High-Resolution Semantic Labeling with Convolutional Neural Networks

Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, Pierre Alliez

High-Resolution Semantic Labeling with Convolutional Neural Networks

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

***Abstract*—Convolutional neural networks (CNNs) have received increasing attention over the last few years. They were initially conceived for image categorization, i.e., the problem of assigning a semantic label to an entire input image.**

In this paper we address the problem of dense semantic labeling, which consists in assigning a semantic label to *every* pixel in an image. Since this requires a high spatial accuracy to determine *where* labels are assigned, categorization CNNs, intended to be highly robust to local deformations, are not directly applicable.

By adapting categorization networks, many semantic labeling CNNs have been recently proposed. Our first contribution is an in-depth analysis of these architectures. We establish the desired properties of an ideal semantic labeling CNN, and assess how those methods stand with regard to these properties. We observe that even though they provide competitive results, these CNNs often underexploit properties of semantic labeling that could lead to more effective and efficient architectures.

Out of these observations, we then derive a CNN framework specifically adapted to the semantic labeling problem. In addition to learning features at different resolutions, it learns how to combine these features. By integrating local and global information in an efficient and flexible manner, it outperforms previous techniques. We evaluate the proposed framework and compare it with state-of-the-art architectures on public benchmarks of high-resolution aerial image labeling.

***Index Terms*—Semantic labeling, convolutional neural networks, deep learning, high-resolution aerial imagery.**

I. Introduction

SEMANTIC labeling is the problem of assigning a semantic class to every individual pixel of an image. In certain application domains, such as high-resolution aerial image analysis, it is of paramount importance to provide fine-grained classification maps where object boundaries are precisely located. For example, with the advent of autonomous driving there is an increasing interest in locating the exact boundaries of roads or even lanes from aerial imagery [1].

Over the last few years, *deep learning* and, in particular, convolutional neural networks (CNNs), have gained significant attention in the image analysis community. These have been originally devised for the image *categorization* problem, i.e., the assignment of one label to an entire image. For example, they have been used to categorize objects in natural scenes (e.g., *airplane*, *bird*, *person*) or land use in the case of aerial images (e.g., *forest*, *beach*, *tennis court*). CNNs jointly learn to extract relevant contextual features and conduct the categorization. In addition to the suppression of the feature design process, which is an interesting advantage itself, this technique has consistently beaten alternative methods in a wide range of problems [2]. Nowadays, one can reasonably expect to find CNN-based techniques scoring the best positions in the leaderboards of online image-related contests.

While neural networks have existed for a few decades, a combination of recent advances has facilitated their development as deep learning techniques. One of these advances is the use of novel architectures. Notably, the novelty in the aforementioned convolutional network is its architecture, which imposes significant restrictions to the neuronal connections compared to previous approaches. While CNNs are thus less general than traditional architectures, the restrictions applied are well grounded in the domain of image analysis, reducing thus the optimization search space in a sensible way. This directs the network to learn a more appropriate function, yielding better categorization results. The lesson learned is that finding the right type of architecture for a given problem often boosts the performance of neural networks. Moreover, fewer computational resources are required for training and conducting labeling.

A sort of “recipe” or meta-architecture for the image categorization problem was incrementally developed in the community. The typical ingredients of a state-of-the-art CNN to categorize images are a combination of convolutional, so-called pooling layers and rectifed linear units, followed by traditional fully-connected layers. However, when it comes to semantic pixel labeling (i.e., assigning a class to every pixel), this categorization recipe cannot be directly transferred. Indeed, while categorization networks are devised to lose spatial precision in order to identify objects that come in different appearances, semantic labeling networks should preserve the spatial resolution to correctly locate object boundaries. This is not straightforward to implement, because of a well-known trade-off between recognition and localization [9], [11], due to the need to keep the networks small (and thus more efficient and easier to train). Since both qualities are required in semantic labeling at the same time, it is important to design specific architectures for this problem.

There have been recent research efforts in this direction, using CNNs for pixel labeling and, in particular, for high-resolution aerial image labeling (e.g., [3], [4]). These networks certainly provide good results and stand as competitive alternatives compared to other methods. However, there is still a

E. Maggiori, Y. Tarabalka and P. Alliez are with Université Côte d’Azur, TITANE team, Inria, 2004 route des Lucioles, BP93 06902 Sophia Antipolis Cedex, France. E-mail: emmanuel.maggiori@inria.fr.

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

Manuscript received ...; revised ...

need for finding *optimal* architectures for semantic labeling, i.e., the "recipe" for suitable semantic labeling networks. We consider that just by doing a proper analysis of the architecture required for our problem we may develop smaller, more efficient networks to achieve equivalent or even better results.

Our first contribution is a detailed analysis of the main families of CNN architectures proposed recently for the semantic labeling problem. We group the different methods into three categories: *dilation* (e.g., [4], [5]), *deconvolution* (e.g., [6], [7], [8], [3]) and *skip* (e.g., [9], [10]) networks. These categories are different from each other in the way of addressing the aforementioned recognition/localization trade-off. For example, while the networks by Long et al. [9] and Marmanisa et al. [10] are substantially different in structure and application domain, they are both *skip* networks in how they manage to provide a high-resolution output. After establishing the desired properties of a semantic labeling architecture, we position the different families of networks with respect to these properties. Let us remark that it is also common to include post-processing modules to increase the resolution of CNN's outputs, such as fully connected CRFs [4], [11], [12]. However, our review focuses on architectures that are specifically designed to provide a high-resolution output.

Our second contribution is a novel semantic labeling network architecture, referred to as MLP (after *multi-layer perceptron*). Derived from the notion of *skip* network, the MLP architecture yields high flexibility and expressiveness by extracting features at different resolutions (and thus at different levels of details), and *learning* how to combine them in order to generate fine-grained classification maps. In the literature, probably the most similar approach is the one in [13] which, though for a different problem, also seeks to learn to combine multi-resolution features. Our MLP architecture exhibits a better performance in aerial image labeling than many other recent techniques, despite being simpler and smaller than them. The design of an appropriate architecture thus leads to a win-win situation, in which both accuracy and computational complexity are improved.

We conduct experiments on two popular benchmarks for high-resolution aerial segmentation: Vaihingen and Potsdam datasets, proposed as part of the ISPRS Semantic Labeling Contest [14]. These datasets highlight the specific challenges of aerial image labeling, requiring to outline small objects with a high spatial precision.

This paper first introduces convolutional neural networks and their use in semantic labeling (Sec. II). An analysis of the different high-resolution labeling schemes is then presented (Sec. III). We later describe our proposed architecture (Sec. IV) and perform an experimental evaluation (Sec. V). Conclusions are drawn in Sec. VI.

II. Convolutional Neural Networks

An artificial neural network is a system of interconnected neurons that pass messages to each other. When the messages are passed from one neuron to the next one without ever going back (i.e., the graph of message passing is acyclic) they network is referred to as feed-forward [15], which is the most common type of network in image categorization. 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:

$$a = \sigma(\mathbf{wx} + b), \quad (1)$$

where $\mathbf{w}$ denotes a weight vector, b a scalar known as *bias* and σ an activation function. Put simply, a neuron computes a weighted sum of its inputs and applies a possibly nonlinear scalar function on the result. The weights $\mathbf{w}$ and biases b are the parameters of the neurons that define the function. The goal of training is to find the optimal values for these parameters, so that the function computed by the neural network performs the best on the task assigned.

The most common activation functions σ are sigmoids, hyperbolic tangents and rectified linear units (ReLU). For image analysis, ReLUs have become the most popular choice due to some practical advantages at training time, but novel activation functions have been recently proposed as well [16].

Despite an apparent simplicity, neural networks are extremely expressive: by using at least one layer of nonlinear activation functions, a sufficiently large network can represent *any* function within a given bounded error [15].

