Convolutional Neural Networks for Large-Scale Remote-Sensing Image Classification

Emmanuel Maggiori, *Student Member, IEEE*, Yuliya Tarabalka, *Member, IEEE*, Guillaume Charpiat, and Pierre Alliez

***Abstract*— We propose an end-to-end framework for the dense, pixelwise classification of satellite imagery with convolutional neural networks (CNNs). In our framework, CNNs are directly trained to produce classification maps out of the input images. We first devise a *fully convolutional* architecture and demonstrate its relevance to the dense classification problem. We then address the issue of imperfect training data through a two-step training approach: CNNs are first initialized by using a large amount of possibly inaccurate reference data, and then refined on a small amount of accurately labeled data. To complete our framework, we design a multiscale neuron module that alleviates the common tradeoff between recognition and precise localization. A series of experiments show that our networks consider a large amount of context to provide fine-grained classification maps.**

***Index Terms*— Classification, convolutional neural networks (CNNs), deep learning, satellite images.**

I. Introduction

THE analysis of remote-sensing images is of paramount importance in many practical applications, such as precision agriculture and urban planning. Recent technological developments have significantly increased the amount of available satellite imagery. Notably, the constellation of Pléiades satellites produces high spatial resolution images that cover the whole Earth in less than a day. The large-scale nature of these data sets introduces new challenges in image analysis. In this paper, we address the problem of pixelwise classification of satellite imagery.

There is a vast literature on classification approaches that consider the spectrum of every individual pixel to assign it to a certain class. Alternatively, more advanced techniques combine information from a few neighboring pixels to enhance the classifiers' performance, often referred to as spectral-spatial classification. These approaches rely on the separability of the different classes based on the spectrum of a single pixel or of some neighboring pixels. In a large-scale setting, however, these approaches are not effective. On the one hand, current large-scale satellite imagery does not use high spectral resolution sensors, making it difficult to distinguish object classes solely by their spectrum. On the other hand, due to the large spatial extent covered by the data sets, classes have a considerable internal variability, which further challenges the class separability when simply observing the spectral signatures of a restricted neighborhood. We argue that a more thorough understanding of the context, such as the shape of objects, is required to aid the classification process.

Convolutional neural networks (CNNs) [1] are, therefore, gaining attention, due to their capability to automatically discover relevant contextual features in image categorization problems. CNNs consist of a stack of learned convolution filters that extract hierarchical contextual image features, and are a popular form of *deep learning* networks. They are already outperforming other approaches in various domains, such as digit recognition [2] and natural image categorization [3].

Our goal is to devise an end-to-end framework to classify satellite imagery with CNNs. The context of large-scale satellite image classification introduces certain challenges that we must address in order to turn CNNs into a relevant classification tool. Notably, we must: 1) design a specific neural network architecture for our problem; 2) acquire large-scale training data and handle its eventual inaccuracies; and 3) generate high-resolution output classification maps.

1) CNN Architecture: CNNs are commonly used for *image categorization*, i.e., for assigning the entire image to a class (e.g., a digit [1] or an object category [3]). In remote sensing, the equivalent problem is to assign a category to an entire image patch, such as "residential" or "agricultural" area. Our context differs in that we wish to conduct a *dense* pixelwise labeling. We must thus design a CNN that outputs a per-pixel classification and not just a category for the entire input.

2) Imperfect Training Data: A sensitive point regarding CNNs is the amount of training data required to properly learn the network parameters. A large source of free-access maps is OpenStreetMap (OSM), a collaborative online mapping platform, but the availability of data is highly variable between areas. In some areas, the coverage is very limited or nonexistent, and an irregular misregistration is prevalent throughout the maps. As we focus on the large-scale application of CNNs for classification, we must explore the use of imperfect training data in order to make our framework applicable to a wide range of geographic areas.

3) High-Resolution Output: The power of CNNs to take a large context to conduct predictions comes at the price of

Manuscript received May 5, 2016; revised August 4, 2016; accepted August 21, 2016. Date of publication October 19, 2016; date of current version December 29, 2016. This work was supported by the Centre National d'Etudes Spatiales.

E. Maggiori, Y. Tarabalka, and P. Alliez are with the TITANE Team, Inria, Université Côte d'Azur, BP93 06902 Sophia Antipolis, France (e-mail: emmanuel.maggiori@inria.fr).

G. Charpiat is with the Tao Team, Inria Saclay–Île-de-France, Laboratoire de Recherche en Informatique, Université Paris-Sud, 91405 Orsay, France.

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

Digital Object Identifier 10.1109/TGRS.2016.2612821

losing resolution for the output. This is because some degree of downsampling of the feature maps along the network is required in order to increase the amount of context without an excessive number of learnable parameters. Such coarse resolution translates into a fuzzy aspect around object edges and corners. One of our challenges is then to alleviate this tradeoff.

A. Related Work

We now review classification methods and the use of CNNs in remote sensing.

In the context of spectral classification, decision trees [4], artificial neural networks [5], [6], and support vector machines (SVMs) [7] are some of the approaches that have been explored, both for multispectral and hyperspectral image analysis. Spectral-spatial methods [8] use contextual information to regularize the classification maps. Different approaches have been presented, for example, Liao *et al.* [9] sequentially apply morphological filters to model different kinds of structural information, and Tarabalka and Rana [10] model spatial interactions with a graphical model. Neural networks have also been used for spectral-spatial classification. In this direction, Kurnaz *et al.* [11] use such network to classify the concatenated spectrum of pixels inside a sliding window, in order to label multispectral images. In a similar fashion, Lloyd *et al.* [12] compute a textural feature, which is concatenated to the pixel spectrum vector, prior the neural network classification. Lu and Weng [13] provide a comprehensive survey on classification methods.

In remote sensing, CNNs have been used to individually classify the pixels of hyperspectral images. This was achieved by performing convolutions in the 1-D domain of the spectrum of each pixel [14]–[16]. Alternatively, a spectral-spatial approach has been taken by convolving in the 1-D flattened spectrum vector of a group of adjacent pixels [17], [18]. Note, however, that these approaches do not learn spatial contextual features, such as the typical shape of the objects of a class. Recent works have incorporated convolutions on the spatial domain after extracting the principal components of the hyperspectral image [19]–[21], and the idea of reasoning at multiple spatial scales has also been exploited, notably for hyperspectral classification [22], [23] and image segmentation [24]. Let us remark that CNNs have also been used for other remote-sensing applications, such as road tracking [25], object detection [26], and land use classification [27], [28].

Mnih [29] proposed a specific architecture to learn large-extent spatial contextual features for aerial image labeling. It is derived from common image categorization networks by increasing the output size of the final layer. Instead of outputting a single value to indicate the category, the final layer produces an entire dense classification patch. This network successfully learns contextual spatial features to better distinguish the object classes. However, this patchwise procedure has the disadvantage of introducing artifacts on the border of the classified patches. Moreover, the last layer of the network introduces an unnecessarily large number of parameters, hampering its efficiency.

B. Contributions

We now summarize our contributions to address the issues presented before and provide then a framework for satellite image classification with CNNs.

1) Fully Convolutional Architecture: We first analyze the CNN architecture proposed by Mnih [29] and the fact that it has a fully connected layer, i.e., connected to *all* the outputs of the previous layer, to produce the output classification patches. We point out that this architectural decision hampers both its accuracy and efficiency.

We then propose a new network architecture that is *fully convolutional* that only involves a series of convolution and deconvolution operations to produce the output classification maps. This architecture solves the issues of the previous patch-based approach by construction. While such a fully convolutional architecture imposes further restrictions to the neuronal connections than the fully connected approach, these restrictions reduce the number of trainable parameters without losing generality. It has been seen multiple times in the literature that reducing the number of parameters under sensible assumptions often implies a simpler error surface and helps reaching better local minima. For example, convolutional networks have fewer connections than multilayer perceptrons but perform better, in practice, for visual tasks [1], and Mnih [29] showed that adding too many layers to a network resulted in poorer results.

We compare the fully convolutional versus fully connected approaches on a data set of publicly available aerial color images over Massachusetts [29] created with the specific purpose of evaluating CNN architectures.

2) Two-Step Training Approach: To deal with the imperfections in training data, we propose a two-step approach. First, we train our fully CNN on raw OSM data to discover the generalities of the data set. Second, we fine-tune the resulting neural networks for a few iterations under a small piece of manually labeled image. Our hypothesis is that, once the network is pretrained on large amounts of imperfect data, we can boost its performance by "showing" it a small amount of accurate labels. Our approach is inspired by a common practice in deep learning: taking pretrained networks designed to solve one problem and fine-tuning them to another problem.

