Pattern Recognition 61 (2017) 539–556

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Towards better exploiting convolutional neural networks for remote sensing scene classification

Keiller Nogueira [a,*], Otávio A.B. Penatti [b], Jefersson A. dos Santos [a]

[a] Departamento de Ciência da Computação, Universidade Federal de Minas Gerais (UFMG), Av. Presidente Antônio Carlos, 6627, Belo Horizonte, MG CEP 31270-901, Brazil

[b] Advanced Technologies Group, SAMSUNG Research Institute, Av. Cambacica, 1200, Building 1, Campinas, SP CEP 13097-160, Brazil

ARTICLE INFO

Article history:
Received 1 February 2016
Received in revised form
1 June 2016
Accepted 1 July 2016
Available online 2 July 2016

Keywords:
Deep learning
Convolutional neural networks
Fine-tune
Feature extraction
Aerial scenes
Hyperspectral images
Remote sensing

ABSTRACT

We present an analysis of three possible strategies for exploiting the power of existing convolutional neural networks (ConvNets or CNNs) in different scenarios from the ones they were trained: full training, fine tuning, and using ConvNets as feature extractors. In many applications, especially including remote sensing, it is not feasible to fully design and train a new ConvNet, as this usually requires a considerable amount of labeled data and demands high computational costs. Therefore, it is important to understand how to better use existing ConvNets. We perform experiments with six popular ConvNets using three remote sensing datasets. We also compare ConvNets in each strategy with existing descriptors and with state-of-the-art baselines. Results point that fine tuning tends to be the best performing strategy. In fact, using the features from the fine-tuned ConvNet with linear SVM obtains the best results. We also achieved state-of-the-art results for the three datasets used.

1. Introduction

Encoding discriminating features from visual data is one of the most important steps in almost any computer vision problem, including in the remote sensing domain. Since manual extraction of these features is not practical in most cases, during years, substantial efforts have been dedicated to develop automatic and discriminating visual feature descriptors [1]. In the early years, most of such descriptors were based on pre-defined algorithms independently of the underlying problem, like color histograms and correlograms [1,2]. Then, descriptors based on visual dictionaries, the so-called Bag of Visual Words (BoVW), attracted the attention and have become the state-of-the-art for many years in computer vision [3–9]. Although the aforementioned visual feature extraction techniques have been successfully applied in several domains [10], due to the specificities of remotely sensed data, many of these classical techniques are not straightforwardly applicable in the remote sensing domain. Indeed, encoding of spatial information in remote sensing images is still considered as an open and challenging task [11]. Thus, there is a huge demand for feature extraction algorithms that are able to effectively encode spectral and spatial information, since it is the key to generating discriminating models for remote sensing images.

Recently, deep learning has become the new state-of-the-art solution for visual recognition. Given its success, deep learning has been intensively used in several distinct tasks of different domains [12,13], including remote sensing [14–16]. In remote sensing, the use of deep learning is growing very quickly, since it has a natural ability to effectively encode spectral and spatial information based mainly on the data itself. Methods based on deep learning have obtained state-of-the-art results in many different remote sensing applications, such as image classification [15], oil spill [17], poverty mapping [18], and urban planning [19].

Deep learning [12,13] is a branch of machine learning that refers to multi-layered interconnected neural networks that can learn features and classifiers at once, i.e., a unique network may be able to learn features and classifiers (in different layers) and adjust the parameters, at running time, based on accuracy, giving more importance to one layer than another depending on the problem. End-to-end feature learning (e.g., from image pixels to semantic labels) is the great advantage of deep learning when compared to previous state-of-the-art methods [20], such as mid-level (BoVW) and global low-level color and texture descriptors. Among all deep learning-based networks, a specific type, called Convolutional (Neural) Networks, ConvNets or CNNs [12,13], is the most popular for learning visual features in computer vision applications, including remote sensing. This sort of network relies on the natural

* Corresponding author.
E-mail addresses: keiller.nogueira@dcc.ufmg.br (K. Nogueira), o.penatti@samsung.com (O.A.B. Penatti), jefersson@dcc.ufmg.br (J.A. dos Santos).

http://dx.doi.org/10.1016/j.patcog.2016.07.001
0031-3203/

stationary property of an image, i.e., the statistics of one part of the image are the same as any other part and information extracted at one part of the image can also be employed to other parts. Furthermore, deep ConvNets usually obtain different levels of abstraction for the data, ranging from local low-level information in the initial layers (e.g., corners and edges), to more semantic descriptors, mid-level information (e.g., object parts) in intermediate layers and high level information (e.g., whole objects) in the final layers.

The exploration of the potentials of deep ConvNets can be a complex task because of several challenges, such as (i) complex tuning, since convergence cannot be totally confirmed (given its highly non-convex property), (ii) "black box" nature, (iii) high computational burden, (iv) proneness to overfitting, and (v) empirical nature of model development, which is over parameterized. However, through years, researches have developed strategies to explore the potential of deep ConvNets in different domains and scenarios.

In this work, we evaluate and analyze three possible strategies of exploiting ConvNets: (i) full-trained ConvNets, (ii) fine-tuned ConvNets, and (iii) pre-trained ConvNets used as feature extractors. In the first strategy (i), a (new or existing) network is trained from scratch obtaining specific visual features for the dataset. This approach is preferable since it gives full control of the architecture and parameters, which tends to yield a more robust and efficient network. However, it requires a considerable amount of data [12,13], since the convergence of network is pruned to overfitting (and small datasets tend to magnify this problem). This drawback makes almost impracticable to fully design and train a network from scratch for most remote sensing problems, since large datasets in this domain are unusual given that training data may require high costs with travel and other logistics [21,22]. In an attempt to overcome the issue of few data to train the network, data augmentation[1] techniques [23] can be employed. However, for small datasets even data augmentation is not enough to avoid overfitting. The other two strategies (ii and iii) rely on the use of pre-trained ConvNets, i.e., we can employ networks that were trained on different data from the data of interest. In fact, these strategies benefit from the property that initial layers of ConvNets tend to be generic filters, like edge or color blob detectors, which are less dependent on the final application and could be used in a myriad of tasks. The second strategy (ii) uses a pre-trained ConvNet and performs fine-tuning of its parameters (filter weights) using the remote sensing data of interest. Usually, in this case, the earlier layers are preserved, as they encode more generic features, and final layers are adjusted to encode specific features of the data of interest [24,25]. The third strategy (iii) simply uses a pre-trained ConvNet as a feature extractor, by removing the last classification layer and considering its previous layer (or layers) as feature vector of the input data.

Thus, in this paper, we analyze the aforementioned strategies of exploiting deep learning in the remote sensing domain. The objective is to elucidate and discuss some aspects of ConvNets which are necessary to take the most advantage of their benefits. To the best of our knowledge, no other study in the literature realizes such analysis neither in remote sensing nor in other computer vision applications. We carried out a systematic set of experiments comparing the results of the different strategies mentioned using *six* existing ConvNets in three remote sensing datasets. In practice, our analysis is designed to address the following research question: what is the strategy that better exploits the benefits of existing ConvNets for the remote sensing domain? Therefore, we can summarize the contributions of our paper as: (i) analysis of the generalization power of ConvNets for other remote sensing datasets, (ii) comparative investigation of ConvNets and handcrafted feature descriptors, (iii) evaluation and investigation of three strategies to exploit existing ConvNets in different scenarios, (iv) comparison with state-of-the-art baselines.

Our results point that fine-tuning tends to be the best performing strategy. In fact, using the features from the fine-tuned ConvNet with linear SVM obtains the best results. In addition, we obtained state-of-the-art classification results for the tree remote sensing datasets used.

The current paper extends two previous works: (i) "Improving spatial feature representation from aerial scenes by using convolutional networks", published at the SIBGRAPI 2015 (Conference on Graphics, Patterns and Images), and (ii) "Do deep features generalize from everyday objects to remote sensing and aerial scenes domains?", published at the EarthVision Workshop in the CVPR 2015 (IEEE Conference on Computer Vision and Pattern Recognition). However, this work differs from the aforementioned papers specially because we evaluate the three strategies for using existing ConvNets, we consider more (six) ConvNets, and we perform experiments in one more dataset. The aforementioned work (i) proposes a new ConvNet and work (ii) evaluates (two) existing ConvNets as feature extractors.

The remainder of this paper is organized as follows. Section 2 presents related work. Some background concepts related to ConvNets are presented in Section 3. Section 4 presents, in detail, all strategies evaluated in this work. The experimental setup, datasets, ConvNets and the descriptors evaluated in this paper are presented in Section 5. In Section 6, we present and discuss the results obtained. Finally, Section 7 concludes the paper.

2. Related work

Considerable efforts have been dedicated to the development of suitable feature descriptors for remote sensing applications [1]. Although several of these visual descriptors have been proposed or successfully used for remote sensing image processing [26,2,27], there are many applications that demand more specific techniques mainly due to the presence of non-visible information provided by multiple spectral bands and the lack of adaptation in relation to aerial scenes (when compared to traditional sort of images). Towards this goal, Convolutional Neural Networks, ConvNets or CNNs, which are the most popular deep learning approach for computer vision, involve machine learning in the process of obtaining the best visual features for a given problem. They are based on an end-to-end learning process, from raw data (e.g., image pixels) to semantic labels, which is an important advantage in comparison with previous state-of-the-art methods [20]. However, during the learning step, only parameters are learned, not the complete deep architecture, like the number and types of layers and how they are organized. Specifically, deep ConvNets have several drawbacks, such as impossibility to confirm convergence, "black box" nature, high computational cost, proneness to overfitting and empirical nature of model development. In an attempt to alleviate these effects, some strategies can be used to better exploit existing ConvNets, which are: (i) fine-tuned ConvNets and (ii) pre-trained ConvNets used as feature extractors. Aside the strategy to fully training a network from scratch, these two approaches assemble the principal strategies when working with ConvNets. Thus, the objective of this paper is to evaluate these three strategies to explore existing deep neural networks. Hence, in this section, we focus on analyzing existing works that also exploit ConvNets similar to what we do.

[1] Data augmentation is a technique that artificially enlarges the training set with the addition of replicas of the training samples under certain types of transformations that preserve the class labels.

Train a (new or existing) network from scratch is preferable since it tends to give specific visual features for the dataset. Also, this strategy gives full control of the architecture and parameters, which tends to yield a more robust network. However, it requires a considerable amount of data [12,13]. During the years, successful ConvNets were the ones trained in large amount of data, such as ImageNet dataset [28], which has been used to train several famous architectures [23,29,30]. AlexNet, proposed by Krizhevsky et al. [23], was the winner of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [28] in 2012 and mainly responsible for the recent popularity of neural networks. GoogLeNet, presented in [30], is the ConvNet architecture that won the ILSVRC-2014 competition (classification and detection tracks) while VGG ConvNets, presented in [29], won the localization and classification tracks of the same competition.

Although huge annotated remote sensing data are unusual, there are many works, usually using reasonable datasets (more than 2000 images), that achieved promising results by proposing the full training of new ConvNets [15,31,16]. Nogueira et al. [15] proposed and fully trained a new ConvNet architecture to classify images from aerial and multispectral images. Makantasis et al. [31] classified hyperspectral images through a new ConvNet with only two convolution layers achieving state-of-the-art in four datasets. In [16], the authors proposed a hybrid method combining principal component analysis, ConvNets and logistic regression to classify hyperspectral image using both spectral and spatial features. Moreover, a myriad of deep learning-based methods appeared exploiting its benefits in the remote sensing community [32–35].

In general, ConvNets have a peculiar property: they all tend to learn first-layer features that resemble either Gabor filters, edge detectors or color blobs. Supported by this characteristic, another strategy to exploit ConvNets is to perform fine-tuning of its parameters using the new data. Specifically, fine-tuning realizes adjustment in the parameters of a pre-trained network by resuming the training of the network from a current setting of parameters but considering a new dataset. In [36], the authors showed that fine-tuning a pre-trained ConvNet on the target data can significantly improve the performance. Specifically, they fine-tuned AlexNet [23] and outperformed results for semantic segmentation. Zhao et al. [37] fine-tuned a couple of networks outperforming state-of-the-art results in classification of traditional datasets. Several works [18,16], in the remote sensing community, also exploit the benefits of fine-tuning pre-trained ConvNets. In [18], the authors evaluated a full-trained ConvNet against a fine-tuned one to detect poverty using remote sensing data. Yue et al. [16] used a fine-tuning method to classify hyperspectral images.

Based on aforementioned characteristics, ConvNets can also be exploited as a feature extractor. Specifically, these features, usually called deep features, are obtained by removing the last classification layer and considering the output of previous layer (or layers). In some recent studies [14,37,38], ConvNets have shown to perform well even in datasets with different characteristics from the ones they were trained with, without performing any adjustment, using them as feature extractors only and using the features according to the application (e.g., classification, retrieval, etc.). In [37], the authors evaluated deep features, combined or not with other features, for classification of traditional images. In remote sensing domains, Penatti et al. [14] evaluated the use of different ConvNets as feature extractors, achieving state-of-the-art results in two remote sensing datasets, outperforming several well-known visual descriptors. Hu et al. [38] extracted features of several pre-trained ConvNets to perform classification of high-resolution remote sensing imagery.

