Neighboring Region Dropout for Hyperspectral Image Classification

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

***Abstract*— Deep neural networks (DNNs) exhibit great performance in the task of hyperspectral image (HSI) classification. However, these models are usually overparameterized and require large amounts of training data in order to properly avoid the curse of dimensionality and the variability of spectral signatures, thus suffering from overfitting problems when very few training samples are available, due to poor generalization ability in this particular case. The traditional regularization dropout (DO) strategy has been shown to be effective in fully connected DNNs but not in convolutional-based ones. This is mainly due to the way these architectures manage the spatial information. In this letter, we introduce a new approach to improve the generalization of convolutional-based models for HSI classification. Specifically, we develop a neighboring region DO technique that selectively cuts off certain neighboring outputs, creating spatial dropped regions. Our experimental results with two well-known HSIs reveal that the newly proposed method helps to achieve better classification accuracy than the traditional DO strategy, with a low computational cost.**

***Index Terms*— Convolutional neural networks (CNNs), dropout (DO), hyperspectral images (HSIs), regularization.**

I. Introduction

HYPERSPECTRAL images (HSIs) comprise big cubes of adjacent spectral bands, where each pixel records the electromagnetic interaction between the incident solar radiation and the observed objects in a spectral signature that can be considered unique for each material on the surface of the earth. This allows a detailed characterization of observed areas and enables their successful exploitation on a wide range of applications [1].

The huge amount of information contained in HSI data cubes has been exploited by a large variety of spectral, spatial, and spectral–spatial classification methods, offering models with good performance in the task of understanding those features and relationship contained in the image. Traditionally, the following three kinds of methods have been established depending on the training procedure.

1) Unsupervised methods do not need to be trained, as they do not use labeled samples to fine-tune the model, being quite popular some clustering method such as k-means [2], linear discriminant analysis (LDA) [3], or probabilistic latent semantic analysis (PLSA) [4].
2) Supervised methods split the available data into labeled and unlabeled samples in order to perform the training and the inference steps. Some widely used supervised classifiers are the multinomial logistic regression (MLR) [5] or the support vector machine (SVM) [6].
3) Semisupervised methods apply different strategies to include unlabeled data, such as active learning approaches [7], or to expand the training set, using, for instance, generative adversarial networks (GANs) [8].

With the release of large and complex HSI data sets, the development of new classification algorithms is required in order to properly interpret the acquired data. Deep learning and convolutional-based approaches have been successfully used for this purpose, reaching excellent performance due to their inherent ability for exploiting different spectral and spatial features through deep and hierarchical architectures made up of stacked feature extractors [9], [10]. However, the performance of these methods for HSI classification is bounded by the high dimensionality of the data, the limited number of available labeled samples, and, generally, the low spatial resolution of HSIs, leading to the curse of dimensionality (Hughes effect), overfitting, and data variability problems [1].

Deep neural networks (DNNs), in general, and convolutional neural networks (CNNs) [11], in particular, can be seen as approximators of the form $f : \mathcal{X} \rightarrow \mathcal{Y}$ that solve an optimization function subject to a loss expression L. In this sense, supervised DNNs are mapping problems where, given an HSI data set, find the corresponding labels by tuning a parameterized model $\mathcal{M}(\mathcal{X}, \theta) = \mathcal{Y}$, whose parameters θ (distributed among the layers' stack) should minimize the error between the predicted and the expected outputs. Recent works claim that the deeper the $\mathcal{M}$, the better the accuracy that can be achieved [12]. This has a direct effect on the number of parameters, imposing severe restrictions on the amount of employed training samples, apart from the data degradation factor that is directly related to the increase of the model's depth. In this regard, several data augmenting and regularization techniques have been explored to avoid these problems. Focusing on the first ones, some works propose to increase the training set by applying slight spectral modifications and

Manuscript received March 14, 2019; revised July 14, 2019; accepted August 28, 2019. Date of publication September 25, 2019; date of current version May 21, 2020. This work was supported in part by Spanish Ministry under Grant FPU14/02012-FPU15/02090 and Grant ESP2016-79503-C2-2-P, in part by Junta de Extremadura under Grant GR18060, and in part by European Union under Grant 734541-EOXPOSURE. *(Corresponding author: Mercedes E. Paoletti.)*