3) Multiscale Architecture: We design a specific neuron module that processes its input at multiple scales, while keeping a low number of parameters. This alleviates the aforementioned tradeoff between the amount of context taken and the resolution of the classification maps. Then, our overall approach constitutes an end-to-end framework for satellite image labeling with CNNs. We evaluate it on a Pléiades image data set over France, where the associated OSM data are significantly inaccurate.

C. Organization of This Paper

In Section II, an introduction to CNNs is presented. In Section III, the fully convolutional architecture is described and evaluated. Section IV presents the two-step training approach and the multiscale architecture, in order to use CNNs as an end-to-end framework for satellite image classification. Finally, the conclusions are drawn in Section V.

II. Convolutional Neural Networks

In machine learning, an artificial neural network is a system of interconnected neurons that pass messages to each other. Neural networks are used to model complex functions and, in particular, as frameworks for classification. In this paper, we deal with the so-called feed-forward networks, whose graph of message passing between neurons is acyclic [30].

An individual neuron takes a vector of inputs $\mathbf{x} = x_1 \dots x_n$ and performs a simple operation to produce an output a. The most common neuron is defined as follows:

$$a = \sigma(\mathbf{w}\mathbf{x} + b) \tag{1}$$

where $\mathbf{x}$ denotes a weight vector, b a scalar known as *bias*, and σ an activation function. The weight vectors and the biases are parameters that define the function computed by a network, and the goal of training is to find the optimal values for these parameters. When using at least one layer of nonlinear activation functions, one can prove that a sufficiently large network can represent any function, suggesting the expressive power of neural networks. The most common activation functions are sigmoids, hyperbolic tangents, and rectified linear units (ReLUs) [3]. ReLUs are known to offer some practical advantages in the convergence of the training procedure.

Even though any function can be represented by a sufficiently large single layer of neurons, it is common to organize them in a set of stacked layers that transform the outputs of the previous layer and feed it to the next layer. This encourages the networks to learn hierarchical features, doing low-level reasoning in the first layers and performing higher level tasks in the last layers. For this reason, the first and last layers are often referred to as lower and upper layers, respectively.

In an image categorization problem, the input of our network is an image (or a set of features derived from an image), and the goal is to predict the correct label associated with the image. Finding the optimal neural network classifier reduces to finding the weights and biases that minimize a loss L between the predicted values and the target values in a training set. If there is a set $\mathcal{L}$ of possible classes, the labels are typically encoded as a vector of length $|\mathcal{L}|$ with value "1" at the position of the correct label and "0" elsewhere. The network then has as many output neurons as possible labels. A softmax normalization is performed on top of the last layer to guarantee that the output is a probability distribution, i.e., the values for every label are between zero and one and add to one. The multilabel problem is then seen as a regression on the desired output label vectors.

The loss function L quantifies the misclassification by comparing the target label vectors $\mathbf{y}^{(i)}$ and the predicted label vectors $\hat{\mathbf{y}}^{(i)}$, for n training samples $i = 1 \dots n$. In this paper, we use the common cross-entropy loss, defined as

$$L = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{|\mathcal{L}|} y_k^{(i)} \log \hat{y}_k^{(i)}. \tag{2}$$

The cross-entropy loss has fast convergence rates when training neural networks (compared with, for instance, the Euclidean distance between $\mathbf{y}$ and $\hat{\mathbf{y}}$) and is numerically stable when coupled with softmax normalization [30].

Note that in the special case of binary labeling, we can produce only one output (with targets "1" for positive and "0" for negative). In this case, a sigmoid normalization and cross-entropy loss are analogously used, albeit a multiclass framework can also be used for two classes.

Once the loss function is defined, the parameters (weights and biases) that minimize the loss must be solved for. Solving is achieved by gradient descent by computing the derivative $(\partial L)/(\partial w_i)$ of the loss function with respect to every parameter w_i, and updating the parameters with a learning rate λ as follows:

$$w_i \leftarrow w_i + \lambda \frac{\partial L}{\partial w_i}. \tag{3}$$

The derivatives $(\partial L)/(\partial w_i)$ are obtained by *backpropagation*, which consists in explicitly computing the derivatives of the loss with respect to the last layer's parameters and using the chain rule to recursively compute the rest of the derivatives. In practice, learning is performed by *stochastic gradient descent*, i.e., by estimating the loss (2) on a small subset of the training set, referred to as a mini-batch.

Despite the fact that neural networks can represent very complex functions, the epigraph of the loss function L can be highly nonconvex, making the optimization difficult via a gradient descent approach. To regularize this loss and improve training, CNNs [1] are a special type of neural networks that impose restrictions that make sense in the context of image processing. In these networks, every neuron is associated with a spatial location (i, j) with respect to the input image. The output a_{ij} associated with location (i, j) is then computed as follows:

$$a_{ij} = \sigma((\mathbf{W} * \mathbf{X})_{ij} + b) \tag{4}$$

where $\mathbf{W}$ denotes a kernel with learned weights, $\mathbf{X}$ the input to the layer, and "$*$" the convolution operation. Note that this is a special case of the neuron in (1) with the following constraints.

1) The connections only extend to a limited spatial neighborhood determined by the kernel size.
2) The same filter is applied to each location, guaranteeing translation invariance.

Typically multiple convolution kernels are learned in every layer, interpreted as a set of spatial feature detectors. The responses to every learned filter are therefore known as a *feature map*.

Departing from the traditional fully connected layer, in which every neuron is connected to all outputs of the previous layer, a convolutional layer dramatically reduces the number of parameters by enforcing the aforementioned constraints. This results in a regularized loss function, easier to optimize, without losing much generality.

Note that the convolution kernels are actually 3-D, because, in addition to their spatial extent, they go through all the feature maps in the previous layers, or through all the bands in the input image. Since the third dimension can be inferred from the previous layer, it is rarely specified in architecture descriptions, only the two spatial dimensions being usually mentioned.

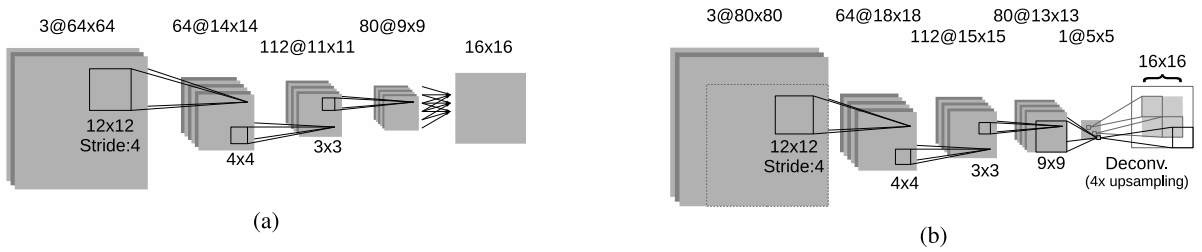


Fig. 1. CNN architectures (e.g., "64@14 × 14" means 64 feature maps of size 14 × 14). (a) Patch-based. (b) Fully convolutional (16 × 16 output).

In addition to convolutional layers, the state-of-the-art networks, such as Imagenet [3], involve some degree of downsampling, i.e., a reduction in the resolution of the feature maps. The goal of downsampling is to increase the so-called *receptive field* of the neurons, which is the part of the input image that neurons can "see." For the predictions to consider a large spatial context, the upper layers should have a large receptive field. This is achieved either by increasing the convolution kernel sizes or by downsampling feature maps to a lower resolution. The first alternative increases the number of parameters and memory consumption, making the training and inference processes prohibitive. The state-of-the-art CNNs then tend to keep the kernels small and add some degree of downsampling instead. This can be accomplished either by including pooling layers (e.g., taking the average or maximum of adjacent locations) or by introducing a so-called *stride*, which amounts to skip some convolutions through applying the filter once every four locations.

Classification networks typically contain a fully connected layer on top of the convolutions/pooling. This layer is designed to have as many outputs as labels, and produces the final classification scores.

The overall success of CNNs lies mostly in the fact that the networks are forced by construction to learn hierarchical contextual translation-invariant features, which are particularly useful for image categorization.

III. CNNs for Dense Classification

In this paper, we address the problem of *dense* classification, i.e., not just the categorization of an entire image, but a full pixelwise labeling into the different categories. We first describe an existing approach, the *patch-based* network, point out its limitations, and propose a *fully convolutional* architecture that addresses these limitations. We restrict our experiments to the binary labeling problem for the *building* versus *not building* classes, but our approach is extensible to an arbitrary number of classes following the formulation described in Section II.

