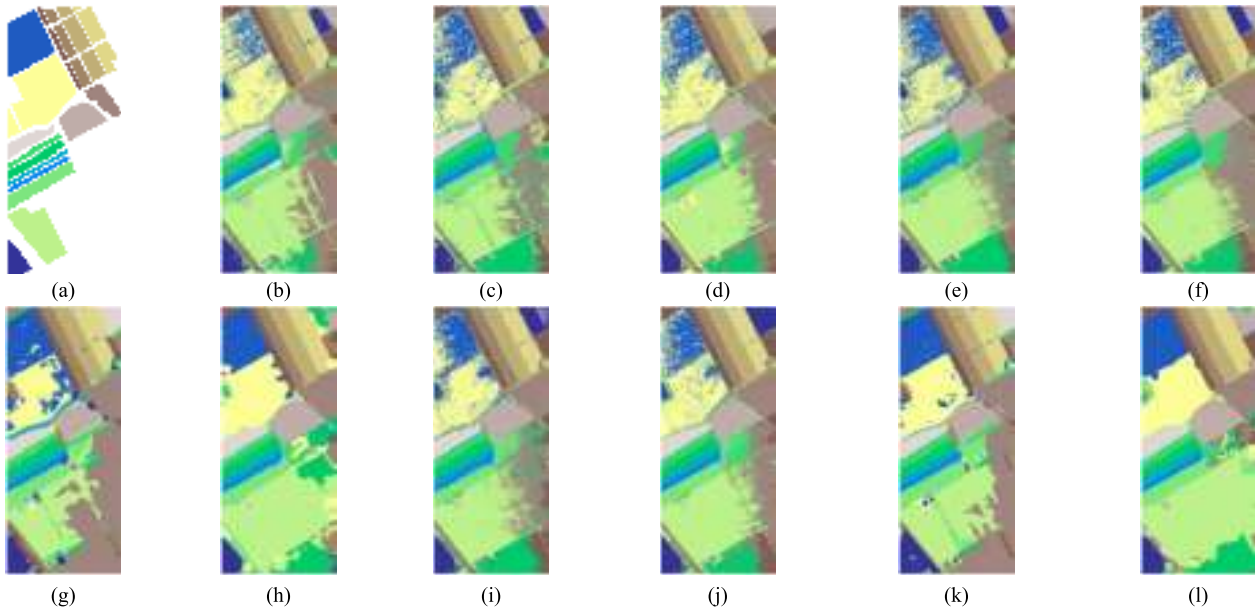


Fig. 10. Classification maps for the SV data set. (a) Ground-truth classification map. (b) RF (88.22%). (c) MLP (89.57%). (d) SVM (91.07%). (e) MLR (89.2%). (f) 1-D CNN (90.85%). (g) 2-D CNN (94.95%). (h) 3-D CNN (97.25%). (i) AL-MLR (91.8%). (j) 1-D B-CNN (93.57%). (k) B-2-D CNN (99.88%). (l) 3-D B-CNN (99.91%). (b)–(l) Classification maps corresponding to Table VII. Note that the overall classification accuracies are shown in brackets.

observe that the BT-criterion is able to reach high OA with a few training samples. Also, the BT-criterion, max-entropy, and MI exhibit similar execution times, being BALD and random the slowest and fastest acquisition functions, respectively.

5) *Experiment 5 (Comparison With Other Traditional Classifiers)*: The fifth and final experiments perform a comparison between the AL implementations described in Sections III-C1–III-C4, with the best OA values for each hyperspectral data set, with traditional classifiers. For the IP data set, the AL-MLR and the 1-D B-CNN with the BT-criterion, 2-D B-CNN with the MI criterion, and 3-D

B-CNN with the BALD criterion have been selected to be compared with the traditional RF, MLP, SVM, and MLR classifiers and also with the standard 1-D CNN, 2-D CNN, and 3-D CNN baselines, which have been implemented with the same parameters and architectures than the proposed AL approaches. In order to train the RF, MLP, SVM, and MLR, and 1-D CNN, 2-D CNN, and 3-D CNN baselines, only two pixels per class have been selected, while the remaining 800 pixels (the selected maximum size of  $\mathcal{D}_{\text{train}}$ ) have been randomly selected. This process has been repeated with the SV data set, selecting the AL-MLR and 2-D B-CNN with BT-criterion, 1-D B-CNN with MI, and 3-D B-CNN with max-