Instead of directly connecting a huge set of neurons to the input, it is common to organize them in groups of stacked layers that transform the outputs of the previous layer and feed it to the next layer. This enforces the networks to learn hierarchical features, performing low-level reasoning in the first layers (such as edge detection) and higher-level tasks in the last layers (e.g. , assembling object parts). 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 the network is an image (or a set of features derived from an image), and the goal is to predict the correct category of the entire image. We can view the pixelwise semantic labeling problem as taking an image patch and categorizing its central pixel. Finding the optimal neural network classifier reduces to finding the weights and biases that minimize a loss L between the predicted labels and the target labels in a training set. Let $\mathcal{L}$ be the set of possible semantic classes; 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 contains thus 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 label values are between zero and one and sum to one. The multi-label 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 work we use the common cross-entropy loss, defined as:

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

Training neural networks by optimizing this criterion converges faster (compared with, for instance, the Euclidean distance between $\mathbf{y}$ and $\hat{\mathbf{y}}$). In addition, it is numerically stable when coupled with softmax normalization [15].

Once the loss function is defined, the parameters (weights and biases) that minimize the loss are found via gradient descent, by computing the derivative $\frac{\partial L}{\partial w_i}$ 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 $\frac{\partial L}{\partial w_i}$ 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 derivatives of each layer's outputs with respect to its weights and inputs (i.e. the lower layer's outputs). In practice, instead of averaging over the full dataset, the loss (2) is estimated from a random small subset of the training set, referred to as a mini-batch. This learning technique is named *stochastic gradient descent*.

A. Convolutional Layers

Convolutional neural networks (CNNs) [17] contain so-called convolutional layers, a specific type of layer that imposes a number of restrictions compared to a more general fully-connected layer (discussed below). These restrictions (e.g., local connectivity) have been introduced for image categorization networks because they make sense in that particular context.

In CNNs, each neuron is associated to a spatial location (i, j) in the input image. The output a_{ij} associated with location (i, j) in a convolutional layer is computed as:

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

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

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

Multiple convolution kernels are usually learned in every layer, interpreted as a set of spatial feature detectors. The responses to every learned filter are thus referred to as *feature maps*. Note that the convolution kernels are actually three-dimensional: in addition to their spatial extent (2D), they span along all the feature maps in the previous layer (or eventually through all the bands in the input image). As this third dimension can be inferred from the previous layer it is rarely mentioned in the architecture descriptions.

Compared to the fully connected layer, in which every neuron is connected to all outputs of the previous layer, a convolutional layer highly reduces the number of parameters by enforcing the aforementioned constraints. This results in an easier optimization problem, without losing much generality. This opened the door to using the image itself as an input without any feature design and selection process, as CNNs discover the relevant spatial features to conduct classification.

B. Increasing the Receptive Field

In CNNs, the *receptive field* denotes the spatial extent of the input image connected to a certain neuron, possibly indirectly through other neurons in previous layers: it is the set of pixels on which this neuron depends. In other words, it quantifies how far a neuron can "see" in the image. In most applications, a large amount of spatial context must be taken into account in order to successfully label the images. For example, to deduce that a certain pixel belongs to a rooftop, it might not be enough to just consider its individual spectrum: we might need to observe a large patch around this pixel, taking into account geometry and structure of the objects, to infer its correct class.

Neural networks for image analysis should thus be designed to accumulate, through their layers, a large enough receptive field. While a straightforward way to do it is to use large convolution kernels, this is not a common practice mostly due to its computational complexity. Besides, this would aim at learning large filters all at once, with millions of parameters. However, it is preferable to learn a hierarchy of small filters instead, reducing the number of parameters while remaining expressive, and thus making the optimization problem easier.

The most common approach to reduce the number of parameters for a given receptive field size is to downsample the feature maps throughout the network. This is commonly achieved progressively through interleaving downsampling layers with convolutional layers. This way, the resolution of the feature maps gets lower and lower as we traverse the layers from input to output. For example, neurons after a chain of two 3×3 convolutions in successive layers would normally have a receptive field of 5×5 pixels, which extends to 12×12 pixels with an accumulated downsampling of factor 4.

To downsample the feature maps, the most popular approach is to use the so-called *max pooling* layer [18]. A max pooling layer takes a group of neighbors in the feature map and condenses them into a single output by computing the maximum of all incoming activations in the window. The pooling windows in general do not overlap, hence the output map is downsampled (see Fig. 1). For instance, if pooling is performed in a 2×2 window, the feature map is reduced to half of its resolution.

Computing the maximum value is inspired by the idea of detecting objects from their parts. For example, in a face detector it is important to identify the constituents of a face, such as *hair* or *nose*, while the exact locations of these components should not be such a determinant factor. The max pooling layer conveys then to which extent there is evidence of the *existence* of a feature in a vicinity. Other less popular forms of downsampling include average pooling and applying convolutions with a *stride*, i.e., "skipping" some of them (e.g., applying every other convolution).

Pooling operations (and downsampling in general) hard-code robustness to spatial deformations, an attribute that boosted the success of CNNs for image categorization. However, spatial precision is lost when downsampling. The increased receptive field (and thus recognition capability) comes at the price of losing localization capability. This well-reported trade-off [9], [11] is a major concern for dense labeling.

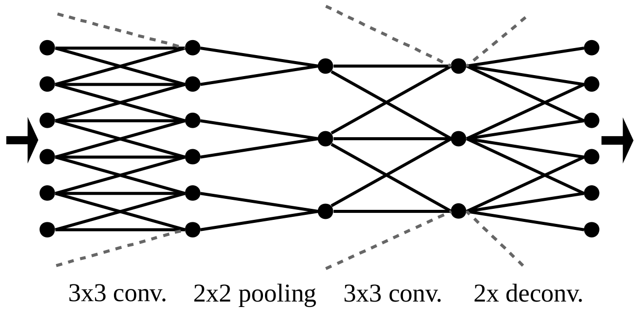

Fig. 1: Lateral view of a fully convolutional network (dashed lines indicate inputs that have been padded in conv. layers and cropped in the deconv. layer to preserve spatial dimensions).

We could still imagine a downsampling network that preserves localization: it would learn features of the type "a corner at the center of the receptive field", "a corner one pixel left of the center of the receptive field", "a corner two pixels left of the center of the receptive field", and so on, multiplying the number of features to be learned. This would however discredit the use of downsampling to gain robustness to spatial variation in the first place. The recognition/localization tradeoff must thus be properly addressed to design a high-resolution semantic labeling network.

C. Fully Convolutional Networks (FCNs)

Image categorization networks are typically written as a series of interleaved convolution and pooling layers that extract spatial features, followed by a few fully connected layers that compute the final classification values. The dense semantic labeling problem can be seen as classifying the central pixel of an image patch, the size of the input patch being the receptive field used to classify it. To label the whole image the prediction must thus be performed on many overlapping image patches, which requires a huge amount of redundant operations.

Fully convolutional networks (FCNs) [9] are especially relevant to semantic labeling. They contain only convolutional layers, i.e., no fully connected layers. Therefore, they can be applied to images with various sizes: inputting a larger image patch produces a larger output, the convolutions being performed on more locations. In contrast, networks with any fully connected layer require a fixed image size, because of the fixed input size of such layers. Using fully convolutional networks also removes any redundancy when computing classification maps on large inputs, as they are applied only once.

The first obvious advantage of FCNs is a reduced computational complexity. Moreover, we can efficiently train on input patches that are larger than the receptive fields, and in turn produce larger classified patches, with more than a single pixel. While the elements inside a contiguous patch are highly correlated, the use of moderately larger patch sizes has been reported to be beneficial [4], [19]. Furthermore, the patch size at training time is decoupled from the one at test time. For example, we could use use small patches to train the network in order to have a highly variable input in every mini-batch, but later conduct predictions on the largest patch size that fits in the GPU. Let us finally remark that a traditional classification network (with fully connected layers) can be in fact easily rewritten as a fully convolutional network [9].

When an FCN has downsampling layers, the output contains fewer elements than the input, since the resolution has been decreased. This gave birth to the so-called deconvolutional (or upconvolutional) layer, which upsamples a feature map by interpolating neighboring elements (as the last layer in Fig. 1). Instead of determining a priori the type of interpolation, e.g., bilinear, the operation is parametrized by a kernel that can also be learned. Deconvolutional layers are typically used to perform a naive interpolation at the very end of the pipeline, on the output classification maps. In the next section we study more advanced ways of providing high-resolution outputs.