A. Patch-Based Network

To perform dense classification of aerial imagery, Mnih [29] proposed a patch-based CNN. Training and inference are performed patchwise: the network takes as input a patch of an aerial image, and generates as output a classified patch. The output patch is smaller, and centered in the input patch, to consider the surrounding context for more accurate predictions. The way to create dense predictions is to increase the number of outputs of the last fully connected classification layer, in order to match the size of the target patch.

Fig. 1(a) shows the patch-based architecture from [29]. The network takes 64×64 patches (on color images of $1m^2$ spatial resolution) and predicts 16×16 centered patches of the same resolution. Three convolutional layers learn 64, 112, and 80 convolution kernels, of 12×12, 4×4, and 3×3 spatial dimensions, respectively. The first convolution is strided (one convolution every four pixels), which implies a downsampling with a factor of 4.

After the three convolutional layers, a fully connected layer transforms the high-level features of the last convolutional layer into a classification map of 256 elements, matching the required 16×16 output patch.

Training is performed by selecting random patches from the training set, and grouping them into mini-batches as required by the stochastic gradient descent algorithm.

B. Limitations of the Patch-Based Framework

We now point out some limitations of the patch-based approach discussed earlier, which motivate the design of an improved network architecture. Let us first analyze the role of the last fully connected layer that constructs the output patches. In the architecture of Fig. 1(a), the size of the feature maps in the last convolutional layer (before the last fully connected one) is 9×9. The resolution of these filters is 1/4 of the resolution of the input image, due to the 4-stride in the first convolution. The output of the fully connected layer is, however, a full-resolution 16×16 classification map. This means that the fully connected layer does not only compute the classification scores, but also learns how to upsample them. Outputting a full-resolution patch is then the result of upsampling and not of an intrinsic high-resolution processing. We also observe that the fully connected layer allows outputs at different locations to have different weights with respect to the previous layer. For example, the weights associated with an output pixel at the top-left corner of a patch can be different to those of a pixel at the bottom right. In other words, the network can learn priors on the position inside a patch. This makes sense in some specific contexts, such as when labeling pictures of outdoor scenes: the system could learn a prior for the sky to be at the top of the image. In our context, however, the partition of an image into patches is arbitrary; hence the

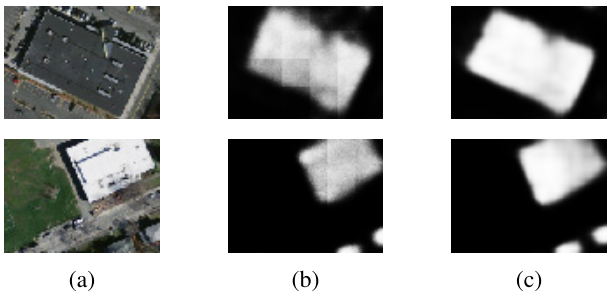


Fig. 2. Patch-based predictions exhibit artifacts on the patch borders while the FCN prevents them by construction. (a) Color. (b) Patch-based. (c) FCN.

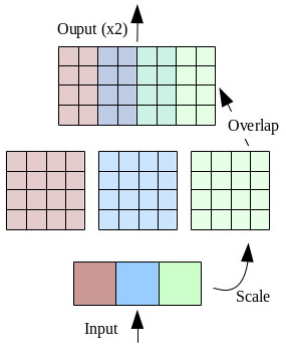


Fig. 3. "Deconvolution" layer for upsampling.

"in-patch location" prior is irrelevant, since allowing different weights at different patch locations may yield undesirable properties. For example, feeding two image patches that are identical but rotated by 90° could yield different classification maps.

When training the network of Fig. 1(a), we expect that, after processing many training cases, the fully connected layer will end up learning a location-invariant function. Fig. 2(a) and (b) shows a fragment of an output score map by using such an architecture. Notice the discontinuities at the border of the patches, which reveal that the network did not succeed in learning to classify pixels independently of their location inside the patch. While this issue is partly addressed in [29] by smoothing the outputs with a conditional random field, we argue that avoiding such artifacts by construction is desirable. In addition, generating similar results regardless of image tiling is an important property for large-scale satellite image processing and an active research topic [31], [32]. Another concern with the fully connected layer is that the receptive field of every patch output is not centered in itself. For example, a prediction near the center of the output patch can "see" about 32 pixels in every direction around it. However, the prediction at the top-left corner of the output patch considers a larger portion of the image to the bottom and to the right than to the top and to the left. Considering that the division into patches is arbitrary, this behavior is hard to justify.

A deeper understanding of the role played by every layer of the network, as described in this section, motivates the design of a more suitable architecture from a theoretical point of view, with the additional goal of boosting the overall performance of the approach.

C. Fully Convolutional Network

We propose a *fully convolutional* network architecture (FCN) to produce dense predictions. We explicitly restrict the process to be location-independent, enforcing the outputs to be the result of a series of convolutions only [see Fig. 1(b)].

A classification network may be "convolutionalized" [33] as follows. We first convert the fully connected layer that carries out the classification to a convolutional layer. The convolution kernel is chosen, so that its dimensions coincide with the previous layer. Thus, its connections are equivalent to a fully connected layer. The difference is that if we enlarge the input image, the output size is also increased, but the number of parameters remains constant. This may be seen as convolving the whole original network around a larger image to evaluate the output at different locations.

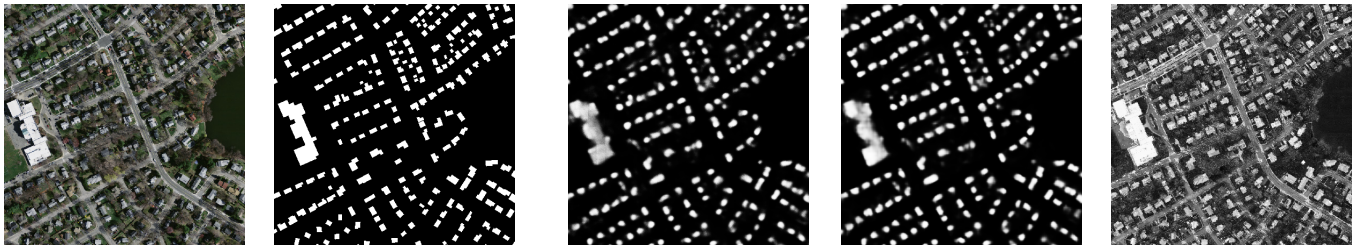
To increase the resolution of the output map, we then add a so-called "deconvolutional" layer [33]. The goal of this layer is to upsample the feature maps from the previous layer, which is achieved by performing an interpolation from a set of nearby points. Such an interpolation is parametrized by a kernel that expresses the extent and amount of contribution from a pixel value to its neighboring positions, only based on their locations. For an effective interpolation, the kernels must be large enough to overlap in the output. The interpolation is then performed by multiplying the values of the kernel by every input and adding the overlapping responses in the output. This process is shown in Fig. 3 for a $2\times$ upsampling. Notice that the scaling step is performed based on a constant 4×4 kernel. In our framework, and as in previous work [33], the interpolation kernel is another set of learnable parameters of the network instead of being determined *a priori*, e.g., setting them to represent a bilinear interpolation. Also note that the upsampled feature map has a central part computed by adding the contribution of two neighboring kernels and an outer border obtained solely by the contribution of one kernel (the two leftmost and rightmost output columns in Fig. 3). The outer border can be seen as an extrapolation of the input while the inner part can be seen as an interpolation. The extrapolated border can be cropped from the output to avoid artifacts.

As compared with a patch-based approach, we can expect our fully convolutional network to exhibit the following advantages:

1) elimination of discontinuities due to patch borders;
2) improved accuracy due to a simplified learning process, with a smaller number of parameters;
3) lower execution time at inference, due to the fast GPU execution of convolution operations.