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

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

Digital Object Identifier 10.1109/LGRS.2019.2940467

spatial transformations to the available data [13], although these approaches are very time-consuming (simpler methods, such as the addition of random noise, do not take into account the spatial characteristics of the image). Haut *et al.* [14] introduced random occlusion (RO) as a new data augmenting method that drops certain areas on the CNN's inputs, maintaining the spatial consistency between the dropped zones. Although this approach exhibits good performance, it is not robust to network parameters, in the sense that relations between the weights of adjacent layers are not encouraged. On the other hand, regularization methods such as dropout (DO) [15] are able to strengthen the model, enforcing independence between adjacent layer weights by setting to zero some randomly selected neural activations during the training stage. In this context, DO is widely used due to its simplicity and low computational cost [9], being effective particularly on the fully connected layers of the CNN's architecture. However, its performance with convolutional layers is not that impressive, due to its random feature clipping, which does not take into account spatial implications. In fact, the effects of DO on a convolutional's output imitate the traditional "salt&pepper" noise, while feature maps remain spatially correlated. In the end, the extracted information is still propagated to the following layers [16].

In this context, inspired by the original DO mechanism and the RO data augmenting approach, this letter introduces the neighboring region DO (NRDO), a new spatially correlated DO mechanism in which random neighboring kernel's activations are dropped, creating occluded areas on the convolutional layers' output volumes that maintain spatial consistency while avoiding a significant increase in computational complexity [16]. The remainder of this letter is organized as follows. Section II describes the proposed method. Section III discusses the performance of the proposed NRDO using two HSIs, demonstrating its accuracy. Section IV concludes with some remarks and hints at plausible future research lines.

II. Methodology

Let us define $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times n_{\text{bands}}}$ as a HSI cube, where $n_1 \times n_2$ are the spatial components and n_{bands} is the number of spectral bands. Each HSI pixel can be represented as a spectral vector $\mathbf{x}_{i,j} \in \mathbb{R}^{n_{\text{bands}}}$. An end-to-end spatial 2-DCNN model normally performs a preprocessing step, encoding the spectral information into one band by using, for instance, principal component analysis (PCA) and extracts, for each 1-D pixel $x_{i,j}$, a neighboring window $\mathbf{p}_{i,j} \in \mathbb{R}^{d \times d \times 1}$ of $d \times d$ spatial dimensions, with $x_{i,j}$ being the central pixel. During the training stage, pairs of patches and labels $\{\mathbf{p}_{i,j}, \mathbf{y}_{i,j}\}$ are used to create the training set, where $\mathbf{y}_{i,j} \in \mathbb{R}^{n_{\text{classes}}}$ represents the label of the (i, j)th patch's central pixel in one-hot encoding way. These patches are fed to the 2-DCNN, which applies a hierarchical stack of feature extraction (FE) stages to obtain different levels of data representation, until reaching an abstract representation at the end, that encodes the more descriptive features and internal nonlinear data relationship, which are employed by the final classifier layers to produce a classification output.

Each FE-stage is usually composed by a set of different layers, being the convolutional layer the major responsable for the extraction. Each layer l defines $K^{(l)}$ filters, with $k^{(l)} \times k^{(l)}$ neurons each. In this sense, the kernel defined by the lth layer computes the operation over the input with sliding-step s, being overlapped on local areas. At the end, the kernel performs the linear convolution $(*)$ between the weights of the neurons $\mathbf{W}^{(l)}$, the input data volume $\mathbf{X}^{(l-1)}$, and the bias $b^{(l)}$, obtaining an output volume $\mathbf{X}^{(l)} \in \mathbb{R}^{n^{(l)} \times n^{(l)} \times K^{(l)}}$ of K feature maps with $n^{(l)} \times n^{(l)}$ extracted features

$$\mathbf{X}^{(l)} = \mathcal{H}(\mathbf{W}^{(l)} * \mathbf{X}^{(l-1)} + b^{(l)})_{K^{(l)} \times k^{(l)} \times k^{(l)}}. \tag{1}$$

After the FE-stage performed by the convolutional layer, a nonlinear activation function $\mathcal{H}(\cdot)$ is applied to the output volume in order to extract the nonlinear features and relationship contained into the volume. In our case, we apply the well-known rectified linear unit (ReLU) function. Also, a downsampling operation (implemented by max or average pooling) is applied in order to reduce the spatial dimensions and summarize the obtained features.

