

# Active Learning with Convolutional Neural Networks for Hyperspectral Image Classification using a new Bayesian Approach

Juan M. Haut, *Student Member, IEEE*, Mercedes E. Paoletti, *Student Member, IEEE*,  
Javier Plaza, *Senior Member, IEEE*, Jun Li, *Senior Member, IEEE*, and Antonio Plaza, *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 semi-supervised) 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 (DNNs) such as convolutional neural networks (CNNs) have recently shown a great potential to yield high performance in 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.

**Index Terms**—Bayesian Convolutional Neural Network, Active learning, hyperspectral remote sensing image classification

## I. INTRODUCTION

This paper was supported by Ministerio de Educación (Resolución de 26 de diciembre de 2014 y de 19 de noviembre de 2015, de la Secretaría de Estado de Educación, Formación Profesional y Universidades, por la que se convocan ayudas para la formación de profesorado universitario, de los subprogramas de Formación y de Movilidad incluidos en el Programa Estatal de Promoción del Talento y su Empleabilidad, en el marco del Plan Estatal de Investigación Científica y Técnica y de Innovación 2013-2016. This work has also been supported by Junta de Extremadura (decreto 297/2014, ayudas para la realización de actividades de investigación y desarrollo tecnológico, de divulgación y de transferencia de conocimiento por los Grupos de Investigación de Extremadura, Ref. GR15005).

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

Jun 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).

TABLE I  
LIST OF ACRONYMS USED IN THIS PAPER

AE	Auto-encoder
AL	Active Learning
ANN	Artificial Neural Network
AVIRIS	Airborne Visible InfraRed Imaging Spectrometer
BALD	Bayesian Active Learning by Disagreement
B-CNN	Bayesian Convolutional Neural Networks
BNN	Bayesian Neural Networks
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
$D_{pool}$	Pool set
$D_{train}$	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
HypIRI	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	Non linearity layer with ReLU
SAE	Stacked Auto-encoder
SSAE	Stacked Sparse Auto-encoder
SVM	Support Vector Machine
RF	Random Forest
ROSI	Reflective Optics System Imaging Spectrometer
RNN	Recurrent Neural Network
SNR	Signal to Noise Ratio

**HYPERSPECTRAL** imaging (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 Airborne Visible InfraRed Imaging Spectrometer -AVIRIS-, the Compact Airborne Spectrographic Imager -CASI-, the Reflective Optics

System Imaging Spectrometer -ROSIS-, the Hymap or the new hyperspectral missions based on HyperCAM technology [2], [3] [4]–[9]) and spaceborne (e.g. the Earth Observing-1 Hyperion or the Fourier Transform HyperSpectral Imager -FTHSI- on MightySat II [10], [11], [12]). Imaging spectrometers are also available on ground-based (stationary or hand-held) 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]–[16], [17]. For instance, we highlight the following spectrometers: AVIRIS [4], which measures the solar reflected spectrum from  $0.4\mu\text{m}$  to  $2.5\mu\text{m}$  at 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  $0.4\mu\text{m}$  to  $2.5\mu\text{m}$  [16], [18], and ROSIS, which collects images with a spectral range from  $0.43\mu\text{m}$  to  $0.96\mu\text{m}$  [7], among others. Also, several new satellite mission will be soon operative and ready to collect data in a very similar spectral range. For instance, the imaging spectrometer included in the NASA Hyperspectral Infrared Imager (HypIRI) [19] is expected to measure the visible to short wave infrared in the range  $0.38\mu\text{m}$  to  $2.5\mu\text{m}$ , or the German Environmental Mapping and Analysis Program (EnMAP [20]) that is expected to collect data in the range  $0.42\mu\text{m}$  to  $2.45\mu\text{m}$ , as well as the Italian Precursore IperSpettrale della Missione Applicativa (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 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 90s [29], [30], artificial neural networks (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 non-parametric (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 non-separable data [33], have made ANNs widely attractive for supervised classification of hyperspectral images as compared to probabilistic methods.

In particular, deep neural networks (DNNs) [34], [35] have recently shown a great potential to yield high performance in image classification tasks [36], [37], [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 multi-layer non-linear transformations applied over DNNs architecture can adaptively extract more meaningful and discriminative features [40]. To date, four DNN models have been the mainstream deep learning (DL) architectures for the analysis of hyperspectral remote sensing images: deep belief networks (DBNs), auto-encoders (AEs), recurrent neural networks (RNNs), and convolutional neural networks (CNNs).

DL 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 hyper-parameters used by the spectral and spectral-spatial DBNs of [43]. Another example is [46] in which a DBN is implemented introducing diversity promoting priors into the pre-training and fine-tuning phases, in order to avoid the co-adaptation of latent factors.

On the other hand, the AE has been traditionally used as 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 stacked AEs (SAEs), for hyperspectral image classification, such as the SAE proposed by [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], a SAE is pre-trained 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 stacked sparse AEs (SSAEs) are proposed to extract spectral and spatial features that are stacked and embed into a support vector machine (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 1D-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 band-to-band fashion [52]. On the other hand, several CNN-based approaches (called Convolutional RNN -CRNN- [53]) have been implemented for hyperspectral classification. In [54], a 1D-CRNN is implemented, where spatial constraints are integrated by linear opinion pools. A similar model is used in [55] where a 1D-CRNN is trained in semi-supervised fashion with labeled and unlabeled data (pixels) using pseudo labels.

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 spectral and spatial information in an easy and natural way. CNNs successively apply convolution filters and pooling operations to the raw input data (which can be 1D, 2D or 3D), 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 an 1D approach, [56] presents a 5-layers 1D-CNN that receives  $n \times 1$  input vectors, where  $n$  is the number of spectral bands, to classify hyperspectral images directly in spectral domain. On the other hand, 2D-CNNs exploit the information from neighboring pixels in order to extract spatial features, whose input data is a patch of  $d \times d$  neighboring pixels [57], normally after applying principal component analysis (PCA) to extract the spectral features [58], [59]. Also, several approaches mix both 1D and 2D-CNNs to extract spectral-space information, respectively [60], [61]. In contrast to these methods, several 3D-CNN models have been proposed that can learn both spatial and spectral features, taking as input data 3D patches from the original hyperspectral data, processing each pixel by means of a 3D 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, the batch size, etc. 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 is 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, 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 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 dataset. In [17] authors provide a method which offers robustness and flexibility in modeling scene images and report the improvements of the accuracy in scene recognition, constructing multi-resolution features and modeling Sparse Features Selection Based Manifold Regularization (SFSMR). In contrast active learning (AL) [70] has been used to facilitate the classification of hyperspectral image data sets, by including intelligently selected unlabeled samples from the original dataset, 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 datasets. Also, similar to multinomial logistic regression, 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 Bayesian neural networks (BNNs) [77], [78], a special kind of ANN that are robust to overfitting, and are able to offer uncertainty estimates and a probabilistic interpretation of deep learning models by inferring distributions over the models' weights, being able to learn from small datasets [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 offer robustness to overfitting on small datasets 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], [82], [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

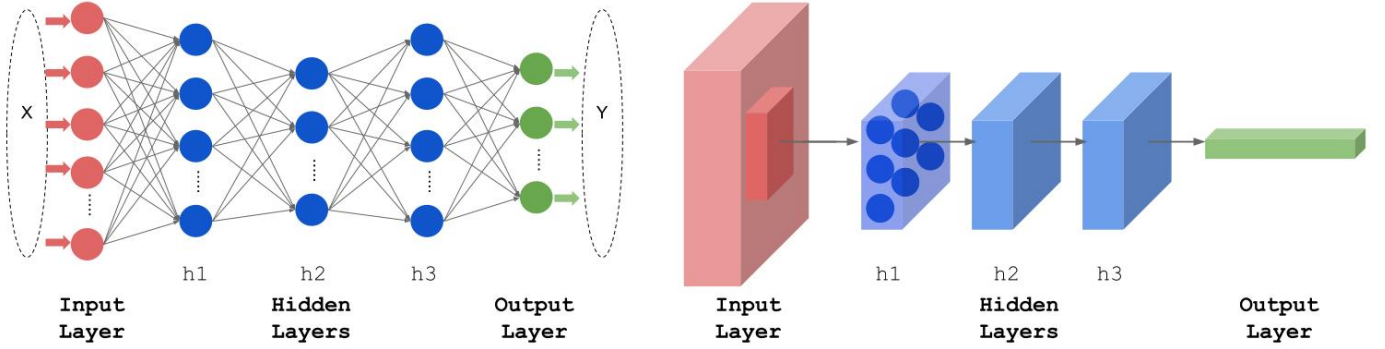


Fig. 1. Comparison between a conventional DNN (a multilayer perceptron or MLP) with 3 hidden fully connected layers and a CNN, also with 3 hidden convolution layers or kernels. The neurons in the CNN create 3-dimensional blocks with sparse connectivity.

classification of hyperspectral remotely sensed data. The main innovative contributions of this paper can be summarized as follows:

- The development (for the first time in the literature) of a dropout-based method (called B-CNN) to extract probabilistic information from 1D, 2D and 3D-CNN models, with the aim of performing accurate spectral-spatial feature-based classification of hyperspectral images using limited training data with different CNN architectures.
- 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.
- The exhaustive analysis and comparison of different acquisition function to perform AL over the implemented B-CNN model, and a detailed comparison between the implemented B-CNNs and the standard 1D, 2D and 3D-CNN classifiers for hyperspectral data, in addition to other traditional hyperspectral data classifiers such as random forest (RF), multilayer perceptron, support vector machine and multinomial logistic regression (MLR).

The remainder of the 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 datasets, highlighting the advantages of the newly proposed classifier. Finally, section IV concludes the paper with some remarks and hints at plausible future research lines.

## II. METHODOLOGY

### A. Active Learning (AL)

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 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  $\mathcal{D}_{train} = [X, Y] = \{\mathbf{x}_i, y_i\}_{i=1}^l$  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, being  $C$  the number of different categories or classes) and by  $\mathcal{D}_{pool} = [X] = \{\mathbf{x}_i\}_{i=l+1}^{l+u} \in \mathbb{R}^d$  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,  $\mathcal{D}_{train}$ ) that iteratively selects new training examples from the pool of candidates ( $\mathcal{D}_{pool}$ ) that provide maximal information about the unlabeled dataset 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,

---

### Algorithm 1 Active Learning (general algorithm)

---

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

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. In [74], the authors 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  $\mathcal{D}_{pool}$ . 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 which is needed for these AL methods. In subsection B we summarize how this problem is solved in [83].

### B. Bayesian Convolutional Neural Networks (B-CNNs)

In contrast to conventional ANNs, the blocks or layers of neurons in CNNs operate like 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 multi-layer perceptron or MLP (see Fig. 1). Each layer actually composes a feature extraction stage that can be of three kinds [64], [84]:

- 1) *Convolution layer* (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  $n$  the deep of the cube (also, spectral bands). Each neuron in one filter that composes the CONV layer (being  $k$  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:

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

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  $X$ ,  $w_{i,j,\hat{t}}$  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 result an output volume  $Z$  is obtained, which will be an array composed by  $k$  1, 2 or 3-dimensional 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 rectified linear unit or ReLU [85]–[87]) which is then applied to each component of the obtained feature map to learn nonlinear representations:  $A = f(Z)$ .
- 3) *Pooling layer*: pooling layers are used to resume the output  $Z$  of several nodes in convolution 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 non-overlapping 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 overfitting problem makes 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 dataset  $\mathcal{D}_{train} = [X, Y]$  composed by inputs  $X = \{x_1, \dots, x_l\}$  (where each  $x_i \in \mathbb{R}^n = [x_{i,1}, x_{i,2}, \dots, x_{i,n}]$ ) and their corresponding outputs  $Y = \{y_1, \dots, y_l\}$  (where each  $y_i = \{1, 2, \dots, c\}$ ), the model posterior' goal is to estimate a function  $y_i = f(x_i)$  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  $f$ ,  $p(Y|X, f)$ . So, the posterior distribution will be  $p(f|X, Y) = p(f|\mathcal{D}_{train})$  that captures the most likely functions given the observed data. In this way, the output  $y^*$  of a new input  $x^*$  can be predicted as the marginal likelihood:

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

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

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

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  $p(\omega|\mathcal{D}_{train})$ . However, the probability distribution  $p(\omega|\mathcal{D}_{train})$  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:

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

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 Eq. (3). The change resides in replacing the intractable probability distribution  $p(\omega|\mathcal{D}_{train})$  by the approximate distribution  $q(\omega)$  that belongs to a tractable family which minimises the Kullback-Leibler divergence,  $D_{KL}(q(\omega)||p(\omega|\mathcal{D}_{train})) = 0$ , a measure that returns the similarity between both 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:

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

being  $\hat{\omega}_t \sim q(\omega)$  called Monte Carlo dropout or 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 convolution 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  $\mathcal{D}_{pool}$  and inputs  $\mathbf{x} \in \mathcal{D}_{pool} \in \mathbb{R}^d$  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  $\mathcal{D}_{train}$ :

$$S = \operatorname{argmax}_{\mathbf{x} \in \mathcal{D}_{pool}} a(\mathbf{x}, \mathcal{M})$$

The paper [89] makes a review and a comparison between different acquisition functions. This work performs a comparison with six different acquisition methods, which 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:

- *Random acquisition* or baseline: it chooses a point  $\mathbf{x}_i$  following a uniformly random distribution from  $\mathcal{D}_{pool} \Rightarrow a(\mathbf{x}_i) = \operatorname{unif}()$  where  $\operatorname{unif}()$  returns a draw from a uniform distribution over the interval  $[0, 1]$ .
- *Mean standard deviation* [90]: for each  $\mathbf{x}_i$ , it calculates the  $\sigma(\mathbf{x}_i) = \frac{1}{C} \sum_c \sigma_c$ , where  $C$  is the number of classes,  $c$  are the classes that  $\mathbf{x}_i$  can take, and

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

- *Maximum entropy* [91]: it chooses the  $\mathbf{x}_i \in \mathcal{D}_{pool}$  with the highest classification uncertainty, i.e. the  $\mathbf{x}_i$  that maximize the predictive entropy

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

- *Bayesian active learning by disagreement* (BALD) [92]: this method chooses the  $\mathbf{x}_i \in \mathcal{D}_{pool}$  that are expected to maximize the mutual information between the predictions and the model posterior

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

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

- *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  $\mathcal{D}_{train}$ . Sample  $\mathbf{x}^{BT}$  is selected from  $\mathcal{D}_{pool}$  by:

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

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

- *Mutual Information criterion* [94], [95]: it measures the mutual dependence between samples. In fact, this function selects the sample  $\mathbf{x}^{MI}$  maximizing the mutual information (MI) between the obtained results and the class labels:

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

where  $\mathbb{I}(\omega; y_i|\mathbf{x}_i) = \frac{1}{2} \log \left( \frac{|H^{MI}|}{H} \right)$  represents the MI between the obtained results and the class label  $y_i$ , being  $H$  the posterior precision matrix and  $H^{MI}$  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, BALD method has been approximated with  $q(\omega)$ , as described by [89]:

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

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

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

Now we can apply Monte Carlo integration as follows:

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

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

Finally, we present the new B-CNN architecture developed in this work. 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 1D-CNN architecture [14], [37], [56].
- 2) Extracting only spatial information implementing a 2D-CNN architecture [57], [96]–[98].
- 3) Extracting spectral-spatial information implementing a 3D-CNN architecture [37], [99].