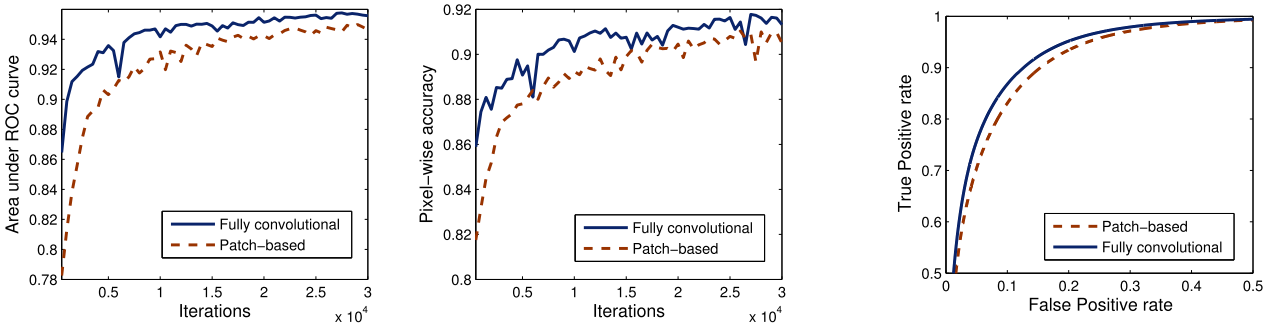
Our FCN network is constructed by convolutionalizing the existing patch-based network shown in Fig. 1(a). We choose an existing framework to benefit from a mature architecture and to carry out a rigorous comparison. The architectural decisions (i.e., the choice of the number of layers and filter sizes) of the base network are described in [29].

Fig. 1(b) shows the resulting FCN. First, we pretend that the output patch of the original network is only of size 1×1, thus just focusing on a single output centered in its receptive field.



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

Fig. 4. Experimental results on a fragment of the Boston data set. (a) Color image. (b) Reference data. (c) Patch-based fuzzy map. (d) FCN fuzzy map. (e) SVM fuzzy map.



(a) (b)

Fig. 5. Evaluation of patch-based and fully CNNs on the Boston test set. (a) Performance evolution. (b) ROC curves.

Second, we rewrite the fully connected layer as a convolutional layer with one feature map and the spatial dimensions of the previous layer (9×9). Third, we add a deconvolutional layer that upsamples its input by a factor of 4 (with a learnable kernel of size 8×8), in order to recover the input resolution. Notice that the tasks of classification and upsampling are now separated.

This new network can take input images of different sizes, with the output size varying accordingly. For example, during the training stage, we wish to output the patches of size 16×16 in order to emulate the learning process as was done in the patch-based network of Fig. 1(a). For this, we require a patch input of size 80×80, as in the architecture of Fig. 1(b). Notice that the input is larger than the original 64×64 patches. This is not because we are taking more context to carry out the predictions, but instead because every output is now centered in its context. At inference time, we can take inputs of arbitrary sizes and feed them to the network to construct the classification maps, and the number of network parameters does not vary.

In the deconvolutional layer shown in Fig. 1(b), the overlapping areas added to produce the output are shown in gray while the excluded extrapolation is in white.

D. Experiments on Fully Convolutional Networks

We implemented the CNNs using the *Caffe* deep learning framework [34]. In a first experiment, we apply our approach to the Massachusetts Buildings data set [29]. This data set consists of color images over the area of Boston with 1 m^2 spatial resolution, covering an area of 340 km^2 for training, 9 km^2 for validation, and 22.5 km^2 for testing. The images are labeled into two classes: *building* and *not building*. A portion of an image and its corresponding reference are shown in Fig. 4(a) and (b).

We train the patch-based and fully convolutional networks [Fig. 1(a) and (b), respectively] for 30 000 stochastic gradient descent iterations, until we observe barely no further improvement on the validation set. The patches are sampled uniformly from the whole training set, with mini-batches of 64 patches each and a learning rate of 0.0001. A momentum and an L2 parameter penalty are introduced to regularize the learning process and avoid overfitting. Momentum adds a fraction of the previous gradient to the current one in order to smooth the descent, while an L2 penalty on the learned parameters discourages neurons to specialize too much on particular training cases [30]. The weights of these regularizers are set to 0.9 and 0.0002, respectively. Further details on these so-called hyperparameters and rationale for selecting them are provided by Mnih [29].

To evaluate the accuracy of the classification, we use two different measures: pixelwise accuracy (proportion of correctly classified pixels, obtained through binary classification of the output probabilities with threshold 0.5) and area under the receiver operating characteristics (ROC) curve [35]. The latter quantifies the relation between true and false positives at different thresholds, and is appropriate to evaluate the overall quality of the fuzzy maps.

Fig. 5(a) shows the evolution of the area under ROC curve (AUC) and pixelwise accuracy in the test set,

across iterations. The FCN consistently outperforms the patch-based network. Fig. 5(b) shows ROC curves for the final networks after convergence, the FCN exhibiting the best relation between true and false positive rates. Fig. 4(c) and (d) shows some visual results.

To further illustrate the benefits of neural networks over other learning approaches, we train an SVM with Gaussian kernel on 1000 randomly selected pixels of each class. We train on the individual pixel spectra without any feature selection. The SVM parameters are selected by fivefold cross validation, as commonly performed in remote-sensing image classification [10]. As shown in Fig. 4(e), the pixelwise SVM classification often confuses roads with buildings due to the fact that their colors are similar, while neural networks better infer and separate the classes by considering the geometry of the context. The accuracy of the SVM on the Boston test data set is 0.6229 and its AUC is 0.5193, i.e., significantly lower than with CNNs, as shown in Fig. 5. If we wished to successfully use an SVM for this task, we should design and select spatial features (e.g., texture) and use them as the input to the classifier instead.

The amplified fragment in Fig. 2 shows that the border discontinuity artifacts present in the patch-based scheme are absent in our fully convolutional setting. This behaves as expected considering that the issues described in Section III-B are addressed by construction in the new architecture. This confirms that imposing sensible restrictions to the connections of a neural network has a positive impact in the performance.

In terms of efficiency, the FCN also outperforms the patch-based CNN. At inference time, instead of carrying out the prediction in a small patch basis, the input of the FCN is simply increased to output larger predictions, better benefiting from the GPU parallelization of convolutions. The execution time required to classify the whole Boston 22.5 km^2 test set (performed on an Intel I7 CPU at 2.7 GHz with a Quadro K3100M GPU) is 82.21 s with the patch-based CNN against 8.47 s with the FCN. The speedup is about $10\times$, a relevant improvement considering the large-scale processing capabilities required by new sensors.

IV. End-to-End Framework

In remote-sensing image analysis, it is a common practice to train classifiers on the spectrum of a small number (a couple of hundreds) of isolated sample pixels [36]. Training relies on the trustworthiness of the reference data and on the fact that classes are reliably separable simply by observing the spectral signature of the sampled pixels. While such training approaches are popular, for example, in hyperspectral image classification, our goals differ as we wish to automatically learn contextual features that can help better identify the classes in satellite imagery. Our goal requires more training data *per se*, as we must show the classifier the many different contexts in which a pixel class can be embedded, and not just its spectral values. In addition, is it well known that massive data might be required to train neural networks, contrary to a common feature selection and classification approach. This led us to analyze and address the dependence of the algorithm on the availability and accuracy of the training data.

In the experiments described in Section III-D, the Massachusetts Buildings data set is used for training and testing. This data set is a hand-corrected version of the OSM vectorial map available over the area covered by the images. Despite the existence of some inaccuracies in the reference data, the coverage of OSM in that region is satisfactory and the errors are minor.

In many other areas of Earth, however, the coverage of OSM is limited. In the samples of Fig. 8, we observe large areas with missing data and a general misregistration of the vectorial maps with respect to the actual structures. In addition, the misregistration is not uniform, and neighboring buildings are often shifted in different directions. Note that in the samples of Fig. 8, the buildings have been delineated in OSM based on the official French cadaster records. However, even the cadaster records are not always accurate up to the meter resolution. Furthermore, satellite images undergo a series of corrections before being aligned to the maps. For example, the use of inexact elevation models for orthorectification might introduce misregistrations throughout the images. As a result, the OSM raw data are imperfect and thus not fully reliable.

The reference data obtained from OSM, as shown in Fig. 8, provide a rough idea of the location of the buildings, but rarely outlines them. In such a setting, CNNs would hardly learn that building boundaries are likely to fall on visible edges, since this is not what the reference data depicts. Under these circumstances, we expect the predictions not to be very confident, especially on the border of the objects. As we will illustrate in Section IV-C, this yields a "blobby" and overly fuzzy aspect to predictions obtained with the network of Section III-C on more challenging data sets.

Our first contribution in this section is a novel approach for tackling the issue of inaccurate labels for CNN training. For this, we propose a two-step approach: 1) the network is first trained on raw OSM data and 2) it is then fine-tuned on a tiny piece of manually labeled image.