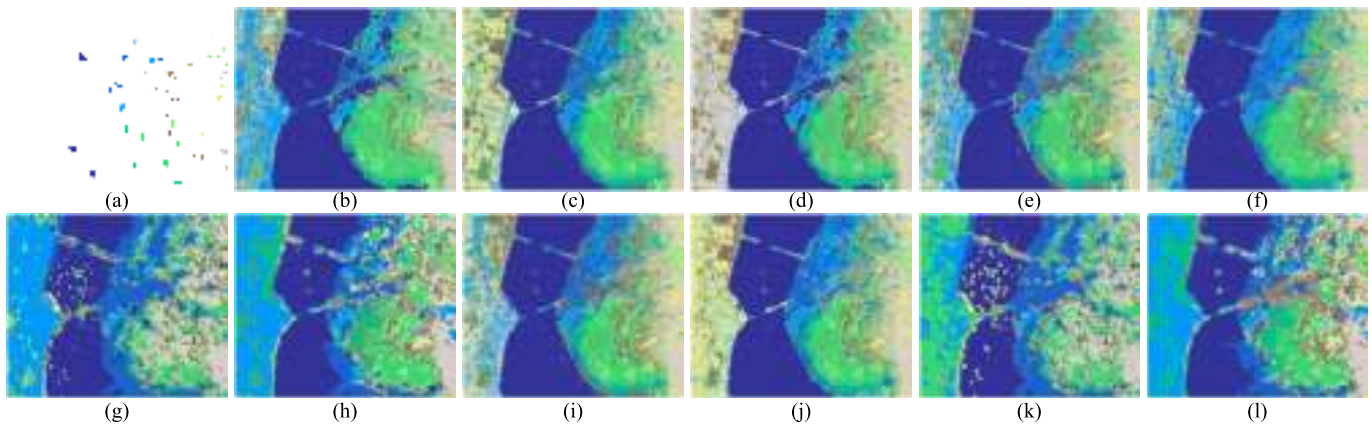


Fig. 11. Classification maps for the KSC data set. The first image (a) represents the ground-truth classification map. (b) RF (89.99%). (c) MLP (93.14%). (d) SVM (94.40%). (e) MLR (91.49%). (f) 1-D CNN (94.84%). (g) 2-D CNN (94.77%). (h) 3-D CNN (98.99%). (i) AL-MLR (95.56%). (j) 1-D B-CNN (97.27%). (k) 2-D B-CNN (99.8%). (l) 3-D B-CNN (100.00%). (b)–(l) Classification maps corresponding to Table VII. Note that the overall classification accuracies are shown in brackets.

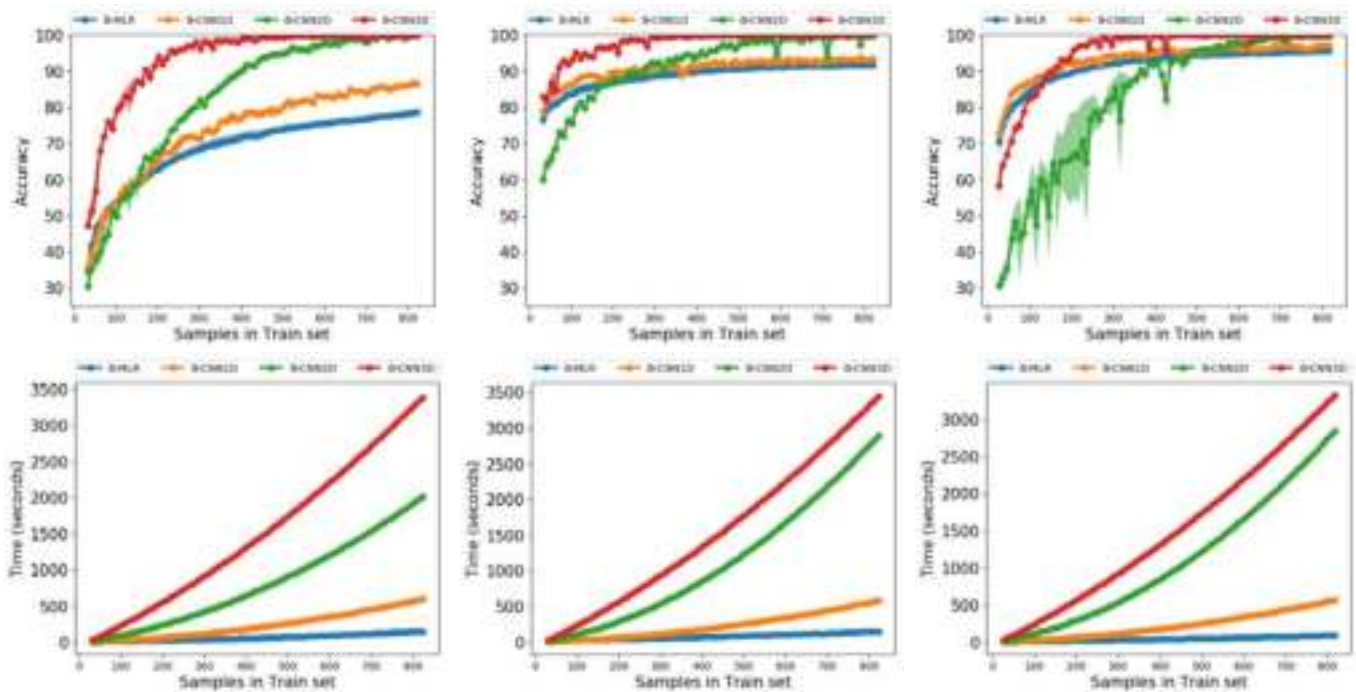


Fig. 12. Comparison of the BT-criterion over spectral AL MLR and spectral, spatial, and spectral-spatial B-CNNs with the IP (first column), SV (second column), and KSC (third column) hyperspectral data sets.

entropy, and with the KSC data set, selecting AL-MLR, 2-D and 3-D B-CNN with BT-criterion and 1-D B-CNN with MI.