In this regard, we emphasize that our B-CNN approach can be applied to 1D, 2D and 3D-CNN architectures. This work 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: 1D, 2D B-CNN and 3D B-CNN.

1) *Spectral B-CNN Architecture*: This model takes advantage of only the spectral information contained in the input hyperspectral image by developing an 1D-CNN architecture and performing a traditional pixel-wise-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 1D 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 into feature maps, capturing the spectral information contained in  $\mathbf{x}_i$ . Eq. (1) can be re-written as:

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

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  $1 \times q$ ,  $w_{\hat{t}}$  is a weight of vector  $W$ ,  $b$  is the bias of the layer and  $s$  is the stride.

In order to compare the implemented 1D B-CNN model with the 1D CNN baseline, the model's architecture has been inspired by work [56]. As we can see in Fig. 2, the proposed

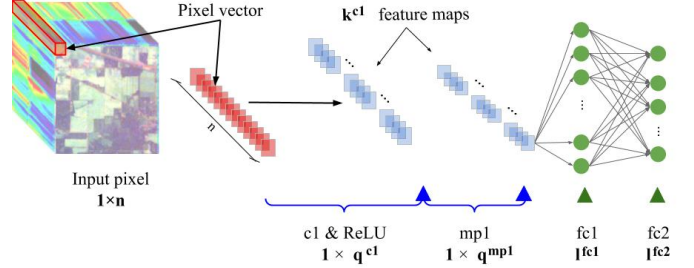


Fig. 2. Proposed spectral Bayesian-Convolutional Neural Network (1D B-CNN) architecture

1D B-CNN model is composed by one input layer that receives the pixel vector with all its spectral bands. This input feeds one convolution 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 Monte-Carlo dropout. Table II shows the details of the 1D B-CNN implementation.

TABLE II  
CONFIGURATION OF OUR THREE B-CNN ARCHITECTURES.  $C$  INDICATES THE NUMBER OF CLASSES CONTAINED IN THE HYPERSPECTRAL DATASET.

	Convolution layers			
	Kernel $k^c \times 1 \times q^c$	Activation Function	Pooling $1 \times q^{mp}$	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	-	
	Convolution layers			
	Kernel $k^c \times l^c \times l^c$	Activation Function	Pooling $l^{mp} \times l^{mp}$	Dropout (%)
2D B-CNN	$50 \times 3 \times 3$	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	-		
	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 (%)
3D B-CNN	$500 \times 3 \times 3 \times n$	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	-		

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 datasets. As result, a 2D-CNN architecture has been implemented, whose input is composed by patches of size  $d \times d \times 1$  extracted from the

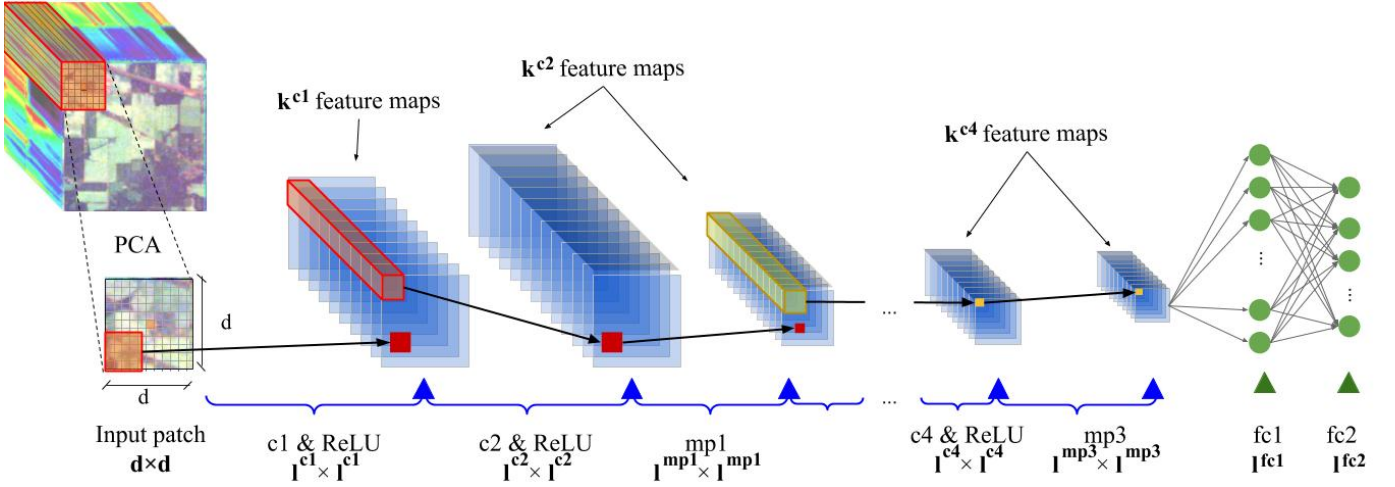


Fig. 3. Proposed spatial Bayesian-Convolutional Neural Network (2D B-CNN) architecture

hyperspectral scene. Normally, CNNs in general (and B-CNNs in particular) receive a completely normalized image prior to classification, i.e. a 3D 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 1D B-CNN, but we also need an additional mechanism in order to include also the spatial information in the 2D 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 2D model accepts volumes of  $d \times d \times 1$  [64], after processing the original scene with PCA. This requires a pre-processing 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 pixel of the patch  $[d/2 + 1, d/2 + 1, n]$ . In this case, Eq. (1) can be re-written 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  $W$ ,  $b$  the bias of the layer and  $s$  the stride.

Fig. 3 shows the implemented 2D B-CNN model. In this case a deeper architecture has been selected. As we can observe, the input layer receives 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 have been added at the end of certain layers in order to model the uncertainty of the network. Table II shows the details of the 2D 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 3D architecture that receives as input data patches of size  $d \times d \times n$ , being  $n$  the number of spectral bands. As in 2D B-CNN model, a pre-processing 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 Eq. (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 convolution layers,  $c1$  and  $c2$  (with ReLU as nonlinear activation function), two maxpool layers at the end of each convolution 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 the uncertainty estimation. Table II provides additional details about the considered spectral-spatial B-CNN architecture.

The parameters of the considered 1D, 2D and 3D architectures, including the number and type of layers or kernel sizes, are one of the design choices of the proposed spectral, spatial, 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 the training and showing its robustness, regardless of the fact that non-optimal or customized topologies are adopted.

Also, we must remark that the proposed 1D, 2D and 3D B-CNN models have been developed as a computation graph using the library for machine intelligence Keras with TensorFlow backend over CUDA toolkit and the library of primitives for deep neural networks cuDNN. This computation graph is composed by connected nodes that represents operations (also called units of computation), while connections (or edges)



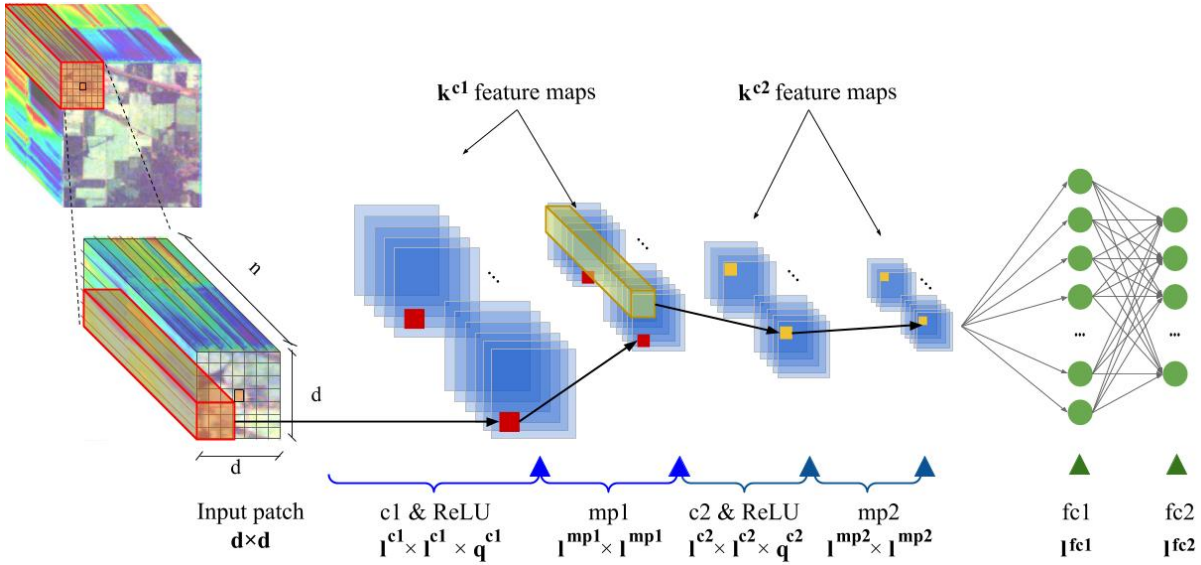


Fig. 4. Proposed spectral-spatial Bayesian-Convolutional Neural Network (3D B-CNN) architecture

represent the data consumed (input connections) and produced (output connections) in the unit. These connections allow us to represent the existing dependencies between different operations, making it possible to identify those operations that can be executed in parallel in an easy way.

The process of our 1D, 2D and 3D 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 datasets: two randomly selected samples per class,  $2 \cdot C$ , will compose the initial *training set*,  $\mathcal{D}_{train}^0$  and the remaining samples will compose the *working set*. From the *working set*, 50% of random selected pixels will compose the initial *pool set*,  $\mathcal{D}_{pool}^0$  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 is pre-processed 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  $\mathcal{D}_{train}^0$  and the pool set  $\mathcal{D}_{pool}^0$  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 2D or 3D models, for each pixel  $x_i \in \mathcal{D}_{train}^0$ , 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:

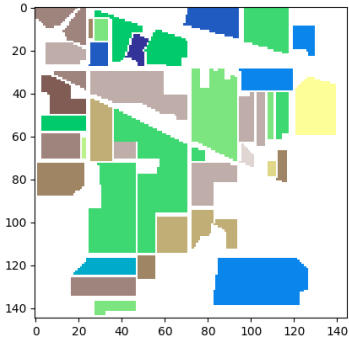
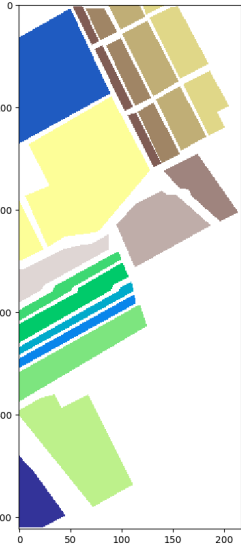
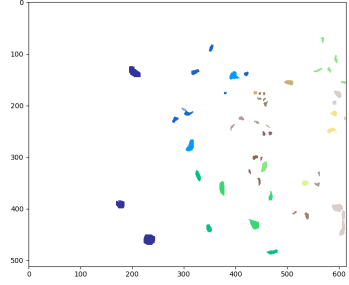









































$$H_y(y') = \sum_i y_i \log(y'_i),$$

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,  $\mathcal{D}_{pool}^0$  is sent as 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  $\mathcal{D}_{pool}^0$ ,  $p(y'_i | x_i, \mathcal{D}_{train})$ . From an implementation point of view,  $\mathcal{D}_{pool}^0$  passes through the network  $T$  times, being  $T$  the number of stochastic forward passes. As a result,  $T$  different outputs  $y'_i$  have been obtained for each  $x_i$  in  $\mathcal{D}_{pool}^0$ . To obtain the final probability, the average between all the outputs is calculated  $y'_i = \frac{1}{T} \sum_{t=1}^T y_i^{(t)}$ , being  $y_i^{(t)}$  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  $\mathcal{D}_{pool}^0$  according to their uncertainty. Then, those samples with higher score are selected, creating the  $\mathcal{D}_{selected}^0$  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 corresponding labels, each  $x_i \in \mathcal{D}_{selected}^0$  is paired with its corresponding label  $y_i$ . Finally, the selected pixels in  $\mathcal{D}_{selected}^0$  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  $\mathcal{D}_{train}^1$ . Also, the selected pixels in  $\mathcal{D}_{selected}^0$  are deleted from the pool set, creating  $\mathcal{D}_{pool}^1$ .

After validating the model, the training process is repeated successively with each  $\mathcal{D}_{train}^\epsilon$  until a satisfactory result is

TABLE III  
NUMBER OF SAMPLES OF THE INDIAN PINES (IP), SALINAS VALLEY (SV) AND KENNEDY S.C. (KSC) HSI DATASETS.

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		Brocoli-green-weeds-1	2009		Scrub	761
	Corn-notill	1428		Brocoli-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

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 4.20GHz (4 cores/8 way multitask processing), 40GB of DDR4 RAM with a serial speed of 2400MHz, a graphical processing unit (GPU) NVIDIA GeForce GTX 1080 with 8GB GDDR5X of video memory and 10Gbps of memory frequency, a Toshiba DT01ACA HDD with 7200RPM and 2TB of capacity, and an ASUS Z170 pro-gaming motherboard.