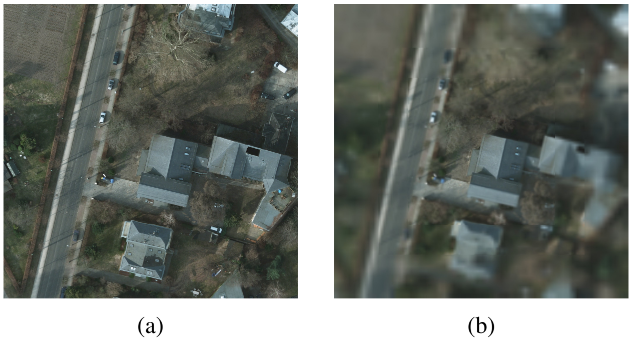

Fig. 2: To classify the central gray pixel of this patch (and not to confuse it, e.g., with an asphalt road), we need to take into account a spatial context (a). However, we do not need a high resolution everywhere in the patch. It can be lower as we go away from the central pixel and still identify the class (b).

III. Analysis of High-Resolution Labeling CNNs

Fully convolutional networks (FCNs), as described in Section II-C, have become the standard in semantic labeling. Nevertheless, the open question is how to conduct fine predictions that provide detailed high-resolution outputs, while still taking large amounts of context into account and without exploding the number of trainable parameters. Simply adding a deconvolutional layer to upsample the output on top of a network provides dense outputs but imprecise labeling results, because the upsampling is performed in a naive way from the coarse classification. This is dissatisfying in many applications, such as high-resolution aerial image labeling, where the goal is to precisely identify and outline tiny objects such as cars.

We now describe what we consider to be the elementary principle from which to derive efficient semantic labeling architectures. Let us then first observe that while our goal is to take large amounts of context into account, we do not need this context at the same spatial resolution everywhere. For example, let us suppose we want to classify the central pixel of the patch in Fig. 2a. Such a gray pixel, taken out of context, could be easily confused with an asphalt road. Considering the whole patch at once helps to infer that the pixel belongs indeed to a gray rooftop. However, two significant issues arise if we take a full-resolution large patch for context: a) it requires many computational resources that are actually not needed for an effective labeling, and b) it does not provide robustness

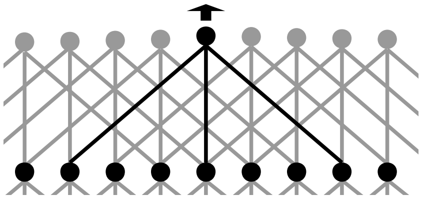

Fig. 3: A dilated convolution (i.e., on non-adjacent inputs) with a dilation factor $S = 4$.

to spatial variation (we might actually not care about the exact location of certain features). Conducting predictions from low-resolution patches instead is not a solution as it produces inaccurate coarse classification maps. Nevertheless, it is actually not necessary to observe all surrounding pixels at full resolution: the farther we go from the pixel we want to label, the lower the requirement to know the exact location of the objects. For example, in the patch of Fig. 2b it is still possible to classify the central pixel, despite the outer pixels being blurry. Therefore, we argue that a combination of reasoning at different resolutions is necessary to conduct fine labeling, if we wish to take a large context into account in an efficient manner.

In the following, we analyze the main families of high-resolution semantic labeling networks that have been proposed in the past two years. For each of them we discuss the following aspects:

- How a solution to the fine-grained labeling problem is provided;
- Where this solution stands with respect to the principle of Fig. 2;
- General advantages and disadvantages, and computational efficiency.

A. Dilation Networks

Dilation networks are based on the shift-and-stitch approach or *à trous* algorithm [9]. This consists in conducting a prediction at different offsets to produce multiple low-resolution outputs, which are then interleaved to compose the final high-resolution result. For example, if the downsampling factor of a network is S, one should obtain S^2 classification maps by shifting the input horizontally and vertically. Such an interleaving can also be implemented directly in the architecture, by using "dilated" operations [20], i.e., performing them on non-contiguous elements of the previous feature maps. This principle is illustrated in Fig. 3.

Dilations have been used with two purposes:

1) as an alternative to upsampling for generating full-resolution outputs [5], [9],
2) as a means to increase the receptive field [11], [20], by enlarging the area covered by a convolution kernel without increasing the number of trainable parameters.

Regarding the first point, we must mention that there is no theoretical improvement compared to an FCN with naive upsampling, because the presence of pooling layers still reduces spatial precision. Executing the prediction multiple times at small offsets still keeps predictions spatially imprecise.

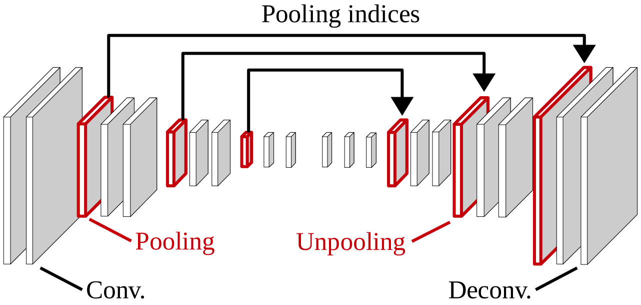

Fig. 4: Deconvolution network. The CNN is "mirrored" to learn the deconvolution.

Regarding the second point, we must remark that while dilated convolutions increase the receptive field, this does not introduce robustness to spatial variation per se. For example, a network with only dilated convolution layers would have a large receptive field but would only be able to learn filters of the type "a building in the center, with a car *exactly* five pixels to the left". This robustness would have to be thus learned, hopefully, by using a larger number of filters.

The use of an interleaved architecture at training time, implemented with dilations, has been however reported to be beneficial. In the context of aerial image labeling, Sherrah [4] recently showed that it outperformed its FCN/upsampling counterpart[1]. The major improvement compared to the FCN/upsampling network was measured in the labeling capabilities of the *car* class, which is a minority class with tiny objects, difficult to recognize [3]. While the dilation strategy is not substantially different from an architectural point of view compared to naive upsampling, some advantages in training might explain the better results: In the upsampling case the network is encouraged to provide a coarse classification that, once upsampled, is close to the ground truth. In the dilation network, on the contrary, the interleaved outputs are directly compared to individual pixels in the ground truth, one by one. The latter seems to better avoid suboptimal solutions that absorb minority classes or tiny objects.

The computational time and memory required by dilation networks are significant, to the point that using GPUs might become impractical even with moderately large architectures. This is because the whole network rationale is applied at many contiguous locations.

Overall, while dilation networks have been reported to exhibit certain advantages, they are computationally demanding and do not particularly address the principle of Fig. 2.

B. Deconvolution Networks (unpooling)

Instead of naively upsampling the classification score maps with one deconvolutional layer, a more advanced approach is to attach a multi-layer network to learn a complex upsampling function. This idea was simultaneously presented by different research groups [6], [7] and later extended to different problems (e.g., [21]). The most hassle-free way to do this is

[1] While such architecture is named a "no-downsampling" network in [4], a more appropriate name would probably be "no-upsampling", because there is indeed downsampling due to the max pooling layers.

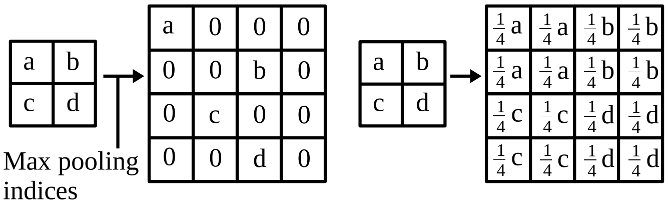
a b
c d
Max pooling
indices
a 0 0 0
0 0 b 0
0 c 0 0
0 0 d 0
a b
c d
$\frac{1}{4}$a $\frac{1}{4}$a $\frac{1}{4}$b $\frac{1}{4}$b
$\frac{1}{4}$a $\frac{1}{4}$a $\frac{1}{4}$b $\frac{1}{4}$b
$\frac{1}{4}$c $\frac{1}{4}$c $\frac{1}{4}$d $\frac{1}{4}$d
$\frac{1}{4}$c $\frac{1}{4}$c $\frac{1}{4}$d $\frac{1}{4}$d

Fig. 5: Max (left) and average (right) unpooling.

to simply "reflect" an existent FCN, with the same number of layers and kernel sizes, to perform the upsampling. The convolutional layers are reflected as deconvolutional layers, and the pooling layers as *unpooling* layers (see Fig. 4). While pooling condenses several activations into one representative value (typically, the maximum activation), unpooling layers must reconstruct the original size of activations. In the case of max unpooling, the location of the maximal activation is recalled from the corresponding pooling layer, and is used to place the activation back into its original pooled location. The other elements in the unpooling window are set to zero, leading to sparse feature maps, as illustrated in Fig. 5. Unpooling was first introduced as part of a framework to analyze and visualize CNN features [2]. The arrows in Fig. 4 represent the communication of the pooling indices from the pooling layer to the unpooling layer. In the case of average pooling, the corresponding unpooling layer simply outputs at every location the input activation value divided by the number of elements in the target unpooling window(see Fig. 5). In this case, there is no need to transmit a location from pooling to unpooling.