Concerning the evaluation of different practices to exploit deep neural networks, Jarrett et al. [39] analyzed the best architecture and training protocol to explore deep neural networks, including unsupervised and supervised feature learning as well as supervised and fine-tuned classification of the instances. Also, Larochelle et al. [25] studied the best way to train neural networks, including greedy layer-wise and fine-tuning. Donahue et al. [40] evaluated the generalization power of deep features extracted from hidden layers of pre-trained ConvNets, but they did not investigate any other strategy of exploiting ConvNets, such as fine-tuning or fully-training. Specifically, they analyze if features extracted from pre-trained deep convolutional network can be repurposed to novel generic tasks, which may differ significantly from the originally trained tasks and do not have sufficient labeled data to conventionally train or adapt a deep architecture to the new tasks. Our paper differs from others in the literature since we evaluate different possible practices to exploit existing ConvNets, including full-training, fine-tuning, and feature extractor. Our work also investigates multiple ConvNets (six) and datasets (three), differing from other works that evaluate only one or a few ConvNets and datasets. To the best of our knowledge, there is no other study in the literature that performs such extensive analysis neither in remote sensing nor in other computer vision applications. Our paper also performs a set of experiments comparing the results of the different strategies mentioned using *six* existing ConvNets in three remote sensing datasets.

3. Background concepts

This section formally presents some background concepts of convolutional neural networks, or simply ConvNets, a specific type of deep learning method. These networks are generally presented as systems of interconnected processing units (neurons) which can compute values from inputs leading to an output that may be used on further units. These neurons work in agreement to solve a specific problem, learning by example, i.e., a network is created for a specific application, such as pattern recognition or data classification, through a learning process. As introduced, ConvNets were initially proposed to work over images, since they try to take leverage from the natural stationary property of an image, i.e., information extracted in one part of the image can also be applied to another region. Furthermore, ConvNets present several other advantages: (i) automatically learn local feature extractors, (ii) are invariant to small translations and distortions in the input pattern, and (iii) implement the principle of weight sharing which drastically reduces the number of free parameters and thus increases their generalization capacity. Next, we present some concept employed in ConvNets.

3.1. Processing units

As introduced, artificial neurons are basically processing units that compute some operation over several input variables and, usually, have one output calculated through the activation function. Typically, an artificial neuron has a weight vector $W = (w_1, w_2, \ldots, w_n)$, some input variables $X = (x_1, x_2, \ldots, x_n)$ and a threshold or bias b. Mathematically, vectors w and x have the same dimension, i.e., w and x are in $\mathfrak{R}^n$. The full process of a neuron may be stated as in the following equation:

$$z = f\left(\sum_{i}^{N} X_i \times W_i + b\right) \tag{1}$$

where z, x, w and b represent output, input, weights and bias, respectively. $f(\cdot)$: $\mathfrak{R} \rightarrow \mathfrak{R}$ denotes an activation function.

Conventionally, a nonlinear function is provided in $f(\cdot)$. There are a lot of alternatives for $f(\cdot)$, such as sigmoid, hyperbolic, and rectified linear function. The latter function is currently the most

used in the literature. Neurons with this configuration have several advantages when compared to others: (i) work better to avoid saturation during the learning process, (ii) induce the sparsity in the hidden units, and (iii) do not face gradient vanishing problem[2] as with sigmoid and tanh function. The processing unit that uses the rectifier as activation function is called Rectified Linear Unit (ReLU) [41]. The first step of the activation function of a ReLU is presented in Eq. (1) while the second one is introduced in the following equation:

$$a = \begin{cases} z, \, if z > 0 \\ 0, \, otherwise \end{cases} \quad \Leftrightarrow \quad a = f(z) = \max(0, z) \tag{2}$$

The processing units are grouped into layers, which are stacked forming multilayer networks. These layers give the foundation to others, such as convolutional and fully-connected layers.

3.2. Network components

Among the different types of layers, the convolutional one is responsible for capturing the features from the images. The first layers usually obtain low-level features (like edges, lines and corners) while the others get high-level features (like structures, objects and shapes). The process made in this type of layer can be decomposed into two phases: (i) the convolution step, where a fixed-size window runs over the image, with some stride,[3] defining a region of interest and (ii) the processing step uses the pixels inside each window as input for the neurons that, finally, perform the feature extraction from the region. Formally, in the latter step, each pixel is multiplied by its respective weight generating the output of the neuron, just like Eq. (1). Thus, only one output is generated concerning each region defined by the window. This iterative process results in a new image (or feature map), generally smaller than the original one, with the visual features extracted. Many of these features are very similar, since each window may have common pixels, generating redundant information. Typically, after each convolutional layer, there are pooling layers that were created in order to reduce the variance of features by computing some operation of a particular feature over a region of the image. Specifically, a fixed-size window runs over the features extracted by the convolutional layer and, at each step, an operation is realized to minimize the amount and optimize the gain of the features. Two operations may be realized on the pooling layers: the max or average operation, which selects the maximum or mean value over the feature region, respectively. This process ensures that the same result can be obtained, even when image features have small translations or rotations, being very important for object classification and detection. Thus, the pooling layer is responsible for sampling the output of the convolutional one preserving the spatial location of the image, as well as selecting the most useful features for the next layers.

After several convolutional and pooling layers, there are the fully-connected ones, which take all neurons in the previous layer and connect them to every single neuron in its layer. The previous layers can be convolutional, pooling or fully-connected, however the next ones must be fully-connected until the classifier layer, because the spatial notion of the image is lost in this layer. Since a fully-connected layer occupies most of the parameters, overfitting can easily happen. To prevent this, the dropout method [42] was employed. This technique randomly drops several neuron outputs, which do not contribute to the forward pass and back propagation anymore. These drops are equivalent to decreasing the number of neurons of the network, improving the speed of training and making model combination practical, even for deep networks. Although this method creates networks with different architectures, those networks share the same weights, permitting model combination and allowing that only one network is needed at test time.

Finally, after all convolution, pooling and fully-connected layers, a classifier layer may be used to calculate the class probability of each instance. The most common classifier layer is the softmax one [13], based on the namesake function. The softmax function, or normalized exponential, is a generalization of the multinomial logistic function that generates a K-dimensional vector of real values in the range $(0, 1)$ which represents a categorical probability distribution. Eq. (3) shows how softmax function predicts the probability for the jth class given a sample vector X.

$$h_{W,b}(X) = P(y = j | X; W, b) = \frac{\exp^{X^T W_j}}{\sum_{k=1}^{K} \exp^{X^T W_k}} \tag{3}$$

where j is the current class being evaluated, X is the input vector, and W represent the weights.

In addition to all these processing layers, there are also normalization ones, such as Local Response Normalization (LRN) [23] layer. This is the most useful one when using processing units with unbounded activations (such as ReLU), because it permits the local detection of high-frequency features with a big neuron response, while damping responses are uniformly large in a local neighborhood.

3.3. Training

After modeling a network, in order to allow the evaluation and improvement of its results, a loss function needs to be defined, even because the goal of the training is to minimize the error of this function, based on the weights and bias, as presented in Eq. (4). Among several functions, the log loss one has become more pervasive because of exciting results achieved in some problems [23]. Eq. (5) presents a general log loss function, without any regularization (or weight decay) term, which is used to prevent overfitting.

$$\underset{W,b}{\operatorname{argmin}} [\mathcal{J}(W, b)] \tag{4}$$

$$\mathcal{J}(W, b) = -\frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} \times \log h_{W,b}\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \times \log\left(1 - h_{W,b}\left(x^{(i)}\right)\right)\right) \tag{5}$$

where y represents a possible class, x is the data of an instance, W is the weights, i is a specific instance, and N represents the total number of instances.

With the cost function defined, the ConvNet can be trained in order to minimize the loss by using some optimization algorithm, such as Stochastic Gradient Descent (SGD), to gradually update the weights and bias in search of the optimal solution:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial \mathcal{J}(W, b)}{\partial W_{ij}^{(l)}}$$

$$b_{i}^{(l)} = b_{i}^{(l)} - \alpha \frac{\partial \mathcal{J}(W, b)}{\partial b_{i}^{(l)}}$$

where α denotes the learning rate, a parameter that determines how much an updating step influences the current value of the weights, i.e., how much the model learns in each step.

[2] The gradient vanishing problem occurs when the propagated errors become too small and the gradient calculated for the back propagation step vanishes, making it impossible to update the weights of the layers and to achieve a good solution.

[3] Stride is the distance between the centers of each window considering two steps.

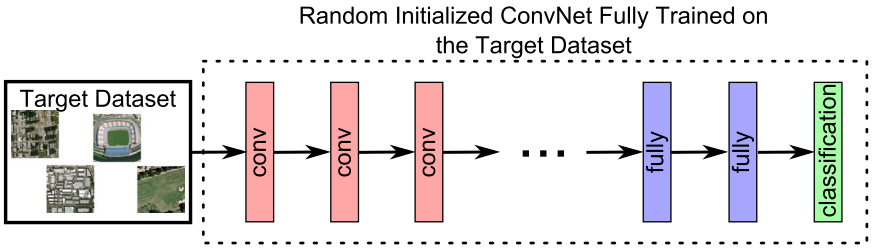


Fig. 1. Illustrative example of a ConvNet being fully trained. Weights from the whole network are randomly initialized and then trained for the target dataset.

However, as presented, the partial derivatives of the cost function, for the weights and bias, are needed. To obtain these derivatives, the back propagation algorithm is used. Specifically, it must calculate how the error changes as each weight is increased or decreased slightly. The algorithm computes each error derivative by first computing the rate at which the error δ changes as the activity level of a unit is changed. For classifier layers, this error is calculated considering the predicted and desired output. For other layers, this error is propagated by considering the weights between each pair of layers and the error generated in the most advanced layer.

The training step of a ConvNet occurs in two steps: (i) the feed-forward one, that passes the information through all the network layers, from the first until the classifier one, usually with high batch size[4] and (ii) the back propagation one, which calculates the error δ generated by the ConvNet and propagates this error through all the layers, from the classifier until the first one. As presented, this step also uses the errors to calculate the partial derivatives of each layer for the weights and bias.

4. Strategies for exploiting ConvNets

This section aims at explaining the most common strategies of employing existing ConvNets in different scenarios from the ones they were trained for. As introduced, training a deep network from scratch requires a considerable amount of data as well as a lot of computational power. In many problems, few labeled data is available, therefore training a new network is a challenging task. Hence, it is common to use a pre-trained network either as a fixed feature extractor for the task of interest or as an initialization for fine-tuning the parameters. We describe these strategies, including their advantages and disadvantages, in the next subsections. Section 4.1 describes about the full training of a new network while Section 4.2 presents the fine-tuning process. Finally, Section 4.3 explains the use of deep ConvNets as a fixed feature extractor.

4.1. Full-trained network

The strategy to train a network from scratch (with random initialization of the filter weights) is the first one to be thought when training ConvNets. This process is useful when the dataset is large enough to make a network converge and has several advantages, such as (i) extractors tuned specifically for the dataset, which tend to generate more accurate features, and (ii) full control of the network. However, fully training a network from scratch requires a lot of computational and data resources [12,13], since the convergence of network is pruned to overfitting. Also, the convergence of a ConvNet cannot be totally confirmed (given its highly non-convex property) making tuning not so trivial.

There are basically two options of training a deep network that we can fit in the case of full training. The first one is by fully designing and training a new architecture, including number of layers, neurons and type of activations, the number of iterations, weight decay, learning rate, etc. The other option is by using an existing architecture and fully training its weights to the target dataset of any size. In this last case, the architecture and parameters (weight decay, learning rate, etc.) are not modified at all.

Full training is expected to achieve better results when compared to other strategies, since the deep network learns specific features for the dataset of interest. In this paper, we do not evaluate the option of creating a whole new ConvNet, because of challenges presented above. Instead, we consider the full training of existing architectures in new datasets. This strategy is illustrated in Fig. 1.

4.2. Fine-tuned network

When the new dataset is reasonable large, but not enough to full train a new network, fine-tuning is a good option to extract the maximum effectiveness from pre-trained deep ConvNets, since it can significantly improve the performance of the final classifier. Fine-tuning is based on a curious property of modern deep neural network: they all tend to learn first-layer features that resemble either Gabor filters, edge or color blob detectors, independently of the training data. More specifically, earlier layers of a network contain generic features that should be useful to many tasks, but later layers become progressively more specific to the details of the classes contained in the original dataset (i.e., the dataset in which the deep ConvNet was originally trained). Supported by this property, the initial layers can be preserved while the final ones should be adjusted to suit the dataset of interest.