This method provides us with a means to deal with the inaccuracy of training data, by increasing the confidence and sharpness of the predictions. However, we still cannot expect it to provide highly precise boundaries with the fully convolutional architecture as described in Section III-C. This is because such network includes a downsampling step, required to capture the long-range spatial dependencies that help recognize the classes. However, downsampling makes the whole system lose spatial precision, and the deconvolutional layer learns a way of naively upsampling the data from a restricted number of neighbors, without reincorporating higher resolution information. What is lost in spatial precision through the network is not recovered. This is a consequence of a well-known tradeoff between the receptive field (how much context is taken to conduct predictions) and the output resolution (how fine is the prediction) if we wish to keep a reasonable number of trainable parameters [33]. Our second contribution is then a new architecture that incorporates information at multiple scales in order to alleviate this tradeoff. Our architecture combines low-resolution long-range features with high-resolution local features that conduct predictions with a higher level of detail. This architecture, when combined with our two-step

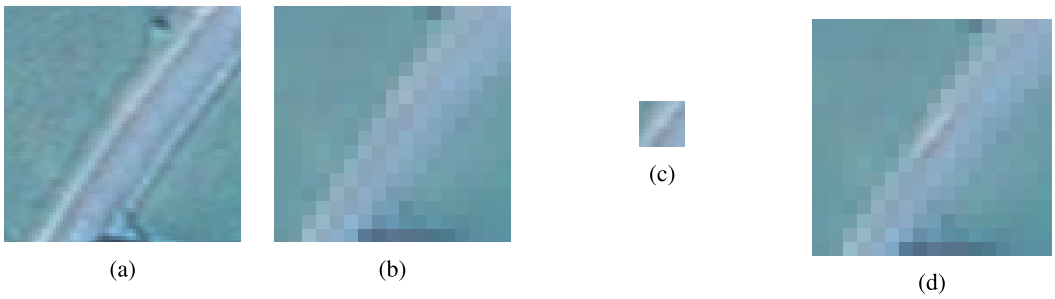


Fig. 6. Different types of context to predict a pixel's class. A multiscale context such as in (d) alleviates the tradeoff between classification accuracy and number of learnable parameters. (a) Large context, high resolution. (b) Large context, low resolution. (c) Small context, high resolution. (d) Combined scales.

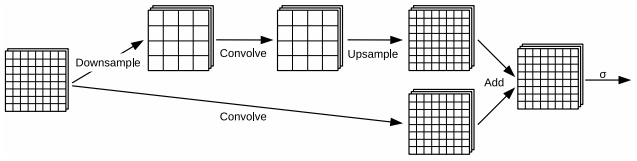


Fig. 7. Two-scale convolutional module that simultaneously combines coarse large-range and fine short-range reasoning.

Fig. 8. Fragments of the Forez training set (red: *building*).

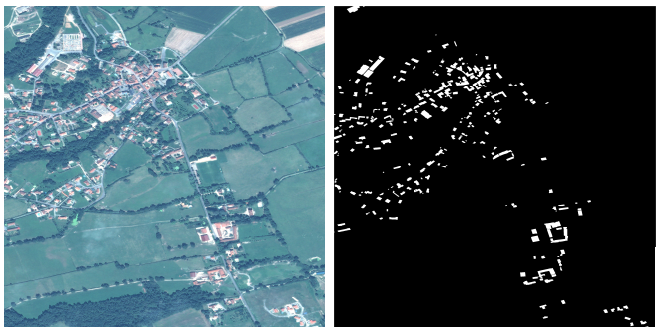


Fig. 9. Manually labeled tile for fine-tuning (3000 × 3000).

Fig. 10. Fragment of the fine-tuning tile. Red borders enclose building areas.

training approach, provides a framework that can be used end-to-end to classify satellite imagery.

A. Fine-Tuning

Fine-tuning is a very common procedure in the neural network literature. The idea is to adapt an existing pretrained model to a different domain by executing a few training iterations on a new data set. The notion of fine-tuning is based on the intuition that low-level information/features can be reused in different applications, without training from scratch. Even when the final classification objective is different, it is also a relevant approach for initializing the learnable parameters close to good local minima, instead of initializing with random weights. After proper fine-tuning, low-level features tend to be quite preserved from one data set to another, while the higher layers' parameters are updated to adapt the network to the new problem [37].

When fine-tuning, the training set for the new domain is usually substantially smaller than the one used to train the original network. This is because one assumes that some generalities of both domains are well conveyed in the pre-trained network (e.g., edge detectors in different directions) and the fine-tuning phase is just needed to conduct the domain adaptation. When the training set used for fine-tuning is very small, additional considerations to avoid overfitting are commonly taken, such as early stopping (executing just a few iterations on the new training data set), fixing the weights at the lower layers, or reducing the learning rate.

We now incorporate the idea of neural network fine-tuning, in order to perform training on imperfect data. Our approach proceeds in two steps. In step 1, large amounts of training data are used to train a fully CNN. This raw training data are extracted directly from OSM, without any hand correction. The goal of this step is to capture the generalities of the data set, such as the representative spectrum of object classes.

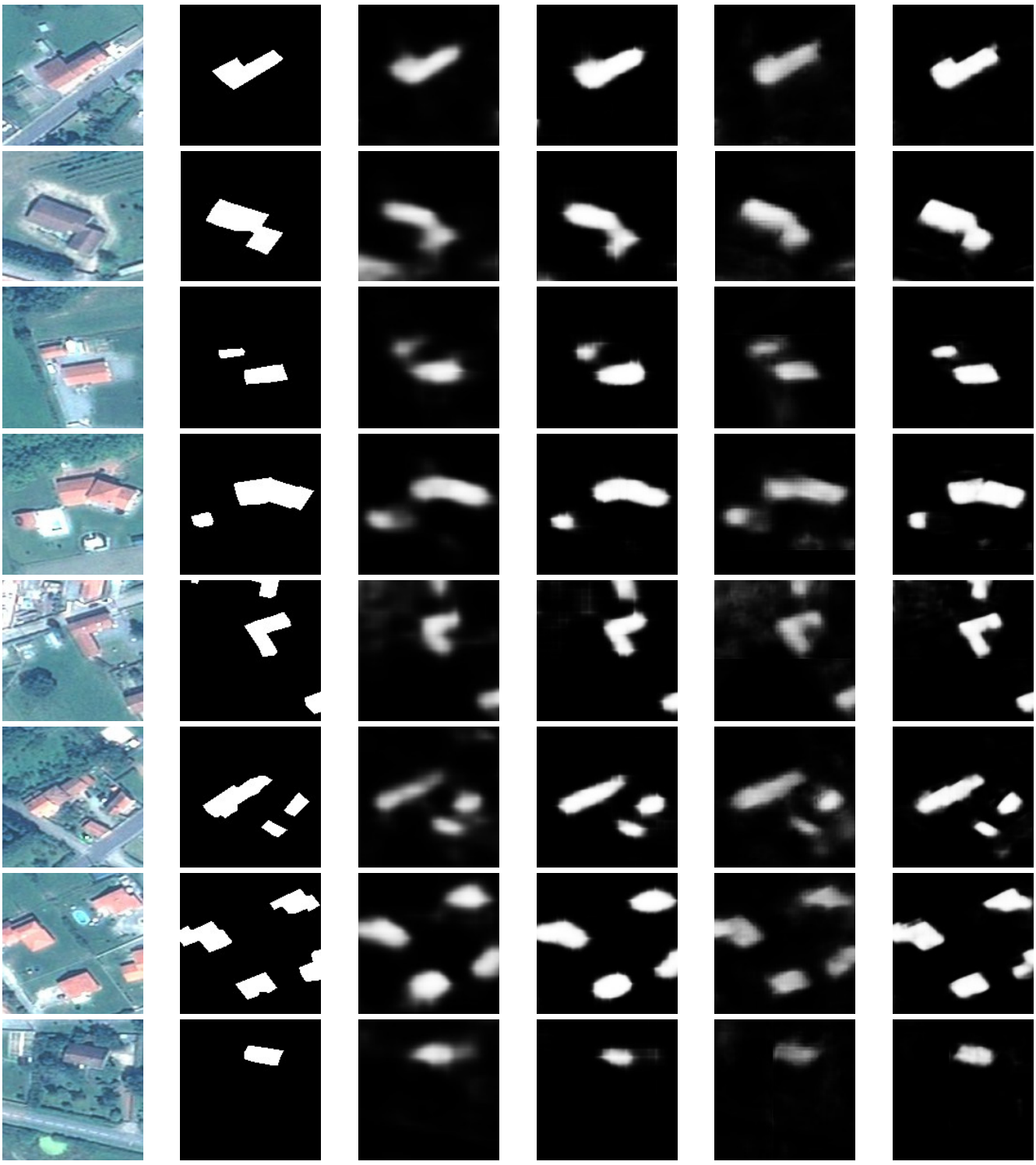


Fig. 11. Classified fragments of the Pléiades test image. Fine-tuning increases the confidence of the predictions, and the two-scale network produces fine-grained classification maps. (a) Color image. (b) Reference. (c) FCN. (d) FCN + fine-tuning. (e) Two-scale FCN. (f) Two-scale FCN + fine-tuning.