From (1), it can be observed that the outputs of previous layers are refined by the following ones, i.e., the CNN's neurons are, in fact, working in a cooperative way [15], [17]. Although this hierarchical mechanism is appealing during the training stage, introduce weak links between the neurons of adjacent layers, and hampering the inference step [9]. In this context, traditional DO [15] is applied between the activation and pooling layers as a regularization method to avoid overfitting and provide some independence between adjacent layers' neurons, by setting to zero some randomly selected neural activations. This improves the backpropagation procedure, where neurons should be adjusted in an individual way, instead of establishing trivial dependencies with other neurons. The main motivation behind this approach is to force the layer's neurons to extract more robust and discriminatory features on their own. Mathematically, we can break down (1) in order to focus on the (i, j)th extracted feature of convolutional layer l in its zth filter (with $z = \{1, \ldots, K^{(l)}\}$), to which a gating 0-1 Bernoulli variable is applied as the DO regularization term $\delta_{i,j}^{(l)}$, following a probability percentage $p^{(l)}$ which is fixed to the lth layer [18]:

$$\delta_{i,j}^{(l)} = \text{Bernoulli}(p^{(l)}) \tag{2a}$$

$$x_{i,j}^{(l)_z} = \mathcal{H}\left(\delta_{i,j}^{(l)} \sum_{\hat{i}=1}^{k^{(l)}} \sum_{\hat{j}=1}^{k^{(l)}} \left(w_{\hat{i},\hat{j}}^{(l)} x_{(is+\hat{i}),(js+\hat{j})}^{(l-1)}\right) + b^{(l)}\right)_z. \tag{2b}$$

Fig. 1 provides a graphical illustration of how the DO regularization method works, using a synthetic feature map given in Fig. 1(a). As it can be observed in Fig. 1(b) and (c), the DO injects random noise to the feature maps in order to disentangle the behavior of adjacent layers' neurons. However, this noise is not structured, which makes it not completely effective in the task of removing semantic information of the feature map, where nearby features still contain related spatial information.

To overcome the limitations of the traditional DO strategy, we propose to inject spatial-structured noise at every feature map by dropping the output of neighboring neural activations, obtaining full-dropped spatial regions on the output feature maps that effectively remove spatial-correlated information. In this sense, the amount of neural activations $\gamma^{(l)}$ that will be dropped, coupled with the surrounding window's spatial size

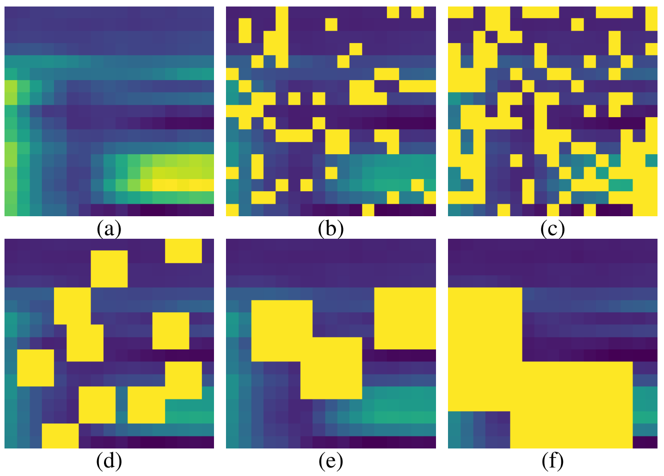

Fig. 1. Visualization of the original DO and the proposed NRDO performance over a feature map of size 17×17. The first row shows (a) original feature map and the feature maps obtained after dropping isolated samples using a DO of (b) $p^{(l)} = 20\%$ and (c) $p^{(l)} = 40\%$. The second row shows the feature map obtained after applying NRDO by configuring the dropping percentage and the dropping windows size to (d) $p^{(l)} = 20\%$ and $d^{(l)} = 3$, (e) $p^{(l)} = 20\%$ and $d^{(l)} = 5$, and (f) $p^{(l)} = 20\%$ with $d^{(l)} = 10$.