Fine-tuning performs a fine adjustment in the parameters of a pre-trained network by resuming the training of the network from a current setting of parameters but considering a new dataset of any size, aiming at accuracy improvements. In other words, fine-tuning uses the parameters learned from a previous training of the network on a specific dataset and, then, adjusts the parameters from the current state for the new dataset, improving the performance of the final classifier.

Based on aforementioned characteristics, there are two possible approaches of performing fine-tuning in a pre-trained network, both exploited in this work: (i) fine-tune all layers, and (ii) fine-tune only higher-level layers keeping some of the earlier layers fixed (due to overfitting concerns). It is important to emphasize that in both scenarios, the search space is bounded to just small variations in each step, since the learning rate is initialized with reduced value. Specifically, in the first case, some layers (usually the final ones, such as the classification layer, since the number of classes tends to be different) have weights ignored, being randomly initialized. These layers have the learning rate increased, so they can learn faster and converge, while the other layers may also change weights by very small variations, since they use the reduced value of the learning rate without any augmentation. By doing this, the first layers can use the information

[4] Batch size is a parameter that determines the number of images that goes through the network before the weights and bias are updated.

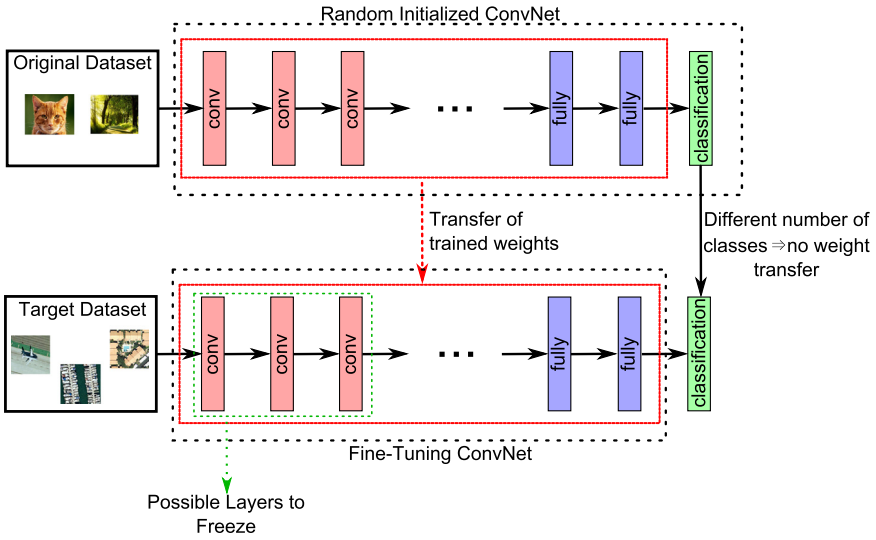


Fig. 2. Illustrative example of two options for the fine-tuning process. In one of them (highlighted in red), all layers are fine-tuned according to the target dataset, but final layers have increased learning rates. In the other option (highlighted in green), weights of initial layers can be frozen and only final layers are tuned. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.).

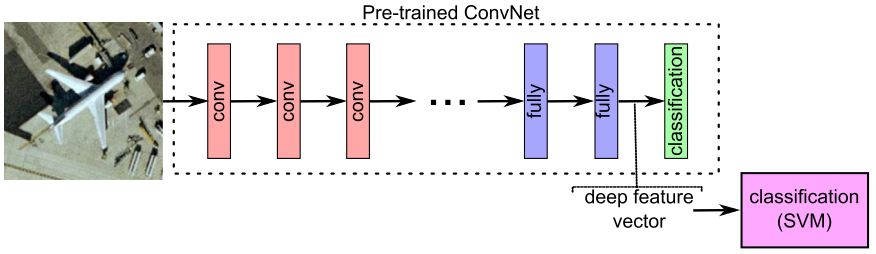


Fig. 3. Illustrative example of the use of a ConvNet as feature extractor. The final classification layer is ignored and one should only choose from which layer to consider the features. The figure shows the use of the features from the last layer before the classification layer, which is commonly used in the literature.

previously learned with just few adjustments to the dataset of interest, and at the same time, the final layers can really learn based only on the new dataset. In the second case, the initial layers are frozen to keep the generic features already learned, while the final layers are adjusted using the increased value of the learning rate.

These two options of fine tuning are illustrated in Fig. 2.

4.3. ConvNet as a feature extractor

A pre-trained network can be used as a feature extractor for any image, since the generic features (learned in earlier layers) are less dependent on the final application and could be used in a myriad of tasks. Specifically, features (usually, called deep features) can be extracted from any layer of a pre-trained network and then used in a given task. Deep features trained on ImageNet (a dataset of everyday objects) have already shown remarkable results in applications like flower categorization [43], human attribute detection [44], bird sub-categorization [45], scene retrieval [46], and many others [37,36], including remote sensing [14,38]. Furthermore, Razavian et al. [47] suggest that features obtained from deep learning should be the primary candidate in most visual recognition tasks.

The strategy of using pre-trained ConvNets as feature extractors is very useful given its simplicity, since no retraining or tuning is necessary. Moreover, one only needs to select the layer to be used, extract the deep features and use them combined with some machine learning technique, in the case of a classification setup. According to previous works [14,47,37], deep features can be extracted from the last layer before the classification layer (usually, a fully-connected one) and, then, used to train a linear classifier, which is the strategy employed in this paper.

Fig. 3 illustrates how to use an existing ConvNet as a feature extractor.

5. Experimental setup

The main objective of this paper is to evaluate which are the most suitable strategies to better exploit the power of deep features for remote sensing image scenes classification. The details about the experiments conducted are presented in the following subsections. The datasets are presented in Section 5.1. Section 5.2 describes the low-level (global) and mid-level (BoVW) descriptors used while Section 5.3 presents the evaluated ConvNets. Finally, Section 5.4 presents the protocol used in the experiments.

5.1. Datasets

We have chosen remote sensing datasets with different visual properties in order to better evaluate the robustness and effectiveness of each strategy. The first one, presented in Section 5.1.1, is a multi-class land-use dataset that contains aerial high resolution scenes in the visible spectrum. The second one, presented in Section 5.1.2, is a multi-class high resolution dataset with images collected from different regions all around the world. The last one, presented in Section 5.1.3, has multispectral high-resolution scenes of coffee crops and non-coffee areas.

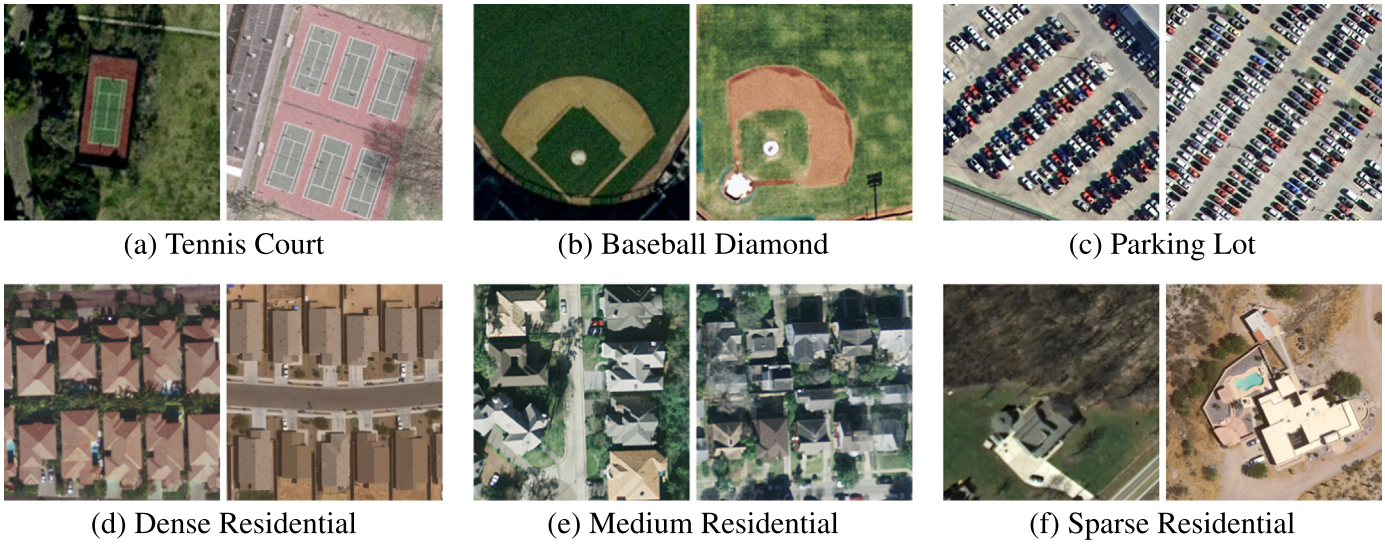


Fig. 4. Examples of the UCMerced land use dataset.

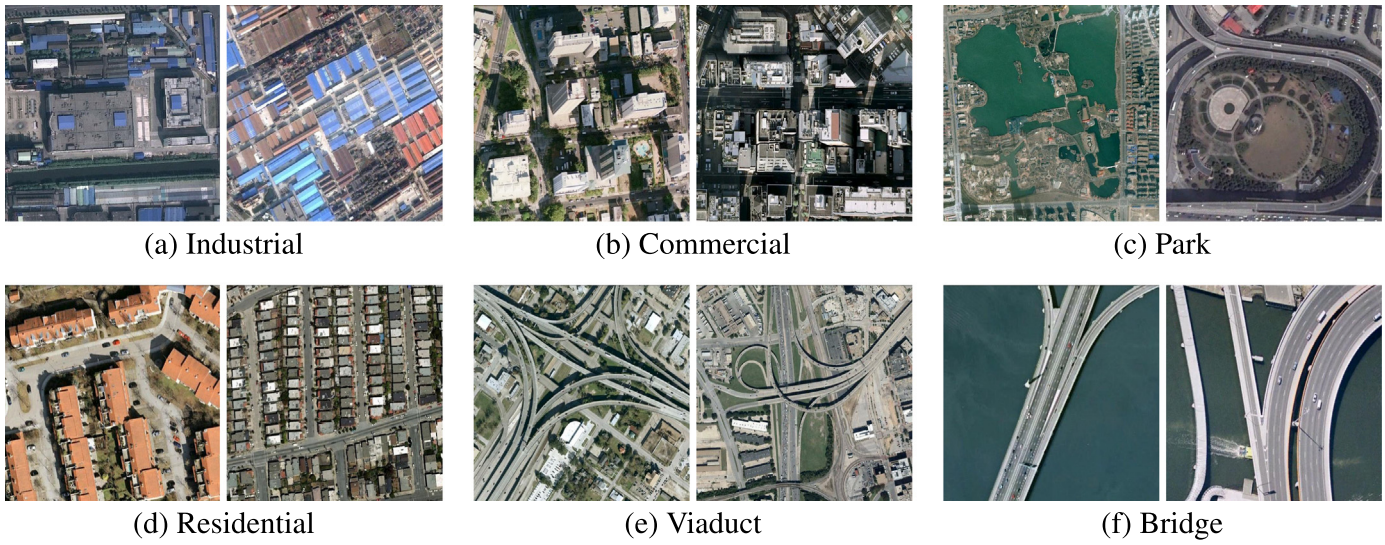


Fig. 5. Examples of the RS19 dataset.

5.1.1. UCMerced land-use

This manually labeled and publicly available dataset [48] is composed of 2100 aerial scene images with 256×256 pixels equally divided into 21 land-use classes selected from the United States Geological Survey (USGS) National Map. Thus, these images were obtained from different US locations, providing diversity to the dataset. The 21 categories are: agricultural, airplane, baseball diamond, beach, buildings, chaparral, dense residential, forest, freeway, golf course, harbor, intersection, medium density residential, mobile home park, overpass, parking lot, river, runway, sparse residential, storage tanks, and tennis courts. Some class samples are shown in Fig. 4. It is remarkable that the overlapping of some classes, such as “dense residential”, “medium residential” and “sparse residential”, mainly differ in the density of structures.

5.1.2. RS19 dataset

This public dataset [49] contains 1005 high-spatial resolution images with 600×600 pixels divided into 19 classes, with approximately 50 images per class. Exported from Google Earth, which provides high-resolution satellite images up to half a meter, this dataset has samples collected from different regions all around the world, which increases its diversity but creates challenges due to the changes in resolution, scale, orientation and illumination of the images. There are 19 classes, including: airport, beach, bridge, river, forest, meadow, pond, parking, port, viaduct, residential area, industrial area, commercial area, desert, farmland, football field, mountain, park and railway station. Fig. 5 presents examples of some classes.

5.1.3. Brazilian Coffee Scenes

This dataset [14] is composed of multi-spectral scenes taken by the SPOT sensor in 2005 over four counties in the State of Minas Gerais, Brazil: Arceburgo, Guaranésia, Guaxupé, and Monte Santo. Images of each county were partitioned into multiple tiles of 64×64 pixels, which generated 2876 images equally divided into 2 classes (coffee and non-coffee). Some samples of this dataset are presented in Fig. 6. It is important to emphasize that these images are composed of green, red, and near-infrared bands, which are the most useful and representative ones for discriminating vegetation areas.