On the other hand, the used software environment is composed by Ubuntu 16.04.4 x64 as operating system, CUDA 8 and cuDNN 5.1.5, Python 2.7 as programming languages.

#### B. Hyperspectral Datasets

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

- The first one is the well-known Indian Pines (IP) dataset (Table III). This dataset was gathered by AVIRIS [4] in 1992, over a set of agricultural fields with regular geometry and with a multiple crops and irregular patches of forest in Northwestern Indiana. The IP scene has  $145 \times 145$  pixels with 224 spectral bands in the range from 400 to 2500nm, with 10nm of spectral resolution, 20m spatial resolution and 16 bits radiometric resolution. After an initial analysis, 4 zero bands and another 20

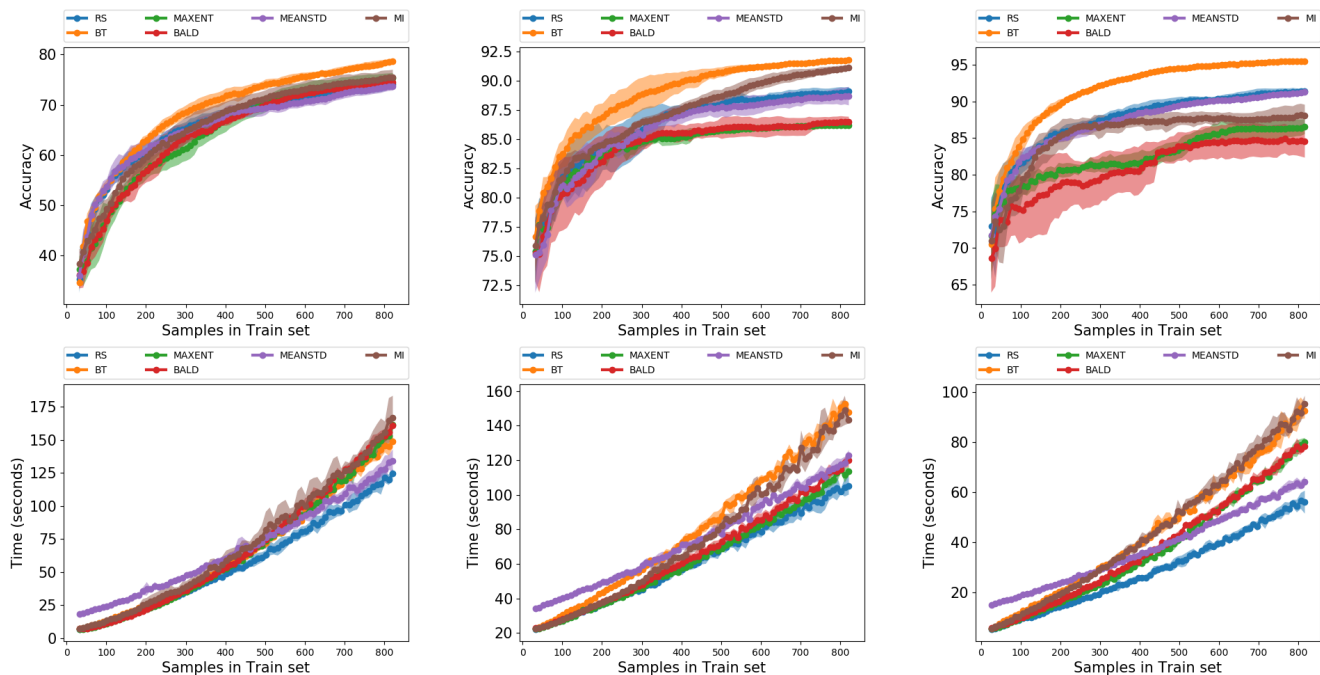


Fig. 5. AL-based performance obtained by the MLR for different acquisition functions with different sizes of  $D_{train}$ . First column: results for the IP dataset. Second column: results for the SV dataset. Third column: results for the KSC dataset.

bands with lower signal-to-noise ratio (SNR) because of atmospheric absorption have been removed, retaining only 200 spectral channels. Moreover, about half of the pixels in the hyperspectral image (10249 of 21025) contain ground-truth information, which comes in the form of a single label assignment having a total of 16 ground-truth classes.

- The second hyperspectral dataset used in experiments was also collected by the AVIRIS instrument, in this case over Salinas Valley (SV), California (Table III). The covered area has  $512 \times 217$  samples and the spatial resolution is 3.7m per pixel. 204 out of the 224 bands are kept after 20 water absorption bands are removed. The ground-truth is composed of 54129 pixels and 16 land-cover classes, including vegetables, bare soils, and vineyard fields.
- The third dataset used in experiments is the Kennedy Space Center (KSC) (Table III), 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 2500nm and with 20m 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 subsection II.C are tested considering an MLR classifier and the proposed 1D, 2D and 3D 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, 1D, 2D and 3D B-CNN models adapted to AL) and the original ones (baseline MLR, 1D, 2D and 3D-CNN models), with the aim of studying the impact of  $D_{train}$  on the performance of both AL and traditional methods.

Also, we remark that each experiment uses the three considered hyperspectral datasets, 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 (LBFGS) optimizer [100], [101] in the case of the MLR, with  $L2$  as penalty and tolerance value fixed to  $1e-18$ , being 1000 the number of maximum iterations, and with the Adam optimizer [102] for the 1D, 2D and 3D 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  $d = 19$  provides an appropriate compromise for the considered images (the selection of other close values of  $d$  did not have a significant impact on the final classification results). Finally, for each  $\epsilon$ , 10 unlabeled pixels have been chosen.

1) *Experiment 1: Performance of different acquisition functions with the MLR:* This experiment implements an AL-based MLR classifier. The output of the classifier gives directly the probability of each sample  $\mathbf{x}_i$  belongs to the class  $c$ , i.e.  $p(y_i = c | \mathbf{x}_i, D_{train})$ . This means that, to obtain the probabilities of  $D_{pool}$ , only one forward pass is needed,  $T = 1$ . Before running each implementation of AL-MLR, the testing set and the validation set are created. In this sense,  $D_{train}^0$  starts with

TABLE IV

AL-BASED CLASSIFICATION RESULTS OBTAINED BY THE MLR FOR DIFFERENT ACQUISITION FUNCTIONS AFTER 80 ITERATIONS AND 10 ACQUISITIONS PER ITERATION.

		Indian Pines					
Class id.	RANDOM	BALD	MAXENT	BT	MI	MEANSTD	
0	23.48 (9.47)	39.57 (13.07)	36.09 (11.30)	31.74 (19.18)	<b>47.39</b> (19.99)	42.17 (11.88)	
1	74.22 (2.45)	76.90 (2.30)	75.28 (3.57)	<b>78.14</b> (2.12)	77.28 (4.17)	69.75 (1.89)	
2	52.94 (1.61)	57.93 (3.83)	57.95 (3.24)	61.25 (4.22)	<b>62.43</b> (3.85)	56.36 (4.61)	
3	41.94 (8.88)	56.88 (9.84)	<b>64.14</b> (6.30)	47.17 (9.11)	46.58 (7.97)	37.38 (9.04)	
4	74.95 (6.43)	66.54 (20.30)	52.59 (22.19)	<b>85.09</b> (2.58)	51.51 (17.99)	79.71 (4.62)	
5	93.78 (1.12)	87.62 (9.74)	94.30 (1.39)	<b>94.79</b> (1.64)	93.12 (2.78)	94.41 (1.70)	
6	60.71 (9.85)	76.43 (12.49)	75.71 (8.57)	65.00 (17.11)	<b>84.29</b> (2.86)	50.71 (20.13)	
7	<b>97.07</b> (1.08)	91.34 (2.23)	89.79 (3.25)	96.99 (1.06)	90.46 (10.56)	94.31 (2.82)	
8	41.00 (17.72)	72.00 (6.78)	<b>74.00</b> (9.20)	63.00 (4.00)	57.00 (17.20)	51.00 (23.11)	
9	64.77 (4.35)	60.78 (4.08)	54.86 (3.97)	<b>68.79</b> (4.12)	64.24 (6.67)	64.01 (1.86)	
10	73.49 (1.93)	78.48 (2.78)	<b>81.65</b> (2.19)	79.23 (2.20)	80.63 (2.73)	74.02 (3.36)	
11	54.03 (2.28)	63.58 (5.48)	63.74 (3.46)	<b>65.87</b> (3.26)	60.57 (2.70)	57.44 (4.76)	
12	92.54 (0.31)	92.78 (5.11)	96.29 (3.50)	<b>98.73</b> (0.66)	97.46 (1.81)	98.24 (0.66)	
13	92.14 (1.18)	84.38 (5.73)	88.70 (6.00)	<b>93.88</b> (1.92)	83.72 (10.80)	93.04 (0.96)	
14	65.85 (6.89)	66.11 (6.20)	72.59 (7.57)	66.99 (4.98)	<b>73.63</b> (4.45)	56.37 (5.52)	
15	85.16 (1.85)	82.37 (2.77)	80.65 (8.27)	79.57 (9.40)	<b>85.59</b> (6.33)	82.15 (5.71)	
OA	74.16 (0.91)	74.64 (0.88)	75.35 (1.12)	<b>78.79</b> (0.53)	74.58 (1.45)	73.90 (0.92)	
AA	68.38 (1.64)	72.11 (1.67)	72.40 (1.00)	<b>73.52</b> (2.12)	72.24 (1.05)	68.82 (1.78)	
K	74.16 (1.08)	74.64 (0.96)	75.35 (1.32)	<b>78.79</b> (0.58)	75.48 (1.58)	73.90 (1.06)	
Time (seconds)	124.39 (3.84)	160.69 (3.17)	161.48 (6.08)	148.59 (4.71)	166.72 (16.54)	133.83 (8.96)	
		Salinas Valley					
Class id.	RANDOM	BALD	MAXENT	BT	MI	MEANSTD	
0	98.26 (0.61)	99.27 (0.79)	<b>99.54</b> (0.25)	99.52 (0.21)	99.27 (0.62)	97.57 (0.73)	
1	99.78 (0.07)	99.80 (1.82)	98.92 (1.48)	99.79 (0.09)	<b>99.88</b> (0.10)	99.73 (0.09)	
2	94.94 (1.82)	99.27 (0.39)	99.49 (0.51)	98.93 (0.58)	<b>99.66</b> (0.19)	94.26 (6.00)	
3	99.24 (0.38)	99.64 (0.06)	<b>99.70</b> (0.18)	99.40 (0.30)	99.68 (0.06)	99.37 (0.11)	
4	97.36 (1.21)	99.65 (0.03)	<b>99.67</b> (0.05)	99.29 (0.31)	99.51 (0.09)	96.65 (2.98)	
5	69.57 (0.18)	99.77 (0.29)	<b>99.85</b> (0.07)	99.67 (0.22)	99.77 (0.18)	99.72 (0.07)	
6	99.66 (0.16)	98.93 (0.65)	99.03 (1.48)	99.74 (0.08)	<b>99.92</b> (0.02)	99.66 (0.06)	
7	81.89 (3.01)	92.97 (3.40)	<b>99.39</b> (0.24)	86.43 (0.66)	89.70 (1.86)	80.42 (1.58)	
8	99.86 (0.07)	99.99 (0.01)	99.99 (0.01)	99.87 (0.07)	99.95 (0.04)	99.78 (0.09)	
9	88.50 (2.12)	<b>97.71</b> (0.22)	97.60 (0.22)	95.33 (1.14)	97.36 (0.58)	89.07 (1.27)	
10	91.95 (3.05)	98.35 (0.45)	98.88 (1.04)	95.30 (0.91)	<b>99.25</b> (0.43)	89.64 (4.28)	
11	99.03 (0.73)	99.92 (0.07)	<b>99.94</b> (0.08)	99.47 (0.11)	99.79 (0.14)	99.25 (0.54)	
12	94.39 (8.09)	99.28 (0.25)	99.43 (0.14)	98.41 (0.95)	<b>99.50</b> (0.32)	97.62 (1.68)	
13	92.26 (1.34)	<b>98.64</b> (0.23)	98.39 (0.27)	96.06 (0.69)	98.06 (4.48)	93.23 (1.08)	
14	60.89 (3.55)	14.46 (0.07)	1.83 (0.51)	<b>65.39</b> (1.02)	53.00 (3.64)	59.31 (6.16)	
15	95.29 (2.48)	99.35 (0.15)	<b>99.42</b> (0.10)	98.48 (0.58)	99.32 (0.25)	96.58 (1.34)	
OA	89.20 (0.30)	86.49 (0.42)	86.25 (0.22)	<b>91.80</b> (0.08)	91.19 (0.22)	88.71 (0.77)	
AA	93.30 (0.59)	93.44 (0.35)	93.19 (0.20)	95.69 (1.11)	<b>95.85</b> (0.12)	93.24 (0.61)	
K	89.20 (0.33)	86.49 (0.49)	86.25 (0.25)	<b>91.80</b> (0.09)	91.19 (0.25)	88.71 (0.87)	
Time (seconds)	56.07 (4.55)	78.33 (0.45)	79.86 (1.78)	92.55 (3.16)	95.20 (3.22)	64.29 (2.00)	
		Kennedy Space Center					
Class id.	RANDOM	BALD	MAXENT	BT	MI	MEANSTD	
0	95.90 (0.87)	70.80 (8.57)	78.29 (4.19)	<b>97.98</b> (0.68)	88.09 (5.47)	95.48 (1.21)	
1	88.81 (1.75)	<b>92.43</b> (3.31)	90.55 (3.40)	89.55 (2.04)	90.37 (3.43)	91.11 (2.20)	
2	87.97 (4.75)	75.23 (6.78)	76.02 (9.98)	<b>93.20</b> (0.72)	79.30 (3.41)	91.72 (1.51)	
3	67.70 (11.40)	79.37 (3.08)	75.32 (8.02)	<b>88.41</b> (1.53)	86.98 (1.21)	62.94 (9.38)	
4	62.11 (8.24)	65.71 (4.67)	65.09 (9.94)	<b>77.76</b> (3.88)	71.55 (1.27)	65.71 (7.69)	
5	71.35 (4.48)	<b>84.80</b> (2.61)	83.06 (1.31)	83.32 (4.00)	83.49 (2.02)	66.90 (4.84)	
6	84.19 (5.51)	63.43 (14.39)	70.86 (16.03)	<b>95.24</b> (0.85)	71.81 (12.51)	77.52 (4.92)	
7	90.35 (1.26)	82.83 (13.54)	89.14 (6.23)	<b>97.54</b> (0.78)	78.70 (9.77)	91.28 (1.82)	
8	96.81 (0.63)	86.08 (3.88)	85.12 (4.86)	<b>98.38</b> (0.58)	90.19 (2.58)	95.38 (2.65)	
9	94.90 (2.79)	84.70 (3.83)	88.27 (3.88)	<b>97.23</b> (0.79)	90.59 (4.65)	95.69 (1.06)	
10	96.95 (0.71)	84.77 (12.51)	91.89 (7.27)	<b>98.23</b> (0.63)	89.79 (14.23)	95.99 (1.04)	
11	92.41 (1.19)	91.81 (1.62)	91.81 (2.29)	<b>94.99</b> (0.66)	93.68 (1.60)	94.12 (1.30)	
12	100.00 (0.00)	98.12 (2.78)	98.10 (1.30)	99.74 (0.13)	93.85 (8.57)	100.00 (0.00)	
OA	91.49 (0.43)	84.36 (2.36)	86.53 (1.21)	<b>95.56</b> (0.28)	88.00 (1.45)	91.35 (0.54)	
AA	86.88 (0.60)	81.55 (2.08)	83.35 (1.47)	<b>93.20</b> (0.30)	85.26 (0.88)	86.45 (0.67)	
K	91.49 (0.47)	84.36 (2.61)	86.53 (1.33)	<b>95.56</b> (0.31)	88.00 (1.61)	91.35 (0.60)	
Time (seconds)	105.23 (5.12)	119.99 (0.66)	113.51 (2.36)	147.96 (4.54)	143.32 (2.12)	122.74 (3.16)	