This concept can be thought of as an "encoder–decoder", where the middle layer is seen as a common representation to images and classification maps, while the "encoder" and "decoder" ensure the translation between this representation and the two modalities. When converting an FCN to a deconvolution network, the final classification layer of the FCN is usually dropped before reflecting the architecture. This way the interface between the encoder and the decoder is a rich representation with multiple features. The first layer of the encoder takes as input as many channels there are in the input image, and usually the last layer of the decoder produces as many feature maps as classes required. In [7], [22], alternatively, the network outputs a larger set of features that are then classified with additional layers.

While pooling is used to add robustness to spatial deformation, the fact of "remembering" the location of the max activation helps to precisely locate objects in the deconvolution steps. For example, the exact location of a road might be irrelevant to do any higher-level reasoning later on, but once the network decides to label the road as a semantic object we need to recover the location information to outline it with high precision. This illustrates how deconvolution networks balance the localization/recognition trade-off.

Note however that if one happens *not* to choose max pooling for downsampling, then the unpooling scheme is not able to recover *per se* the lost spatial resolution. There is no memory about the location of the higher resolution feature. Even though max pooling is very common, it has been shown that average or other types of pooling might be more effective in certain applications [18]. In fact, recent results [23] suggested that max pooling can be emulated with a strided convolution and achieve similar performance. The deconvolution network idea is however leveraged when max pooling is the downsampling mechanism used.

This certainly does not mean that a deconvolutional network is incapable of learning without max pooling layers. Convolution/deconvolution architectures without max pooling have been successfully used in different domains [3], [24]. For example, a recent submission to the ISPRS Semantic Labeling Challenge [3] is such type of network. The recognition/localization trade-off is not really alleviated in this case: the encoder should encode features of the type "an object boundary 5 (or 7, 10...) pixels away to the left", so that the decoder can really leverage that information and reconstruct a high-resolution classification map.

The depth of deconvolution networks is significantly larger, roughly twice the one of the associated FCN. This often implies a slower and more difficult optimization, due to the increase in trainable parameters introduced by deconvolutional layers. While the decoding part of the network can be simplified [8], this adds arbitrariness to the design.

To conclude, the deconvolution scheme does address the recognition/localization trade-off, but only in the case where max pooling is used for downsampling. The increased network depth can be a concern for an effective training.

C. Skip Networks

In the original paper about fully convolutional networks, Long et al. [9] proposed the so-called "skip" architecture to generate high-resolution classification outputs. The idea is to build the final classification map by combining multiple classification maps, obtained from intermediate features of the network at different resolutions (and not just the last one).

The last layer of an FCN outputs as many feature maps as classes, which are interpreted as score or "heat" maps for every class. Intermediate layers, however, tend to have many more features than the number of classes. Therefore, skip networks add extra layers that convert the arbitrarily large number of features of intermediate layers into the desired number of heat maps. This approach allows us to extract multiple score maps for each class from a single network, at different resolutions. The lower-level score maps are fine but have a small receptive field, while the higher-level ones can see farther but with less detail. As a result, we have a pool of score maps.

The score maps are then combined pairwise, from the lower scales to the higher scales. At every step, the lower-resolution score maps are upsampled to match the higher-resolution ones. They are then added elementwise. This is repeated until all intermediate maps are processed. The overall combination of resolutions forms a directed acyclic graph, with links that "skip" ahead from lower layers to higher ones. A skip network is illustrated in Fig. 6.

Skip networks address the trade-off between localization and precision quite explicitly: the information at different resolutions is extracted and combined. The original paper

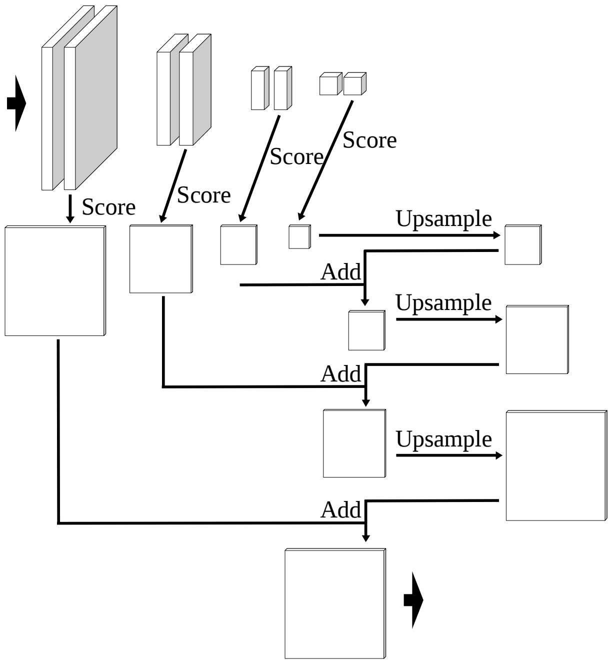

Fig. 6: Skip network: multiple classification scores are obtained from intermediate CNN features at different resolutions, and are combined by element-wise adding and upsampling.

introduces this methodology as "combining what and where". This approach is closer to the principle described in Fig. 2 than the previous approaches reviewed above. The skip network mixes observations at different resolutions, without unnecessarily increasing the depth or width of the architecture (as in deconvolution and dilation networks respectively) and it does not impose a particular type of downsampling (as in deconvolution networks).

While the idea of extracting different resolutions is certainly very relevant, the skip model seems to be inflexible and arbitrary in how to combine them. First of all, it combines classification verdicts, instead of a rich set of features, coming from each of the resolutions. For example, it combines how a layer evaluates that an object is a building by using low-level information, with how another layer evaluates whether the same object is a building by using higher-level information. Let us recall that we use deep multi-layer schemes with downsampling because we actually consider that certain objects can only be detected at the upper layers of the network, when a large amount of context has been taken into account and at a high level of abstraction. It seems thus contradictory to try to refine the boundaries of an object detected at a high level, by using a classification conducted at a lower level, where the object might not be detected at all. Moreover, the element-wise addition restricts the combination of resolutions to be simply a linear combination. The skip links to combine resolutions are in fact parameterless (besides the addition of the scoring layers). We could certainly imagine classes that require a more complex nonlinear combination of high- and low-level information to be effectively classified.

It is worth noting that the creation of the intermediate score maps has also been referred to as a dimensionality reduction step [22]. It is however not by chance that the amount of reduced features coincides with the amount of classes: even though it is technically a dimensionality reduction, its spirit is to create a partial classification, not just to reduce the number of features. This is confirmed by the name of these layers in the public implementation of [9] : "score" layers. Moreover, if this operation were indeed intended to be just a reduction of dimensionality, we could imagine outputting different amounts of feature maps from different resolutions. However, in that case there would be no way of adding them element by element as suggested.

To conclude, the skip network architecture provides an efficient solution to address the localization/recognition trade-off, yet this could be done in a more flexible way that allows a more complex combination of the features.

IV. Learning to Combine Resolutions

In this section we propose an alternative scheme for high-resolution labeling, derived as a natural consequence of our observations about the other families of methods. In particular, this architecture leverages the benefits of the skip network described in Section III-C while addressing its potential limitations.