This dataset is very challenging for several different reasons: (i) high intraclass variance, caused by different crop management

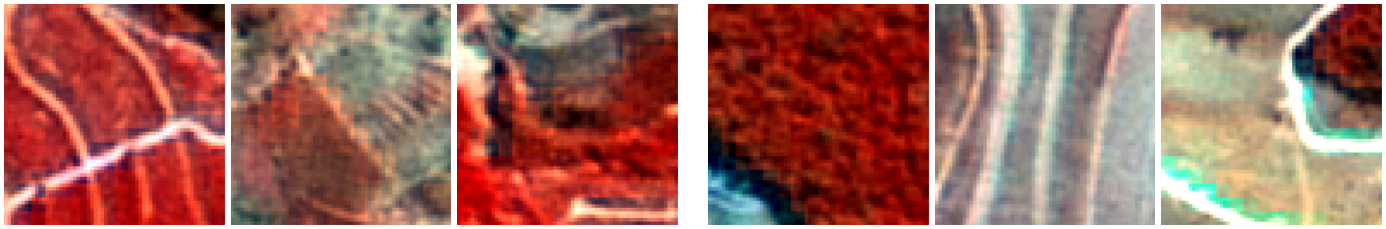


Fig. 6. Examples of coffee and non-coffee samples in the Brazilian Coffee Scenes dataset. The similarity among samples of opposite classes is notorious. The intraclass variance is also perceptive. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

techniques, (ii) scenes with different plant ages, since coffee is an evergreen culture, and (iii) images with spectral distortions caused by shadows, since the South of Minas Gerais is a mountainous region.

5.2. Classical feature extraction strategies

Several previously state-of-the-art descriptors have been selected to be evaluated, based on preceding works [26,2,7,50–52], in which they were evaluated for remote sensing image classification, texture and color image retrieval/classification, and web image retrieval. Our selection includes simple global low-level descriptors, like descriptors based on color histograms and variations, and also descriptors based on bags of visual words (BoVW).

5.2.1. Low-level descriptors

There is a myriad of descriptors available in the literature [50] that can be used to represent visual elements. Clearly, different descriptors may provide distinct information about images producing contrastive results. Thus, we tested a diverse set of 7 descriptors based on color, texture, and gradient properties, in order to extract visual features from each image and evaluate the potential of deep features when compared them. The global low-level descriptors considered are: Auto-Correlogram Color (*ACC*) [53], Border/Interior Pixel Classification (*BIC*) [54], Local Color Histogram (*LCH*) [55], Local Activity Spectrum (*LAS*) [56], Statistical Analysis of Structural Information (*SASI*) [57], Histogram of Oriented Gradients (*HOG*) [58], and *GIST* [59].

The implementations of ACC, BIC, LCH, SASI, and LAS descriptors follow the specifications of [50]. GIST implementation is the one used in [60] with the parameters discussed therein.[5] The implementation of Histogram of Oriented Gradients (HOG) was obtained from the VLFeat framework [61]. We used HOG in different configurations, varying the cell size in 14×14, 20×20, 40×40 and 80×80 pixels, but keeping the orientation binning in 9 bins.

5.2.2. Mid-level descriptors

Bag of visual words (*BoVW*) and their variations [5,6,8,62–64,9] are considered mid-level representations, since these methods create a codebook of visual discriminating patches (visual words), and then, compute statistics (using the codebook) about the visual word occurrences in the test image. BoVW descriptors have been the state-of-the-art for several years in the computer vision community and are still important candidates to perform well in many tasks.

We tested BoVW in several different configurations:

- Sampling: sparse (Harris–Laplace detector) or dense (grid of circles with 6 pixels of radius)
- Low-level descriptor: SIFT and OpponentSIFT [7]
- Visual codebooks of size: 100, 1000, 5000, and 10,000
- Coding: hard or soft (with $\sigma = 90$ or 150)
- Pooling: average, max pooling or WSA [63].

To differentiate them in the experiments, we used the following naming: BX^{w}_{cp}, where X is S (sparse sampling) or D (dense sampling); w is the codebook size; c refers to the coding scheme used, h (hard), s (soft); p refers to the pooling technique used, a (average), m (max), or W (WSA).

The low-level feature extraction of BoVW descriptors was based on the implementation of van de Sande et al. [65]. For BoVW, in UCMerced and RS19 datasets, we used SIFT [66] to describe each patch, but in the Brazilian Coffee dataset, we used OpponentSIFT [7], as color should provide more discriminating power.

Table 1
Some statistics about the deep networks evaluated in this work.

Networks	# Parameters (millions)	# Connections (millions)
OverFeat$_S$	145	2810
OverFeat$_L$	144	5369
AlexNet	60	630
CaffeNet	60	630
GoogLeNet	5	>10, 000
VGG$_{16}$	138	4096
PatreoNet	15	200

5.3. ConvNets

As the main goal of the paper is to evaluate the strategies for better exploiting existing deep ConvNets, we selected some of the most popular ConvNets available nowadays. All networks presented in this section, except for the OverFeat ones, were implemented in Convolutional Architecture for Fast Feature Embedding [67], or simply Caffe, a fully open-source framework that affords clear and easy implementations of deep architectures. The OverFeat ConvNets were proposed and implemented by the namesake framework [68]. Furthermore, as introduced, all networks exploited in this work, for fine-tuning and feature extractors, were pre-trained on the ImageNet 2012 training set [69]. It is worth mentioning that we also evaluated three ConvNets (the ones with publicly available pre-trained models, which includes AlexNet [23], GoogLeNet [30] and *VGG*$_{16}$ [29]) trained on Places205 dataset [70]. However, we chose not to report these results, since the results with the ConvNets trained on ImageNet, reported in this paper, were still superior.

Some information about the networks evaluated in this work is presented in Table 1. GoogLeNet [30] is the biggest network, with higher number of connections, followed OverFeat$_L$ and *VGG*$_{16}$. In fact, these networks, OverFeat and *VGG*$_{16}$, are also the ones with higher number of parameters, which require more memory during

[5] http://lear.inrialpes.fr/software (as of March 14th, 2015).

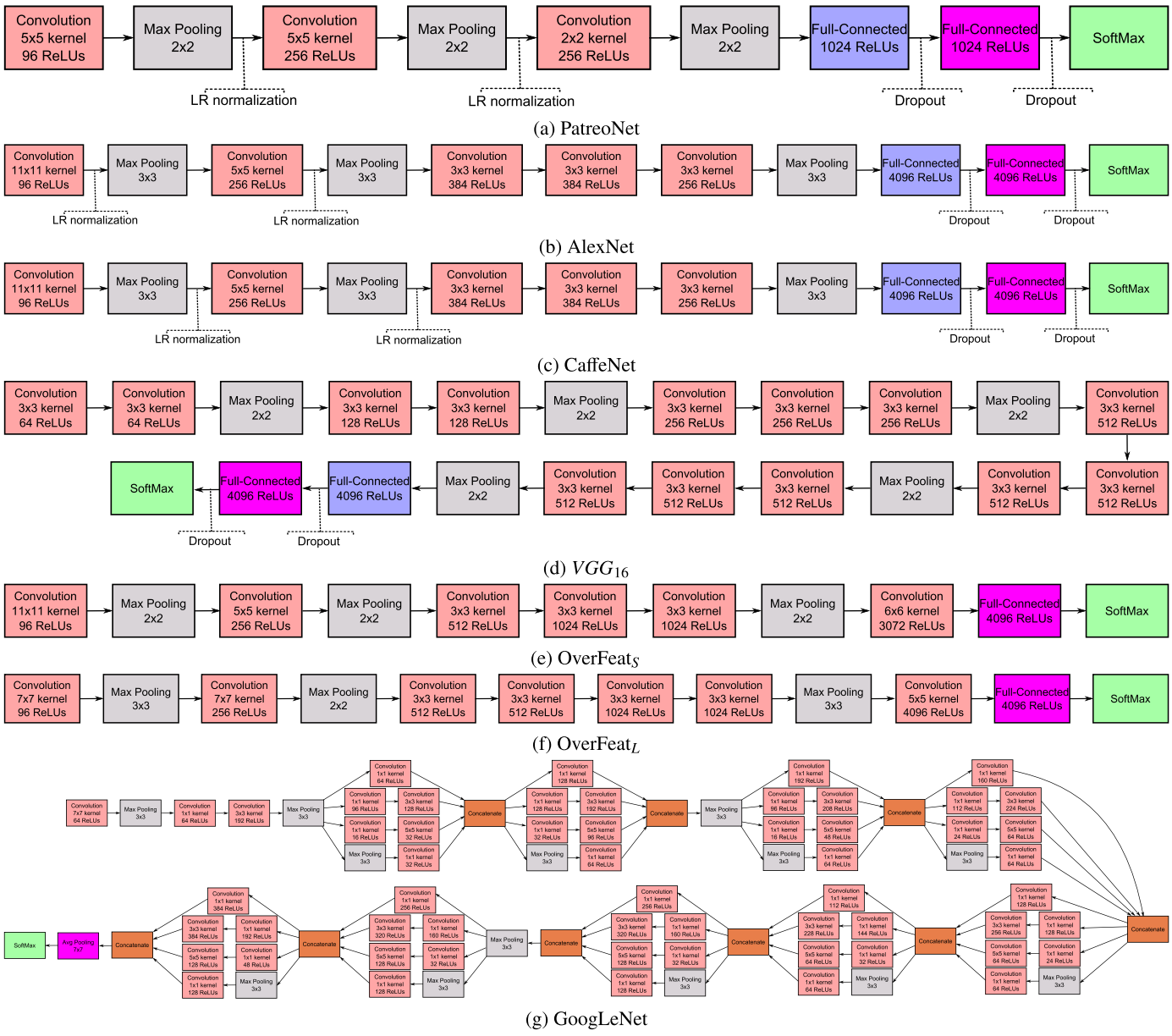


Fig. 7. Architectures of different ConvNets evaluated in this work. Purple boxes indicate the layers from where features were extracted in the case of using the ConvNets as feature extractors. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.).

the training process. We describe some properties of each ConvNet in the following subsections.

5.3.1. *PatreoNet*

PatreoNet, presented in [15], is a network capable of learning specific spatial features from remote sensing images, without any pre-processing step or descriptor evaluation. This network, which architecture can be seen in Fig. 7a, has 3 convolutional layers, 3 pooling ones and 3 fully-connected ones (considering the softmax). PatreoNet was only used in full-training strategy, since it has no model pre-trained in large datasets.

5.3.2. *AlexNet*

AlexNet, proposed by Krizhevsky et al. [23], was the winner of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [28] in 2012. This ConvNet, that has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final softmax. Its final architecture can be seen in Fig. 7b. It was a breakthrough work since it was the first to employ non-saturating neurons, GPU implementation of the convolution operation and dropout to prevent overfitting.

In the experiments, AlexNet was used as a feature extractor network by extracting the features from the last fully-connected layer (purple one in Fig. 7b), which results in a feature vector of 4096 dimensions. AlexNet was also fine-tuned for all the datasets used in the experiments. For UCMerced and RS19 datasets the network was fine-tuned by giving more importance to the final softmax layer (without freezing any layer), while for the Coffee Dataset, the first three layers were frozen and the final ones participate normally in the learning process. Finally, AlexNet was fully trained from scratch for all datasets under the same configurations of the original model [23].

5.3.3. *CaffeNet*

CaffeNet [67] is almost a replication of AlexNet [23] with some important differences: (i) training has no relighting data-augmentation and (ii) the order of pooling and normalization layers is

switched (in CaffeNet, pooling is done before normalization). Thus, this network, which architecture can be seen in Fig. 7c, has the same number of parameters, neurons and layers of the AlexNet. Given its similarity to the AlexNet architecture, in the experiments, CaffeNet were exploited in the same way of the aforementioned network.

5.3.4. GoogLeNet

GoogLeNet, presented in [30], is the ConvNet architecture that won the ILSVRC-2014 competition (classification and detection tracks). Its main peculiarity is the use of inception modules, which reduce the complexity of the expensive filters of traditional architectures allowing multiple filter, with different resolutions, to be used in parallel. GoogLeNet has two main advantages: (i) utilization of filters of different sizes at the same layer, which maintains more spatial information and (ii) reduction of the number of parameters of the network, making it less prone to overfitting and allowing it to be deeper. In fact, GoogLeNet has 12 times fewer parameters than AlexNet, i.e., approximately 5 millions of parameters. Specifically, the 22-layer GoogLeNet architecture, which can be seen in Fig. 7g, has more than 50 convolutional layers distributed inside the inception modules.

In our experiments, GoogLeNet was used as a feature extractor network by extracting the features from the last pooling layer (purple one in Fig. 7g), which results in a feature vector of 1024 dimensions. GoogLeNet was fine-tuned for all datasets just like AlexNet, with exact the same strategies for each dataset. Finally, GoogLeNet was fully trained from scratch for all datasets under the same configurations of the original model [30].