$d^{(l)}$ (that will be set to zero), must be defined at each layer l. Following the DO method, for each position (i, j) of the input feature map $\mathbf{X}^{(l-1)}$, our NRDO applies the gating variable $\delta_{i,j}^{(l)}$, obtained by the Bernoulli distribution with probability $\gamma^{(l)}$. In addition, for the zero variables $\delta_{i,j}^{(l)}$, a spatial square patch centered on (i, j) is obtained as a zero-mask with dimensions $d^{(l)} \times d^{(l)}$. Finally, this mask is overlapped and applied over the input volume $\mathbf{X}^{(l-1)}$, dropping the corresponding window in all the K filters of the lth layer. However, instead of setting a direct dropping probability, $\gamma^{(l)}$ is obtained as a correction of the traditional DO percentage $p^{(l)}$, the dropping window's size $d^{(l)}$, and the spatial dimension of the obtained feature map $\mathbf{X}^{(l)} \in \mathbb{R}^{n^{(l)} \times n^{(l)} \times K^{(l)}}$ by the lth convolutional layer. In this context, $\gamma^{(l)}$ is obtained as

$$\gamma^{(l)} = \frac{p^{(l)}}{(d^{(l)})^2} \frac{(n^{(l)})^2}{(n^{(l)} - d^{(l)} + 1)^2}. \tag{3}$$

It is recommened that $d^{(l)}$ is not be greater than $n^{(l)}$. In fact, (3) makes an approximation between the desired amount of dropped data, indicated by the known $p^{(l)}$, and the dropped neighborhood for each zero variable $\delta_{i,j}^{(l)}$, to make an equitable balance between the pixels and their surrounding windows to be dropped and the desired amount of spatial-structured noise to be injected.

Algorithm 1 gives a general overview of the proposed NRDO method, which is applied between the convolutional and nonlinear activation layers, following the scheme given in (2). An interesting aspect is the computation of the window to be dropped. As some HSI data sets are characterized by their low spatial resolution, our strategy can help in this particular case since the dropped neighborhoods are adapted to the feature map's margins, taking advantage of all the available features as we can observe in Fig. 1(d)–(f), where the dropped neighborhoods have been adjusted to the feature's edges. Moreover, these dropped windows can be also spatially overlapped, as it can be observed in Fig. 1(f), where two dropped regions that are slightly overlapped can be appreciated.

Algorithm 1 NRDO

```
1: procedure NRDO($\mathbf{X}^{(l)} \in \mathbb{R}^{n^{(l)} \times n^{(l)} \times K^{(l)}}$: obtained feature map from $l$-th layer, $p^{(l)}$: dropping percentage, $d^{(l)}$: dropping window's size)
2:     $\gamma^{(l)} = \frac{p^{(l)}}{(d^{(l)})^2} \frac{(n^{(l)})^2}{(n^{(l)} - d^{(l)} + 1)^2}$    ▷ Dropping probability
3:     $\mathbf{M} = \text{Ones}(n^{(l)}, n^{(l)}, K^{(l)})$    ▷ Initializing mask
4:     for $i, j$ in $n^{(l)}$ do
5:         if $\left(\delta_{i,j}^{(l)} = \text{Bernoulli}(\gamma^{(l)})\right) == 0$ then
6:             $\mathbf{M} = \text{Dropped\_Window\_on\_M}(i, j, d^{(l)})$    ▷ For each zero $\delta_{i,j}^{(l)}$, a zeroed square window $d^{(l)} \times d^{(l)}$ is set on the mask $\mathbf{M}$ with the center on the $(i, j)$ position
7:         end if
8:     end for
9:     $\hat{\mathbf{X}}^{(l)} = \mathbf{M} \cdot \mathbf{X}^{(l)}$
       return $\hat{\mathbf{X}}^{(l)}$
10: end procedure
```