Focusing on the IP data set, we can observe the performance of pixelwise classifiers, being RF and the baseline MLR the classifiers that provide lower OA. We can observe how AL-MLR is better than the baseline MLR, but worse than SVM, while the spectral B-CNN model improves the classification results over the baseline 1-D CNN and the other pixelwise methods. Looking at spatial classifiers, the 2-D B-CNN is able to outperform the 2-D CNN results in 6.11 percentage points, improving also the generalization power. We can also observe this behavior with spectral-spatial classifiers, where the proposed B-CNN model outperforms the

3-D CNN baseline. Moreover, we can observe that, after adding spectral-spatial information, the classifier is able to improve its accuracy results. Fig. 9 presents these results in a graphical form, showing the classification maps obtained for each classifier. In Fig. 12, we can observe the performance of AL-MLR, 1-D B-CNN, 2-D B-CNN, and 3-D B-CNN with BT-criterion for the IP data set. We can see that the spectral-spatial B-CNN is able to reach a good classification accuracy with fewer training samples than the other AL-based classifiers, although it is the slowest method. Moreover, in Table VIII, we can observe the number of training samples that each classifier needs to reach the accuracy percentage, being spectral-spatial B-CNN the one that needs

TABLE VIII  
NUMBER OF SAMPLES THAT EACH MODEL IN FIG. 12 NEEDS  
TO REACH A GIVEN % OF ACCURACY

Algorithm	Indian Pines						
	Accuracy						
	70%	75%	80%	85%	90%	95%	99%
AL-MLR	342	522	—	—	—	—	—
1D B-CNN	252	352	502	662	—	—	—
2D B-CNN	222	252	292	352	402	512	662
3D B-CNN	72	82	112	152	172	232	402
Algorithm	Salinas Valley						
	Accuracy						
	70%	75%	80%	85%	90%	95%	99%
MLR	32	32	52	132	412	—	—
CNN1D	32	32	42	62	232	—	—
CNN2D	72	92	122	162	272	412	622
CNN3D	32	32	32	52	72	112	292
Algorithm	Kennedy Space Center						
	Accuracy						
	70%	75%	80%	85%	90%	95%	99%
AL-MLR	26	36	66	116	216	616	—
1D B-CNN	26	36	56	76	156	386	—
2D B-CNN	226	246	286	306	366	496	666
3D B-CNN	56	86	96	126	166	206	276

less training data, only 402 samples (i.e., the 3.92% of the ground truth) to reach 99% OA.

The results for the SV data set are similar. In Table VII, we can see that the pixelwise classifiers based on AL outperform their baseline methods, being the AL-MLR better than MLR, while the spectral B-CNN is also better than the 1-D CNN baseline. Also, the spatial B-CNN outperforms the 2-D CNN baseline, being around 4.93 perceptual points better. Finally, the spectral-spatial B-CNN classifier is much better than the 3-D CNN baseline with 2.66 perceptual points better. Fig. 10 shows the classification maps obtained by each classifier. Also, in Fig. 12, we can observe how B-CNNs are able to outperform the results of AL-MLR, standing out 150 training samples in the case of the 2-D B-CNN. In Table VIII, we can see that the spectral-spatial B-CNN needs less training data than the other classifier in order to reach 99% of accuracy.

The results obtained for the KSC data set are also quite similar to those obtained for the SV data set. In Table VII, we can see that the pixelwise classifiers based on AL outperform their respective baseline methods, as well as the RF, SVM, and MLP methods. Also, the spatial B-CNN model obtains better results than the baseline 2-D CNN, while the spectral-spatial B-CNN also outperforms the 3-D CNN baseline. These classification results can be observed in the graphical form in Fig. 11. In Fig. 12, we can observe in the third column the implemented AL-based methods with the BT-criterion as an acquisition function over the KSC data set. As we can see, the spectral-spatial B-CNN is able to reach good values with a few training samples, and in fact, this model can reach 99% accuracy with only 276 samples (i.e., 5.30% of the KSC's ground truth), as shown in Table VIII.

