Learning Hierarchical Features for Scene Labeling

Clément Farabet, Camille Couprie, Laurent Najman, Yann LeCun

Abstract—Scene labeling consists in labeling each pixel in an image with the category of the object it belongs to. We propose a method that uses a multiscale convolutional network trained from raw pixels to extract dense feature vectors that encode regions of multiple sizes centered on each pixel. The method alleviates the need for engineered features, and produces a powerful representation that captures texture, shape and contextual information. We report results using multiple post-processing methods to produce the final labeling. Among those, we propose a technique to automatically retrieve, from a pool of segmentation components, an optimal set of components that best explain the scene; these components are arbitrary, *e.g.* they can be taken from a segmentation tree, or from any family of over-segmentations. The system yields record accuracies on the Sift Flow Dataset (33 classes) and the Barcelona Dataset (170 classes) and near-record accuracy on Stanford Background Dataset (8 classes), while being an order of magnitude faster than competing approaches, producing a 320×240 image labeling in less than a second, including feature extraction.

Index Terms—Convolutional networks, deep learning, image segmentation, image classification, scene parsing.

1 Introduction

IMAGE UNDERSTANDING is a task of primary importance for a wide range of practical applications. One important step towards understanding an image is to perform a *full-scene labeling* also known as a *scene parsing*, which consists in labeling every pixel in the image with the category of the object it belongs to. After a perfect scene parsing, every region and every object is delineated and tagged. One challenge of scene parsing is that it combines the traditional problems of detection, segmentation, and multi-label recognition in a single process.

There are two questions of primary importance in the context of scene parsing: how to produce good internal representations of the visual information, and how to use contextual information to ensure the self-consistency of the interpretation.

This paper presents a scene parsing system that relies on deep learning methods to approach both questions. The main idea is to use a convolutional network [27] operating on a large input window to produce label hypotheses for each pixel location. The convolutional net is fed with raw image pixels (after band-pass filtering and contrast normalization), and trained in supervised mode from fully-labeled images to produce a category for each pixel location. Convolutional networks are composed of multiple stages each of which contains a filter bank module, a non-linearity, and a spatial pooling module. With end-to-end training, convolutional networks can automatically learn hierarchical feature representations.

Unfortunately, labeling each pixel by looking at a small region around it is difficult. The category of a pixel may depend on relatively short-range information (*e.g.* the presence of a human face generally indicates the presence of a human body nearby), but may also depend on long-range information. For example, identifying a grey pixel as belonging to a road, a sidewalk, a gray car, a concrete building, or a cloudy sky requires a wide contextual window that shows enough of the surroundings to make an informed decision. To address this problem, we propose to use a *multi-scale convolutional network*, which can take into account large input windows, while keeping the number of free parameters to a minimum.

Common approaches to scene parsing first produce segmentation hypotheses using graph-based methods. Candidate segments are then encoded using engineered features. Finally, a conditional random field (or some other type of graphical model), is trained to produce labels for each candidate segment, and to ensure that the labelings are globally consistent.

A striking characteristic of the system proposed here is that the use of a large contextual window to label pixels reduces the requirement for sophisticated post-processing methods that ensure the consistency of the labeling.

More precisely, the proposed scene parsing architecture is depicted on Figure 1. It relies on two main components:

1) Multi-scale, convolutional representation: our multi-scale, dense feature extractor produces a series of feature vectors for regions of multiple sizes centered around every pixel in the image, covering a large context. The multi-scale convolutional net contains multiple copies of a single network (all sharing the same weights) that are applied to different scales of a Laplacian pyramid version of the input image. For each pixel, the networks collectively encode the information present in a large contextual window around the given pixel (184×184 pixels in the system described here). The convolutional network is fed with raw pixels and trained end to end, thereby alleviating the need for hand-engineered features. When properly trained, these features produce a representation that captures texture, shape and contextual information. While using

- *Clément Farabet, Camille Couprie, and Yann LeCun are with the Courant Institute of Mathematical Sciences, New York University (New York, NY 10003, USA).*
- *Clément Farabet and Laurent Najman are with the Laboratoire d'Informatique Gaspard-Monge, Université Paris-Est, Equipe A3SI, ESIEE Paris (93160 Noisy-le-Grand, France).*
E-mails: cfarabet@cs.nyu.edu, ccouprie@cs.nyu.edu, l.najman@esiee.fr, yann@cs.nyu.edu

a multiscale representation seems natural for FSL, it has rarely been used in the context of feature learning systems. The multiscale representation that is learned is sufficiently complete to allow the detection and recognition of all the objects and regions in the scene. However, it does not accurately pinpoint the boundaries of the regions, and requires some post-processing to yield cleanly delineated predictions.