5.3.5. VGG ConvNets

VGG ConvNets, presented in [29], won the localization and classification tracks of the ILSVRC-2014 competition. Several networks have been proposed in this work, but two have become more successful: VGG_{16} and VGG_{19}. Giving the similarity of both networks, we choose to work with the former one because of its simpler architecture and slightly better results. However, similar results obtained by VGG_{16} should be also yielded by VGG_{19}. This network, which architecture can be seen in Fig. 7d, has 13 convolutional layers, 5 pooling ones and 3 fully-connected ones (considering the softmax).

VGG_{16} was used as a feature extractor network by extracting the features from the last fully-connected layer (purple one in Fig. 7d), which results in a feature vector of 4096 dimensions. VGG_{16} was also fine-tuned in the UCMerced and RS19 datasets by giving more importance to the final softmax layer, without freezing any layer. However, this network could not be fine-tuned for the Brazilian Coffee dataset (the most different one) as well as could not be fully trained from scratch for any dataset. This problem is due to the large amount of memory required by this network, as presented in Table 1, which allows only small values of batch size to be used during the training process. Since larger values of batch size, combined with other parameters (weight decay, learning rate), help the convergence of a ConvNet in the training process [12,13], there was no convergence in the aforementioned scenarios.

5.3.6. OverFeat ConvNets

OverFeat [71], a deep learning framework focused on ConvNets and winner of the detection track of ILSVRC 2013, has two ConvNet models available, which can be used to extract features and/or to classify images. There is a small (*fast* – OverFeat_S) and a larger network (*accurate* – OverFeat_L), both having similarities with AlexNet [23]. The main differences are: (i) no response normalization and (ii) non-overlapping pooling regions. OverFeat_L, whose architecture can be seen in Fig. 7f, has more differences including: (i) one more convolutional layer and (ii) the number and size of feature maps, since different number of kernels and stride were used for the convolutional and the pooling layers. In the other way around, OverFeat_S, whose architecture can be seen in Fig. 7e, is more similar to the AlexNet, differing only in the number and size of feature maps. The main differences between the two OverFeat networks are the stride of the first convolution, the number of stages and the number of feature maps [71].

These ConvNets are only used as feature extractor, since no model was provided in order to perform fine-tuning or full-training of the networks. Considering this strategy, a feature vector of 4096 dimensions is obtained from the last fully-connected layer, which are illustrated as purple layers in Fig. 7e and f, for OverFeat_S and OverFeat_L, respectively.

Table 2
Parameters utilized in fine-tuning and full-training strategies.

Strategy	# Iterations	Learning rate
Fine-tuning	20,000	0.001
Full-training	50,000	0.01

5.4. Experimental protocol

We carried out all experiments with a 5-fold cross-validation protocol. Therefore, the dataset was arranged into fivefolds with almost same size, i.e., the images are almost equally divided into five non-overlapping sets. Specifically, the UCMerced and RS19 datasets have 5-folds, unbalanced in terms of the number of samples per class, with 420 and 201 images, respectively. For the Brazilian Coffee Scenes dataset, 4-folds have 600 images each and the 5th has 476 images, all folds are balanced with coffee and non-coffee samples (50% each).

When performing fine-tuning or training a network for scratch, at each run, three-folds are used as training-set, one as validation (used to tune the parameters of the network) and the remaining one is used as test-set. It is important to mention that when changing the folds which are train, validation and test (during the cross-validation process), the full training or fine tuning of the network starts from the beginning. Five different networks are obtained, one for each step of the 5-fold cross-validation process. That is, there is no contamination of the training set with testing data. When using the ConvNets as feature extractors, four sets are used as training while the last is the test-set. Still considering this strategy, we always used linear SVM as the final classifier.

When fine-tuning or full-training a network, we basically preserve the parameters of the original author, varying only two according to Table 2. It is important to highlight that there is no training when using a pre-trained ConvNet (without fine-tuning) as feature extractor, thus there no parameters to vary.

The results are reported in terms of average accuracy and standard deviation among the 5-folds. For a given fold, we compute the accuracy for each class and then compute the average accuracy among the classes. This accuracy is used to compute the final average accuracy among the 5-folds.

All experiments were performed on a 64 bits Intel i7 4960X machine with 3.6 GHz of clock and 64 GB of RAM memory. Two GPUs were used: a GeForce GTX770 with 4 GB of internal memory and a GeForce GTX Titan X with 12 GB of memory, both under a 7.5 CUDA version. Ubuntu version 14.04.3 LTS was used as an operating system.

6. Results and discussion

In this section, we present and discuss the experimental results. Firstly, we discuss the power of generalization of ConvNets

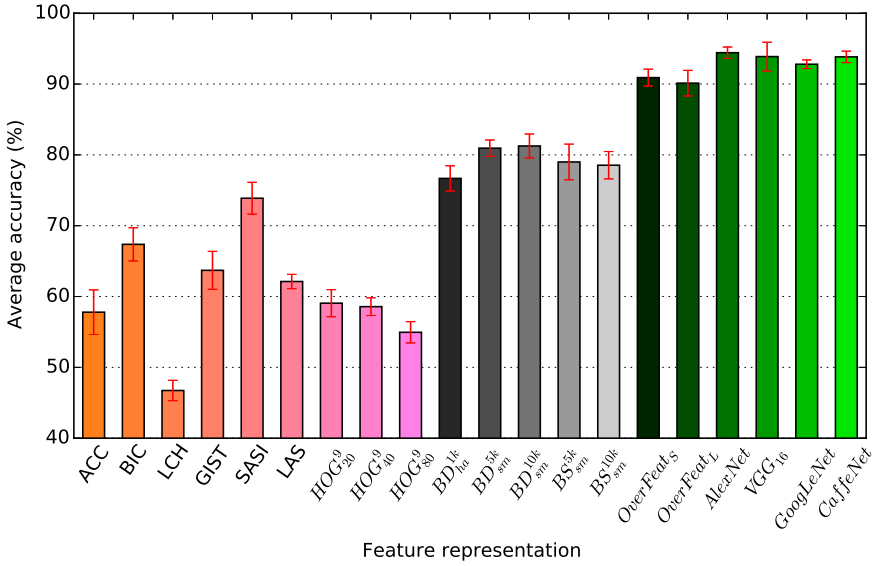


Fig. 8. Average accuracy of pre-trained ConvNets used as feature extractors and low- and mid-level descriptors for the UCMerced land-use dataset.

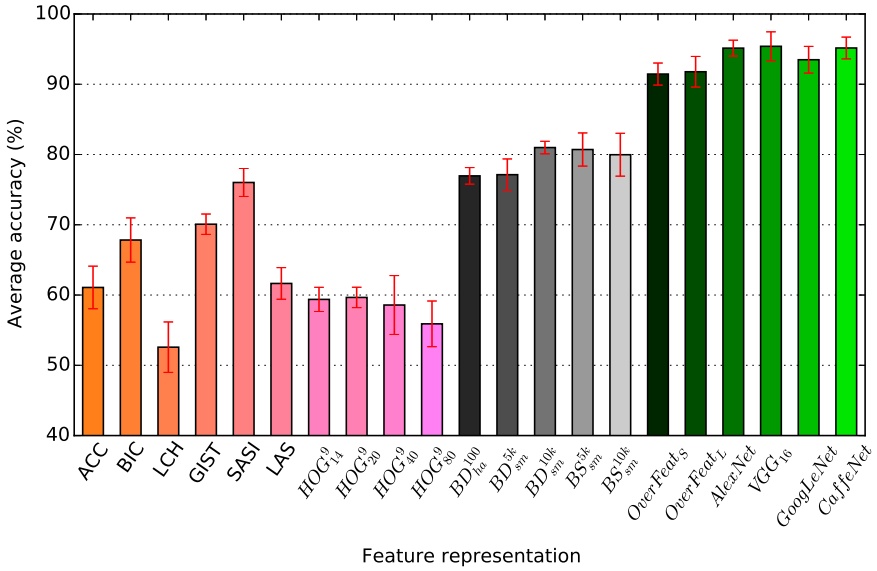


Fig. 9. Average accuracy of pre-trained ConvNets used as feature extractors and low- and mid-level descriptors for the RS19 Dataset.

as feature descriptors and compare them with low-level and mid-level representations (Section 6.1). Then, we compare the performance of the three different strategies for exploiting the existing ConvNets (Section 6.2). Finally, we compare the most accurate ConvNets against some of the state-of-the-art methods for each dataset (Section 6.3).

6.1. Generalization power evaluation

In this subsection, we compare six pre-trained ConvNets used as descriptors against low-level and mid-level representations for aerial and remote sensing scene classification. We conducted several experiments in order to evaluate the best BoVW configurations, but only the top-5 best were reported. It is also important to highlight that the ConvNets results in this subsection refer to their use as feature extractors, by considering the features of the last layer before softmax as input for a linear SVM. So, the original network was not used to classify the samples. The objective is to observe how deep features perform in datasets from different domains they were trained.

In Fig. 8, we show the average accuracy of each descriptor and ConvNet for the UCMerced dataset. We can notice that ConvNet features achieve the highest accuracy rates ($\geq 90\%$). CaffeNet, AlexNet, and VGG_{16} yield the highest average accuracies (more than 93%). GoogLeNet achieves $92.80 \pm 0.61\%$ and OverFeat achieves $90.91 \pm 1.19\%$ for the small and $90.13 \pm 1.81\%$ for the large network. SASI is the best global descriptor ($73.88 \pm 2.25\%$), while the best BoVW configurations are based on dense sampling, 5 or 10 thousand visual words and soft assignment with max pooling (~81%).

In Fig. 9, we show the average accuracies for the RS19 dataset. ConvNets again achieved the best results ($\geq 90\%$). The best global descriptor was again SASI and the best BoVW configurations have 5 or 10 thousand visual words with soft assignment and max pooling, but in this dataset, sparse sampling also achieved similar accuracies to dense sampling.

The results with UCMerced and RS19 datasets illustrate the capacity of ConvNet features to generalize to the aerial domain.

In Fig. 10, we show the average accuracies for the Brazilian Coffee Scenes dataset. In this dataset, the results are different from

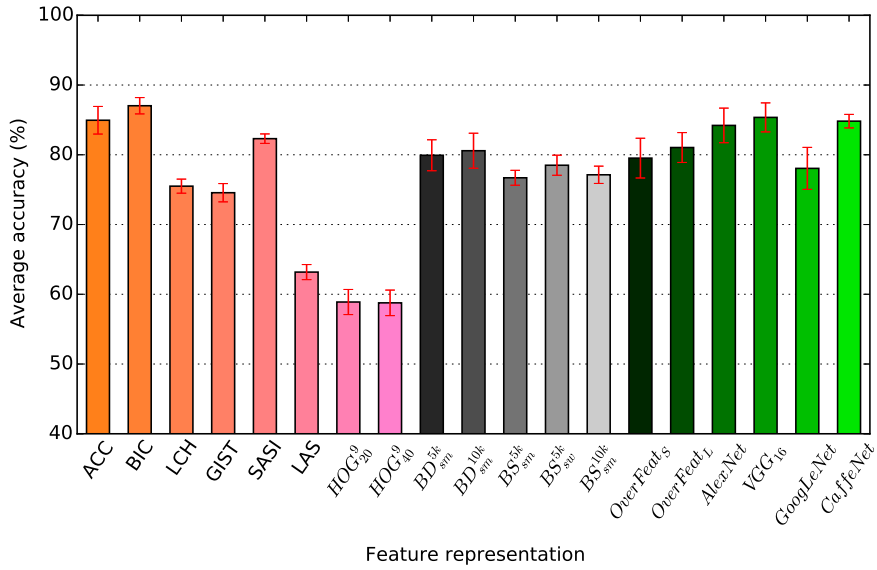


Fig. 10. Average accuracy of pre-trained ConvNets used as feature extractor and low- and mid-level descriptors for the Brazilian Coffee Scenes dataset.