TABLE I
ARCHITECTURAL DETAILS OF THE PROPOSED MODEL

Layer ID	Kernel/Neurons	DO/NRDO	Max Pooling	Act. function
Conv1	$50 \times 3 \times 3 \times 1$	No	No	ReLU
Conv2	$100 \times 5 \times 5 \times 50$	Yes	2×2	ReLU
Conv3	$200 \times 5 \times 5 \times 100$	Yes	2×2	ReLU
Conv4	$400 \times 2 \times 2 \times 200$	No	No	ReLU
FC1	300	No	-	ReLU
FC2	$n_{classes}$	No	-	Softmax

On the other hand, the application of a rigorous NRDO with a fixed value of $p^{(l)}$ can negatively affect the performance of the network, while the implementation of a soft NRDO may not provide the desired robustness. In order to overcome the limitations, our model is trained with a $p^{(l)}$ whose value increases linearly and progressively through the epochs [16], from zero probability to the maximum indicated value of $p^{(l)}$, with the goal of progressively adapting the performance, extracting more robust and independent features at each epoch.

III. EXPERIMENTS

A. Experimental Configuration and Data Sets

In order to test the performance of the proposed regularization technique, a deep 2-DCNN model has been implemented for HSI classification. Inspired by previous works in [14], the proposed network is composed of four convolutional layers and two fully connected layers. Focusing on the convolutional layers, the second and third layers implement one of the two available dropping mechanism, DO or NRDO, for comparative purposes. Table I describes the details of the configuration of the network, which has been executed on a hardware environment composed by a sixth-generation Intel Core i7-6700K processor with 8M of Cache and up to 4.20 GHz (four cores/eight way multi-task processing), an ASUS Z170 progamming motherboard, a GPU NVIDIA GeForce GTX 1080 with 8-GB GDDR5X of video memory and 10 Gbps of memory frequency, 40 GB of DDR4 RAM with a serial speed of 2400 MHz and a Toshiba DT01ACA

TABLE II

COMPARISON BETWEEN DO AND NRDO, WITH DIFFERENT PERCENTAGES OF $p^{(l)}$ AND SETTING $d^{(l)} = 3$

	T. Size	2DCNN	*2DCNN + Non-Spatially Structured Dropout*			*2DCNN + Neighboring Region Dropout*		
			20%	40%	80%	20%	40%	80%
INDIAN P.	1%	52.85 ±1.90	52.69 ±2.18	53.61 ±1.40	56.84 ±1.16	56.24 ±1.08	56.72 ±2.63	**59.64** ±1.41
	3%	70.97 ±1.57	74.70 ±1.29	75.10 ±1.48	80.44 ±1.83	79.08 ±0.98	82.93 ±0.69	**85.81** ±1.05
	5%	81.45 ±1.24	84.40 ±2.18	86.46 ±1.33	90.39 ±0.78	89.61 ±1.05	91.58 ±0.84	**93.78** ±0.53
	10%	92.43 ±1.03	94.98 ±0.72	96.35 ±0.23	97.90 ±0.26	97.52 ±0.19	97.85 ±0.51	**98.49** ±0.29
	15%	96.42 ±1.04	98.18 ±0.10	98.62 ±0.30	99.34 ±0.17	99.03 ±0.21	99.24 ±0.23	**99.53** ±0.15
	20%	98.13 ±0.42	99.20 ±0.32	99.25 ±0.18	99.69 ±0.16	99.51 ±0.08	99.69 ±0.05	**99.83** ±0.06
U. PAVIA	1%	87.47 ±0.54	87.55 ±0.97	87.49 ±0.87	90.75 ±0.89	89.94 ±0.47	91.92 ±0.39	**92.39** ±0.54
	3%	95.62 ±0.41	95.56 ±0.37	96.42 ±0.21	98.14 ±0.16	97.35 ±0.33	97.90 ±0.21	**98.70** ±0.17
	5%	97.64 ±0.20	98.18 ±0.07	98.50 ±0.26	99.16 ±0.16	99.07 ±0.20	99.38 ±0.08	**99.57** ±0.08
	10%	99.20 ±0.14	99.51 ±0.08	99.62 ±0.08	99.82 ±0.05	99.79 ±0.07	99.89 ±0.02	**99.93** ±0.02
	15%	99.57 ±0.11	99.84 ±0.05	99.89 ±0.01	99.94 ±0.02	99.93 ±0.01	99.96 ±0.01	**99.98** ±0.01
	20%	99.81 ±0.02	99.92 ±0.01	99.95 ±0.02	99.98 ±0.02	99.98 ±0.01	99.99 ±0.01	99.99 ±0.01