2-C labeled samples. This means that IP and SV start with 32 labeled pixels and KSC starts with 26, while 50% of the remaining ground-truth samples is used to initialize the pool set  $\mathcal{D}_{pool}^0$ , i.e., IP's  $\mathcal{D}_{pool}^0$  starts with 5109 unlabeled pixels and SV with 27049. With the KSC dataset, the percentage has been changed in order to reserve more data for the pool set, so that 85% of the remaining ground-truth samples have been selected to form  $\mathcal{D}_{pool}^0$  with 4407 samples, instead of 2592. After  $\mathcal{D}_{train}^0$  and  $\mathcal{D}_{pool}^0$  are created, 5% of the remaining ground-truth data is used for validation and the remaining 95% for testing.

Table IV shows the accuracy results of each acquisition function after 80 iterations, i.e.  $\epsilon = 80$ , with 10 acquisitions at each iteration. Focusing on the IP dataset, the AL-MLR classifier with BT-criterion obtains the best overall accuracy (OA) when compared with the other functions, reaching 78.79% with only 8.12% of the ground-truth, while the lowest OA is reached by the mean standard deviation (mean STD), with 73.90%. We can observe this behavior at different  $\mathcal{D}_{train}$  sizes in Fig. 5, which shows the evolution of the AL-MLR accuracy

TABLE V

AL-BASED CLASSIFICATION RESULTS OBTAINED BY THE 1D B-CNN FOR DIFFERENT ACQUISITION FUNCTIONS AFTER 80 ITERATIONS AND 10 ACQUISITIONS PER ITERATION.

		Indian Pines					
Class id.	RANDOM	BALD	MAXENT	BT	MI	MEANSTD	
0	55.80 (6.72)	<b>89.13</b> (4.70)	63.77 (9.78)	71.74 (15.17)	50.00 (32.54)	71.74 (15.47)	
1	82.05 (1.63)	76.61 (3.32)	76.98 (1.50)	<b>82.12</b> (4.13)	79.06 (3.89)	80.58 (2.80)	
2	61.77 (4.61)	71.49 (4.85)	69.72 (0.93)	<b>72.97</b> (3.73)	71.85 (2.14)	63.25 (5.48)	
3	63.01 (8.78)	82.98 (4.38)	79.47 (4.63)	83.12 (3.10)	<b>84.67</b> (7.11)	70.46 (1.79)	
4	90.82 (1.67)	<b>95.65</b> (0.45)	89.99 (3.83)	94.13 (1.98)	77.78 (25.00)	91.24 (0.43)	
5	98.31 (0.74)	95.75 (1.84)	<b>98.90</b> (0.19)	98.26 (0.26)	97.95 (1.17)	97.58 (0.81)	
6	77.38 (8.42)	<b>94.05</b> (1.68)	89.29 (2.92)	82.14 (10.51)	91.67 (4.45)	63.10 (10.24)	
7	<b>98.68</b> (0.71)	96.51 (1.37)	97.77 (1.29)	97.42 (1.73)	95.47 (5.00)	98.61 (1.24)	
8	66.67 (10.27)	93.33 (6.24)	96.67 (2.36)	96.67 (10.27)	96.67 (4.71)	53.33 (20.95)	
9	84.43 (3.88)	67.28 (12.64)	71.40 (4.80)	<b>86.01</b> (3.99)	78.84 (2.23)	76.30 (5.75)	
10	76.78 (1.82)	81.60 (5.70)	85.36 (1.44)	<b>83.54</b> (6.24)	<b>85.99</b> (1.63)	84.15 (3.50)	
11	78.98 (4.03)	80.94 (7.36)	<b>78.98</b> (7.34)	86.90 (3.10)	<b>89.60</b> (1.95)	76.50 (3.90)	
12	98.21 (1.15)	97.40 (2.04)	<b>98.37</b> (1.61)	94.47 (4.53)	98.05 (1.83)	96.42 (3.68)	
13	94.60 (2.28)	95.86 (0.32)	<b>96.73</b> (0.94)	95.34 (2.97)	94.52 (3.97)	96.63 (1.45)	
14	53.89 (1.68)	<b>69.43</b> (4.88)	67.53 (3.25)	65.72 (2.32)	65.63 (1.82)	53.89 (4.48)	
15	90.32 (2.63)	<b>97.49</b> (0.51)	92.11 (1.01)	92.47 (2.32)	95.70 (0.88)	89.25 (1.76)	
OA	81.83 (0.78)	82.94 (1.08)	83.85 (0.41)	<b>86.14</b> (0.42)	84.78 (1.62)	82.94 (2.00)	
AA	79.48 (0.03)	<b>86.60</b> (0.97)	84.57 (0.69)	85.81 (0.07)	84.59 (0.46)	78.94 (1.01)	
K	81.83 (0.91)	82.94 (1.25)	83.85 (0.50)	<b>86.14</b> (0.50)	84.78 (1.86)	82.94 (1.17)	
Time (seconds)	<b>496.82</b> (1.27)	620.43 (3.09)	608.47 (3.11)	612.55 (4.08)	609.77 (6.13)	663.72 (5.10)	
		Salinas Valley					
Class id.	RANDOM	BALD	MAXENT	BT	MI	MEANSTD	
0	99.00 (0.42)	<b>99.98</b> (0.02)	99.40 (0.84)	97.05 (1.92)	99.80 (0.22)	99.75 (0.04)	
1	99.95 (0.00)	<b>99.98</b> (0.03)	99.97 (0.02)	99.91 (0.05)	99.97 (0.02)	99.78 (0.24)	
2	97.79 (0.56)	96.54 (4.05)	98.52 (1.66)	<b>98.82</b> (1.02)	99.68 (0.21)	98.46 (0.42)	
3	98.76 (0.96)	99.47 (0.07)	99.69 (0.12)	99.59 (0.12)	<b>99.71</b> (0.10)	99.45 (0.09)	
4	96.98 (1.18)	99.56 (0.02)	<b>99.64</b> (0.06)	99.10 (0.21)	98.51 (1.51)	97.82 (0.64)	
5	99.80 (0.13)	99.97 (0.02)	<b>100.00</b> (0.00)	99.93 (0.03)	99.97 (0.00)	99.88 (0.07)	
6	96.88 (0.09)	100.00 (0.00)	100.00 (0.00)	99.87 (0.07)	99.97 (0.00)	99.83 (0.18)	
7	83.43 (3.15)	<b>87.47</b> (8.97)	78.67 (4.87)	79.67 (7.92)	86.61 (5.95)	74.61 (6.37)	
8	99.26 (0.43)	99.95 (0.04)	<b>99.99</b> (0.02)	99.91 (0.08)	99.96 (0.01)	99.63 (0.24)	
9	93.49 (2.15)	<b>98.98</b> (0.25)	98.75 (0.32)	98.08 (0.78)	98.59 (0.48)	95.90 (1.68)	
10	94.48 (1.99)	99.47 (0.16)	<b>99.81</b> (0.15)	98.91 (0.50)	98.94 (0.94)	96.60 (0.42)	
11	<b>99.97</b> (0.05)	99.88 (0.05)	99.88 (0.09)	99.65 (0.23)	99.48 (0.26)	99.81 (0.02)	
12	98.25 (0.62)	94.54 (0.09)	98.36 (1.01)	97.63 (2.04)	<b>99.49</b> (0.19)	97.78 (0.95)	
13	91.03 (1.75)	96.60 (1.00)	98.63 (0.97)	97.60 (1.13)	<b>99.19</b> (0.23)	94.92 (1.96)	
14	66.41 (7.54)	59.81 (26.88)	76.45 (3.55)	<b>83.63</b> (6.10)	74.95 (9.27)	74.98 (7.19)	
15	98.34 (0.57)	99.76 (0.09)	<b>99.89</b> (0.08)	99.79 (0.79)	99.56 (0.24)	97.14 (1.00)	
OA	90.85 (0.77)	91.58 (2.24)	92.15 (0.60)	93.06 (0.89)	<b>93.57</b> (0.29)	90.53 (0.63)	
AA	94.79 (0.64)	95.75 (1.51)	96.73 (0.16)	96.77 (0.30)	<b>97.15</b> (0.33)	95.40 (0.23)	
K	90.85 (0.87)	91.58 (2.55)	92.15 (0.66)	93.06 (0.97)	<b>93.57</b> (0.32)	90.53 (0.68)	
Time (seconds)	<b>484.01</b> (1.91)	607.08 (4.15)	597.70 (4.37)	597.47 (5.31)	600.83 (3.40)	644.05 (4.00)	
		Kennedy Space Center					
Class id.	RANDOM	BALD	MAXENT	BT	MI	MEANSTD	
0	97.33 (0.16)	97.33 (0.78)	<b>98.73</b> (1.00)	98.64 (1.19)	98.69 (0.77)	96.01 (2.29)	
1	93.42 (1.16)	93.28 (1.85)	95.06 (0.89)	91.22 (2.74)	<b>95.61</b> (3.03)	90.81 (4.90)	
2	86.85 (8.15)	96.22 (1.61)	<b>98.05</b> (0.64)	93.67			

although the random function reaches better OA than the mean STD, its average accuracy (AA) is slightly worse, i.e. the mean STD generalizes better than the random function, as it can be observed with Oats and Alfalfa (class 0, with 46 pixels), where mean STD reaches better accuracy than the random function when less pixels are available.

A similar behavior can be observed for the SV dataset, where distance-based methods reach better OA than random and entropy-based ones. In particular the BT-criterion is able to reach 91.80% accuracy with only 1.53% of the SV ground-truth, being max-entropy the function with lowest OA, 86.25%. In general, entropy-based methods reach poorer OAs with SV dataset than the random function, due to the spectral characteristics of the image and the number of pixels per class, which is quite balanced, being Lettuce-romaine-6wk, class 12 the one with less pixels, 916, where max-entropy and BALD reach the best OA with lower standard deviation than MI. Moreover, although BALD reaches a lower OA than the random function, it obtains better AA, so it can generalize better. Looking at Fig. 5, the second column shows the OAs and execution times of SV at different  $D_{train}$  sizes. BT-criterion and MI stand out with 100 training samples, being the slowest methods, where random and max-entropy are the fastest acquisition functions.

Finally, in KSC scene the BT-criterion reaches the best OA with good generalization power, achieving 95.56% and 93.20%, respectively, with 15.85% of the ground-truth as training pixels and standing out with less than 100 training samples. On the other part, the slowest functions are those based on distances, i.e. the BT-criterion and MI, while random and mean STD are the fastest ones.

2) *Experiment 2: Performance of different acquisition functions with the 1D B-CNN:* Our second experiment performs a comparison between all the acquisition functions for the proposed 1D B-CNN model, whose architecture is described in Table II. In this case, 300 Monte-Carlo iterations ( $T = 300$ ) have been implemented, while  $D_{train}^0$  and  $D_{pool}^0$  follow the same initialization described in MLR experiment.

Table V shows the obtained results in the 80-th iteration of the model ( $\epsilon = 80$ ), with 10 acquisitions at each iteration. Focusing on the IP scene, we can observe that the 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  $D_{train}$  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 very similar performance.

For the SV dataset, 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,

TABLE VI  
AL-BASED CLASSIFICATION RESULTS OBTAINED BY THE 2D B-CNN FOR DIFFERENT ACQUISITION FUNCTIONS AFTER 80 ITERATIONS AND 10 ACQUISITIONS PER ITERATION.