#### IV. CONCLUSION

In this paper, we have developed a new AL model with B-CNNs for hyperspectral image classification using spectral,

spatial, and spectral-spatial features. The proposed approach offers robustness to overfitting on small labeled sets and improves the generalization capacity by including intelligently selected unlabeled training samples, integrating the spatial and the spectral information contained in the original hyperspectral image. To the best of our knowledge, this is the first time in the literature that AL is combined with CNNs (via BNNs) to perform robust hyperspectral image classification with very limited training sets. In this paper, we report very high classification accuracies using very limited labeled samples, avoiding the curse of dimensionality and the overfitting problems introduced by these kinds of networks. Our results also indicate that, by the proper selection of the acquisition function, AL offers a very good solution to avoid the aforementioned problems of overfitting with supervised deep networks. Future work will focus on improving the results obtained from the viewpoint of computational complexity, drawing additional comparisons with other established methods for spatial-spectral classification of remotely sensed hyperspectral and also validating the proposed techniques using multispectral data [101]. Finally, the inclusion of postprocessing methods, such as conditional random field [102], will also be studied in the future developments as a way to improve and smooth the classification maps obtained by the different tested methods.

#### ACKNOWLEDGMENT

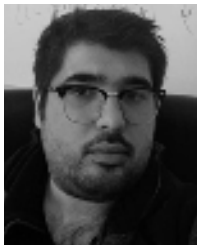
The authors would like to thank the editors and reviewers for their outstanding comments and suggestions, which greatly helped us to improve the technical quality and presentation of this paper.

#### REFERENCES

- [1] A. F. H. Goetz, G. Vane, J. E. Solomon, and B. N. Rock, "Imaging spectrometry for earth remote sensing," *Science*, vol. 228, no. 4704, pp. 1147–1153, 1985.
- [2] R. Lucas, A. Rowlands, O. Niemann, and R. Merton, *Advanced Image Processing Techniques for Remotely Sensed Hyperspectral Data*. Berlin, Germany: Springer, 2004.
- [3] G. Vane, D. L. Evans, and A. B. Kahle, "Recent advances in airborne terrestrial remote sensing with the NASA airborne visible/infrared imaging spectrometer (AVIRIS), airborne synthetic aperture radar (SAR), and thermal infrared multispectral scanner (TIMS)," in *Proc. 12th Can. Symp. Remote Sens. Geosci. Remote Sens. Symp.*, Jul. 1989, pp. 942–943.
- [4] 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. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0034425798000649>
- [5] X. She, L. Zhang, C. Huang, and S. Wang, "Comparison of hyperspectral vegetation indices based on CASI airborne data," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2016, pp. 4532–4534.
- [6] J. Chen *et al.*, "Compact Airborne Spectrographic Imager (CASI) used for mapping biophysical parameters of boreal forests," *J. Geophys. Res.*, vol. 104, no. D22, pp. 27945–27958, Nov. 1999.
- [7] 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. 868, pp. 134–142, Apr. 1988.
- [8] E. Bedini, F. van der Meer, and F. van Ruitenbeek, "Use of HyMap imaging spectrometer data to map mineralogy in the Rodalquilar caldera, southeast Spain," *Int. J. Remote Sens.*, vol. 30, no. 2, pp. 327–348, 2009.
- [9] E. Puckrin, C. S. Turcotte, M.-A. Gagnon, J. Bastedo, V. Farley, and M. Chamberland, "Airborne infrared hyperspectral imager for intelligence, surveillance, and reconnaissance applications," *Proc. SPIE*, vol. 8360, pp. 836004-1–836004-10, May 2012.