TABLE III

OBTAINED OA RESULTS FOR EACH CONSIDERED CLASSIFIER

	T. Size	RF	SVM RBF	MLP	ELM	K-ELM	1DCNN	2DCNN			
								RAW	RO	DO	NRDO
INDIAN P.	1%	-	-	-	-	-	-	52.85	54.90	56.84	**59.64**
	3%	-	-	-	-	-	-	70.97	78.74	80.44	**85.81**
	5%	69.00	74.52	77.13	72.23	80.38	75.37	81.45	89.46	90.39	**93.78**
	10%	75.58	81.00	83.10	78.88	84.85	82.66	92.43	96.96	97.90	**98.49**
	15%	78.19	83.91	85.28	81.59	86.81	86.13	96.42	98.45	99.34	**99.53**
PAVIA U.	1%	81.35	88.91	88.11	80.28	82.05	85.70	87.47	88.69	90.75	**92.39**
	3%	-	-	-	-	-	-	95.62	96.94	98.14	**98.70**
	5%	87.36	93.43	93.19	85.35	87.07	91.01	97.64	98.64	99.16	**99.57**
	10%	89.51	94.45	94.36	86.72	88.69	94.13	99.20	99.68	99.82	**99.93**
	15%	90.48	94.89	94.91	87.24	89.41	95.29	99.57	99.92	99.94	**99.98**

HDD with 7200RPM and 2 TB of storage capacity. In addition, and in order to efficiently implement the proposed approach, it has been parallelized over the GPU using CUDA language over Pytorch framework. Finally, all the codes and examples presented in this letter are available online.[1] The proposed method has been tested over two widely used HSI data sets. The first one is the AVIRIS's Indian Pines (IP) scene, which has 145 × 145 samples with low spatial resolution of 20mpp and 200 spectral bands in the wavelength range from 0.4 to 2.5 μm. It was captured over an agricultural and forest area and its ground truth is composed of 16 different classes. The second one is the ROSIS's University of Pavia (UP) scene, which contains 610 × 340 samples with higher (1.3 mpp) spatial resolution and 113 spectral bands in the wavelength range from 0.43 to 0.86 μm. It was captured over an urban area and its ground truth is composed of nine different classes.

B. Experimental Results and Discussion

1) Comparison Between Dropout and Neighboring Region Dropout: First, experiment compares the performance of the 2-DCNN model with regularization method, considering the original nonspatially structured DO and the proposed NRDO. Each model has been trained over IP and UP with 1%, 3%, 5%, 10%, 15%, and 20% of randomly selected samples, input patch size of 11 × 11, and different dropping percentages ($p^{(l)} = \{20\%, 40\%, 80\%\}$), fixing the dropping window's size to $d^{(l)} = 3$ in the case of NRDO. Table II shows the obtained results. Focusing on DO, this strategy is highly beneficial when the scene is spectrally mixed and contains few regular spatial structures (as it is the case with the IP scene). Moreover, the bigger $p^{(l)}$, the larger the overall accuracy (OA) improvement. However, with the UP scene, the effectiveness of DO is appreciably lower than with the IP scene (in fact, only in the case of $p^{(l)} = 80\%$, the OA values rise by more than 1% point for small training sets), even reducing the overall performance with limited training samples (1% of IP and 3% of UP employing $p^{(l)} = 20\%$). In this sense, the proposed method exhibits a more consistent behavior with both data sets, being able to outperform the results obtained by DO and significantly improving the results obtained by the original 2-DCNN without regularization method and exhibiting a lower standard deviation. The effectiveness of this method is visibly high in IP and UP, in particular, when small training sets are considered, reaching the best OA performances when $p^{(l)} = 80\%$. It must be noted that, since NRDO occludes entire windows, it prevents the model from seeing all the complete features of the input data, forcing the network to look for more robust parameters.