Class id.	Indian Pines					
	RANDOM	BALD	MAXENT	BT	MI	MEANSTD
0	93.48 (6.52)	98.91 (1.09)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	97.83 (0.00)
1	94.40 (0.63)	99.86 (0.07)	99.72 (0.28)	99.86 (0.14)	<b>100.00 (0.00)</b>	96.81 (1.79)
2	92.17 (0.84)	98.19 (1.20)	99.76 (0.12)	99.64 (0.24)	<b>99.82 (0.18)</b>	88.55 (4.70)
3	97.47 (1.69)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	99.58 (0.42)	91.56 (0.00)
4	90.79 (1.14)	99.38 (0.62)	75.36 (11.59)	<b>99.69 (0.31)</b>	98.55 (1.45)	89.23 (0.41)
5	95.27 (0.48)	99.04 (0.41)	<b>100.00 (0.00)</b>	99.93 (0.07)	99.79 (0.21)	97.53 (1.92)
6	96.43 (3.57)	100.00 (0.00)	100.00 (0.00)	91.07 (8.93)	100.00 (0.00)	96.43 (3.57)
7	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	99.79 (0.21)
8	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	97.30 (2.50)
9	89.87 (1.08)	94.29 (3.55)	98.79 (1.08)	99.49 (0.51)	<b>99.54 (0.05)</b>	89.92 (1.75)
10	97.13 (0.61)	95.97 (1.71)	99.39 (0.37)	98.98 (0.86)	<b>99.98 (0.02)</b>	96.37 (1.75)
11	95.03 (0.76)	<b>99.92 (0.08)</b>	99.75 (0.08)	98.65 (0.51)	99.33 (0.67)	92.16 (1.94)
12	99.76 (0.24)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	98.54 (0.49)
13	99.29 (0.71)	97.91 (2.09)	99.96 (0.04)	<b>100.00 (0.00)</b>	99.01 (0.99)	97.91 (0.36)
14	98.45 (0.51)	100.00 (0.00)	100.00 (0.00)	98.45 (1.55)	100.00 (0.00)	97.02 (0.39)
15	94.09 (4.84)	<b>100.00 (0.00)</b>	97.85 (1.08)	98.39 (1.61)	99.46 (0.54)	95.16 (2.69)
OA	95.58 (0.41)	97.96 (0.35)	98.46 (0.39)	99.46 (0.36)	<b>99.68 (0.18)</b>	94.99 (0.76)
AA	95.85 (0.19)	98.97 (0.45)	98.15 (0.58)	99.01 (0.68)	<b>99.69 (0.17)</b>	95.14 (0.14)
K	95.58 (0.47)	97.96 (0.40)	98.46 (0.44)	99.46 (0.41)	<b>99.68 (0.20)</b>	94.99 (0.86)
Time (seconds)	<b>1434.98 (24.12)</b>	2731.62 (48.75)	2077.61 (37.93)	2051.92 (26.48)	2065.19 (26.50)	2080.78 (24.13)
Class id.	Salinas Valley					
	RANDOM	BALD	MAXENT	BT	MI	MEANSTD
0	99.85 (0.15)	<b>100.00 (0.00)</b>	99.95 (0.05)	99.55 (0.45)	99.10 (0.90)	97.49 (0.27)
1	94.15 (1.45)	96.01 (2.51)	<b>99.95 (0.05)</b>	99.72 (0.28)	96.14 (1.18)	90.77 (0.19)
2	99.62 (0.03)	95.32 (4.68)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	98.36 (0.38)
3	99.86 (0.14)	99.96 (0.04)	100.00 (0.00)	99.89 (0.11)	99.78 (0.22)	100.00 (0.00)
4	99.79 (0.06)	99.98 (0.02)	100.00 (0.00)	100.00 (0.00)	99.93 (0.04)	99.07 (0.78)
5	99.73 (0.21)	100.00 (0.00)	99.95 (0.05)	100.00 (0.00)	100.00 (0.00)	99.44 (0.33)
6	99.09 (0.15)	98.39 (0.68)	100.00 (0.00)	100.00 (0.00)	99.94 (0.03)	99.57 (2.46)
7	92.31 (1.01)	98.10 (1.39)	99.25 (0.51)	<b>99.97 (0.03)</b>	99.53 (0.38)	88.20 (0.69)
8	99.84 (0.06)	99.66 (0.34)	99.95 (0.05)	100.00 (0.00)	100.00 (0.00)	97.85 (1.39)
9	96.19 (2.81)	100.00 (0.00)	100.00 (0.00)	99.82 (0.18)	100.00 (0.00)	96.92 (1.83)
10	96.82 (0.84)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	99.91 (0.09)	96.96 (0.80)
11	99.82 (0.18)	<b>99.97 (0.03)</b>	99.84 (0.16)	99.92 (0.08)	99.87 (0.13)	98.75 (0.99)
12	98.42 (1.15)	100.00 (0.00)	99.95 (0.05)	100.00 (0.00)	100.00 (0.00)	98.69 (1.31)
13	96.82 (0.00)	100.00 (0.00)	100.00 (0.00)	99.63 (0.09)	100.00 (0.00)	95.33 (3.18)
14	84.74 (1.35)	96.35 (3.61)	94.50 (1.33)	<b>99.64 (0.30)</b>	97.61 (2.25)	90.54 (1.33)
15	85.78 (0.39)	98.01 (0.94)	99.64 (0.36)	99.81 (0.19)	<b>100.00 (0.00)</b>	93.05 (1.74)
OA	94.95 (0.07)	98.45 (0.52)	99.07 (0.05)	<b>99.88 (0.08)</b>	99.26 (0.16)	94.35 (0.30)
AA	96.43 (0.23)	98.86 (0.56)	99.56 (0.03)	<b>99.87 (0.07)</b>	99.49 (0.07)	96.18 (0.05)
K	94.95 (0.08)	98.45 (0.58)	99.07 (0.06)	<b>99.88 (0.09)</b>	99.26 (0.18)	94.35 (0.33)
Time (seconds)	<b>2312.63 (17.19)</b>	3625.83 (64.68)	2963.12 (23.25)	2956.60 (18.43)	3071.39 (28.29)	2986.14 (24.65)
Class id.	Kennedy Space Center					
	RANDOM	BALD	MAXENT	BT	MI	MEANSTD
0	95.93 (0.53)	98.55 (0.13)	99.21 (0.66)	<b>99.80 (0.20)</b>	99.47 (0.39)	96.98 (1.05)
1	95.93 (0.53)	100.00 (0.00)	99.79 (0.21)	99.38 (0.62)	99.38 (0.62)	87.65 (0.41)
2	86.72 (0.00)	<b>100.00 (0.00)</b>	98.44 (1.56)	99.80 (0.20)	99.61 (0.39)	92.19 (3.91)
3	89.29 (0.40)	100.00 (0.00)	99.80 (0.20)	99.40 (0.60)	100.00 (0.00)	93.85 (1.79)
4	97.83 (2.17)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)
5	94.10 (3.28)	100.00 (0.00)	99.56 (0.44)	100.00 (0.00)	100.00 (0.00)	91.70 (2.62)
6	78.57 (0.48)	<b>99.52 (0.48)</b>	99.05 (0.00)	99.05 (0.00)	97.14 (0.95)	70.95 (17.62)
7	89.41 (1.04)	<b>100.00 (0.00)</b>	99.88 (0.12)	99.54 (0.00)	99.77 (0.23)	90.26 (5.57)
8	94.54 (0.38)	<b>100.00 (0.00)</b>	99.23 (0.58)	<b>100.00 (0.00)</b>	99.62 (0.38)	97.02 (0.48)
9	96.91 (1.86)	100.00 (0.00)	100.00 (0.00)	99.88 (0.12)	99.88 (0.12)	95.54 (1.24)
10	98.21 (0.12)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	98.81 (1.19)
11	94.73 (3.28)	99.90 (0.10)	100.00 (0.00)	99.60 (0.20)	100.00 (0.00)	98.71 (0.89)
12	99.68 (0.32)	100.00 (0.00)	99.89 (0.11)	100.00 (0.00)	98.98 (1.02)	99.73 (0.27)
OA	94.77 (0.47)	99.54 (0.02)	99.64 (0.14)	<b>99.80 (0.01)</b>	99.57 (0.01)	95.63 (0.51)
AA	92.77 (0.41)	99.67 (0.04)	99.60 (0.16)	<b>99.73 (0.06)</b>	99.53 (0.16)	93.34 (1.66)
K	94.77 (0.52)	99.54 (0.02)	99.64 (0.14)	<b>99.80 (0.01)</b>	99.57 (0.01)	95.63 (0.57)
Time (seconds)	<b>2273.24 (13.11)</b>	3520.89 (20.20)	2910.52 (27.14)	2901.06 (20.47)	2913.41 (27.04)	2932.83 (23.41)

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 regards to those obtained for the SV dataset.

With these datasets we also can observe how 1D 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 few samples are used. In this case, the model is not able to obtain good uncertainty values for these acquisition functions due 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 2D B-CNN:* The third experiment performs a

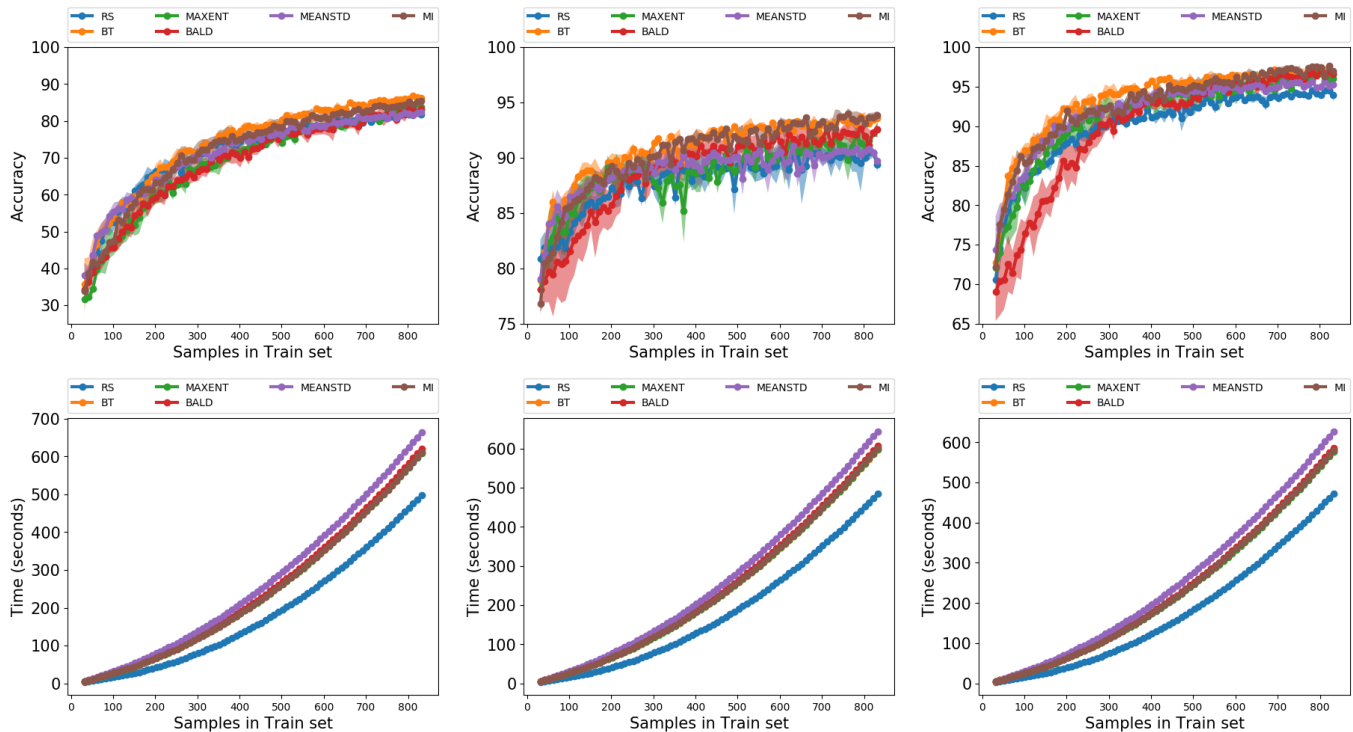


Fig. 6. AL-based performance obtained by the 1D B-CNN for different acquisition functions with different sizes of  $D_{train}$ . First column: results for the IP dataset. Second column: results for the SV dataset. Third column: results for the KSC dataset.

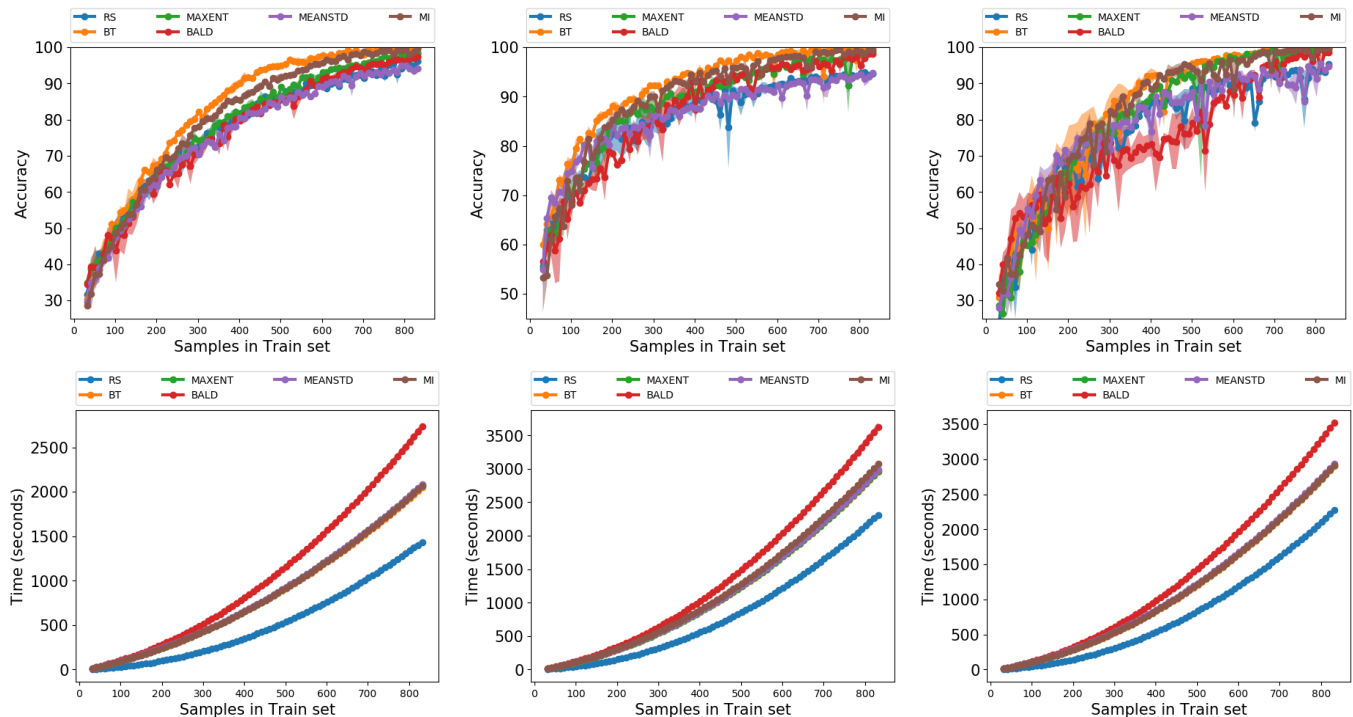


Fig. 7. AL-based performance obtained by the 2D B-CNN for different acquisition functions with different sizes of  $D_{train}$ . First column: results for the IP dataset. Second column: results for the SV dataset. Third column: results for the KSC dataset.

comparison of different acquisition function with the proposed 2D B-CNN model, whose architecture is described in Table II. The parameter  $T$  has been set to 300, and the initialization of  $D_{train}$ ,  $D_{pool}$ , test and validation sets are the same as in our experiments with the 1D B-CNN and MLR.