In step 2, we fine-tune the network by using a small part of carefully labeled image. This phase is designed to compensate for the inaccuracy of labels obtained in step 1, by fine-tuning the network on small yet consistent target outputs. Assuming that most of the generalities have been captured during the initial training step, the fine-tuning step should locally correct the network parameters to output more accurate classifications. The efforts of fine-tuning are thus limited to manually labeling a small data set, while the large inaccurate data set is automatically extracted from OSM.

B. Conducting Fine Predictions

The resolution at which the networks proposed in Section III operate yields probability maps that, once upsampled, are coarse in terms of spatial accuracy. A naive way to increase the resolution of the network would be to use higher resolution filters, which requires to increase their dimensions if we want to preserve the receptive field. For example, instead of applying a 5×5 filter at the fourth of the image resolution, one could use a 20×20 filter at full resolution, hence covering the same spatial extent. However, such an increase in filter sizes is prohibitive, hampering the spatial and temporal efficiency of the algorithm and producing less accurate results due to the difficulty of optimizing so many parameters.

Nevertheless, we observed that we do not need full-resolution filters to conduct accurate predictions. One requires a higher resolution only in the center of the convolution filters (assuming that the pixel, we wish to predict, is in the center of the context of interest). A large spatial extent is indeed required to capture contextual information, but it is not necessary to conduct this analysis at full resolution. For example, the presence of two parallel bands of grass can help identify a road (and distinguish it from, for instance, a building with a gray rooftop), but a precise localization of the grass is not necessary. On the contrary, at the center of the convolution filter, a higher resolution analysis is required to specifically locate the boundary of the aforementioned road.

Fig. 6 shows this observation. In Fig. 6(a), we observe the area around a pixel whose class we wish to predict, at full resolution. A filter taking such an amount of context with that resolution would be prohibitive in the number of parameters, as well as unnecessary. Fig. 6(b) shows the same context at a quarter of the resolution. Notice that it is still possible to visually infer that there is a road. However, identifying the precise location of the boundaries of the road becomes difficult. Alternatively, Fig. 6(c) shows a small patch but at full resolution. We can now better locate the precise boundary of the object, but with so little context, it is difficult to identify that the object is indeed a road. Large filters at low resolution—see Fig. 6(b) or small filters at high resolution—see Fig. 6(c), which would both have a reasonable number of parameters, are bad alternatives: the first filter is too coarse and the second filter is using too little context.

We propose convolutional filters that combine multiple scales instead. In Fig. 6(d), the large-size low-resolution context of Fig. 6(b) is combined with the small high-resolution context of Fig. 6(d). This provides us with a means to simultaneously infer the class by observing the surroundings at a coarse scale, and determine the precise boundary location by using a finer context. This way, the amount of parameters is kept small while the tradeoff between recognition and localization is alleviated.

Let us denote by $\mathcal{S}$ a set of levels of detail expressed as a fraction of the original resolution. For example, $\mathcal{S} = \{1, 1/2\}$ is a set comprising two scales: full resolution and half of the full resolution. We denote by $\mathbf{x}_s$ a feature map $\mathbf{x}$ downsampled to a certain level $s \in \mathcal{S}$. For example, $\mathbf{x}_{1/2}$ is a feature map downsampled to half of the original resolution. Inspired in (1), we design a special type of neuron that adds the responses to a set of filters applied at different scales of the feature maps in the previous layer

$$a = \sigma\left(\sum_{s \in \mathcal{S}} \mathbf{w}_s \mathbf{x}_s + b\right). \tag{5}$$

Notice that individual filters $\mathbf{w}_s$ are learned for every scale s. Such a filter is easily implemented by using a combination of elementary convolutional, downsampling, and upsampling layers. Fig. 7 shows this process in the case of a two-scale ($\mathcal{S} = \{1, 1/2\}$) module. In our implementation, we average neighboring elements in a window for downsampling and perform bilinear interpolation for upsampling, but other approaches are also applicable. The kernel sizes of the convolutions at both scales are set to be equal (e.g., 3×3), yet the amount of context taken varies from one path to the other due to the different scales. The addition is an elementwise operation, followed by the nonlinear activation function.

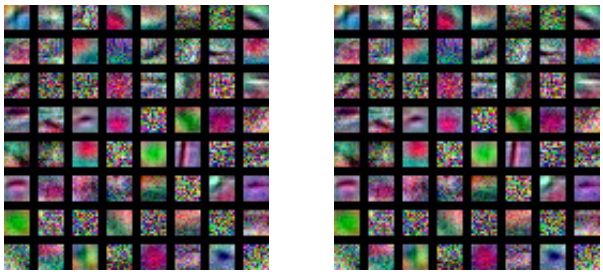


Fig. 12. First-layer filters before and after fine-tuning.

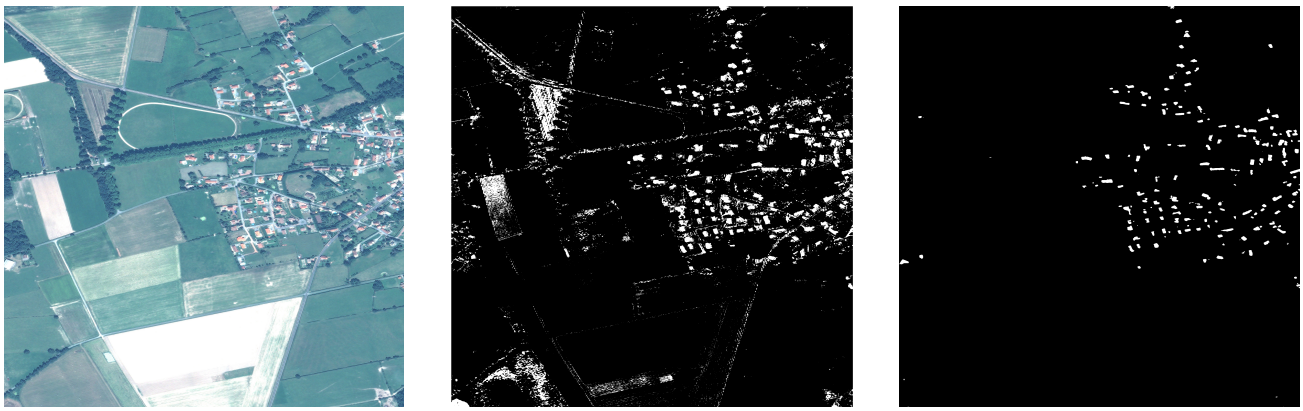
C. Experiments on the End-to-End Classification Framework

We conduct our experiments on a Pléiades image over the area of Forez, France. An orthorectified color pansharpened version of the image is used, at a spatial resolution of 0.5 m^2. Our training subset amounts to 22.5 km^2. The criterion to construct the training set was to choose ten 3000×3000 tiles with at least some coverage of OSM. The shape files were rasterized with GDAL[1] to create the binary reference maps. Fig. 8 shows some fragments of the reference data. Inconsistent misregistrations and considerable omissions are observed all over.

We manually labeled a 2.25 km^2 tile for FCN fine-tuning, and a different 2.25 km^2 tile for testing. The manual labeling takes about 2 h for each of the tiles. The entire tuning tile is shown in Fig. 9 and a close-up is shown in Fig. 10.

The fully convolutional network (FCN) described in Section III-C, which was used for the Massachusetts data set, is now trained with the Forez set, under a similar experimental setting. Note that this FCN was designed for images, which have a 1 m^2 resolution, while Pléiades imagery features a 0.5 m^2 resolution. In order for the architectural decisions of FCN to be valid in our new data set, one must preserve the receptive field size in terms of meters, not pixels. We thus downsample Pléiades images prior to entering the first layer of the FCN, and bilinearly upsample the output classification maps. Even though a new network directly tailored

[1] http://www.gdal.org



(a) (b) (c)

Fig. 13. Binary classification maps on the Forez test image. (a) Color pansharpened input. (b) SVM on individual pixels. (c) FCN (two scales + fine-tuning).

to the Pléiades resolution could be designed, we favor this proven architecture to conduct our experiments. The concepts described in this paper are, however, general and can be used to design other networks.