2) Comparison Between Neighboring Region Dropout and Several Classifiers: Second, experiment performs a comparison between the proposal NRDO, with $p^{(l)} = 40\%$ and $d^{(l)} = 3$, and six different classifiers: 1) random forest (RF); 2) SVM with radial basis function; 3) shallow multilayer perceptron (MLP); 4) basic and kernel extreme learning machines (ELM and K-ELM); 5) spectral 1-DCNN; and 6) spatial 2-DCNN. In addition, four 2-DCNN models have been considered: without data augmenting or DO methods (original data), with RO [14], with $p^{(l)} = 80\%$ of DO, and with

[1]https://github.com/mhaut/DeepNRD

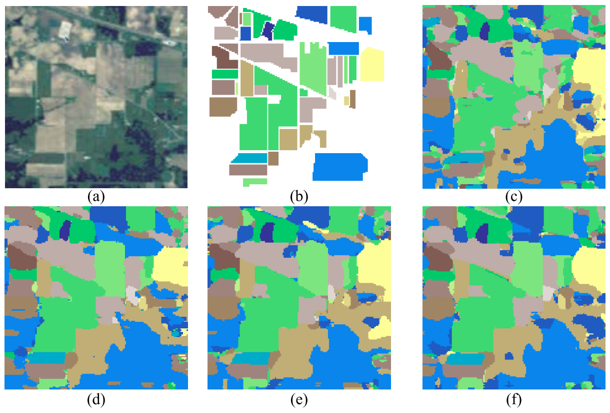

Fig. 2. Classification maps of IP, being (a) simulated RGB composition of the scene, (b) ground truth, and from (c) to (f), the obtained classification maps corresponding to the 2-DCNN models of Table III. (a) RGB. (b) GT. (c) 2-DCNN (92.43%). (d) 2-DCNN-RO% (96.96%). (e) 2-DCNN-DO% (97.90%). (f) 2-DCNN-NRDO% (98.49%).

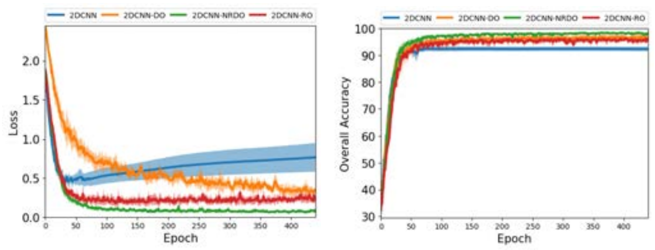

Fig. 3. Evolution of the (Left) loss and (Right) OA as a function of the number of training epochs when using the 2-DCNN with and without DO and NRDO regularization methods and RO data augmenting method, over IP data set, with 10% of training data and setting $p^{(l)} = 80\%$ and $d^{(l)} = 3\%$.

$p^{(l)} = 80\%$ of NRDO. Table III reveals that spatial models are able to greatly outperform spectral methods, reaching 90% and 99% of OA when classifying IP and UP scenes, respectively. Focusing in spatial models, the proposed NRDO is able to reduce the overfitting problem when lower training percentages are employed, achieving the best result in all the experiments. This suggests that neurons are able to learn independently while retaining a spatial context, so the final classification becomes more robust. Fig. 2 shows classification maps obtained by the spatial classifiers. It can be seen that the proposed method is able to correctly classify even the smallest and more complex classes, thus providing a more detailed map. Finally, Fig. 3 shows the evolution of the loss and OA with increasing epochs obtained by the 2-DCNN without any data augmenting/regularization method, and with RO, DO, and NRDO. Looking at the raw model, the loss grows as the epochs increase, indicating a clear overfitting problem. Although the RO decreases the loss faster than DO, it also suffers the overfitting in the final epochs. However, the proposed method is able to achieve lower and stable loss scores than DO and RO. This is also observed in the evolution of OA, where the NRDO enables a better tuning of the result.