Table VI shows the obtained results over the three considered hyperspectral datasets at iteration  $\epsilon = 80$ . Focusing on the IP dataset, 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 proposed spatial model with different  $\mathcal{D}_{train}$  sizes. 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.

SV dataset 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 lowest OA results. The execution times are rather similar to IP dataset, being the processing of SV slowest than IP. Again, BALD is the slowest method and random the fastest one, being BT-criterion, max-entropy and MI very similar.

Finally, for the KSC dataset the best OA is reached with BT-criterion as 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. The execution times, on the other hand, are very similar to those achieved in the experiments with the SV dataset.

4) *Experiment 4: Performance of different acquisition functions with the 3D B-CNN:* Our fourth experiment implements the proposed spectral-spatial BCNN classifier using the six considered acquisition functions. Table VII and Fig. 8 show the obtained results.

With the IP dataset, 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 lowest OA. In Fig. 8 we can observe how the BT-criterion stands out with 100 training samples, achieving good results with 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, BALD stands out with 100 training samples, until it is reached by MI and max-entropy. Again, BALD is the slowest method 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 lowest OA (98.99% and 98.56%, respectively). In Fig. 8 we can observe that the BT-criterion is able to reach high OA with 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 experiment performs a comparison

TABLE VII  
AL-BASED CLASSIFICATION RESULTS OBTAINED BY THE 3D B-CNN FOR DIFFERENT ACQUISITION FUNCTIONS AFTER 80 ITERATIONS AND 10 ACQUISITIONS PER ITERATION.

Class id.	Indian Pines					
	RANDOM	BALD	MAXENT	BT	MI	MEANSTD
0	97.83 (2.17)	98.91 (1.09)	100.00 (0.00)	100.00 (0.00)	98.91 (1.09)	96.74 (1.09)
1	96.29 (1.33)	99.51 (0.21)	<b>99.93</b> (0.07)	99.09 (0.77)	97.72 (1.51)	98.49 (0.18)
2	99.82 (0.06)	99.76 (0.24)	99.28 (0.72)	99.64 (0.36)	<b>99.94</b> (0.06)	96.14 (3.86)
3	99.16 (0.84)	100.00 (0.00)	100.00 (0.00)	99.79 (0.21)	99.79 (0.21)	98.31 (1.27)
4	96.17 (1.76)	100.00 (0.00)	99.79 (0.21)	99.59 (0.41)	100.00 (0.00)	96.69 (1.66)
5	99.59 (0.41)	99.73 (0.27)	99.93 (0.07)	<b>100.00</b> (0.00)	99.93 (0.07)	99.59 (0.14)
6	91.07 (8.93)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	98.21 (1.79)	100.00 (0.00)
7	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)
8	85.00 (15.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	97.50 (2.50)
9	96.60 (1.24)	99.69 (0.31)	<b>99.95</b> (0.05)	99.90 (0.10)	99.69 (0.31)	96.66 (0.46)
10	98.90 (0.16)	99.98 (0.02)	98.90 (0.98)	<b>100.00</b> (0.00)	99.86 (0.06)	98.31 (0.75)
11	96.12 (0.51)	99.92 (0.08)	99.83 (0.17)	99.83 (0.17)	99.92 (0.08)	97.72 (1.26)
12	99.51 (0.49)	100.00 (0.00)	100.00 (0.00)	98.78 (1.22)	100.00 (0.00)	98.29 (1.71)
13	99.80 (0.04)	99.64 (0.36)	100.00 (0.00)	99.96 (0.04)	100.00 (0.00)	99.25 (0.75)
14	95.73 (4.27)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	99.87 (0.13)	94.17 (4.27)
15	96.77 (0.00)	97.31 (1.61)	<b>98.92</b> (0.00)	95.16 (2.69)	95.70 (1.08)	73.12 (26.88)
OA	98.14 (0.00)	<b>99.78</b> (0.07)	99.63 (0.29)	99.73 (0.06)	99.55 (0.18)	97.79 (0.41)
AA	96.77 (1.68)	99.65 (0.20)	<b>99.78</b> (0.11)	99.48 (0.10)	99.35 (0.30)	96.31 (0.68)
K	98.14 (0.01)	<b>99.78</b> (0.08)	99.63 (0.33)	99.73 (0.07)	99.55 (0.21)	97.79 (0.47)
Time (seconds)	<b>1265.17</b> (8.06)	4497.65 (15.16)	3328.13 (25.08)	3331.44 (28.67)	3304.41 (10.91)	3325.62 (10.61)
Class id.	Salinas Valley					
	RANDOM	BALD	MAXENT	BT	MI	MEANSTD
0	99.98 (0.02)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)
1	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)
2	100.00 (0.00)	99.90 (0.10)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	99.97 (0.03)
3	97.49 (2.22)	99.89 (0.04)	<b>99.96</b> (0.04)	99.61 (0.18)	99.89 (0.04)	99.32 (0.61)
4	99.07 (0.93)	<b>99.96</b> (0.00)	99.91 (0.02)	99.91 (0.06)	99.93 (0.04)	99.51 (0.15)
5	99.55 (0.45)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	98.28 (1.72)
6	99.90 (0.10)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)
7	91.98 (6.45)	100.00 (0.00)	100.00 (0.00)	99.88 (0.12)	100.00 (0.00)	99.37 (0.44)
8	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	99.98 (0.02)	100.00 (0.00)
9	99.44 (0.50)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	99.79 (0.21)	99.27 (0.34)
10	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	99.95 (0.05)
11	99.87 (0.03)	99.51 (0.49)	100.00 (0.00)	99.97 (0.03)	100.00 (0.00)	100.00 (0.00)
12	99.51 (0.49)	100.00 (0.00)	100.00 (0.00)	99.95 (0.05)	100.00 (0.00)	99.84 (0.05)
13	94.07 (5.84)	99.95 (0.05)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	97.62 (2.01)
14	98.50 (4.60)	99.99 (0.00)	99.99 (0.01)	99.66 (0.34)	<b>100.00</b> (0.00)	93.95 (0.59)
15	99.11 (0.89)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	99.06 (0.22)
OA	97.25 (0.90)	99.97 (0.02)	<b>99.99</b> (0.00)	99.91 (0.08)	99.98 (0.01)	98.76 (0.23)
AA	98.40 (0.66)	99.95 (0.04)	<b>99.99</b> (0.00)	99.94 (0.05)	99.97 (0.01)	99.13 (0.02)
K	97.25 (0.99)	99.97 (0.03)	<b>99.99</b> (0.00)	99.91 (0.09)	99.98 (0.02)	98.76 (0.25)
Time (seconds)	<b>1405.82</b> (10.05)	4662.36 (55.90)	3465.04 (15.60)	3444.14 (20.19)	3445.91 (15.95)	3467.78 (20.80)
Class id.	Kennedy Space Center					
	RANDOM	BALD	MAXENT	BT	MI	MEANSTD
0	98.49 (0.73)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	99.41 (0.20)
1	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	97.53 (0.82)
2	94.53 (2.34)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	93.75 (0.78)
3	95.63 (3.97)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	93.65 (0.40)
4	99.69 (0.31)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	95.03 (1.86)
5	96.94 (0.87)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	92.79 (1.09)
6	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)
7	99.65 (0.35)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)
8	99.81 (0.19)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	98.65 (0.19)
9	98.64 (1.36)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)
10	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)
11	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	99.80 (0.20)
12	99.95 (0.05)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)
OA	98.99 (0.47)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	98.56 (0.06)
AA	98.72 (0.55)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	97.74 (0.15)
K	98.99 (0.52)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	98.56 (0.06)
Time (seconds)	<b>1273.81</b> (7.18)	4533.24 (14.71)	3327.97 (11.50)	3325.30 (11.48)	3327.47 (11.27)	3348.26 (9.04)

between the AL implementations described in previous subsections, with the best OA values for each hyperspectral dataset, with traditional classifiers. For the IP dataset, the AL-MLR and the 1D B-CNN with BT-criterion, 2D B-CNN with MI criterion and 3D B-CNN with BALD criterion have been selected to be compared with the traditional random forest (RF), multilayer perceptron (MLP), support vector machine (SVM) and MLR classifiers, and also with the standard 1D-CNN, 2D-CNN and 3D-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, 1D-CNN, 2D-CNN and 3D-CNN baselines, only two pixels per class have been selected, while the remaining 800 pixels (the selected maximum size of  $\mathcal{D}_{train}$ ) have been randomly selected. This process has been repeated with the SV dataset, selecting the AL-MLR and 2D B-CNN with BT-criterion, 1D B-CNN with MI and 3D B-CNN with max-entropy, and with the KSC dataset, selecting AL-MLR, 2D and 3D B-CNN with BT-criterion and 1D B-CNN with MI.

Focusing on the IP dataset, we can observe the performance of pixel-wise 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,

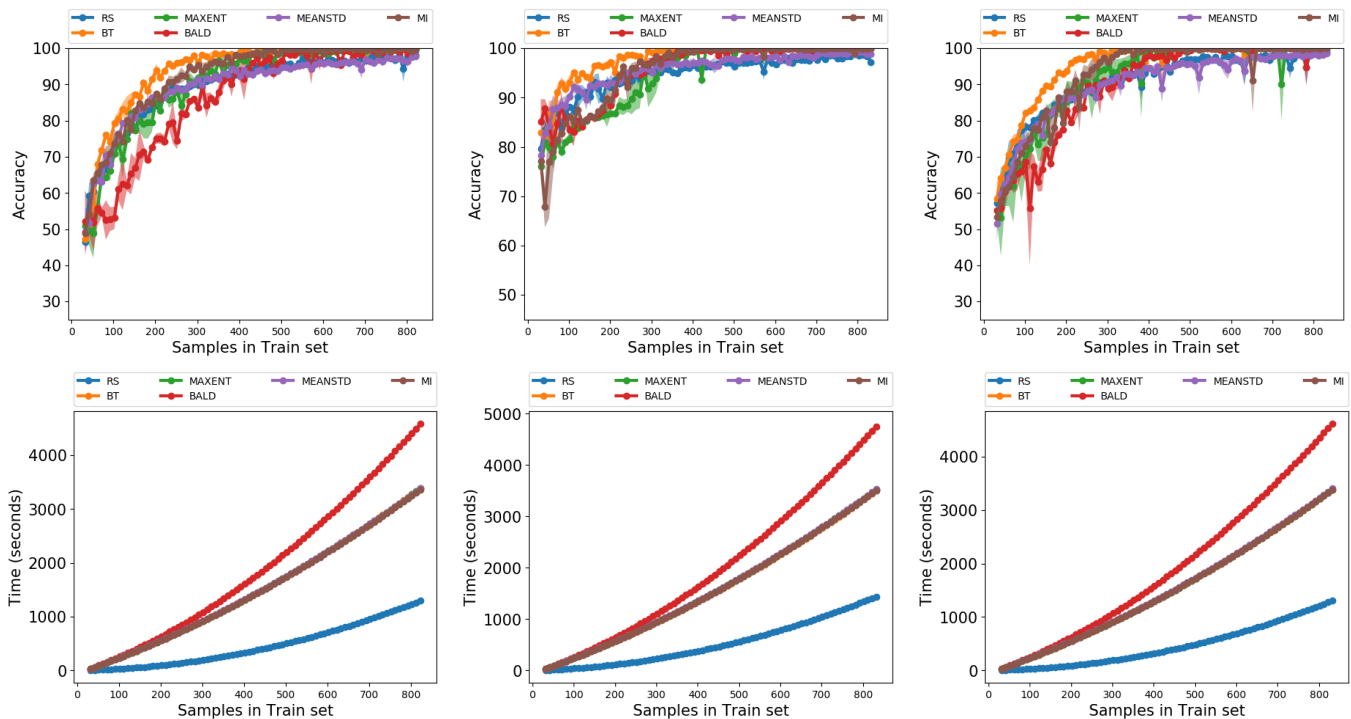


Fig. 8. AL-based performance obtained by the 3D B-CNN for different acquisition functions with different sizes of  $D_{train}$ . First column: results for the IP dataset. Second column: results for the SV dataset. Third column: results for the KSC dataset.