Taking multiple intermediate features at different resolutions and combining them seems to be a sensible approach to specifically address the localization/recognition trade-off, as done with skip networks. In such a scheme, the high-resolution features have a small receptive field, while the low-resolution ones have a wider receptive field. Combining them constitutes indeed an efficient use of resources, since we do not actually *need* the high-resolution filters to have a wide receptive field, following the principle of Fig. 2.

The skip network combines *predictions* derived from the different resolutions, i.e., score maps for each of the classes. For example, we try to refine the "blobby" building outputted by the coarse classifier, via a higher-resolution classification. However, it is unclear how effectively the higher-resolution classifier detects the building in question, considering its reduced receptive field and shallow reasoning.

We thus argue that the most appropriate way of performing fine semantic labeling is to combine features, not classification maps. For example, to refine the boundaries of a coarse building, we would use high-resolution edge detectors (and not high-resolution building detectors).

In our proposed scheme, intermediate features are extracted from the network and treated equally, creating a pool of features that emanate from different resolutions. A neural network then learns how to combine these features to give the final classification verdict. This adds flexibility to learn more complex relations between the different resolutions and generalizes the element-wise addition of the skip architecture.

The overall process is depicted in Fig. 7. First, a subset of intermediate features are extracted from the network. These are naively upsampled to match the resolution of the higher-resolution features. They are then concatenated to create the pool of features. Notice that while the spatial dimensions of the feature maps are all the same, they originally come from

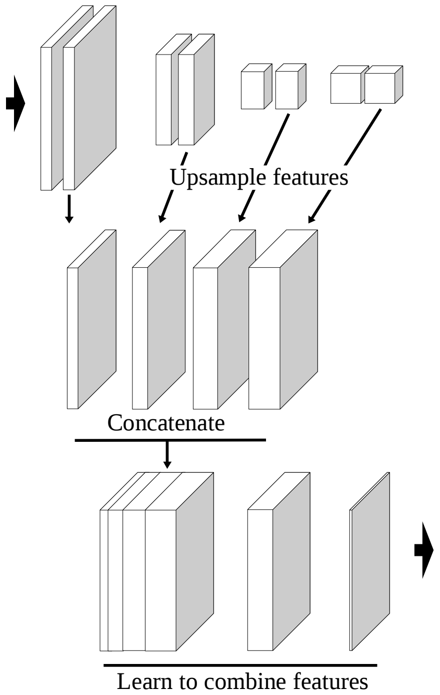

Fig. 7: MLP network: intermediate CNN features are concatenated, to create a pool of features. Another network learns how to combine them to produce the final classification.

different resolutions. This way, the variation of the feature responses across space will be smoother in certain maps while sharper in others. Note that while it is practical to store in memory the upsampled responses, this is not intrinsically necessary. For example, we could imagine a system that answers to a high-resolution query by outputting the nearest neighbor in the coarser map or by interpolating neighboring values on the fly.

From the pool of features, a neural network predicts the final classification map (we could certainly use other classifiers, but this lets us train the system end to end). We assume that all the spatial reasoning has been conveyed in the features computed by the initial CNN. This is why we operate on a pixel-by-pixel basis to combine the features. Any need to look at neighbors should be expressed in the spatial filters of the CNN. This way we conceptually and architecturally separate the extraction of spatial features from their combination.

We can think of the multi-layer perceptron (MLP) with one hidden layer and a non-linear activation function as a minimal system to learn how to combine the pool of features. Such MLPs can learn to approximate any function and, since we do not have any particular constraints, it seems an appropriate choice. In practice, this is implemented as a succession of convolutional layers with 1×1 kernels, since we want the same MLP to be applied at every location. By introducing the MLP and executing it at a fine resolution, we must expect an overhead in processing time compared to the skip network.

The proposed technique is intended to learn how to combine information at different resolutions, not how to upsample a low-resolution classification. An example of the type of relation we are able to convey in this scheme is as follows: "label a pixel as *building* if it is red and belongs to a larger red rectangular structure, which is surrounded by areas of green vegetation and near a road".

Finally, let us discuss the CNN from which features are extracted (the topmost part of Fig. 7). The different features are extracted from intermediate layers of a single CNN. This assumes that the higher-level features can be derived from the lower-level ones. It is basically a part-based model [25], where we consider that an object can be detected by its parts, and we are using those same parts as the higher-resolution features inputted to the MLP. This seems to be a sensible assumption, yet we must mention that we could eventually think of separate networks to detect features at different resolutions instead of extracting intermediate representations of a single network (as, e.g., in [26]). While we adopt the model of Fig. 7 in this work, the alternative could be also considered. It would be certainly interesting to study to which extent it is redundant to learn the features in separate networks and, conversely, how results could be eventually improved by doing it.

V. Experiments

A. Datasets and Evaluation Metrics

We evaluate the aforementioned architectures on two benchmarks of aerial image labeling: Vaihingen and Potsdam, provided by Commission III of the ISPRS [14]. The Vaihingen dataset is composed of 33 image tiles (of average size 2494×2064), out of which 16 are fully annotated with class labels. The spatial resolution is 9 cm. Near infrared (NIR), red (R) and green (G) bands are provided, as well as a digital surface model (DSM), normalized and distributed by [27]. We select 5 images for validation (IDs: 11, 15, 28, 30, 34) and the remaining 11 images for training, following [4], [3], [12].

Potsdam dataset consists of 38 tiles of size 6000×6000 at a spatial resolution of 5 cm, out of which 24 are annotated. It provides an additional blue channel and the normalized DSM. We select the same 7 validation tiles as in [4] (IDs: 2_11, 2_12, 4_10, 5_11, 6_7, 7_8 7_10) and the remaining 17 tiles for training. Both datasets are labeled into the following six classes: *impervious surface*, *building*, *low vegetation*, *tree*, *car* and *clutter/background*.

In order to account for labeling mistakes, another version of the ground truth with eroded boundaries is provided, on which accuracy is measured. To evaluate the overall performance, overall accuracy is used, i.e., the percentage of correctly classified pixels. To evaluate class-specific performance, the F1-score is used, computed as the harmonic mean between precision and recall [28]. We also include the mean F1 measure among classes, since overall accuracy tends to be less sensitive to minority classes in imbalanced datasets.

B. Network Architectures

To conduct our experiments we depart from a base fully convolutional network (FCN) and derive other architectures from it. Table I summarizes our base FCN for the Vaihingen dataset. The architecture is borrowed from [19], except for the fact that we increased the size of the filters from 3 to 5 in the first layer, since it is a common practice to use larger filters

TABLE I: Architecture of our base FCN.

Layer	Filter size	Number of filters	Stride	Padding
Conv-1_1	5	32	2	2
Conv-1_2	3	32	1	1
Pool-1	2		2	
Conv-2_1	3	64	1	1
Conv-2_2	3	64	1	1
Pool-2	2		2	
Conv-3_1	3	96	1	1
Conv-3_2	3	96	1	1
Pool-3	2		2	
Conv-4_1	3	128	1	1
Conv-4_2	3	128	1	1
Pool_4	2		2	
Conv-Score	1	5	1	

if there is a stride. Every convolutional layer (except the last one) is followed by a batch normalization layer [29] and a ReLU activation. We did not optimize the architecture of the base FCN.

The total downsampling factor is 16, out of which 8 is the result of the max pooling layers and 2 of the stride in the first layer. The conversion of the last set of features to classification maps (the "score" layer) is performed by a 1×1 convolution. To produce a dense pixel labeling we must add a deconvolutional layer to upsample the predictions by a factor of 16, thus bringing them back to the original resolution.

To implement a *skip* network, we extract the features of layers *Conv-*_2*, i.e., produced by the last convolution in each resolution and before max pooling. Additional scoring layers are added to produce classification maps from the intermediate features. The resulting score maps are then combined as explained in Section III-C. Our *MLP* network was implemented by extracting the same set of features. As no intermediate scores are needed, we remove layer 'Conv-Score' from the base FCN. The features are combined as explained in Section IV. The added multi-layer perceptron contains one hidden layer with 1024 neurons.