2) Graph-based classification:

An over-segmentation is constructed from the image, and is used to group the feature descriptors. Several over-segmentations are considered, and three techniques are proposed to produce the final image labeling.

2.a. Superpixels: The image is segmented into disjoint components, widely over-segmenting the scene. In this scenario, a pixelwise classifier is trained on the convolutional feature vectors, and a simple vote is done for each component, to assign a single class per component. This method is simple and effective, but imposes a fixed level of segmentation, which can be suboptimal.

2.b. Conditional random field over superpixels: a conditional random field is defined over a set of superpixels. Compared to the previous, simpler method, this post-processing models joint probabilities at the level of the scene, and is useful to avoid local aberrations (*e.g.* a person in the sky). That kind of approach is widely used in the computer vision community, and we show that our learned multiscale feature representation essentially makes the use of a global random field much less useful: most scene-level relationships seem to be already captured by it.

2.c. Multilevel cut with class purity criterion: A family of segmentations is constructed over the image to analyze the scene at multiple levels. In the simplest case, this family might be a segmentation tree; in the most general case it can be any set of segmentations, for example a collection of superpixels either produced using the same algorithm with different parameter tunings or produced by different algorithms. Each segmentation component is represented by the set of feature vectors that fall into it: the component is encoded by a spatial grid of aggregated feature vectors. The aggregated feature vector of each grid cell is computed by a component-wise max pooling of the feature vectors centered on all the pixels that fall into the grid cell. This produces a scale-invariant representation of the segment and its surrounding. A classifier is then applied to the aggregated feature grid of each node. This classifier is trained to estimate the histogram of all object categories present in the component. A subset of the components is then selected such that they cover the entire image. These components are selected so as to minimize the average "impurity" of the class distribution in a procedure that we name "optimal cover". The class "impurity" is defined as the entropy of the class distribution. The choice of the cover thus attempts to find a consistent overall segmentation in which each segment contains pixels belonging to only one of the learned categories. This simple method allows us to consider full families of segmentation components, rather than a unique, predetermined segmentation (*e.g.* a single set of superpixels).

All the steps in the process have a complexity linear (or almost linear) in the number of pixels. The bulk of the computation resides in the convolutional network feature extractor. The resulting system is very fast, producing a full parse of a 320×240 image in less than a second on a conventional CPU, and in less than 100ms using dedicated hardware, opening the door to real-time applications. Once trained, the system is parameter free, and requires no adjustment of thresholds or other knobs.

An early version of this work was first published in [7]. This journal version reports more complete experiments, comparisons and higher results.

2 Related work

The scene parsing problem has been approached with a wide variety of methods in recent years. Many methods rely on MRFs, CRFs, or other types of graphical models to ensure the consistency of the labeling and to account for context [19], [39], [15], [25], [32], [44], [30]. Most methods rely on a pre-segmentation into superpixels or other segment candidates, and extract features and categories from individual segments and from various combinations of neighboring segments. The graphical model inference pulls out the most consistent set of segments which covers the image.

[43] proposed a method to aggregate segments in a greedy fashion using a trained scoring function. The originality of the approach is that the feature vector of the combination of two segments is computed from the feature vectors of the individual segments through a trainable function. Like us, they use "deep learning" methods to train their feature extractor. But unlike us, their feature extractor operates on hand-engineered features.

One of the main question in scene parsing is how to take a wide context into account to make a local decision. [32] proposed to use the histogram of labels extracted from a coarse scale as input to the labeler that looks at finer scales. Our approach is somewhat simpler: our feature extractor is applied densely to an image pyramid. The coarse feature maps thereby generated are upsampled to match that of the finest scale. Hence with three scales, each feature vector has multiple fields which encode multiple regions of increasing sizes and decreasing resolutions, centered on the same pixel location.

Like us, a number of authors have used families of segmentations or trees to generate candidate segments by aggregating elementary segments. The approaches of [39], [30] rely on inference algorithms based on Graph Cuts to label images using trees of segmentation. Other strategies using families of segmentations appeared in [36], [5]. None of the previous strategies for scene labeling used a purity criterion on the class distributions. Combined to the optimal cover strategy, this purity criterion is general, efficient and could be applied to solve different problems.