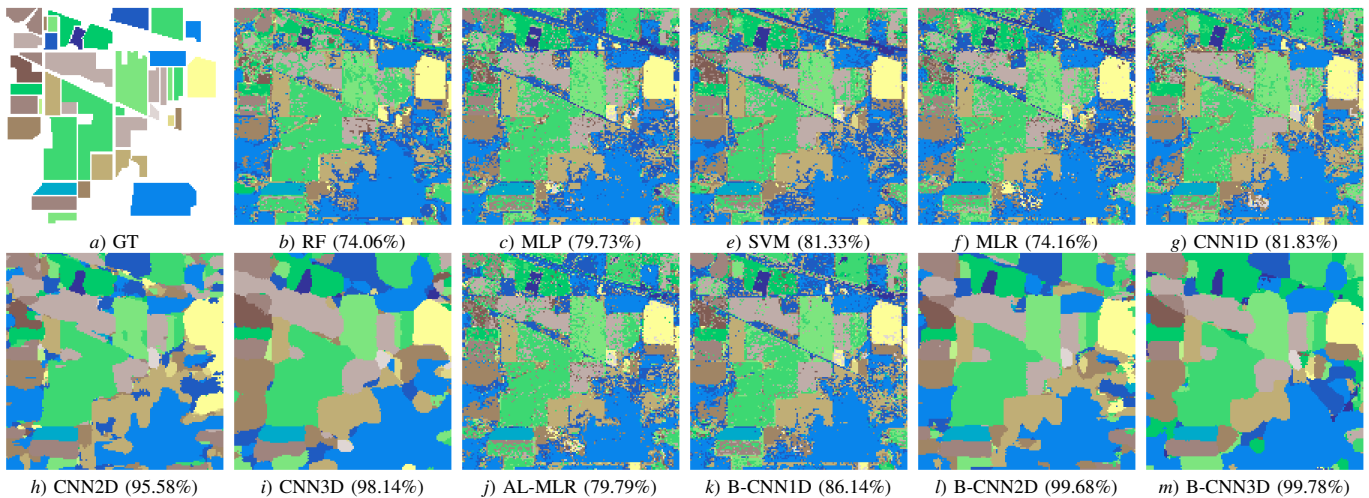


Fig. 9. Classification maps for the Indian Pines (IP) dataset. The first image (a) contains the ground-truth classification map. Finally, images from (b) to (m) provide the classification maps corresponding to Table VIII. Note that the overall classification accuracies are shown in brackets.

while the spectral B-CNN model improves the classification results over the baseline 1D-CNN and the other pixel-wise methods. Looking at spatial classifiers, the 2D B-CNN is able to outperform the 2D-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 3D-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, 1D B-CNN, 2D B-CNN and 3D B-CNN with BT-criterion for

the IP dataset. We can see that the spectral-spatial B-CNN is able to reach good classification accuracy with fewer training samples than the other AL-based classifiers, although it is the slowest method. Moreover, in Table IX 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 less training data, only 402 samples (i.e. the 3.92% of the ground-truth) to reach 99% OA.

The results for the SV dataset are similar. In Table VIII we can see that the pixel-wise 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 1D-CNN baseline. Also spatial B-CNN outperforms the 2D-CNN



TABLE VIII  
CLASSIFICATION RESULTS OBTAINED BY AL-BASED PRESENTED METHODOLOGIES IN COMPARISON WITH THOSE OBTAINED WITH TRADITIONAL  
HYPERSPETRAL 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	83.53 (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 (0.99)	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)	

baseline, being around 4.93 perceptual points better. Finally, spectral-spatial B-CNN classifier is much better than 3D-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 2D B-CNN. In table IX we can see that spectral-spatial B-CNN needs less training data than the other classifier in order to reach the 99% of accuracy.

The results obtained for the KSC dataset are also quite similar to those obtained for the SV dataset. In Table VIII we can see that the pixel-wise 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 2D-CNN, while the spectral-spatial B-CNN also outperforms the 3D-CNN baseline. These classification results can be observed in graphical form in Fig. 11. In Fig. 12 we can observe in the third column the implemented AL-based methods with BT-criterion as acquisition function over the KSC dataset. As we can see, the spectral-spatial B-CNN is able to reach good values with few training samples, 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 IX.

#### IV. CONCLUSIONS AND FUTURE LINES

In this paper, we have developed a new active learning model with Bayesian convolutional neural networks for hyperspectral image classification using spectral, spatial and

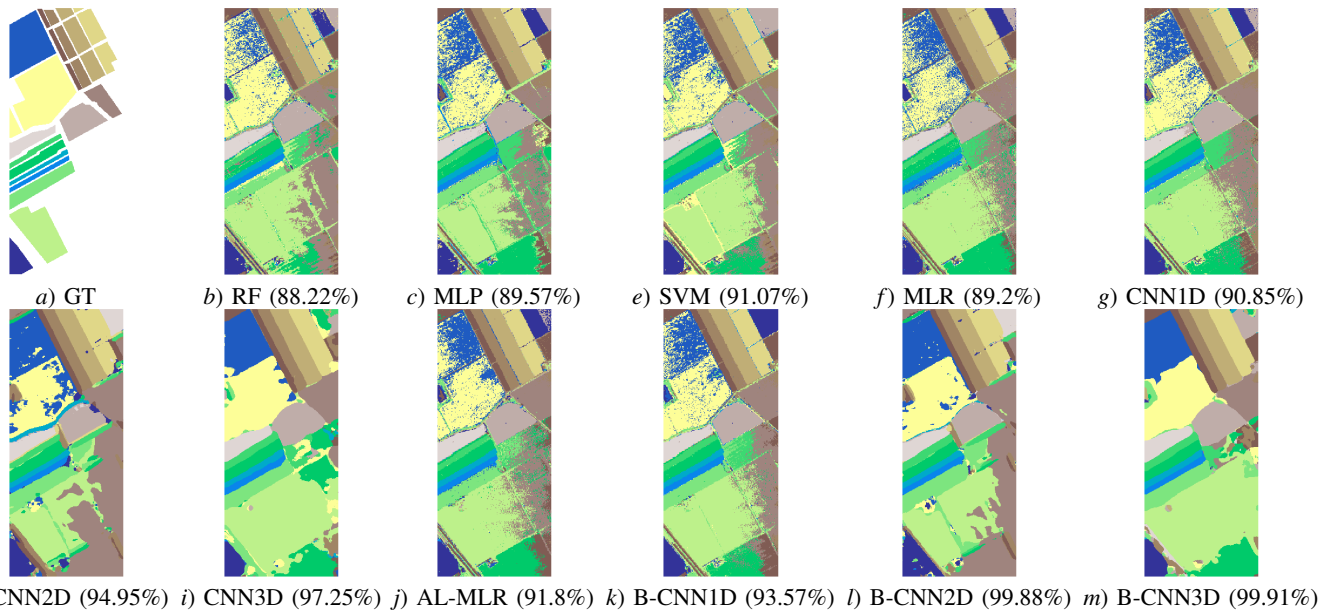


Fig. 10. Classification maps for the Salinas Valley (SV) dataset. The first image contains the ground-truth classification map. Finally, images from (b) to (m) provide the classification maps corresponding to Table VIII. Note that the overall classification accuracies are shown in brackets.

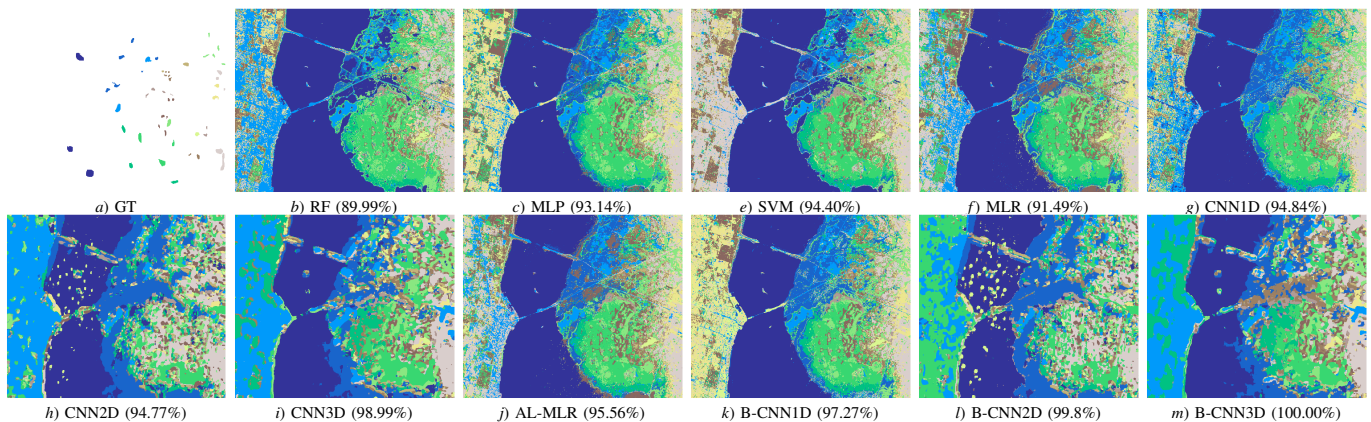


Fig. 11. Classification maps for the Kennedy Space Center (KSC) dataset. The first image (a) represents the ground-truth classification map. Finally, images from (b) to (m) provide the classification maps corresponding to Table VIII. Note that the overall classification accuracies are shown in brackets.

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 active learning is combined with convolutional neural networks (via Bayesian neural networks) to perform robust hyperspectral image classification with very limited training sets. In our work, we report very high classification accuracies using very limited labeled samples, avoiding the curse of dimensionality and the overfitting problems introduced by these kind of networks. Our results also indicate that, by a proper selection of the acquisition function, active learning 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 [103]. Finally, the inclusion of post-processing methods, such as conditional random field (CRF) [104], will be also studied in future developments as a way to improve and smooth the classification maps obtained by the different tested methods.

## 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, Heidelberg: Springer Berlin Heidelberg, 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 *12th Canadian Symposium on Remote Sensing Geoscience and Remote Sensing Symposium*, 1989, pp. 942–943.

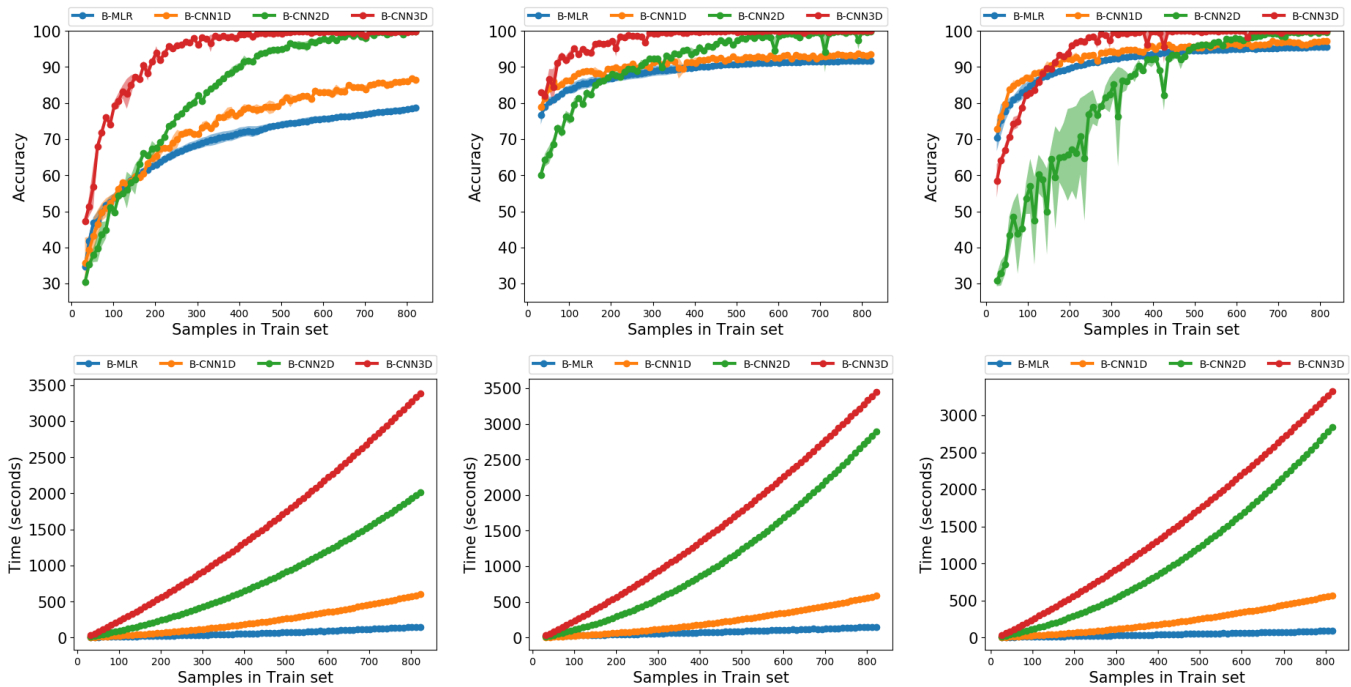


Fig. 12. Comparison of the BT-criterion over spectral active learning multinomial logistic regression and spectral, spatial, spectral-spatial bayesian convolutional neural networks with the IP (first column), SV (second column) and KSC (third column) hyperspectral datasets.

TABLE IX

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	—	—	—	—	—
AL-CNN1D	252	352	502	662	—	—	—
AL-CNN2D	222	252	292	352	402	512	662
AL-CNN3D	72	82	112	152	172	232	402
Algorithm	Salinas Valley						
	70	75	80	85	90	95	99
MLR	32	32	52	132	412	—	—
AL-CNN1D	32	32	42	62	232	—	—
AL-CNN2D	72	92	122	162	272	412	622
AL-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	—
AL-CNN1D	26	36	56	76	156	386	—
AL-CNN2D	226	246	286	306	366	496	666
AL-CNN3D	56	86	96	126	166	206	276

- [4] R. O. Green, M. L. Eastwood, C. M. Sarture, T. G. Chrien, M. Aronsson, B. J. Chippendale, J. A. Faust, B. E. Pavri, C. J. Chovit, M. Solis, M. R. Olah, and O. Williams, "Imaging spectroscopy and the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS)," *Remote Sensing of Environment*, vol. 65, no. 3, pp. 227–248, 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 *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, July 2016, pp. 4532–4534.
- [6] J. Chen, S. Leblanc, J. Miller, J. Freemantle, S. E. Loechel, C. Walthall,

K. Innanen, and H. White, "Compact airborne spectrographic imager (casi) used for mapping biophysical parameters of boreal forests," *Journal of Geophysical Research*, V, vol. 104, pp. 27,94527,958, 11 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," in *Proc. SPIE 0868 Optoelectronic technologies for remote sensing from space*, J. Seeley and S. Bowyer, Eds., 1988, p. 8.
- [8] E. Bedini, F. van der Meer, and F. van Ruitenbeek, "Use of hynam imaging spectrometer data to map mineralogy in the rodalquilar caldera, southeast spain," *International Journal of Remote Sensing*, 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," in *Proc. SPIE 8360 Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications IX*, 2012, p. 10.
- [10] I. Vorovencii, "The Hyperspectral Sensors used in Satellite and Aerial Remote Sensing," *Bulletin of the Transilvania University of Braşov*, vol. 2, no. 51, 2009.
- [11] R. C. Olsen, *Remote sensing from air and space*. Bellingham, Washington USA: SPIE press, 2007.
- [12] S. Yarbrough, T. R. Caudill, E. T. Kouba, V. Osweiler, J. Arnold, R. Quarles, J. Russell, L. J. Otten, B. A. Jones, A. Edwards, J. Lane, A. D. Meigs, R. B. Lockwood, and P. S. Armstrong, "MightySat II.1 hyperspectral imager: summary of on-orbit performance7510 a currently with Ball Aerospace and Technology Corporation2649 b currently with Rio Grande Medical Technologies," in *Proc. SPIE 4480, Imaging Spectrometry VII*, 2002, p. 12.
- [13] C.-I. Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. Springer US, 2003.
- [14] P. Ghamisi, J. Plaza, Y. Chen, J. Li, and A. Plaza, "Advanced Spectral Classifiers for Hyperspectral Images: A Review," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 1, pp. 8–32, 2017.
- [15] M. Teke, H. S. Deveci, O. Haliloğlu, S. Zübeyde Gürbüz, and U. Sakarya, "A Short Survey of Hyperspectral Remote Sensing Applications in Agriculture," in *Recent Advances in Space Technologies (RAST)*, 2013.
- [16] A. Plaza, J. Plaza, A. Paz, and S. Sanchez, "Parallel Hyperspectral Image and Signal Processing," *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 119–126, 2011.