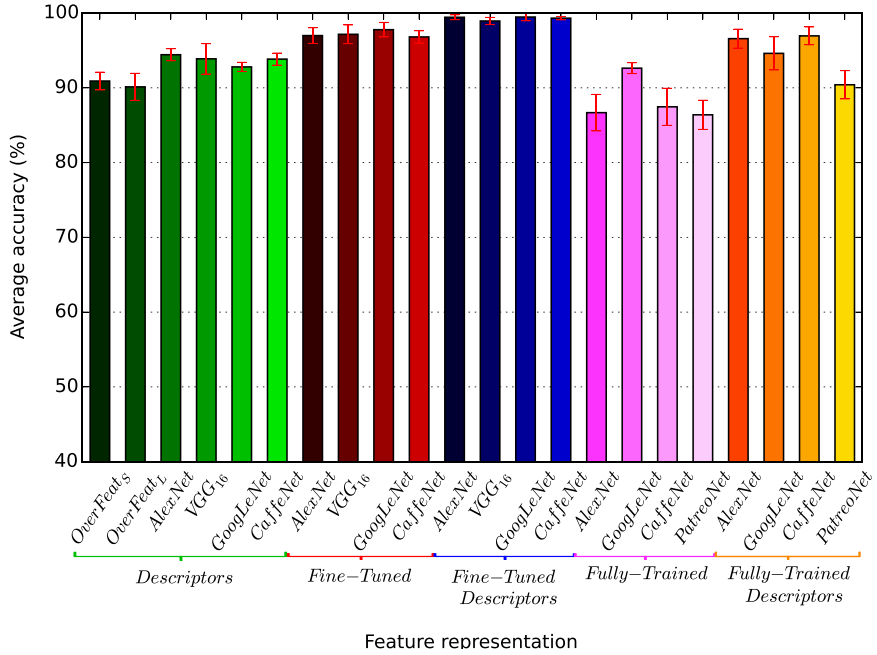


Fig. 11. Average accuracy considering all possible strategies to exploit ConvNets for the UCMerced Land-use dataset. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.).

in the other two datasets already mentioned. We can see that, although most of the ConvNets achieve accuracy rates above 80%, with VGG_{16} achieving $85.36 \pm 2.08\%$, BIC and ACC also achieved high accuracies. BIC achieved the highest average accuracy ($87.03 \pm 1.17\%$) outperforming all the descriptors including the ConvNets in this dataset. The BIC algorithm for classifying pixels in border or interior basically separates the images into homogeneous and textured regions. Then, a color histogram is computed for each type of pixel. As for the Brazilian Coffee Scenes dataset, the differences between classes may be not only in texture but also in color properties, BIC could encode well such differences. The best BoVW configurations are again based on dense sampling, 5 or 10 thousand visual words and soft assignment with max pooling, and they have comparable results to OverFeat.

A possible reason for the deep features to perform better in aerial dataset than in the agricultural one is due to the particular intrinsic properties of each dataset. The aerial datasets have more complex scenes, composed of a lot of small objects (e.g., buildings, cars, airplanes). Many of these objects are composed of similar visual patterns in comparison with the ones found in the dataset used to train the ConvNets, with salient edges and borders.

Concerning the Brazilian Coffee Scenes dataset, it is composed of finer and more homogeneous textures where the patterns are much more overlapping visually and more different than everyday objects. The color/spectral properties are also important in this dataset, which fit with results reported in other works [52,72].

6.2. Comparison of ConvNets strategies

In this section, we compare the performance of the three different strategies for exploiting the existing ConvNets: full training,

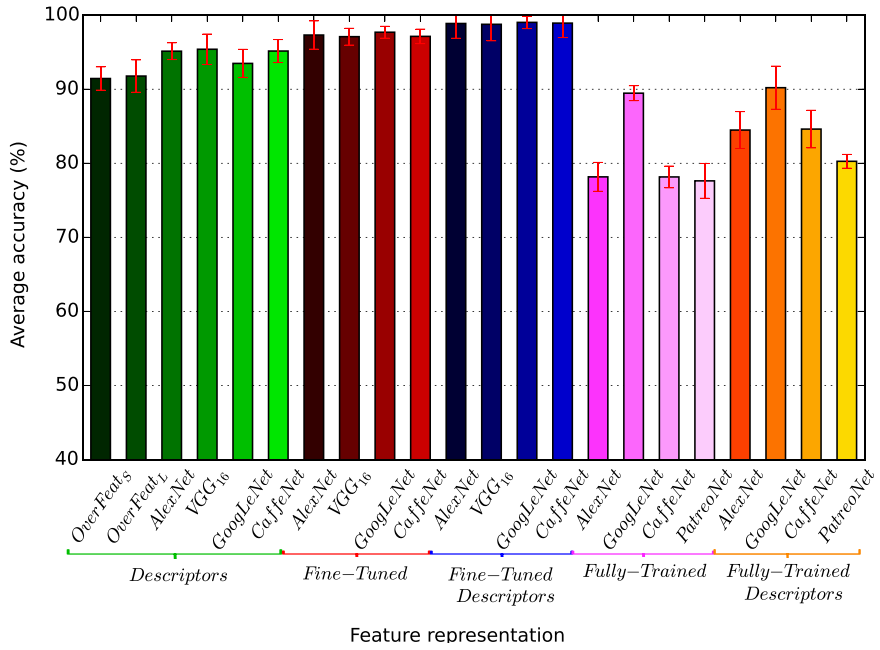


Fig. 12. Average accuracy considering all possible strategies to exploit ConvNets for the RS19 dataset. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.).

fine tuning, and using as feature extractors. Figs. 11–13 show the comparison of the strategies in terms of average classification accuracy. In such figures, the suffix "Descriptors" refers to when we use the features from the last layer before the softmax layer as input for another classifier, which was a linear SVM in our case. However, it is worth mentioning that SVM was used only after the fine-tuning or the full-training process, not during the training process.

There are several interesting aspects illustrated in the graphs. The first one is that fine tuning (red and blue bars) is usually the best strategy, outperforming the other two in all the datasets. The difference was higher for UCMerced and RS19 datasets (Figs. 11 and 12). For the Coffee Scenes dataset, this difference was small, however, the Fine-Tuned Descriptors (blue bars) were slightly superior.

This advantage of fine-tuned networks, when compared to the full-trained ones, is maybe due to a better initialization in the search space. This can be noticed in Fig. 14, where even with more iterations, full-trained networks stick in worse local minimum than the fine-tuned ones, what demonstrates that a better initialization of the filter weights tends to provide better results.

Other aspect to highlight is that full training (orange bars) was not a good strategy for datasets UCMerced and RS19. For RS19 dataset specially, there was a drop in accuracy in relation even to the original feature extractors (green bars), which were trained on ImageNet. However, for Coffee Scenes dataset, the full-training strategy improved results in comparison with using the original ConvNets as feature extractors.

Another aspect of the results is that replacing the last softmax layer by SVM of almost every ConvNet was a better solution. The Fine-Tuned Descriptors (blue bars) and Fully-Trained Descriptors (orange bars) were usually superior than their counterparts with the softmax layer (red and pink bars, respectively). In the Coffee Scenes, however, this difference was smaller.

Comparing the different ConvNets, we can see that their results are very similar in all the datasets. GoogLeNet seems to be less affected by the full training process, as their results decreased less than the other ConvNets when comparing the Fine-Tuned and the Full-Trained versions. One possible reason is that GoogLeNet has less parameters to be learned, and as the datasets used are very small considering the requirements of the full training process, GoogLeNet was less affected.

Comparing the results of the ConvNets as feature extractors (green bars) in relation to fine tuning (red and blue bars), we can see that the original feature extractors, trained on ImageNet, are not too worse than the fine-tuned version of the ConvNets, specially for the UCMerced and RS19 datasets. The reason is that in such datasets, the edges and local structures of the images are more similar to everyday objects than in the Coffee Scenes dataset, in which the difference was higher in favor of fine-tuned ConvNets. In the Coffee Scenes dataset, the textures and local structures are very different than everyday objects.

Comparing the results among the three datasets, we can see that the Coffee Scenes dataset has a different behavior for the ConvNets. Full-trained networks achieve better accuracy in this dataset than in the others. This maybe be motivated by the huge difference between the datasets, since UCMerced and RS19 datasets are aerial ones while Coffee scenes is a multi-spectral one.

Considering the computational load, using a pre-trained ConvNet as feature extractor is the most efficient strategy, since no training over the network is required. Fine-tuning strategy is less efficient since it requires a little training over the parameters of the network, but this method, generally, yields better results. The strategy that needs more computational requirements is full-training, since a network is trained from scratch (with randomly initialized weights and bias), requiring more time and resources to be trained. Specifically for these two last strategies, in our experiments, five ConvNets are trained (one for each fold), which makes the process time consuming. Considering the fine-tuning technique, the whole process takes around 5 h to be completed, while the fully-trained strategy takes, approximately, 9 h. In both cases, we consider an average training time of all three datasets on the GeForce GTX Titan X.

As summary, we can recommend *fine-tuning* as the strategy that tends to be more promising in different situations. In addition, fine tuning is less costly than full training, which can represent another advantage when efficiency is a constraint. On top of that, we can also recommend the use of the features extracted from the

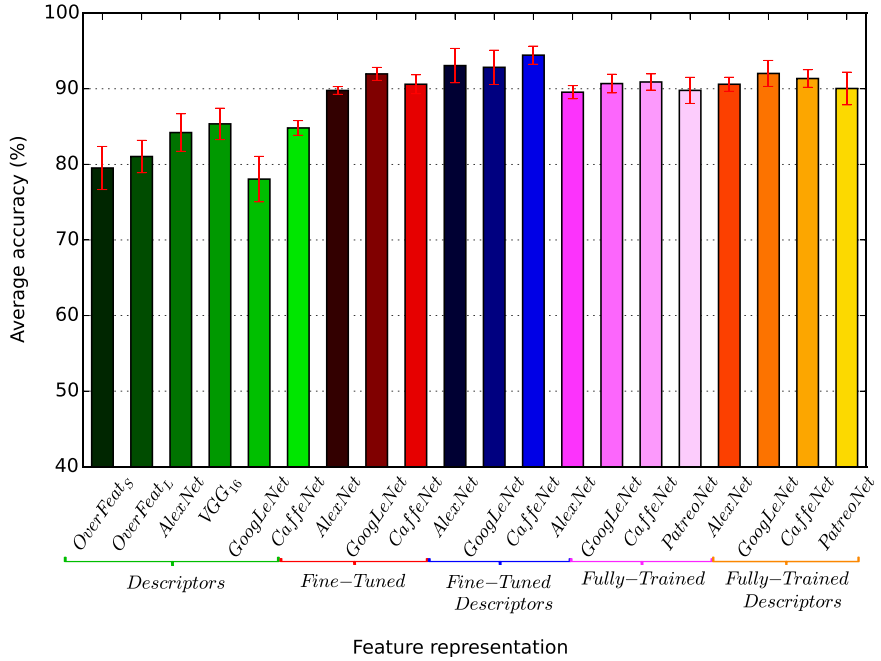
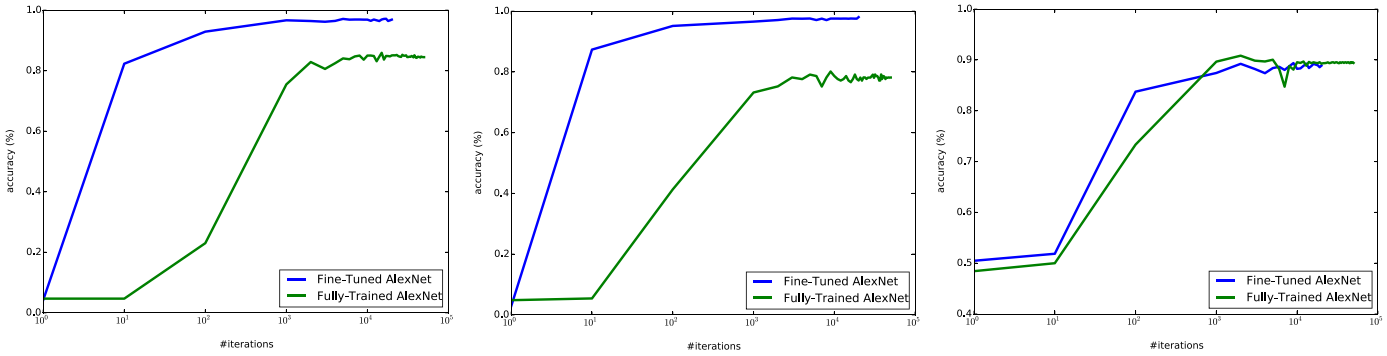


Fig. 13. Average accuracy considering all possible strategies to exploit ConvNets for the Brazilian Coffee Scenes dataset. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.).



(a) UCMerced Land-use dataset (b) RS19 dataset (c) Brazilian Coffee Scenes dataset

Fig. 14. Examples of the convergence of AlexNet for all datasets, considering Fold 1.

last layer of the fine-tuned network and then using SVM for the classification task, instead of using the softmax layer.

As shown in the experimental results, the best ConvNet configurations classify almost 100% of the aerial images (UCMerced and RS19). Notwithstanding, the wrong classified images are really difficult, as can be noted in the examples shown in Fig. 15. Notice how these misclassified samples are quite similar visually.

6.3. Comparison with baselines

In this section, we compare the performance of the best results of each strategy for exploiting the existing ConvNets and state-of-the-art baselines. Figs. 16–18 show the comparison in terms of average accuracy. As in previous section, the suffix "Descriptors" specify when ConvNets were used as feature extractor, being the deep features classified with linear SVM.