Contrary to the previously cited approaches using engineered features, our system extracts features densely from a multiscale pyramid of images using a convolutional network (ConvNet) [27]. These networks can be fed with raw pixels and can automatically learn low-level and mid-level features, alleviating the need for

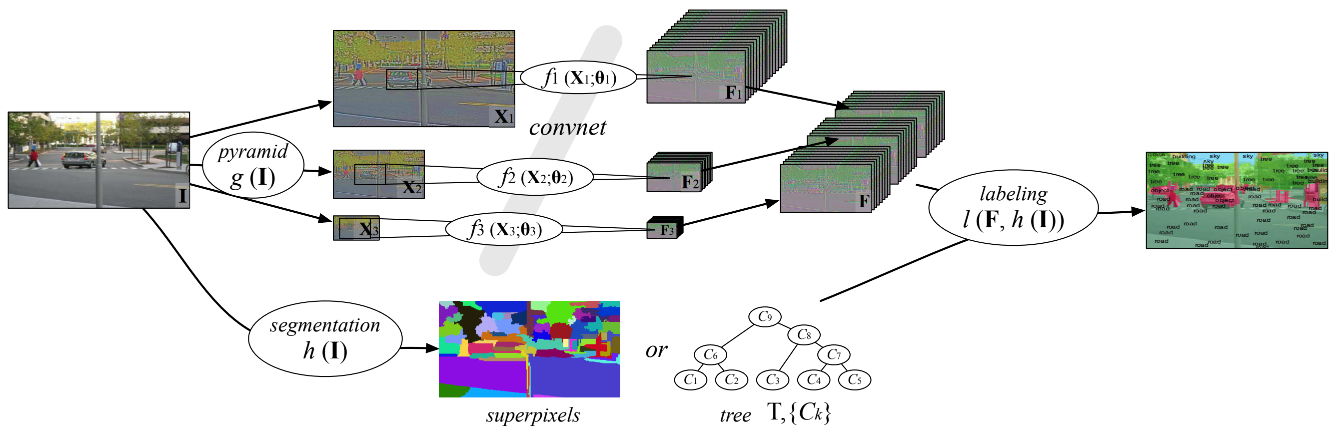

Fig. 1. Diagram of the scene parsing system. The raw input image is transformed through a Laplacian pyramid. Each scale is fed to a 3-stage convolutional network, which produces a set of feature maps. The feature maps of all scales are concatenated, the coarser-scale maps being upsampled to match the size of the finest-scale map. Each feature vector thus represents a large contextual window around each pixel. In parallel, a single segmentation (*i.e.* superpixels), or a family of segmentations (*e.g.* a segmentation tree) are computed to exploit the natural contours of the image. The final labeling is produced from the feature vectors and the segmentation(s) using different methods, as presented in section 4.

hand-engineered features. One of their advantage is the ability to compute dense features efficiently over large images. They are best known for their applications to detection and recognition [47], [14], [35], [21], but they have also been used for image segmentation, particularly for biological image segmentation [34], [20], [46].

The only previously published work on using convolutional networks for scene parsing is that of [17]. While somewhat preliminary, their work showed that convolutional networks fed with raw pixels could be trained to perform scene parsing with decent accuracy. Unlike [17] however, our system uses a boundary-based hierarchy of segmentations to align the labels produced by the network to the boundaries in the image and thus produces representations that are independent of the size of the segments through feature pooling. Slightly after [8], Schulz and Behnke proposed a similar architecture of a multiscale convolutional network for scene parsing [40]. Unlike us, they use pairwise class location filters to predict the final segmentation, instead of using the image gradient that we found to be more accurate.

3 Multiscale feature extraction for scene parsing

The model proposed in this paper, depicted on Figure 1, relies on two complementary image representations. In the first representation, an image patch is seen as a point in $\mathbb{R}^P$, and we seek to find a transform $f : \mathbb{R}^P \to \mathbb{R}^Q$ that maps each patch into $\mathbb{R}^Q$, a space where it can be classified linearly. This first representation typically suffers from two main problems when using a classical convolutional network, where the image is divided following a grid pattern: (1) the window considered rarely contains an object that is properly centered and scaled, and therefore offers a poor observation basis to predict the class of the underlying object, (2) integrating a large context involves increasing the grid size, and therefore the dimensionality P of the input; given a finite amount of training data, it is then necessary to enforce some invariance in the function f itself. This is usually achieved by using pooling/subsampling layers, which in turn degrades the ability of the model to precisely locate and delineate objects. In this paper, f is implemented by a multiscale convolutional network, which allows integrating large contexts (as large as the complete scene) into local decisions, yet still remaining manageable in terms of parameters/dimensionality. This multiscale model, in which weights are shared across scales, allows the model to capture long-range interactions, without the penalty of extra parameters to train. This model is described in Section 3.1.