- [17] X. Lu, X. Li, and L. Mou, "Semi-supervised multitask learning for scene recognition," *IEEE Transactions on Cybernetics*, vol. 45, no. 9, pp. 1967–1976, Sept 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 (HypSIRI)*. Dordrecht: Springer Netherlands, 2013, pp. 117–130.
- [20] H. Kaufmann, L. Guanter, K. Segl, S. Hofer, K.-P. Foerster, T. Stuffer, A. Mueller, R. Richter, H. Bach, and P. Hostert, "Environmental Mapping and Analysis Program (EnMAP) Recent Advances and Status," *IEEE International Geoscience & Remote Sensing Symposium, IGARSS*, vol. 4, pp. 109–112, 2008.
- [21] C. Galeazzi, A. Sacchetti, A. Cisbani, and G. Babini, "The PRISMA Program," in *IGARSS 2008 - 2008 IEEE International Geoscience and Remote Sensing Symposium*, 2008, pp. IV – 105–IV – 108.
- [22] A. Plaza, J. A. Benediktsson, J. W. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, A. Gualtieri, M. Marconcini, J. C. Tilton, and G. Trianni, "Recent advances in techniques for hyperspectral image processing," *Remote Sensing of Environment*, vol. 113, pp. S110 – S122, 2009, imaging Spectroscopy Special Issue. [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," *Transactions in GIS*, vol. 20, no. 4, pp. 463–490, 2016.
- [24] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.
- [25] J. Haut, M. Paoletti, J. Plaza, and A. Plaza, "Cloud implementation of the K-means algorithm for hyperspectral image analysis," *Journal of Supercomputing*, vol. 73, no. 1, 2017.
- [26] Y. Tarabalka, J. A. Benediktsson, and J. Chanussot, "SpectralSpatial Classification of Hyperspectral Imagery Based on Partitional Clustering Techniques," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 8, pp. 2973–2987, 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 classification*. Stanford Research Institute, 1965.
- [28] M. Paoletti, J. Haut, J. Plaza, and A. Plaza, "Yinyang K-means clustering for hyperspectral image analysis," in *Proceedings of the 17th International Conference on Computational and Mathematical Methods in Science and Engineering*, J. Vigo-Aguiar, Ed., Rota, 2017, pp. 1625–1636.
- [29] C. M. Bishop, *Neural Networks for Pattern Recognition*. Clarendon Press, 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," *International Journal of Remote Sensing*, vol. 18, no. 4, pp. 699–709, 1997. [Online]. Available: <http://dx.doi.org/10.1080/014311697218700>
- [31] J. A. Benediktsson, P. H. Swain, and O. K. Ersoy, "Conjugate gradient neural networks in classification of very high dimensional remote sensing data," *International Journal of Remote Sensing*, 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," *International Journal of Remote Sensing*, vol. 20, no. 1, pp. 97–110, 1999. [Online]. Available: <http://dx.doi.org/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, PhD thesis, Purdue Univ., School of Elect. Eng. West Lafayette, IN, 1990.
- [34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [35] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, p. 436444, 2015.
- [36] L. Deng and D. Yu, "Deep Learning: Methods and Applications," *Foundations and Trends® in Signal Processing*, vol. 7, no. 34, pp. 197–387, 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 Transactions on Geoscience and Remote Sensing*, vol. 54, no. 10, pp. 6232–6251, 2016. [Online]. Available: <http://ieeexplore.ieee.org/document/7514991/>
- [38] K. Makantasis, K. Karantzalos, A. Doulamis, and N. Doulamis, "Deep supervised learning for hyperspectral data classification through convolutional neural networks," in *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2015, pp. 4959–4962.
- [39] Y. Bengio, "Learning Deep Architectures for AI," *Machine Learning*, 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, 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. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, 2006.
- [43] Y. Chen, X. Zhao, and X. Jia, "Spectral-Spatial Classification of Hyperspectral Data Based on Deep Belief Network," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 6, pp. 2381–2392, 2015.
- [44] T. Li, J. Zhang, and Y. Zhang, "Classification of hyperspectral image based on deep belief networks," in *Proc. IEEE Int. Conf. Image Proces.*, 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, R. Boyle, B. Parvin, D. Koracin, I. Pavlidis, R. Feris, T. McGraw, M. Elendt, R. Kopper, E. Ragan, Z. Ye, and G. Weber, Eds. Cham: Springer International Publishing, 2015, pp. 682–692.
- [46] P. Zhong, Z. Gong, S. Li, and C. B. Schnlieb, "Learning to diversify deep belief networks for hyperspectral image classification," *IEEE Trans Geosci. 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 *2014 6th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, 2014, pp. 1–4.
- [48] J. Li, L. Bruzzone, and S. Liu, "Deep feature representation for hyperspectral image classification," in *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2015, pp. 4951–4954.
- [49] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep Learning-Based Classification of Hyperspectral Data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2094–2107, 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 Geoscience and Remote Sensing Letters*, vol. 12, no. 12, pp. 2438–2442, 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 Geoscience and Remote Sensing Letters*, vol. 11, no. 10, pp. 1797–1801, 2014.
- [52] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep recurrent neural networks for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 3639–3655, 2017.
- [53] Z. Zuo, B. Shuai, G. Wang, X. Liu, X. Wang, B. Wang, and Y. Chen, "Convolutional recurrent neural networks: Learning spatial dependencies for image representation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2015, pp. 18–26.
- [54] H. Wu and S. Prasad, "Convolutional Recurrent Neural Networks for Hyperspectral Data Classification," *Remote Sensing*, vol. 9, no. 3, p. 298, 2017.
- [55] —, "Semi-supervised deep learning using pseudo labels for hyperspectral image classification," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1259–1270, March 2018.
- [56] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *Journal of Sensors*, 2015.
- [57] S. Yu, S. Jia, and C. Xu, "Convolutional neural networks for hyperspectral image classification," *Neurocomputing*, vol. 219, pp. 88–98, 1 2017. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S09252321216310104>
- [58] Z. Zheng, Y. Zhang, L. Li, M. Zhu, Y. He, M. Li, Z. Guo, Y. He, Z. Yu, X. Yang, X. Liu, J. Luo, T. Yang, Y. Liu, and J. Li, "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 *IEEE Int. Geosci. Remote Sens. Symp.*, 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, Jan. 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 Transactions on Geoscience and Remote Sensing*, vol. PP, no. 99, pp. 1–12, 2017.
- [63] Y. Li, H. Zhang, and Q. Shen, "Spectralspatial classification of hyperspectral imagery with 3d convolutional neural network," *Remote Sens.*, vol. 9, 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 Journal of Photogrammetry and Remote Sensing*, 2017.
- [65] L. Breiman, "Heuristics of instability and stabilization in model selection," *Annals of Statistics*, vol. 24, no. 6, 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 Sensing of Environment*, vol. 103, no. 2, pp. 179–189, 2006.
- [67] M. Khodadadzadeh, J. Li, A. Plaza, H. Ghassemian, J. M. Bioucas-Dias, and X. Li, "SpectralSpatial Classification of Hyperspectral Data Using Local and Global Probabilities for Mixed Pixel Characterization," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 10, pp. 6298–6314, 2014.
- [68] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE Transactions on Information Theory*, vol. 14, no. 1, pp. 55–63, January 1968.
- [69] C.-I. Chang, *Hyperspectral data exploitation : theory and applications*, C.-I. Chang, Ed. Wiley-Interscience, 2007.
- [70] D. J. MacKay, "Information-Based Objective Functions for Active Data Selection," *Neural Computation*, vol. 4, no. 4, pp. 590–604, 1992.
- [71] D. Tuia, F. Ratle, F. Pacifici, M. F. Kanevski, and W. J. Emery, "Active Learning Methods for Remote Sensing Image Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 7, pp. 2218–2232, 2009.
- [72] S. Rajan, J. Ghosh, and M. M. Crawford, "An Active Learning Approach to Hyperspectral Data Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 4, pp. 1231–1242, 2008.
- [73] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Semisupervised Hyperspectral Image Segmentation Using Multinomial Logistic Regression With Active Learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 11, pp. 4085–4098, 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 Journal of Selected Topics in Signal Processing*, vol. 5, no. 3, pp. 606–617, 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 Transactions on Geoscience and Remote Sensing*, vol. 51, no. 2, pp. 844–856, 2013.
- [76] C. M. Bishop, "Bayesian neural networks," *Journal of the Brazilian Computer Society*, vol. 4, no. 1, 1997.
- [77] D. J. C. MacKay, "A Practical Bayesian Framework for Backpropagation Networks," *Neural Computation*, vol. 4, no. 3, pp. 448–472, 1992.
- [78] R. M. Neal, *Bayesian Learning for Neural Networks*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1996.
- [79] Y. Gal, "Uncertainty in Deep Learning," Ph.D. dissertation, University of Cambridge, 2016.
- [80] J. S. Denker and Y. leCun, "Transforming neural-net output levels to probability distributions," in *Proceedings of the 1990 Conference on Advances in Neural Information Processing Systems 3*, ser. NIPS-3. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, pp. 853–859.
- [81] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," *CoRR*, 2015.
- [82] R. Islam, "Active Learning for High Dimensional Inputs using Bayesian Convolutional Neural Networks," Ph.D. dissertation, University of Cambridge, 2016.
- [83] Y. Gal and Z. Ghahramani, "Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference," *CoRR*, vol. abs/1506.02158, 2015.
- [84] L. Zhang, L. Zhang, and B. Du, "Deep Learning for Remote Sensing Data Advances in Machine Learning for Remote Sensing and Geosciences," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 22–40, 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 *ICCV*. IEEE, 2009, pp. 2146–2153.
- [86] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, Johannes Fürnkranz and Thorsten Joachims, Ed. Omnipress, 2010, pp. 807–814.
- [87] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, Geoffrey J. Gordon and David B. Dunson, Ed. Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011, pp. 315–323.
- [88] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight Uncertainty in Neural Networks," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning*, vol. 37. Lille, France: JMLR.org, 2015, pp. 1613–1622.
- [89] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," *CoRR*, vol. abs/1703.02910, 2017.
- [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 *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2016, pp. 680–688.
- [91] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 26, no. 4, pp. 623–656, 1948.
- [92] N. Houlsby, F. Huszár, and Z. Ghahramani, "Bayesian Active Learning for Classification and Preference Learning," *ArXiv e-prints*, 2011.
- [93] T. Luo, K. Kramer, S. Samson, A. Remsen, D. B. Goldgof, L. O. Hall, and T. Hopkins, "Active learning to recognize multiple types of plankton," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 3, Aug 2004, pp. 478–481 Vol.3.
- [94] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Hyperspectral Image Segmentation Using a New Bayesian Approach With Active Learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 10, pp. 3947–3960, 2011.
- [95] D. J. C. MacKay, "Information-based objective functions for active data selection," *Neural Computation*, vol. 4, no. 4, pp. 590–604, July 1992.
- [96] Q. Liu, R. Hang, H. Song, F. Zhu, J. Plaza, and A. Plaza, "Adaptive Deep Pyramid Matching for Remote Sensing Scene Classification," *CoRR*, vol. abs/1611.03589, 2016. [Online]. Available: <http://arxiv.org/abs/1611.03589>
- [97] W. Zhao and S. Du, "Learning multiscale and deep representations for classifying remotely sensed imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 128, pp. 223–239, 2016.
- [98] 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," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 116, pp. 24–41, 2016.
- [99] Y. Li, H. Zhang, and Q. Shen, "SpectralSpatial Classification of Hyperspectral Imagery with 3D Convolutional Neural Network," *Remote Sensing*, vol. 9, no. 1, p. 67, 2017.
- [100] J. Nocedal, "Updating Quasi-Newton Matrices With Limited Storage," *Math. of Computation*, vol. 35, no. 151, pp. 773–782, 1980.
- [101] J. Haut, M. Paoletti, A. Paz-Gallardo, J. Plaza, and A. Plaza, "Cloud implementation of logistic regression for hyperspectral image classification," in *Proceedings of the 17th International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE 2017*, J. Vigo-Aguiar, Ed., Costa Ballena (Rota), Cádiz, Spain, 2017, pp. 1063–2321.
- [102] D. P. Kingma and J. L. Ba, "ADAM: {A} method for stochastic optimization," *CoRR*, 2014. [Online]. Available: <http://arxiv.org/abs/1412.6980>
- [103] M. Volpi and V. Ferrari, "Semantic segmentation of urban scenes by learning local class interactions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2015, pp. 1–9.
- [104] X. Wei, Y. Guo, X. Gao, M. Yan, and X. Sun, "A new semantic segmentation model for remote sensing images," in *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, July 2017, pp. 1776–1779.