For the UCMerced dataset, we select three state-of-the-art baselines: (i) GCLBP [73], (ii) With-Sal [35], and (iii) Dense-Sift [74]. The results presented in Fig. 16 show that the baselines were outperformed by all strategies, except the full training one. Furthermore, the classification of deep features extracted from the fine-tuned GoogLeNet with linear SVM achieve the best result of all ($99.47 \pm 0.50\%$), being closely followed by the fine-tuned GoogLeNet, which yielded $97.78 \pm 0.97\%$, in terms of average accuracy.

For the RS19 dataset, we compare the best results of each strategy to the GCLBP [73] approach, which yielded $91.0 \pm 1.5\%$. The results presented in Fig. 17 confirm the results obtained in the UCMerced dataset, since these two datasets are very similar: using linear SVM to classify deep features extracted from a fine-tuned GoogLeNet yielded the best result. However, different from the previous dataset, the fully trained network did not outperform the baseline, being statistically similar.

For the Brazilian Coffee Scenes dataset, the only state-of-the-art result available is the one which was released with the dataset in our previous work [14], using the BIC descriptor, that we also present here in Section 6.1. Now, the best result for this dataset and current state-of-the-art is achieved by extracting deep features from the fine-tuned CaffeNet ($94.45 \pm 1.20\%$), as presented in Fig. 18. Note that although BIC outperforms the ConvNet used as a descriptor, it is not true for the Full-Trained and Fine-Tuned. It means that we can adjust the domain by using a full trained ConvNet. However, by using parameters obtained in other domain is useful to yield even better results.

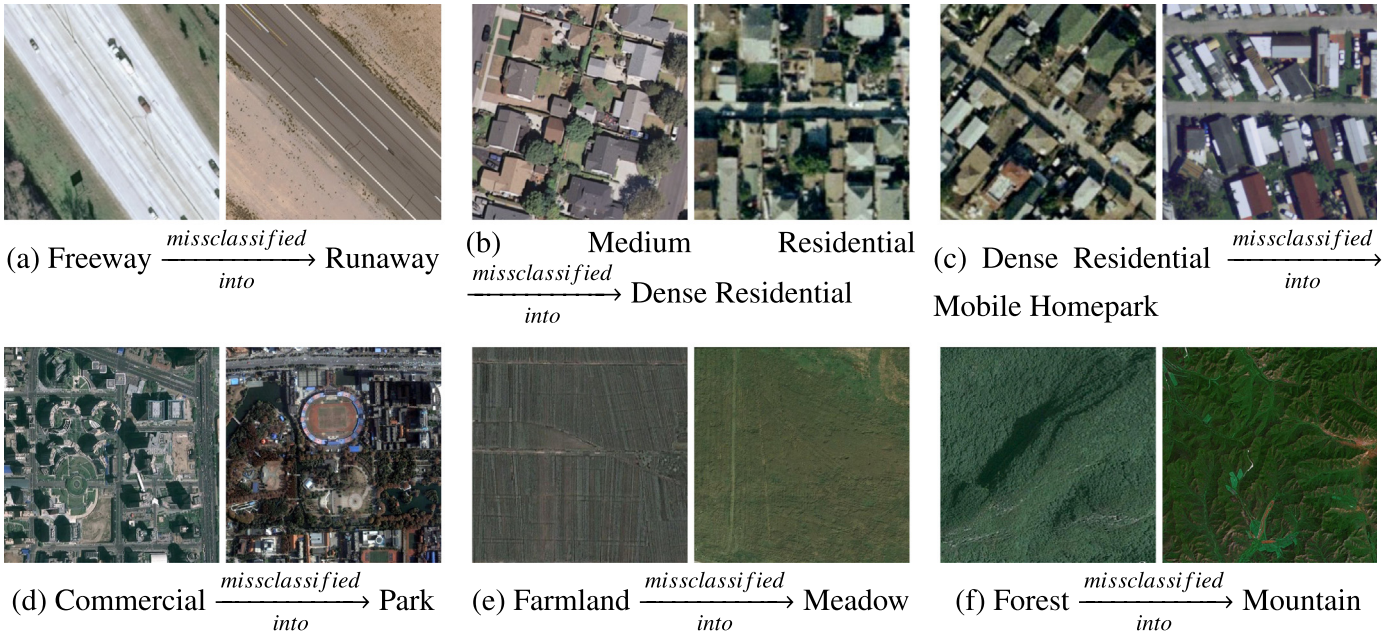


Fig. 15. Three examples of wrong predictions of each aerial dataset, UCMerced and RS19, for a fine-tuned AlexNet. (a)–(f) The first image is the misclassified one, while the second is a sample of the predicted class. Notice the similarity between the classes.

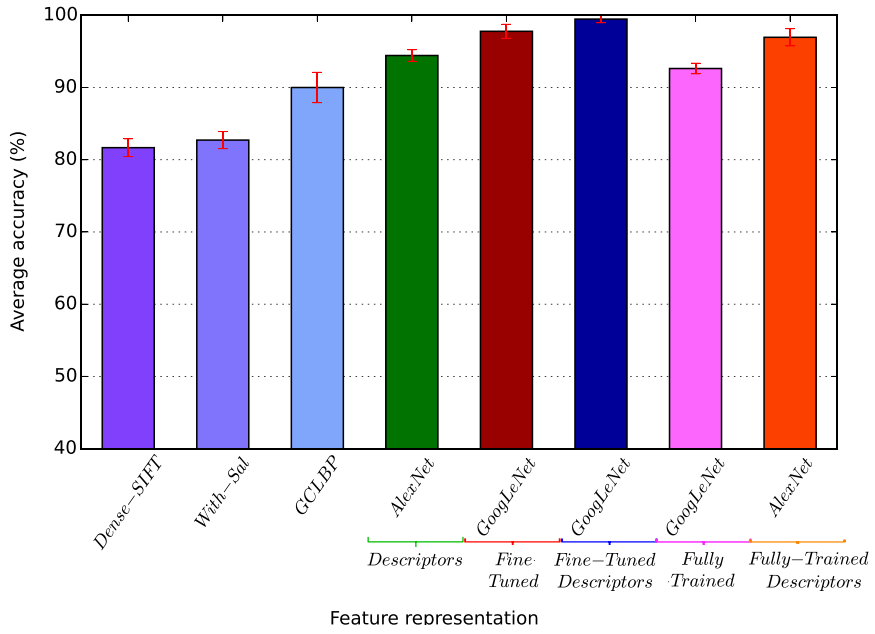


Fig. 16. Comparison between state-of-the-art baselines and the best results of each strategy to exploit ConvNets for the UCMerced Land-use Dataset. The Fine-Tuned Descriptors extracted by GoogLeNet achieved the highest accuracy rates.

7. Conclusions

In this paper, we evaluated three strategies for exploiting existing ConvNets in different scenarios from the ones they were trained. The objective was to understand the best way to obtain the most benefits from these state-of-the-art deep learning approaches in problems that usually are not suitable for the design and creation of new ConvNets from scratch. Such scenarios reflect many existing applications, in which there is few labeled data. We performed experiments evaluating the following strategies for exploiting the ConvNets: full training, fine tuning, and using as feature extractors. The experiments considered six popular ConvNets (OverFeat networks [71], AlexNet [23], CaffeNet [67], GoogLeNet [30], VGG_{16} [29], and PatreoNet [15]) in three remote sensing datasets.

The results point that *fine tuning* tends to be the best strategy in different situations. Specially, using the features of the fine-tuned network with an external classifier, linear SVM in our case, provides the best results.

As additional contributions of this work, we can point the evaluation of different ConvNets in each strategy mentioned in three remote sensing datasets, comparing their results with traditional low- and mid-level descriptors, as well as with state-of-the-art baselines of each dataset. We can also understand the generalization power of the existing ConvNets when used as feature descriptors. And finally, we obtained state-of-the-art results

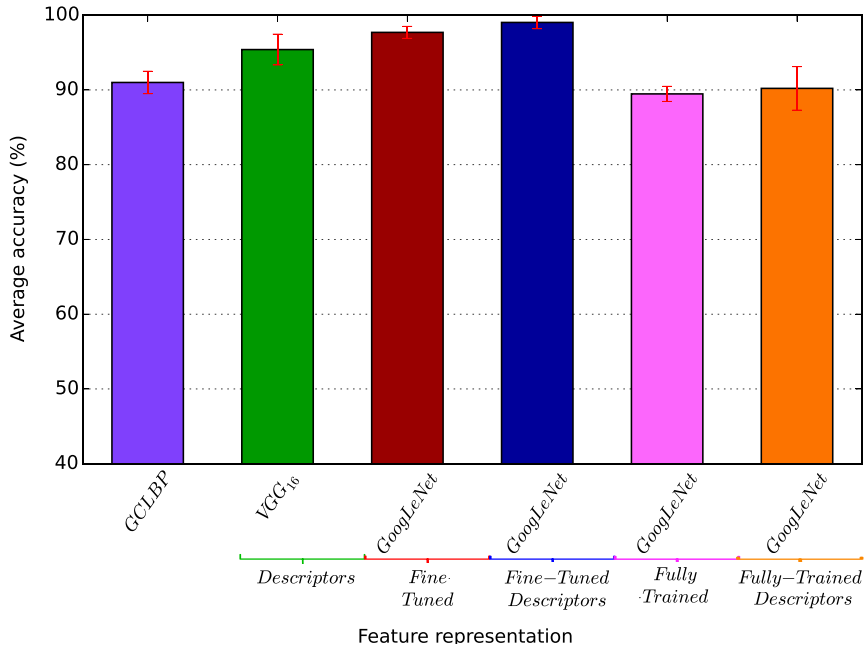


Fig. 17. Comparison between state-of-the-art baselines and the best results of each strategy to exploit ConvNets for the RS19 dataset. The Fine-Tuned Descriptors extracted by GoogLeNet achieved the highest accuracy rates.

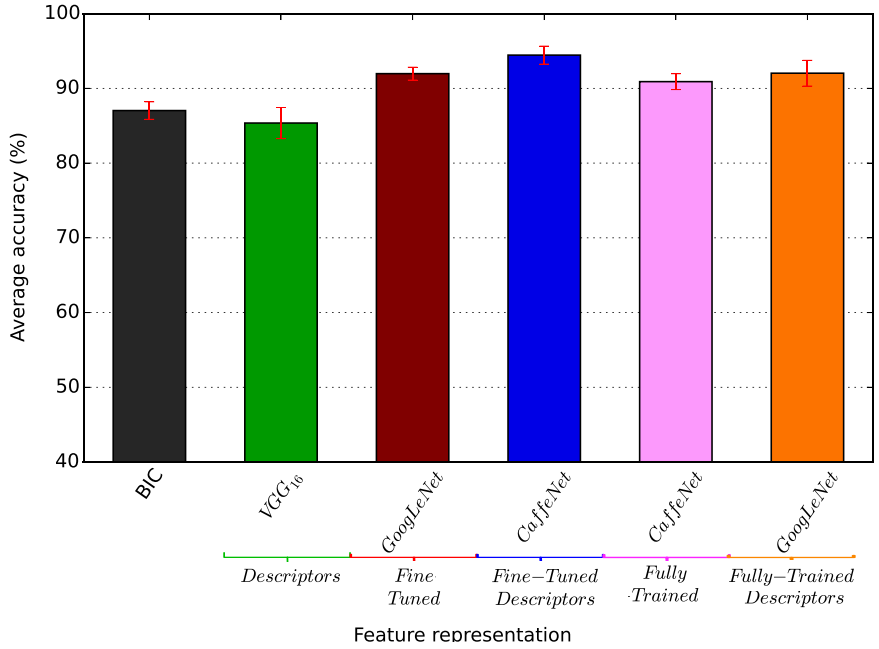


Fig. 18. Comparison between state-of-the-art baselines and the best results of each strategy to exploit ConvNets for the Brazilian Coffee Scenes dataset. The Fine-Tuned Descriptors extracted by CaffeNet achieved the highest accuracy rates.

in the three datasets used (UCMerced land use, RS19, and Brazilian Coffee Scenes).

The presented conclusions open new opportunities towards a better spectral–spatial feature representation, which is still needed for remote sensing applications, such as agriculture or environmental monitoring. We also believe that these conclusions also apply to other domains, however, we would like to perform such evaluation as future work. Another interesting opportunity for future work is to analyze the relation between the number of classes in the dataset, the number of parameters in the ConvNet, and their impact on the discrepancy between fine tuning and full training processes.

Acknowledgments

This work was partially financed by CNPq (grant 449638/2014-6), CAPES, and Fapemig (APQ-00768-14). We thank the RECOD lab of Institute of Computing, University of Campinas, Brazil, for the infrastructure for running part of the experiments. The authors gratefully acknowledge the support from NVIDIA Corporation with the donation of the GeForce GTX TITAN X GPU used for this research.