In the second representation, the image is seen as an edge-weighted graph, on which one or several over-segmentations can be constructed. The components are spatially accurate, and naturally delineate the underlying objects, as this representation conserves pixel-level precision. Section 4 describes multiple strategies to combine both representations. In particular, we describe in Section 4.3 a method for analyzing a family of segmentations (at multiple levels). It can be used as a solution to the first problem exposed above: assuming the capability of assessing the quality of all the components in this family of segmentations, a system can automatically choose its components so as to produce the best set of predictions.

3.1 Scale-invariant, scene-level feature extraction

Good internal representations are hierarchical. In vision, pixels are assembled into edglets, edglets into motifs, motifs into parts, parts into objects, and objects into scenes. This suggests that recognition architectures for vision (and for other modalities such as audio and natural language) should have multiple trainable stages stacked on top of each other, one for each level in the feature hierarchy. Convolutional Networks (ConvNets) provide a simple framework to learn such hierarchies of features.

Convolutional Networks [26], [27] are trainable architectures composed of multiple stages. The input and output of each stage are sets of arrays called *feature maps*. For example, if the input is a color image, each feature map would be a 2D array containing a color channel of the input image (for an audio input each feature map would be a 1D array, and for a video or volumetric image, it would be a 3D array). At the output, each feature map represents a particular feature extracted at all locations on the input. Each stage is composed of three layers: a *filter bank layer*, a *non-linearity layer*, and a *feature pooling layer*. A typical ConvNet is composed of one, two or three such 3-layer stages, followed by a classification module. Because they are trainable, arbitrary input modalities can be modeled, beyond natural images.

Our feature extractor is a three-stage convolutional network. The first two stages contain a bank of filters producing multiple feature maps, a point-wise non-linear mapping and a spatial pooling followed by subsampling of each feature map. The last layer only contains a bank of filters. The filters (convolution kernels) are subject to training. Each filter is applied to the input feature maps through a 2D convolution operation, which detects local features at all locations on the input. Each filter bank of a convolutional network produces features that are equivariant under shifts, *i.e.* if the input is shifted, the output is also shifted but otherwise unchanged.

While convolutional networks have been used successfully for a number of image labeling problems, image-level tasks such as full-scene understanding (pixel-wise labeling, or any dense feature estimation) require the system to model complex interactions at the scale of complete images, not simply within a patch. To view a large contextual window at full resolution, a convolutional network would have to be unmanageably large.

The solution is to use a multiscale approach. Our multiscale convolutional network overcomes these limitations by extending the concept of spatial weight replication to the scale space. Given an input image $\mathbf{I}$, a multiscale pyramid of images $\mathbf{X}_s$, $\forall s \in \{1,\dots,N\}$ is constructed, where $\mathbf{X}_1$ has the size of $\mathbf{I}$. The multiscale pyramid can be a Laplacian pyramid, and is typically pre-processed, so that local neighborhoods have zero mean and unit standard deviation. Given a classical convolutional network f_s with parameters θ_s, the multiscale network is obtained by instantiating one network per scale s, and sharing all parameters across scales: $\theta_s = \theta_0$, $\forall s \in \{1,\dots,N\}$.

We introduce the following convention: banks of images will be seen as three dimensional arrays in which the first dimension is the number of independent feature maps, or images, the second is the height of the maps and the third is the width. The output state of the L-th stage is denoted $\mathbf{H}_L$.

The maps in the pyramid are computed using a scaling/normalizing function g_s as $\mathbf{X}_s = g_s(\mathbf{I})$, for all $s \in \{1,\dots,N\}$.