IV. Conclusion

This letter evaluates a spatial-structured regularization technique for HSI data classification, which is based on randomly drop squared-windows of the convolutional-extracted feature maps, retaining the spatial consistency and allowing a strongest, deep and independent learning of the layer's neurons. Obtained results demonstrate that not only the proposed approach efficiently deals with the overfitting problem when low training data are available but is also able to reach a better performance than other compared techniques. Moreover, as the proposal improves the performance of the convolutional layer, it can be effectively used in more complex models, such as ResNets and DenseNets. Finally, since the approach is not restricted to spatial classifiers, in the future, we plan to incorporate it to spatial–spectral models too.

References

[1] M. E. Paoletti *et al.*, "Capsule networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 4, pp. 2145–2160, Apr. 2019.
[2] J. Haut, M. Paoletti, J. Plaza, and A. Plaza, "Cloud implementation of the K-means algorithm for hyperspectral image analysis," *J. Supercomput.*, vol. 73, no. 1, pp. 514–529, 2017.
[3] R. Bahmanyar, D. Espinoza-Molina, and M. Datcu, "Multisensor earth observation image classification based on a multimodal latent Dirichlet allocation model," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 3, pp. 459–463, Mar. 2018.
[4] R. Fernandez-Beltran, J. M. Haut, M. E. Paoletti, J. Plaza, A. Plaza, and F. Pla, "Remote sensing image fusion using hierarchical multimodal probabilistic latent semantic analysis," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 12, pp. 4982–4993, Dec. 2018.
[5] M. Khodadadzadeh, J. Li, A. Plaza, and J. M. Bioucas-Dias, "A subspace-based multinomial logistic regression for hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 12, pp. 2105–2109, Dec. 2014.
[6] Z. Shao, L. Zhang, X. Zhou, and L. Ding, "A novel hierarchical semisupervised SVM for classification of hyperspectral images," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 9, pp. 1609–1613, Sep. 2014.
[7] J. M. Haut, M. E. Paoletti, J. Plaza, J. Li, and A. Plaza, "Active learning with convolutional neural networks for hyperspectral image classification using a new Bayesian approach," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 11, pp. 6440–6461, Nov. 2018.
[8] L. Zhu, Y. Chen, P. Ghamisi, and J. A. Benediktsson, "Generative adversarial networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 9, pp. 5046–5063, Sep. 2018.
[9] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza, "A new deep convolutional neural network for fast hyperspectral image classification," *ISPRS J. Photogram. Remote Sens.*, vol. 145, Part A, pp. 120–147.
[10] L. Zhang, Q. Zhang, B. Du, X. Huang, Y. Y. Tang, and D. Tao, "Simultaneous spectral-spatial feature selection and extraction for hyperspectral images," *IEEE Trans. Cybern.*, vol. 48, no. 1, pp. 16–28, Jan. 2018.
[11] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the Art," *IEEE Geosci. Remote Sens. Mag.*, vol. 4, no. 2, pp. 22–40, Jun. 2016.
[12] M. E. Paoletti, J. M. Haut, R. Fernandez-Beltran, J. Plaza, A. J. Plaza, and F. Pla, "Deep pyramidal residual networks for spectral–spatial hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 740–754, Feb. 2019.
[13] W. Li, C. Chen, M. Zhang, H. Li, and Q. Du, "Data augmentation for hyperspectral image classification with deep CNN," *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 4, pp. 593–597, Apr. 2019.
[14] J. M. Haut, M. E. Paoletti, J. Plaza, J. Li, and A. Plaza, "Hyperspectral image classification using random occlusion data augmentation," *IEEE Geosci. Remote Sens. Lett.*, to be published.
[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[16] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Dropblock: A regularization method for convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 10750–10760.
[17] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," Jul. 2012, *arXiv:1207.0580*. [Online]. Available:https://arxiv.org/abs/1207.0580
[18] P. Baldi and P. J. Sadowski, "Understanding dropout," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2814–2822.