We also created a deconvolution network that exactly reflects the base FCN (as in [6]). This is straightforward, with deconvolutional and unpooling layers associated to every convolutional and pooling layer. The only particularity is that the last layer outputs as many maps as required classes and not as input channels. We here call it *unpooling* network, to differentiate it in the experiments from another method that uses a stack of deconvolutions but without unpooling [3], which we simply refer to as *deconvolution* network. To cover the last family of architectures of Sec. III, the *dilation* network, we incorporate the results recently presented by Sherrah [4].

In both datasets we use the same four input channels: DSM, NIR, R and G. Notice that we simply add the DSM as an extra band. While for Potsdam dataset the blue channel is also available, we here excluded it for simplicity. In the case of Vaihingen we predict five classes, ignoring the clutter class, due to the lack of training data for that class. In the case of Potsdam we predict all six classes.

Considering the difference in resolution in both datasets, in the case of Potsdam we downsample the input and linearly upsample the output by a factor of 2 (following [4]). We use the same architecture as for Vaihingen (besides the different number of output classes) between the downsampling and upsampling layers. This is intended to cover similar receptive field in terms of meters (and not pixels) for both datasets.

C. Training

The networks are trained by stochastic gradient descent [15]. In every iteration a group of patches is fed to the network for backpropagation. We sample random patches from the images, performing random flips (vertically, horizontally or both) and transpositions, augmenting the data 8 times. At every iteration we group five patches in the mini-batch, of size 256×256 for Vaihingen dataset and 512×512 for Potsdam (to roughly cover the same geographical area, considering the difference in resolution). In all cases, gradient descent is run with a momentum of 0.9, and an L2 penalty on the network's parameters of 0.0005. Weights are initialized following [30] and, since we use batch normalization layers before ReLUs, there is no need to normalize the input channels.

We start from a base learning rate of 0.1 and anneal it with an exponential decay. The decay rate is set so that the learning rate is divided by ten every 10,000 iterations in the case of Vaihingen and every 20,000 iterations in Potsdam. We decrease the learning rate more slowly in the case of Potsdam because the total surface covered by the dataset is larger, thus we assume it must take longer to explore. Training is stopped after 45,000 iterations in the first dataset and 90,000 in the second one, when the error stagnates on the validation set.

To train the unpooling, skip and MLP networks we initialize the weights with the pretrained base FCN, and jointly retrain the entire architecture. We start this second training phase with a learning rate of 0.01, and stop after 30,000 and 65,000 iterations for Vaihingen and Potsdam datasets respectively. We verified that the initialization with the pretrained weights is indeed beneficial compared to training from scratch.

D. Numerical Results

In this section we first present how our base FCN network compares to its derived architectures: unpooling, skip and MLP. We then position MLP with respect to other results reported in the literature, including a dilation network, thus completing the evaluation over all four families of techniques. We finally discuss our submission to the ISPRS contest.

a) Comparison of a base FCN to its derived unpooling, skip and MLP networks: The classification performances on the validation sets are included in Tables II and III, for Vaihingen and Potsdam datasets, respectively. The MLP network exhibits the best performance in almost every case. The skip network effectively enhances the results compared with the base network, yet it does not outperform MLP. Let us remark that the unpooling strategy does not necessarily constitute an improvement to the base FCN. This might be a result of the increased training difficulty due to the depth of the network and the sparsity of the unpooled maps. We tried to modify the training scheme, yet we could not improve its performance.

Overall, the numerical results show that the injection of lower-resolution features significantly improves the classification accuracy. MLP is the most competitive method, boosting the performance by learning how to combine these features.

TABLE II: Numerical evaluation of architectures derived from our base FCN on the Vaihingen validation set.

	Imp. surf.	Building	Low veg.	Tree	Car	Mean F1	Overall acc.
Base FCN	91.46	94.88	79.19	87.89	72.25	85.14	88.61
Unpooling	91.17	95.16	79.06	87.78	69.49	84.54	88.55
Skip	91.66	95.02	79.13	88.11	77.96	86.38	88.80
MLP	**91.69**	**95.24**	**79.44**	**88.12**	**78.42**	**86.58**	**88.92**

TABLE III: Numerical evaluation of architectures derived from our base FCN on the Potsdam validation set.

	Imp. surf.	Building	Low veg.	Tree	Car	Clutter	Mean F1	Overall acc.
Base FCN	88.33	93.97	84.11	80.30	86.13	75.35	84.70	86.20
Unpooling	87.00	92.86	82.93	78.04	84.85	72.47	83.03	84.67
Skip	89.27	94.21	84.73	**81.23**	93.47	75.18	86.35	86.89
MLP	**89.31**	**94.37**	**84.83**	81.10	**93.56**	**76.54**	**86.62**	**87.02**

b) Comparison with other methods: Tables IV and V (for Vaihingen and Potsdam datasets respectively) incorporate the numerical results reported by other authors using the same training and validation sets. Since not every method was applied to both datasets, the tables do not display exactly the same techniques. The MLP approach also outperforms the dilation strategy, in both datasets, thus positioning it as the most competitive category among those presented in Sections III, IV (dilation, unpooling, skip, MLP).

In the case of Vaihingen dataset, we also report the results of the *deconvolution* network [3], commented in Sec. III-B, which performs upsampling by using a series of deconvolutional layers. Contrary to the *unpooling* network, the decoder does not exactly reflect the encoder and no unpooling operations are used. Additionally, we include the performance of other methods recently presented in the literature: the CNN+RF approach [12], which combines a CNN with a random forest classifier; the CNN+RF+CRF approch, which adds CRF post-processing to CNN+RF; and Dilation+CRF [4], which adds CRF post-processing to the dilation network. As depicted in the table, the MLP approach outperforms these other methods too.

For Potsdam dataset, Table V reports the performance of two other methods, presented in [4]. In both cases, a pretrained network based on VGG [31] is applied to the IR-R-G channels of the image, and another FCN is applied to the DSM, resulting in a huge hybrid architecture. An ordinary version (with upsampling at the end) and a *dilation* version are considered (‘VGG pretr.’ and ‘VGG+Dilation’ in Table V, respectively). In the latter version, the dilation strategy could only be applied partially as it is too memory intensive. While MLP outperforms the non-pretrained simpler dilation network, the *VGG+Dilation* variants exhibits the best overall performance (though not on all of the individual classes). This suggests that the VGG component might be adding a competitive edge, though the authors stated that this is not the case on the Vaihingen dataset.

Overall, MLP provides better accuracies than most techniques presented in the literature, including dilation networks, ensemble approaches and CRF post-processing.

c) Submission to the ISPRS challenge: We submitted the result of executing MLP on the Vaihingen test set to the ISPRS server (ID: ‘INR’), which can be accessed online [14]. Our method scored second out of 29 methods, with an overall accuracy of 89.5. Note that the MLP technique described in this paper is very simple compared to other methods in the leaderboard, yet it scored better than them. For example, an ensemble of two *skip* CNNs was pretrained on large natural image databases [10], with over 20 convolutional layers and separate paths for the image and the DSM. Despite being simpler, our MLP network outperforms it in the benchmark.

E. Visual Results

We include visual comparisons on closeups of classified images of both datasets in Fig. 8. As expected, the base FCN tends to output “blobby” objects, while the other methods provide sharper results. This is particularly noticeable for the cars of Rows 2, 5 and 6, and for the thin road at the lower left corner of Row 4. We also observe that the incorporation of reasoning at lower resolutions allows the derived networks to discover small objects that are otherwise lost. This is particularly noticeable in the 4th row, where there is a set of small round/cross-shaped objects of the *clutter* class (in red) that are omitted or grouped together by the base FCN.

The unpooling technique seems to be prone to outputting artifacts. These are often very small in size, even isolated pixels. This is well observed for example for the car of Row 3. This effect could be a natural consequence of the max unpooling mechanism, as depicted in Fig. 5, which upsamples into sparse matrices and delegates the task of reconstructing a smoother output to the deconvolutional layers.

At first sight it is more challenging to visually assess why MLP beats the skip network in almost every case in the numerical evaluation. Taking a closer look we can however observe that boundaries tend to be more accurate at a fine level in the case of MLP. For example, the “staircase” shape of one of the buildings in Row 1 is noticeably better outlined by the MLP network.