- [10] I. Vorovencii, "The hyperspectral sensors used in satellite and aerial remote sensing," *Bull. Transilvania Univ. Braşov*, vol. 2, p. 51, Jan. 2009.
- [11] R. C. Olsen, *Remote Sensing From Air and Space*. Bellingham, WA, USA: SPIE, 2007.
- [12] S. Yarbrough *et al.*, "MightySat II.1 hyperspectral imager: Summary of on-orbit performance," *Proc. SPIE, Imag. Spectrometry VII*, vol. 4480, Jan. 2002, doi: 10.1117/12.453339.
- [13] C.-I. Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. New York, NY, USA: Springer, 2003.
- [14] 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.
- [15] M. Teke, H. S. Deveci, O. Halilöglü, S. Z. Gürbüüz, and U. Sakarya, "A short survey of hyperspectral remote sensing applications in agriculture," in *Proc. Recent Adv. Space Technol. (RAST)*, Jun. 2013, pp. 171–176.
- [16] A. Plaza, J. Plaza, A. Paz, and S. Sánchez, "Parallel hyperspectral image and signal processing [applications corner]," *IEEE Signal Process. Mag.*, vol. 28, no. 3, pp. 119–126, May 2011.
- [17] X. Lu, X. Li, and L. Mou, "Semi-supervised multitask learning for scene recognition," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1967–1976, Sep. 2015.
- [18] A. B. Pour and M. Hashim, "ASTER, ALI and Hyperion sensors data for lithological mapping and ore minerals exploration," *SpringerPlus*, vol. 3, no. 1, p. 130, 2014.
- [19] M. J. Abrams and S. J. Hook, *NASA's Hyperspectral Infrared Imager (HyspIRI)*. Dordrecht, The Netherlands: Springer, 2013, pp. 117–130.
- [20] H. Kaufmann *et al.*, "Environmental Mapping and Analysis Program (EnMAP)—Recent advances and status," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, vol. 4, Jul. 2008, pp. 109–112.
- [21] C. Galeazzi, A. Sacchetti, A. Cisbani, and G. Babini, "The PRISMA program," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2008, pp. IV-105–IV-108.
- [22] A. Plaza *et al.*, "Recent advances in techniques for hyperspectral image processing," *Remote Sens. Environ.*, vol. 113, pp. S110–S122, Sep. 2009. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0034425709000807>
- [23] D. Chutia, D. K. Bhattacharyya, K. K. Sarma, R. Kalita, and S. Sudhakar, "Hyperspectral remote sensing classifications: A perspective survey," *Trans. GIS*, vol. 20, no. 4, pp. 463–490, 2016.
- [24] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.
- [25] J. M. 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, Jan. 2017.
- [26] Y. Tarabalka, J. A. Benediktsson, and J. Chanussot, "Spectral-spatial classification of hyperspectral imagery based on partitioning clustering techniques," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 8, pp. 2973–2987, Aug. 2009. [Online]. Available: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4840429>
- [27] G. H. Ball and D. J. Hall, *ISODATA: A Novel Method of Data Analysis and Pattern Classification*. Menlo Park, CA, USA: Stanford Research Institute, 1965.
- [28] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "Yinyang K-means clustering for hyperspectral image analysis," in *Proc. 17th Int. Conf. Comput. Math. Methods Sci. Eng.*, J. Vigo-Aguiar, Ed. Cadiz, Spain: Rota, 2017, pp. 1625–1636.
- [29] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Clarendon, 1995. [Online]. Available: [https://books.google.es/books?id=-aAwQO\\_rXwC](https://books.google.es/books?id=-aAwQO_rXwC)
- [30] P. M. Atkinson and A. R. L. Tatnall, "Introduction neural networks in remote sensing," *Int. J. Remote Sens.*, vol. 18, no. 4, p. 699, 1997, doi: 10.1080/014311697218700.
- [31] J. A. Benediktsson, P. H. Swain, and O. K. Ersoy, "Conjugate-gradient neural networks in classification of multisource and very-high-dimensional remote sensing data," *Int. J. Remote Sens.*, vol. 14, no. 15, pp. 2883–2903, 1993. [Online]. Available: <http://www.tandfonline.com/doi/abs/10.1080/01431169308904316>
- [32] H. Yang, "A back-propagation neural network for mineralogical mapping from AVIRIS data," *Int. J. Remote Sens.*, vol. 20, no. 1, pp. 97–110, 1999, doi: 10.1080/014311699213622.
- [33] J. A. Benediktsson and P. H. Swain, "Statistical methods and neural network approaches for classification of data from multiple sources," Ph.D. dissertation, School Elect. Eng., Purdue Univ., West Lafayette, IN, USA, 1990.
- [34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
- [35] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436–444, May 2015.
- [36] L. Deng and D. Yu, "Deep learning: Methods and applications," *Found. Trends Signal Process.*, vol. 7, nos. 3–4, pp. 197–387, Jun. 2014.
- [37] 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. [Online]. Available: <http://ieeexplore.ieee.org/document/7514991/>
- [38] K. Makantasis, K. Karantzas, A. Doulamis, and N. Doulamis, "Deep supervised learning for hyperspectral data classification through convolutional neural networks," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2015, pp. 4959–4962.
- [39] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.
- [40] Y. Feng, Y. Yuan, and X. Lu, "Learning deep event models for crowd anomaly detection," *Neurocomputing*, vol. 219, pp. 548–556, Jan. 2017.
- [41] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.
- [42] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [43] Y. Chen, X. Zhao, and X. Jia, "Spectral-spatial classification of hyperspectral data based on deep belief network," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 6, pp. 2381–2392, Jun. 2015.
- [44] T. Li, J. Zhang, and Y. Zhang, "Classification of hyperspectral image based on deep belief networks," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2014, pp. 5132–5136.
- [45] J. H. Le, A. P. Yazdanpanah, E. E. Regentova, and V. Muthukumar, "A deep belief network for classifying remotely-sensed hyperspectral data," in *Advances in Visual Computing*, G. Bebis *et al.*, Eds. Cham, Switzerland: Springer, 2015, pp. 682–692.
- [46] P. Zhong, Z. Gong, S. Li, and C.-B. Schönlieb, "Learning to diversify deep belief networks for hyperspectral image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 55, no. 6, pp. 3516–3530, Jun. 2017.
- [47] A. Okan, B. Özdemir, B. E. Gedik, C. Yasemin, and Y. Çetin, "Hyperspectral classification using stacked autoencoders with deep learning," in *Proc. 6th Workshop Hyperspectral Image Signal Process., Evol. Remote Sens. (WHISPERS)*, 2014, pp. 1–4.
- [48] J. Li, L. Bruzzone, and S. Liu, "Deep feature representation for hyperspectral image classification," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2015, pp. 4951–4954.
- [49] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2094–2107, Jun. 2014.
- [50] C. Tao, H. Pan, Y. Li, and Z. Zou, "Unsupervised spectral-spatial feature learning with stacked sparse autoencoder for hyperspectral imagery classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 12, pp. 2438–2442, Dec. 2015.
- [51] X. Chen, S. Xiang, C.-L. Liu, and C.-H. Pan, "Vehicle detection in satellite images by hybrid deep convolutional neural networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 10, pp. 1797–1801, Oct. 2014.
- [52] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep recurrent neural networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3639–3655, Jul. 2017.
- [53] Z. Zuo *et al.*, "Convolutional recurrent neural networks: Learning spatial dependencies for image representation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2015, pp. 18–26.
- [54] H. Wu and S. Prasad, "Convolutional recurrent neural networks for hyperspectral data classification," *Remote Sens.*, vol. 9, no. 3, p. 298, 2017.