References

[1] G. Kumar, P.K. Bhatia, A detailed review of feature extraction in image processing systems, in: Advanced Computing and Communication Technologies,

IEEE, 2014, pp. 5–12.
[2] J.A. dos Santos, O.A.B. Penatti, R. da S. Torres, Evaluating the potential of texture and color descriptors for remote sensing image retrieval and classification, in: International Conference on Computer Vision Theory and Applications, 2010, pp. 203–208.
[3] C.-F. Tsai, Bag-of-words representation in image annotation: a review, ISRN Artif. Intell. (2012).
[4] M. Faraji, J. Shanbehzadeh, Bag-of-visual-words, its detectors and descriptors: a survey in detail, Adv. Comput. Sci.: Int. J. 4 (2) (2015) 8–20.
[5] J. Sivic, A. Zisserman, Video Google: a text retrieval approach to object matching in videos, in: International Conference on Computer Vision, vol. 2, 2003, pp. 1470–1477.
[6] J.C. van Gemert, C.J. Veenman, A.W.M. Smeulders, J.-M. Geusebroek, Visual word ambiguity, Trans. Pattern Anal. Mach. Intell. 32 (2010) 1271–1283.
[7] K.E.A. van de Sande, T. Gevers, C.G.M. Snoek, Evaluating color descriptors for object and scene recognition, Trans. Pattern Anal. Mach. Intell. 32 (9) (2010) 1582–1596.
[8] Y.-L. Boureau, F. Bach, Y. LeCun, J. Ponce, Learning mid-level features for recognition, in: Computer Vision and Pattern Recognition, 2010, pp. 2559–2566.
[9] F. Perronnin, J. Sánchez, T. Mensink, Improving the Fisher kernel for large-scale image classification, in: European Conference on Computer Vision, 2010, pp. 143–156.
[10] C.-H. Chen, L.F. Pau, P.S.-P. Wang, Handbook of Pattern Recognition And Computer Vision, vol. 2, World Scientific, 1993.
[11] J.A. Benediktsson, et al., Advances in very-high-resolution remote sensing, Proc. IEEE 101 (3) (2013).
[12] Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016, in preparation. URL ⟨http://www.deeplearningbook.org⟩.
[13] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.
[14] O.A. Penatti, K. Nogueira, J.A. dos Santos, Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? in: Computer Vision and Pattern Recognition Workshop, IEEE, 2015.
[15] K. Nogueira, W.O. Miranda, J.A. Dos Santos, Improving spatial feature representation from aerial scenes by using convolutional networks, in: 2015 28th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), IEEE, 2015, pp. 289–296.
[16] J. Yue, W. Zhao, S. Mao, H. Liu, Spectral-spatial classification of hyperspectral images using deep convolutional neural networks, Remote Sens. Lett. 6 (6) (2015) 468–477.
[17] M. Fingas, C. Brown, Review of oil spill remote sensing, Mar. Pollut. Bull. 83 (1) (2014) 9–23.
[18] M. Xie, N. Jean, M. Burke, D. Lobell, S. Ermon, Transfer learning from deep features for remote sensing and poverty mapping, arXiv:1510.00098.
[19] A. Tayyebi, B.C. Pijanowski, A.H. Tayyebi, An urban growth boundary model using neural networks, GIS and radial parameterization: an application to Tehran, Iran, Landsc. Urban Plan. 100 (1) (2011) 35–44.
[20] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[21] D. Tuia, M. Volpi, L. Copa, M. Kanevski, J. Munoz-Mari, A survey of active learning algorithms for supervised remote sensing image classification, IEEE J. Sel. Top. Signal Process. 5 (3) (2011) 606–617.
[22] J.A. dos Santos, P.-H. Gosselin, S. Philipp-Foliguet, R.d.S. Torres, A.X. Falcao, Interactive multiscale classification of high-resolution remote sensing images, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 6 (4) (2013) 2020–2034.
[23] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: Neural Information Processing Systems, 2012, pp. 1106–1114.
[24] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[25] H. Larochelle, Y. Bengio, J. Louradour, P. Lamblin, Exploring strategies for training deep neural networks, J. Mach. Learn. Res. 10 (2009) 1–40.
[26] Y. Yang, S. Newsam, Comparing SIFT descriptors and Gabor texture features for classification of remote sensed imagery, in: International Conference on Image Processing, 2008, pp. 1852–1855.
[27] R. Bouchiha, K. Besbes, Comparison of local descriptors for automatic remote sensing image registration, Signal Image Video Process. 9 (2) (2013) 463–469.
[28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: a large-scale hierarchical image database, in: Computer Vision and Pattern Recognition, IEEE, 2009, pp. 248–255.
[29] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556.
[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, arXiv:1409.4842.
[31] K. Makantasis, K. Karantzalos, A. Doulamis, N. Doulamis, Deep supervised learning for hyperspectral data classification through convolutional neural networks, in: IEEE International Geoscience and Remote Sensing Symposium, IEEE, Milan, Italy, 2015, pp. 4959–4962.
[32] H. Guan, Y. Yu, Z. Ji, J. Li, Q. Zhang, Deep learning-based tree classification using mobile LiDAR data, Remote Sens. Lett. 6 (11) (2015) 864–873.
[33] Y. Chen, Z. Lin, X. Zhao, G. Wang, Y. Gu, Deep learning-based classification of hyperspectral data, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 7 (6) (2014) 2094–2107.
[34] O. Firat, G. Can, F. Yarman Vural, Representation learning for contextual object and region detection in remote sensing, in: International Conference on Pattern Recognition, 2014, pp. 3708–3713.
[35] F. Zhang, B. Du, L. Zhang, Saliency-guided unsupervised feature learning for scene classification, IEEE Trans. Geosci. Remote Sens. 53 (4) (2015) 2175–2184.
[36] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Computer Vision and Pattern Recognition, IEEE, 2014, pp. 580–587.
[37] K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the devil in the details: delving deep into convolutional nets, arXiv:1405.3531.
[38] F. Hu, G.-S. Xia, J. Hu, L. Zhang, Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery, Remote Sens. 7 (11) (2015) 14680–14707.
[39] K. Jarrett, K. Kavukcuoglu, M. Ranzato, Y. LeCun, What is the best multi-stage architecture for object recognition? in: International Conference on Computer Vision, IEEE, 2009, pp. 2146–2153.
[40] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, Decaf: a deep convolutional activation feature for generic visual recognition, arXiv:1310.1531.
[41] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: International Conference on Machine Learning, 2010, pp. 807–814.
[42] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[43] N. Sunderhauf, C. McCool, B. Upcroft, P. Tristan, Fine-grained plant classification using convolutional neural networks for feature extraction, in: Working Notes of CLEF 2014 Conference, 2014.
[44] K. Hara, V. Jagadeesh, R. Piramuthu, Fashion apparel detection: the role of deep convolutional neural network and pose-dependent priors, arXiv:1411.5319.
[45] Z. Ge, C. McCool, C. Sanderson, A. Bewley, Z. Chen, P. Corke, Fine-grained bird species recognition via hierarchical subset learning, in: 2015 IEEE International Conference on Image Processing (ICIP), IEEE, 2015, pp. 561–565.
[46] A. Babenko, A. Slesarev, A. Chigorin, V. Lempitsky, Neural codes for image retrieval, in: Computer Vision—ECCV 2014, Springer, Zurich, Switzerland, 2014, pp. 584–599.
[47] A.S. Razavian, H. Azizpour, J. Sullivan, S. Carlsson, CNN features off-the-shelf: an astounding baseline for recognition, in: Computer Vision and Pattern Recognition Workshop, 2014, pp. 512–519.
[48] Y. Yang, S. Newsam, Bag-of-visual-words and spatial extensions for land-use classification, in: ACM International Conference on Advances in Geographic Information Systems, 2010, pp. 270–279.
[49] G.-S. Xia, W. Yang, J. Delon, Y. Gousseau, H. Sun, H. Maître, Structural high-resolution satellite image indexing, in: ISPRS TC VII Symposium-100 Years ISPRS, vol. 38, 2010, pp. 298–303.
[50] O.A.B. Penatti, E. Valle, R. da, S. Torres, Comparative study of global color and texture descriptors for web image retrieval, J. Vis. Commun. Image Represent. 23 (2) (2012) 359–380.
[51] Y. Yang, S. Newsam, Geographic image retrieval using local invariant features, IEEE Trans. Geosci. Remote Sens. 51 (2) (2013) 818–832.
[52] J.A. dos Santos, O.A.B. Penatti, P.-H. Gosselin, A.X. Falcao, S. Philipp-Foliguet, R. da, S. Torres, Efficient and effective hierarchical feature propagation, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 7 (12) (2014) 4632–4643.
[53] J. Huang, S.R. Kumar, M. Mitra, W. Zhu, R. Zabih, Image indexing using color correlograms, in: Computer Vision and Pattern Recognition, 1997, pp. 762–768, ISBN 0-8186-7822-4.
[54] R. de O. Stehling, M.A. Nascimento, A.X. Falcao, A compact and efficient image retrieval approach based on border/interior pixel classification, in: International Conference on Information and Knowledge Management, 2002, pp. 102–109.
[55] M.J. Swain, D.H. Ballard, Color indexing, Int. J. Comput. Vision. 7 (1) (1991) 11–32, ISSN 0920-5691.
[56] B. Tao, B.W. Dickinson, Texture recognition and image retrieval using gradient indexing, J. Vis. Commun. Image Represent. 11 (3) (2000) 327–342, ISSN 1047-3203.
[57] A. Çarkacioglu, F. Yarman-Vural, SASI: a generic texture descriptor for image retrieval, Pattern Recognit. 36 (11) (2003) 2615–2633.
[58] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Computer Vision and Pattern Recognition, 2005, pp. 886–893.
[59] A. Oliva, A. Torralba, Modeling the shape of the scene: a holistic representation of the spatial envelope, Int. J. Comput. Vision. 42 (3) (2001) 145–175, ISSN 0920-5691.
[60] M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, C. Schmid, Evaluation of GIST descriptors for web-scale image search, in: ACM International Conference on Image and Video Retrieval, 2009, pp. 19:1–19:8.
[61] A. Vedaldi, B. Fulkerson, VLFeat: An open and portable library of computer vision algorithms, 2008, ⟨http://www.vlfeat.org/⟩.
[62] S. Lazebnik, C. Schmid, J. Ponce, Beyond Bags of features: spatial pyramid matching for recognizing natural scene categories, in: Computer Vision and Pattern Recognition, 2006, pp. 2169–2178.
[63] O.A.B. Penatti, F.B. Silva, E. Valle, V. Gouet-Brunet, R. da, S. Torres, Visual word spatial arrangement for image retrieval and classification, Pattern Recognit. 47 (2) (2014) 705–720, ISSN 0031-3203.
[64] S. Avila, N. Thome, M. Cord, E. Valle, A. de, A. Araújo, Pooling in image representation: the visual codeword point of view, Comput. Vision. Image Underst. 117 (5) (2013) 453–465.
[65] K.E.A. van de Sande, T. Gevers, C.G.M. Snoek, Empowering visual categorization with the GPU, Trans. Multimed. 13 (1) (2011) 60–70.
[66] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J.

Comput. Vision. 60 (2) (2004) 91–110.
[67] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, arXiv:1408.5093.
[68] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun, OverFeat: integrated recognition, localization and detection using convolutional networks, in: International Conference on Learning Representations, CBLS, 2014.
[69] O. Russakovsky J. Deng H. Su J. Krause S. Satheesh S. Ma Z. Huang A. Karpathy A. Khosla M. Bernstein A.C. Berg L. Fei-Fei, ImageNet Large Scale Visual Recognition Challenge, 2014.
[70] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, A. Oliva, Learning deep features for scene recognition using places database, Neural Inf. Process. Syst. (2014) 487–495.
[71] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun, OverFeat: integrated recognition, localization and detection using convolutional networks, arXiv:1312.6229v4.
[72] F. Faria, D. Pedronette, J. dos Santos, A. Rocha, R. Torres, Rank aggregation for pattern classifier selection in remote sensing images, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 7 (4) (2014) 1103–1115.
[73] C. Chen, L. Zhou, J. Guo, W. Li, H. Su, F. Guo, Gabor-filtering-based completed local binary patterns for land-use scene classification, in: IEEE International Conference on Multimedia Big Data (BigMM), IEEE, Beijing, China, 2015, 324–329.
[74] A.M. Cheriyadat, Unsupervised feature learning for aerial scene classification, IEEE Trans. Geosci. Remote Sens. 52 (1) (2014) 439–451.

Keiller Nogueira is a Doctoral candidate in the Computer Science Department at Universidade Federal de Minas Gerais. He received the master's degree in Computer Science by the same institution while B.Sc. degree was granted by the Universidade Federal de Viçosa. His research interests include Machine Learning and Pattern Recognition.

Otávio A.B. Penatti received his PhD in Computer Science in 2012 from University of Campinas, Brazil. He is currently a researcher at Samsung Research Institute Brazil. His research interests include computer vision, pattern recognition, machine learning, and multimedia geocoding.

Jefersson A. dos Santos received the PhD in Computer Science from the University of Campinas and from the University of Cergy-Pontoise. He is currently a professor in the Department of Computer Science at the Universidade Federal de Minas Gerais. His research interests include remote sensing, machine learning, and information retrieval.