After training on the raw OSM Forez data set, we fine-tune the weights on the manually labeled tuning tile. The training hyperparameters are kept similar in the fine-tuning step, but an early stopping criterion interrupts it after 200 iterations.

To assess the performance of fine-tuning, we use as criteria pixelwise accuracy and AUC, as described in Section III-D. Since there are many more nonbuilding pixels than building pixels in this data set, these accuracy measures might seem overly high, a well-known issue of pixelwise accuracy in imbalanced data sets [38]. We then add the intersection over union (IoU) criterion, an object-based overlap measure typically used for imbalanced data sets [38]. In our case, it is defined as the number of pixels labeled as buildings both in the classified image and in the ground truth, divided by the total amount of pixels labeled as such in either of them. These criteria are evaluated on the manually labeled test set, which is used neither for training nor for fine-tuning. The first two rows of Table I show that fine-tuning enhances the quality of the predictions in terms of accuracy, AUC, and IoU. To confirm the significance of the accuracy, McNemar's test [39] proved that the improvement is not a result of mere luck with a probability greater than 0.99999. Besides, the IoU is improved by over a third with the fine-tuned network.

Fig. 11(a)–(d) shows the impact of fine-tuning on several amplified fragments of the test set. A greater confidence in the fine-tuned network predictions is observed. The objects exhibit better alignment to the objects of the image, albeit the boundaries could better line up to the underlying edges.

Fig. 12 shows the first-layer convolutional filters learned by the initial and fine-tuned networks. We observe a combination of low- and high-frequency filters, a behavior typically observed in CNNs. We also observe edge and color blob detectors. These filters remain unchanged after fine-tuning, even though no constraints are introduced to enforce this. Fine-tuning corrects the weights in the high-level layers, which suggests that the initial low-level features were useful indeed, but the inaccuracy in the labels was introducing fuzziness in the upper layers of the network.

TABLE I
PERFORMANCE EVALUATION ON THE Pléiades TEST SET

Method	Accuracy	AUC	IoU
FCN	0.99126	0.99166	0.48
FCN + Fine-tuning	0.99459	0.99699	0.66
Two-scale FCN	0.99129	0.98154	0.47
Two-scale FCN + Fine-tuning	0.99573	0.99836	0.72

We now evaluate the performance of a two-scale network. The FCN architecture described in Section III-C is replaced by three two-scale stacked modules, with scales $\mathcal{S} = \{1, 1/4\}$. We select $\mathcal{S} = 1/4$ as it corresponds to the degree of downsampling of the original FCN network, and $\mathcal{S} = 1$ is added to refine the predictions. The three modules learn 3×3 filters in both scales. The first two modules generate 64 feature maps and the last module generates a single map with the building/nonbuilding prediction.

The two-scale network is trained and fine-tuned in a similar setting as the FCN network. The results summarized in the last two rows of Table I show that fine-tuning significantly enhances the classification performance, and that the fine-tuned two-scale network outperforms the single scale network. Notably, IoU goes from 0.48 to 0.72, implying that objects overlap with the ground truth 50% better by adding a scale and performing fine-tuning. Note that if a scale is added but no fine-tuning is done, there is actually a slight decrease in performance. A possible explanation for this is that including a finer scale adds even more confusion to the training algorithm if only noisy misregistered labels are provided.

Fig. 11(e) and (f) shows the results on visual fragments of the test set. The two-scale network yields classification maps that better correspond to the actual image objects, and exhibit sharper angles and straighter lines. The entire classified test tile for the fine-tuned two-scale network is shown in Fig. 13(c). The time required to generate this result corresponds to 3 h for training on the OSM data set, 2 h to manually label an image tile, and about 1 min for fine-tuning. The prediction of the 3000×3000 test tile using the hardware described

in Section III-D takes 3.2 s, and it grows linearly in the size of the image. As in Section III-D, we ran an SVM on the individual pixel values [see the classification map in Fig. 13(b)]. Accuracy is 0.9487 and IoU 0.19, yielding poorer results than the presented CNN-based approaches.

As validated by the experiments, the issue of not having large amounts of high-quality reference data can be alleviated by providing the network with a small amount of accurate data in a fine-tuning step. Our multiscale neurons combine reasoning at different resolutions to effectively produce fine predictions, while keeping a reasonable number of parameters. Such a framework can be used end-to-end to perform the classification task directly from input imagery. More scales can be easily be added and, besides the fact of being fully convolutional, there are little constraints on the architecture itself, admitting a different number of classes, input bands or number of feature maps.

V. Concluding Remarks

CNNs have become a popular classifier in the context of image analysis due to their potential to automatically learn relevant contextual features. Initially devised for the categorization of natural images, these networks must be revisited and adapted to tackle the problem of pixelwise labeling in remote-sensing imagery.

We proposed a fully convolutional network architecture by analyzing a state-of-the-art model and solving its concerns by construction. Despite their outstanding learning capability, the lack of accurate training data might limit the applicability of CNN models in realistic remote-sensing contexts. We, therefore, proposed a two-step training approach combining the use of large amounts of raw OSM data and a small sample of manually labeled reference. The last ingredient we needed to provide a usable end-to-end framework for remote-sensing image classification was to produce fine-grained classification maps, since typical CNNs tend to hamper the fineness of the output as a side effect of taking large amounts of context. We proposed a type of neuron module that simultaneously reasons at different scales.

Experiments showed that our fully convolutional network outperforms the previous model in multiple aspects: the accuracy of the results is improved; the visual artifacts are removed, and the inference time is reduced by a factor of 10. The use of our architecture then constitutes a win–win situation in which no aspect is compromised for the others. This was achieved by analyzing the role played by every layer in the network in order to propose a more appropriate architecture, showing that a deep understanding of how CNNs work is important for their success. Further experimentation showed that the two-step training approach effectively combines imperfect training data with manually labeled data to capture the data set's generalities and its precise details. Moreover, the multiscale modules increase the level of detail of the classification without making the number of parameters explode, attenuating the tradeoff between detection and localization.

Our overall framework then shows that CNNs can be used end-to-end to process large amounts of satellite images and provide accurate pixelwise classifications.

As future work, we plan to extend our experiments to multiple object classes and study the possibility of directly inputting nonpansharpened imagery, in order to avoid this preprocessing step. We also plan to study the introduction of shape priors in the learning process and the vectorization of the classification maps.

Acknowledgment

The authors would like to thank ©CNES for initializing the study and providing Pléiades data. All Pléiades images are CNES (2012 and 2013), distribution Airbus DS / SpotImage.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[2] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. IEEE CVPR*, Jun. 2012, pp. 3642–3649.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.

[4] A. S. Kumar and K. Majumder, "Information fusion in tree classifiers," *Int. J. Remote Sens.*, vol. 22, no. 5, pp. 861–869, 2001.

[5] J. Mas and J. Flores, "The application of artificial neural networks to the analysis of remotely sensed data," *Int. J. Remote Sens.*, vol. 29, no. 3, pp. 617–663, 2008.

[6] T. Villmann, E. Merényi, and B. Hammer, "Neural maps in remote sensing image analysis," *Neural Netw.*, vol. 16, nos. 3–4, pp. 389–403, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608003000212

[7] G. Camps-Valls and L. Bruzzone, "Kernel-based methods for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 6, pp. 1351–1362, Jun. 2005.

[8] M. Fauvel, Y. Tarabalka, J. A. Benediktsson, J. Chanussot, and J. C. Tilton, "Advances in spectral-spatial classification of hyperspectral images," *Proc. IEEE*, vol. 101, no. 3, pp. 652–675, Mar. 2013.

[9] W. Liao, M. D. Mura, J. Chanussot, R. Bellens, and W. Philips, "Morphological attribute profiles with partial reconstruction," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1738–1756, Mar. 2016.

[10] Y. Tarabalka and A. Rana, "Graph-cut-based model for spectral-spatial classification of hyperspectral images," in *Proc. IEEE IGARSS*, Jul. 2014, pp. 3418–3421.

[11] M. N. Kurnaz, Z. Dokur, and T. Ölmez, "Segmentation of remote-sensing images by incremental neural network," *Pattern Recognit. Lett.*, vol. 26, no. 8, pp. 1096–1104, Jun. 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865504003113

[12] C. D. Lloyd, S. Berberoglu, P. Curran, and P. Atkinson, "A comparison of texture measures for the per-field classification of mediterranean land cover," *Int. J. Remote Sens.*, vol. 25, no. 19, pp. 3943–3965, 2004.