- [55] H. Wu and S. Prasad, "Semi-supervised deep learning using pseudo labels for hyperspectral image classification," *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1259–1270, Mar. 2018.
- [56] 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.
- [57] S. Yu, S. Jia, and C. Xu, "Convolutional neural networks for hyperspectral image classification," *Neurocomputing*, vol. 219, pp. 88–98, Jan. 2017. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S0925231216310104>
- [58] Z. Zheng *et al.*, "Classification based on deep convolutional neural networks with hyperspectral image," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2017, pp. 1828–1831.
- [59] H. Liang and Q. Li, "Hyperspectral imagery classification using sparse representations of convolutional neural network features," *Remote Sens.*, vol. 8, no. 2, p. 99, 2016.
- [60] J. Yang, Y. Zhao, J. C.-W. Chan, and C. Yi, "Hyperspectral image classification using two-channel deep convolutional neural network," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2016, pp. 5079–5082.
- [61] H. Zhang, Y. Li, Y. Zhang, and Q. Shen, "Spectral-spatial classification of hyperspectral imagery using a dual-channel convolutional neural network," *Remote Sens. Lett.*, vol. 8, no. 5, pp. 438–447, 2017.
- [62] 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.
- [63] Y. Li, H. Zhang, and Q. Shen, "Spectral-spatial classification of hyperspectral imagery with 3D convolutional neural network," *Remote Sens.*, vol. 9, no. 1, p. 67, Jan. 2017.
- [64] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "A new deep convolutional neural network for fast hyperspectral image classification," *ISPRS J. Photogramm. Remote Sens.*, to be published, doi: [10.1016/j.isprsjprs.2017.11.021](https://doi.org/10.1016/j.isprsjprs.2017.11.021).
- [65] L. Breiman, "Heuristics of instability and stabilization in model selection," *Ann. Statist.*, vol. 24, no. 6, pp. 2350–2383, 1996.
- [66] G. M. Foody and A. Mathur, "The use of small training sets containing mixed pixels for accurate hard image classification: Training on mixed spectral responses for classification by a SVM," *Remote Sens. Environ.*, vol. 103, no. 2, pp. 179–189, Jul. 2006.
- [67] M. Khodadadzadeh, J. Li, A. Plaza, H. Ghassemian, J. M. Bioucas-Dias, and X. Li, "Spectral-spatial classification of hyperspectral data using local and global probabilities for mixed pixel characterization," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 10, pp. 6298–6314, Oct. 2014.
- [68] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE Trans. Inf. Theory*, vol. TIT-14, no. 1, pp. 55–63, Jan. 1968.
- [69] C.-I. Chang, Eds., *Hyperspectral Data Exploitation: Theory and Applications*. Hoboken, NJ, USA: Wiley, 2007.
- [70] D. J. C. MacKay, "Information-based objective functions for active data selection," *Neural Comput.*, vol. 4, no. 4, pp. 590–604, Jul. 1992.
- [71] D. Tuia, F. Ratle, F. Pacifici, M. F. Kanevski, and W. J. Emery, "Active learning methods for remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 7, pp. 2218–2232, Jul. 2009.
- [72] S. Rajan, J. Ghosh, and M. M. Crawford, "An active learning approach to hyperspectral data classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 4, pp. 1231–1242, Apr. 2008.
- [73] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Semisupervised hyperspectral image segmentation using multinomial logistic regression with active learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 11, pp. 4085–4098, Nov. 2010.
- [74] D. Tuia, M. Volpi, L. Copa, M. Kanevski, and J. Munoz-Mari, "A survey of active learning algorithms for supervised remote sensing image classification," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 3, pp. 606–617, Jun. 2011.
- [75] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Spectral-spatial classification of hyperspectral data using loopy belief propagation and active learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 2, pp. 844–856, Feb. 2013.
- [76] C. M. Bishop, "Bayesian neural networks," *J. Brazilian Comput. Soc.*, vol. 4, no. 1, pp. 61–68, Jul. 1997.
- [77] D. J. C. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural Comput.*, vol. 4, no. 3, pp. 448–472, May 1992.