We can also observe that the ground truth itself is often not very precise. For example, the car in Row 3 does not seem to be labeled accurately, hence it is difficult to imagine that a network would learn to finely label that class. In Row 5, an entire lightwell between buildings has apparently been omitted in the ground truth (labeled as part of the building), yet recognized as an impervious surface by the CNNs.

The general recognition capabilities of CNNs can also be well appreciated in these fragments. For example, in Row 4, while there are tiny round objects both on the roof of the building and outside the building, CNNs correctly label as *building* the ones on the roof and as *clutter* the other ones.

TABLE IV: Comparison of MLP with other methods on the Vaihingen validation set.

	Imp. surf.	Build.	Low veg.	Tree	Car	F1	Acc.
CNN+RF [12]	88.58	94.23	76.58	86.29	67.58	82.65	86.52
CNN+RF+CRF [12]	89.10	94.30	77.36	86.25	71.91	83.78	86.89
Deconvolution [3]						83.58	87.83
Dilation [4]	90.19	94.49	77.69	87.24	76.77	85.28	87.70
Dilation + CRF [4]	90.41	94.73	78.25	87.25	75.57	85.24	87.90
MLP	**91.69**	**95.24**	**79.44**	**88.12**	**78.42**	**86.58**	**88.92**

TABLE V: Comparison of MLP with other methods on the Potsdam validation set.

	Imp. surf.	Build.	Low veg.	Tree	Car	Clutter	F1	Acc.
Dilation [4]	86.52	90.78	83.01	78.41	90.42	68.67	82.94	84.14
VGG pretr. [4]	89.84	93.80	85.43	83.61	88.00	74.48	85.86	87.42
VGG+Dilation [4]	**89.95**	93.73	**85.91**	**83.86**	**94.31**	74.62	**87.06**	**87.69**
MLP	89.31	**94.37**	84.83	81.10	93.56	**76.54**	86.62	87.02

TABLE VI: Execution times.

	Train [s]		Test [s/ha]	
	Vaih.	Pots.	Vaih.	Pots.
Base FCN	3.9	9.8	0.81	1.44
Unpooling	8.4	21.0	1.38	1.84
Skip	6.6	16.9	0.81	1.48
MLP	10.0	24.5	1.70	2.0
Dilation*	62	400	4.81	17.2

*As reported in [4], using a faster machine (see details in Sec. V-F.)

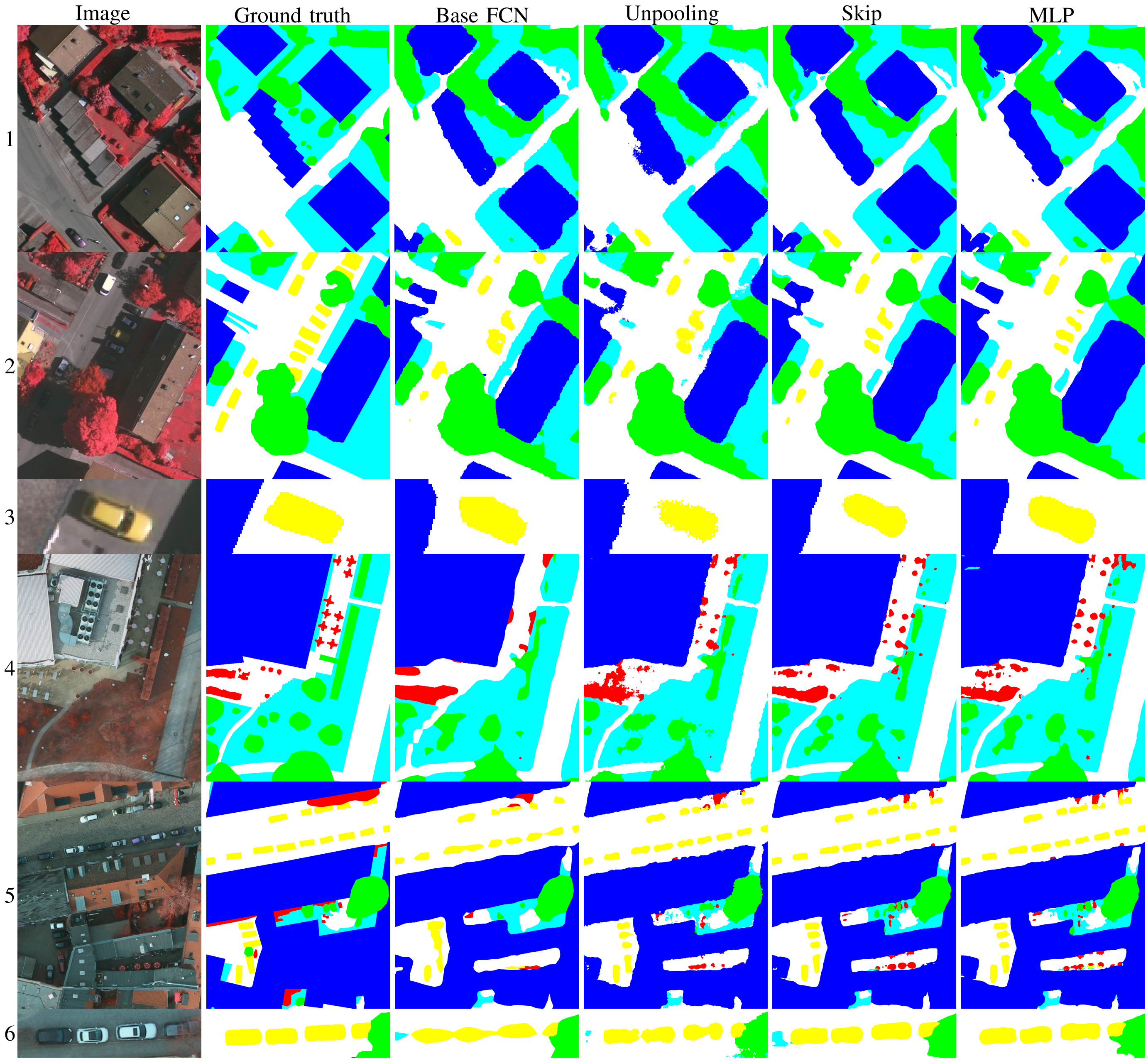

Fig. 8: Classification of closeups of Vaihingen (1–3) and Potsdam (4–6) validation sets. Classes: Impervious surface (white), Building (blue), Low veget. (cyan), Tree (green), Car (yellow), Clutter (red).

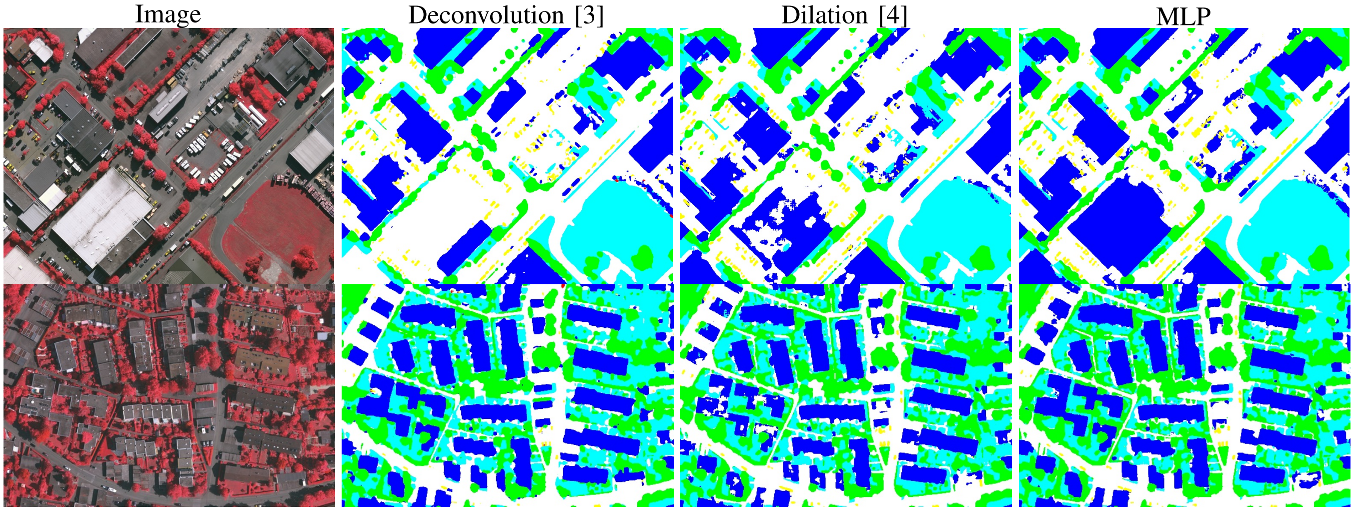

Fig. 9: Classification of entire tiles of the Vaihingen test set.