[13] D. Lu and Q. Weng, "A survey of image classification methods and techniques for improving classification performance," *Int. J. Remote Sens.*, vol. 28, no. 5, pp. 823–870, 2007.

[14] M. E. Midhun, S. R. Nair, V. T. N. Prabhakar, and S. S. Kumar, "Deep model for classification of hyperspectral image using restricted boltzmann machine," in *Proc. Int. Conf. Interdiscipl. Adv. Appl. Comput.*, 2014, p. 35.

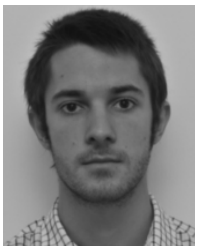
[15] T. Li, J. Zhang, and Y. Zhang, "Classification of hyperspectral image based on deep belief networks," in *Proc. IEEE ICIP*, Oct. 2014, pp. 5132–5136.

[16] V. Slavkovikj, S. Verstockt, W. De Neve, S. Van Hoecke, and R. Van de Walle, "Hyperspectral image classification with convolutional neural networks," in *Proc. 23rd ACM Int. Conf. Multimedia*, 2015, pp. 1159–1162.

[17] Y. Chen, X. Zhao, and X. Jia, "Spectral–spatial classification of hyperspectral data based on deep belief network," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 6, pp. 2381–2392, Jun. 2015.

[18] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2094–2107, Jun. 2014.

[19] J. Yue, W. Zhao, S. Mao, and H. Liu, "Spectral–spatial classification of hyperspectral images using deep convolutional neural networks," *Remote Sens. Lett.*, vol. 6, no. 6, pp. 468–477, 2015.
[20] K. Makantasis, K. Karantzalos, A. Doulamis, and N. Doulamis, "Deep supervised learning for hyperspectral data classification through convolutional neural networks," in *Proc. IEEE IGARSS*, Jul. 2015, pp. 4959–4962.
[21] W. Zhao and S. Du, "Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach," *IEEE Tran. Geosci. Remote Sens.*, vol. 54, no. 8, pp. 4544–4554, Aug. 2016.
[22] W. Zhao, Z. Guo, J. Yue, X. Zhang, and L. Luo, "On combining multiscale deep learning features for the classification of hyperspectral remote sensing imagery," *Int. J. Remote Sens.*, vol. 36, no. 13, pp. 3368–3379, 2015.
[23] W. Zhao and S. Du, "Learning multiscale and deep representations for classifying remotely sensed imagery," *ISPRS J. Photogram. Remote Sens.*, vol. 113, pp. 155–165, Mar. 2016.
[24] E. Basaeed, H. Bhaskar, P. Hill, M. Al-Mualla, and D. Bull, "A supervised hierarchical segmentation of remote-sensing images using a committee of multi-scale convolutional neural networks," *Int. J. Remote Sens.*, vol. 37, no. 7, pp. 1671–1691, 2016.
[25] J. Wang, J. Song, M. Chen, and Z. Yang, "Road network extraction: A neural-dynamic framework based on deep learning and a finite state machine," *Int. J. Remote Sens.*, vol. 36, no. 12, pp. 3144–3169, 2015.
[26] X. Chen, S. Xiang, C.-L. Liu, and C.-H. Pan, "Vehicle detection in satellite images by hybrid deep convolutional neural networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 10, pp. 1797–1801, Oct. 2014.
[27] I. Ševo and A. Avramović, "Convolutional neural network based automatic object detection on aerial images," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 5, pp. 740–744, May 2016.
[28] F. P. S. Luus, B. P. Salmon, F. Van Den Bergh, and B. Maharaj, "Multiview deep learning for land-use classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 12, pp. 2448–2452, Dec. 2015.
[29] V. Mnih, "Machine learning for aerial image labeling," Ph.D. dissertation, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2013.
[30] C. M. Bishop, *Neural Networks for Pattern Recognition*. London, U.K.: Oxford Univ. Press, 1995.
[31] J. Michel, D. Youssefi, and M. Grizonnet, "Stable mean-shift algorithm and its application to the segmentation of arbitrarily large remote sensing images," *IEEE Tran. Geosci. Remote Sens.*, vol. 53, no. 2, pp. 952–964, Feb. 2015.
[32] P. Lassalle, J. Inglada, J. Michel, M. Grizonnet, and J. Malik, "A scalable tile-based framework for region-merging segmentation," *IEEE Tran. Geosci. Remote Sens.*, vol. 53, no. 10, pp. 5473–5485, Oct. 2015.
[33] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. CVPR*, 2015, pp. 3431–3440.
[34] Y. Jia *et al.* (Jun. 2014). "Caffe: Convolutional architecture for fast feature embedding." [Online]. Available: https://arxiv.org/abs/1408.5093
[35] C. Ferri and J. Hernández-Orallo, and P. A. Flach, "A coherent interpretation of AUC as a measure of aggregated classification performance," in *Proc. ICML*, 2011, pp. 1–8.
[36] Y. Tarabalka, M. Fauvel, J. Chanussot, and J. A. Benediktsson, "SVM- and MRF-based method for accurate classification of hyperspectral images," *IEEE Geosci. Remote Sens. Lett.*, vol. 7, no. 4, pp. 736–740, Oct. 2010.
[37] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. NIPS*, 2014, pp. 3320–3328.
[38] G. Csurka, D. Larlus, F. Perronnin, and F. Meylan, "What is a good evaluation measure for semantic segmentation?" in *Proc. BMVC*, vol. 27, 2013.
[39] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, Jun. 1947.



Emmanuel Maggiori (S'15) received the Engineering degree in computer science from the National University of Central Buenos Aires, Tandil, Argentina, in 2014. He is currently pursuing the Ph.D. degree with the TITANE Team, with a focus in machine learning techniques for large-scale processing of satellite imagery.

In 2014, he joined the AYIN and STARS teams with the Inria Sophia Antipolis-Méditerranée, Valbonne, France, as a Research Intern, where he is involved in remote sensing image processing.

Yuliya Tarabalka (S'08–M'10) received the B.S. degree in computer science from Ternopil Ivan Puluj National Technical University, Ternopil, Ukraine, in 2005, the M.Sc. degree in signal and image processing from the Grenoble Institute of Technology (INPG), Grenoble, France, in 2007, and the joint Ph.D. degree in signal and image processing from INPG and in electrical engineering from the University of Iceland, Reykjavik, Iceland, in 2010.

From 2007 to 2008, she was a Researcher with the Norwegian Defence Research Establishment, Kjeller, Norway. From 2010 to 2011, she was a Post-Doctoral Research Fellow with the Computational and Information Sciences and Technology Office, NASA Goddard Space Flight Center, Greenbelt, MD, USA. In 2012, she was a Post-Doctoral Research Fellow with the French Space Agency, Toulouse, France and the Inria Sophia Antipolis-Méditerranée, Valbonne, France. She is currently a Researcher with the TITANE Team, Inria Sophia Antipolis-Méditerranée. Her current research interests include image processing, pattern recognition, and development of efficient algorithms.

Guillaume Charpiat received the B.S. degree in mathematics and physics from the École Normale Supérieure (ENS) at Paris, France, the M.Sc. degree in computer vision and machine learning, and theoretical physics from ENS at Cachan, France, and the Ph.D. degree in computer science at ENS in 2006. His Ph.D. thesis was on the distance-based shape statistics for image segmentation with priors.

He was with the Max-Planck Institute for Biological Cybernetics, Tübingen, Germany, where he was involved in medical imaging (MR-based PET prediction) and automatic image colorization. He was a Researcher with the Inria Sophia Antipolis-Méditerranée, Valbonne, France, where he was involved in image segmentation and optimization techniques. He is currently a Researcher with the TAO Team, Inria Saclay, Palaiseau, France, where he is involved in machine learning, in particular on building a theoretical background for neural networks.

Pierre Alliez is currently a Senior Researcher and the Team Leader with Inria Sophia Antipolis-Méditerranée, Valbonne, France. He has authored scientific publications and several book chapters on mesh compression, surface reconstruction, mesh generation, surface remeshing, and mesh parameterization.

Dr. Alliez was a recipient of the EUROGRAPHICS Young Researcher Award in 2005 for his contributions to computer graphics and geometry processing and a Starting Grant from the European Research Council on Robust Geometry Processing in 2011. He was the Co-Chair of the Symposium on Geometry Processing in 2008, Pacific Graphics in 2010, and Geometric Modeling and Processing 2014. He is currently an Associate Editor of the Computational Geometry Algorithms Library and an Associate Editor of the *ACM Transactions on Graphics*.