- [78] R. M. Neal, *Bayesian Learning for Neural Networks*. Secaucus, NJ, USA: Springer-Verlag, 1996.
- [79] Y. Gal, "Uncertainty in Deep Learning," Ph.D. dissertation, Dept. Eng., Univ. Cambridge, Cambridge, U.K., 2016.
- [80] J. S. Denker and Y. LeCun, "Transforming neural-net output levels to probability distributions," in *Proc. 3rd Conf. Adv. Neural Inf. Process. Syst. (NIPS-3)*. San Francisco, CA, USA: Morgan Kaufmann, 1990, pp. 853–859.
- [81] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, New York, NY, USA, vol. 48, Jun. 2016, pp. 1050–1059.
- [82] R. Islam, "Active learning for high dimensional inputs using Bayesian convolutional neural networks," Ph.D. dissertation, Dept. Eng., Univ. Cambridge, Cambridge, U.K., 2016.
- [83] Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with Bernoulli approximate variational inference," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR) Workshop Track*, May 2016.
- [84] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the art," *IEEE Geosci. Remote Sens. Mag.*, vol. 4, no. 2, pp. 22–40, Jun. 2016. [Online]. Available: <http://ieeexplore.ieee.org/document/7486259/>
- [85] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. ICCV*, Sep./Oct. 2009, pp. 2146–2153.
- [86] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, J. Fürnkranz and T. Joachims, Eds. Madison, WI, USA: Omnipress, 2010, pp. 807–814.
- [87] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2011, pp. 315–323.
- [88] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proc. 32nd Int. Conf. Int. Conf. Mach. Learn.*, vol. 37, Lille, France: JMLR, 2015, pp. 1613–1622.
- [89] Y. Gal, R. Islam, and Z. Ghahramani, "Deep Bayesian active learning with image data," in *Proc. Workshop Bayesian Deep Learn. NIPS*, Barcelona, Spain, 2016.
- [90] M. Kampffmeyer, A.-B. Salberg, and R. Jenssen, "Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun./Jul. 2016, pp. 680–688.
- [91] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 26, no. 4, pp. 623–656, 1948.
- [92] N. Houlsby, F. Huszar, Z. Ghahramani, and M. Lengyel. (2011). "Bayesian active learning for classification and preference learning." [Online]. Available: <https://arxiv.org/abs/1112.5745>
- [93] T. Luo *et al.*, "Active learning to recognize multiple types of plankton," in *Proc. 17th Int. Conf. Pattern Recognit. (ICPR)*, vol. 3, Aug. 2004, pp. 478–481.
- [94] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Hyperspectral image segmentation using a new Bayesian approach with active learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 10, pp. 3947–3960, Oct. 2011.
- [95] Q. Liu, R. Hang, H. Song, F. Zhu, J. Plaza, and A. Plaza. (2016). "Adaptive deep pyramid matching for remote sensing scene classification." [Online]. Available: <https://arxiv.org/pdf/1611.03589.pdf>
- [96] W. Zhao and S. Du, "Learning multiscale and deep representations for classifying remotely sensed imagery," *ISPRS J. Photogramm. Remote Sens.*, vol. 128, pp. 223–239, Mar. 2016.
- [97] P. Zhang, M. Gong, L. Su, J. Liu, and Z. Li, "Change detection based on deep feature representation and mapping transformation for multi-spatial-resolution remote sensing images," *Photogram. Remote Sens.*, vol. 116, pp. 24–41, Sep. 2016.
- [98] J. Nocedal, "Updating quasi-Newton matrices with limited storage," *Math. Comput.*, vol. 35, no. 151, pp. 773–782, 1980.
- [99] J. Haut, M. Paoletti, A. Paz-Gallardo, J. Plaza, and A. Plaza, "Cloud implementation of logistic regression for hyperspectral image classification," in *Proc. 17th Int. Conf. Comput. Math. Methods Sci. Eng. (CMMSE)*, J. Vigo-Aguiar, Ed. Cádiz, Spain: Costa Ballena (Rota), 2017, pp. 1063–2321.
- [100] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, Banff, AB, Canada, Apr. 2014.
- [101] M. Volpi and V. Ferrari, "Semantic segmentation of urban scenes by learning local class interactions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2015, pp. 1–9.
- [102] X. Wei, Y. Guo, X. Gao, M. Yan, and X. Sun, "A new semantic segmentation model for remote sensing images," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2017, pp. 1776–1779.