In Fig. 9 we show the classification of entire tiles of the Vaihingen set, obtained from the test set submissions. We include the *deconvolution* [3] and *dilation* [4] network results, together with our MLP. We can see, for instance, that a large white building in the first image is recognized by MLP but misclassified or only partially recovered by the other methods. In the second tile, the Dilation method outputs some holes in the buildings which are not present in the MLP results. A better combination of the information coming from different resolutions might explain why MLP successfully recognizes that these entire surfaces do belong to the same object.

F. Running Times

Table VI reports the running times for training and testing on both datasets. The training time of the architectures derived from the base FCN comprises the time to pretrain the base FCN first and the time to then train the whole system altogether (see details in Sec. V-C). The architectures were implemented using Caffe [32] and run on an Intel I7 CPU @ 2.7Ghz with a Quadro K3100M GPU (4 GB RAM). We also add for comparison the results reported by the author of the Dilation network [4], run on a larger 12 GB RAM GPU. To classify large images we crop them into tiles with as much overlap as the amount of padding in the network, to avoid tile border effects.

As reported in the table, the unpooling, skip and MLP networks introduce an overhead to the base FCN. MLP is the slowest of the derived networks, followed by the unpooling and skip networks. MLP, which provides the highest accuracy, classifies the entire Vaihingen validation set in about 30 seconds and the Postdam validation set in 2 minutes. This is much faster than the dilation network. Incorporating the principle of Fig. 2 allows us to better allocate computational resources, not spending too much time and space in conducting a high-resolution analysis where it is not needed, boosting accuracy and performance.

VI. Concluding Remarks

Convolutional neural networks (CNNs) are becoming the leading choice for high-resolution semantic labeling. The biggest concern with this technique is the spatial coarseness of the outputs. Most of the work has moderately modified or post-processed well-known CNN architectures in order to counteract this issue. We decided, however, to rethink CNNs from a semantic labeling perspective.

For this purpose, we first analyzed different families of semantic labeling CNN prototypes. This analysis bears some similarity with the reasoning that gave birth to CNNs themselves: we study which relevant constraints can be imposed in the architecture by construction, reducing the number of parameters and improving the optimization. We observed that existing networks often spend efforts in learning invariances that could be otherwise guaranteed, and reason at a high resolution even when it is not needed. While previous methods are already competitive, we can devise more optimal approaches.

We derived a model in which spatial features are learned at multiple resolutions (and thus different levels of detail) and a specific CNN module learns how to combine them. In our experiments on aerial imagery, such a model proved to be more effective than the other approaches to conduct high-resolution labeling. It provides a better accuracy with low computational requirements, leading to a win-win situation. Some of the outperformed methods are in fact significantly more complex than our approach, proving once again that striving for simplicity is often the way to go when using CNN architectures.

We hope that our architectural prototype will be used as a basis for semantic labeling networks. Our future plan is to create a large-scale aerial image dataset, covering dissimilar areas of the earth, on which to conduct semantic labeling with convolutional neural networks.

Acknowledgment

The authors would like to thank CNES for initializing and funding this study.

REFERENCES

[1] Gellert Mattyus, Shenlong Wang, Sanja Fidler, and Raquel Urtasun, "Hd maps: Fine-grained road segmentation by parsing ground and aerial images," in *IEEE CVPR*, 2016.

[2] Matthew D Zeiler and Rob Fergus, "Visualizing and understanding convolutional networks," in *ECCV*. Springer, 2014.

[3] Michele Volpi and Devis Tuia, "Dense semantic labeling of subdecimeter resolution images with convolutional neural networks," *arXiv preprint arXiv:1608.00775*, 2016.

[4] Jamie Sherrah, "Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery," *arXiv preprint arXiv:1606.02585*, 2016.

[5] Anastasia Dubrovina, Pavel Kisilev, Boris Ginsburg, Sharbell Hashoul, and Ron Kimmel, "Computational mammography using deep neural networks," *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, pp. 1–5, 2016.

[6] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han, "Learning deconvolution network for semantic segmentation," in *IEEE CVPR*, 2015, pp. 1520–1528.

[7] Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling," *arXiv preprint arXiv:1505.07293*, 2015.

[8] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[9] Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[10] D Marmanisa, JD Wegnera, S Gallianib, K Schindlerb, M Datcuc, and U Stillad, "Semantic segmentation of aerial images with an ensemble of cnns," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 473–480, 2016.

[11] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014.

[12] Sakrapee Paisitkriangkrai, Jamie Sherrah, Pranam Janney, Van-Den Hengel, et al., "Effective semantic pixel labelling with convolutional networks and conditional random fields," in *IEEE CVPR Workshops*, 2015.

[13] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik, "Hypercolumns for object segmentation and fine-grained localization," in *IEEE CVPR*, 2015, pp. 447–456.

[14] ISPRS, "http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html," .

[15] Christopher M Bishop, *Neural networks for pattern recognition*, Oxford university press, 1995.

[16] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[18] Y-Lan Boureau, Jean Ponce, and Yann LeCun, "A theoretical analysis of feature pooling in visual recognition," in *ICML*, 2010, pp. 111–118.

[19] Michael Kampffmeyer, Arnt-Borre Salberg, and Robert Jenssen, "Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks," in *IEEE CVPR Workshops*, 2016.

[20] Fisher Yu and Vladlen Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[21] Jimei Yang, Brian Price, Scott Cohen, Honglak Lee, and Ming-Hsuan Yang, "Object contour detection with a fully convolutional encoder-decoder network," *arXiv preprint arXiv:1603.04530*, 2016.

[22] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *arXiv preprint arXiv:1511.00561*, 2015.

[23] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[24] Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa, "Learning to simplify: fully convolutional networks for rough sketch cleanup," *ACM Transactions on Graphics*, vol. 35, no. 4, pp. 121, 2016.

[25] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, 2010.

[26] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, 2013.

[27] Markus Gerke, "Use of the stair vision library within the isprs 2d semantic labeling benchmark (vaihingen)," Tech. Rep., Technical report, University of Twente, 2015.

[28] Russell G Congalton and Kass Green, *Assessing the accuracy of remotely sensed data: principles and practices*, CRC press, 2008.

[29] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *IEEE CVPR*, 2015, pp. 1026–1034.

[31] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, 2014.

[32] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

Emmanuel Maggiori (S'15) received the Engineering degree in computer science from Central Buenos Aires Province National University (UNCPBA), Tandil, Argentina, in 2014. The same year he joined AYIN and STARS teams at Inria Sophia Antipolis-Méditerranée as a research intern in the field of remote sensing image processing. Since 2015, he has been working on his Ph.D. within TITANE team, studying machine learning techniques for large-scale processing of satellite imagery.

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

From July 2007 to January 2008, she was a researcher with the Norwegian Defence Research Establishment, Norway. From September 2010 to December 2011, she was a postdoctoral research fellow with the Computational and Information Sciences and Technology Office, NASA Goddard Space Flight Center, Greenbelt, MD. From January to August 2012 she was a postdoctoral research fellow with the French Space Agency (CNES) and Inria Sophia Antipolis-Méditerranée, France. She is currently a researcher with the TITANE team of Inria Sophia Antipolis-Méditerranée. Her research interests are in the areas of image processing, pattern recognition and development of efficient algorithms. She is Member of the IEEE Society.

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 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 Senior Researcher and team leader at 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.