**Juan Mario Haut** (S'17) received the B.Sc. and M.Sc. degrees in computer engineering from the University of Extremadura, Cáceres, Spain, in 2011 and 2014, respectively, where he is currently pursuing the Ph.D. degree with the Hyperspectral Computing Laboratory, Department of Computers and Communications, within the University Teacher Training Program from the Spanish Ministry of Education.

His research interests include remote sensing and analysis of very high spectral resolution with the current focus on deep learning and cloud computing.



**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 Hyperspectral Computing Laboratory, Department of Computers and Communications, within the University Teacher Training Program from the Spanish Ministry of Education

Her research interests include remote sensing and analysis of very high spectral resolution with the current focus on deep learning and high-performance computing.



**Javier Plaza** (M'09–SM'15) received the M.Sc. and Ph.D. degrees in computer engineering from the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura, Cáceres, Spain, in 2004 and 2008, respectively.

He is currently a member of the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, University of Extremadura. He has authored over 150 publications, including over 50 journal citation report papers,

10 book chapters, and 90 peer-reviewed conference proceeding papers. His research interests include hyperspectral data processing and parallel computing of remote sensing data.

Dr. Plaza received the Best Paper Award from the IEEE International Conference on Space Technology and the IEEE Symposium on Signal Processing and Information Technology. He was a recipient of the Outstanding Ph.D. Dissertation Award from the University of Extremadura in 2008, the Most Highly Cited Paper Award (2005–2010) from the *Journal of Parallel and Distributed Computing*, and the Best Column Award from the *IEEE Signal Processing Magazine* in 2015. He has guest edited four special issues on hyperspectral remote sensing for different journals. He is currently an Associate Editor of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS and the IEEE Remote Sensing Code Library. <http://www.umbc.edu/rssipl/people/jplaza>.



**Jun Li** (SM'16) was born in Lodi, Hunan, China, in 1982. She received the Engineering Degree in geographical information systems from Hunan Normal University, Changsha, China, in 2004, the M.Sc. degree in remote sensing and photogrammetry from Peking University, Beijing, China, in 2007, and the Ph.D. degree in electrical and computer engineering from the Instituto Superior Técnico, Technical University of Lisbon, Lisbon, Portugal, in 2011.

From 2011 to 2012, she was a Post-Doctoral Researcher with the Department of Technology of Computers and Communications, University of Extremadura, Cáceres, Spain. She is currently a Professor with the School of Geography and Planning, Sun Yat-sen University, Guangzhou, China, where she founded her own research group on hyperspectral image analysis in 2013. Since 2013, she has been receiving several prestigious funding grants at the national and international levels. She has authored or co-authored a total of 69 journal citation report papers, 48 conference international conference papers, and one book chapter. Her research interests include remotely sensed hyperspectral image analysis, signal processing, supervised/semisupervised learning, and active learning.

Dr. Li's students have also obtained important distinctions and awards at international conferences and symposia. She has been serving as an Associate Editor for the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING since 2014. She has served as a Guest Editor of a special issue in the prestigious PROCEEDINGS OF THE IEEE journal and a special issue in the prestigious *ISPRS Journal of Photogrammetry and Remote Sensing* journal. She has received a significant number of citations to her published works with several papers distinguished as Highly Cited Papers in Thomson Reuters Web of Science—Essential Science Indicators.



**Antonio Plaza** (M'05–SM'07–F'15) received the M.Sc. and Ph.D. degrees in computer engineering from the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, 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 over 600 publications, including over 200 journal citation report papers (over 160 in IEEE journals), 23 book chapters, and around 300 peer-reviewed conference proceeding papers. His research interests include hyperspectral data processing and parallel computing of remote sensing data.

Dr. Plaza 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, and the Steering Committee of the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING (JSTARS). He is a fellow of the IEEE for his contributions to hyperspectral data processing and parallel computing of earth observation data. He received the Best Paper Award from the IEEE International Conference on Space Technology and the IEEE Symposium on Signal Processing and Information Technology. 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, and a recognition as an Outstanding Associate Editor of the IEEE ACCESS in 2017. He was a recipient of the Most Highly Cited Paper Award (2005–2010) from the *Journal of Parallel and Distributed Computing*, the 2013 Best Paper Award of the JSTARS journal, and the Best Column Award from the *IEEE Signal Processing Magazine* in 2015. He has guest edited 10 special issues on hyperspectral remote sensing for different journals. He has reviewed over 500 manuscripts for over 50 different journals. He served as the Director of education activities for the IEEE Geoscience and Remote Sensing Society (GRSS) from 2011 to 2012 and the President for the Spanish Chapter of the IEEE GRSS from 2012 to 2016. He has served as an Associate Editor for the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS and the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING from 2007 to 2012. He is an Associate Editor of the IEEE ACCESS. He served as the Editor-in-Chief for the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING journal from 2013 to 2017. Additional information: <http://www.umbc.edu/rssipl/people/aplaza>.