For each scale s, the convolutional network f_s can be described as a sequence of linear transforms, interspersed with non-linear symmetric squashing units (typically the $\tanh$ function [28]), and pooling/subsampling operators. For a network f_s with L layers, we have:

$$f_s(\mathbf{X}_s;\theta_s) = \mathbf{W}_L \mathbf{H}_{L-1}, \tag{1}$$

where the vector of hidden units at layer l is

$$\mathbf{H}_l = \mathrm{pool}(\tanh(\mathbf{W}_l \mathbf{H}_{l-1} + \mathbf{b}_l)) \tag{2}$$

for all $l \in \{1,\dots,L-1\}$, with $\mathbf{b}_l$ a vector of bias parameters, and $\mathbf{H}_0 = \mathbf{X}_s$. The matrices $\mathbf{W}_l$ are Toeplitz matrices, therefore each hidden unit vector $\mathbf{H}_l$ can be expressed as a regular convolution between kernels from $\mathbf{W}_l$ and the previous hidden unit vector $\mathbf{H}_{l-1}$, squashed through a $\tanh$, and pooled spatially. More specifically,

$$\mathbf{H}_{lp} = \mathrm{pool}(\tanh(b_{lp} + \sum_{q \in \mathrm{parents}(p)} \mathbf{w}_{lpq} * \mathbf{H}_{l-1,q})). \tag{3}$$

The filters $\mathbf{W}_l$ and the biases $\mathbf{b}_l$ constitute the trainable parameters of our model, and are collectively denoted θ_s. The function $\tanh$ is a point-wise non-linearity, while pool is a function that considers a neighborhood of activations, and produces one activation per neighborhood. In all our experiments, we use a max-pooling operator, which takes the maximum activation within the neighborhood. Pooling over a small neighborhood provides built-in invariance to small translations.

Finally, the outputs of the N networks are upsampled and concatenated so as to produce $\mathbf{F}$, a map of feature vectors of size N times the size of $\mathbf{f}_1$, which can be seen as local patch descriptors and scene-level descriptors

$$\mathbf{F} = [\mathbf{f}_1, u(\mathbf{f}_2), \dots, u(\mathbf{f}_N)], \tag{4}$$

where u is an upsampling function.

As mentioned above, weights are shared between networks f_s. Intuitively, imposing complete weight sharing across scales is a natural way of forcing the network to learn scale invariant features, and at the same time reduce the chances of over-fitting. The more scales used to jointly train the models $f_s(\theta_s)$ the better the representation becomes for all scales. Because image content is, in principle, scale invariant, using the same function to extract features at each scale is justified.

3.2 Learning discriminative scale-invariant features

As described in Section 3.1, feature vectors in $\mathbf{F}$ are obtained by concatenating the outputs of multiple networks f_s, each taking as input a different image in a multiscale pyramid.

Ideally a linear classifier should produce the correct categorization for all pixel locations i, from the feature vectors $\mathbf{F}_i$. We train the parameters θ_s to achieve this goal, using the multiclass *cross entropy* loss function. Let $\hat{\mathbf{c}}_i$ be the normalized prediction vector from the linear classifier for pixel i. We compute normalized predicted probability distributions over classes $\hat{\mathbf{c}}_{i,a}$ using the *softmax* function, *i.e.*

$$\hat{\mathbf{c}}_{i,a} = \frac{e^{\mathbf{w}_a^T \mathbf{F}_i}}{\sum_{b \in \text{classes}} e^{\mathbf{w}_b^T \mathbf{F}_i}}, \tag{5}$$

where $\mathbf{w}$ is a temporary weight matrix only used to learn the features. The cross entropy between the predicted class distribution $\hat{\mathbf{c}}$ and the target class distribution $\mathbf{c}$ penalizes their deviation and is measured by

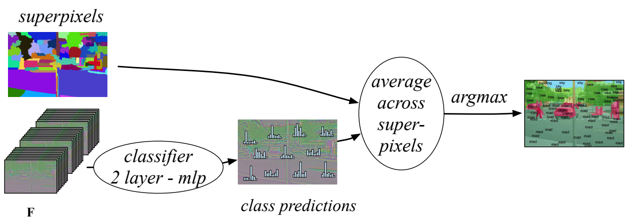

Fig. 2. First labeling strategy from the features: using superpixels as described in Section 4.1.

$$L_{\text{cat}} = -\sum_{i \in \text{pixels}} \sum_{a \in \text{classes}} \mathbf{c}_{i,a} \ln(\hat{\mathbf{c}}_{i,a}). \tag{6}$$

The true target probability $\mathbf{c}_{i,a}$ of class a to be present at location i can either be a distribution of classes at location i, in a given neighborhood or a hard target vector: $\mathbf{c}_{i,a} = 1$ if pixel i is labeled a, and 0 otherwise. For training maximally discriminative features, we use hard target vectors in this first stage.

Once the parameters θ_s are trained, the classifier in Eq 5 is discarded, and the feature vectors $\mathbf{F}_i$ are used using different strategies, as described in Section 4.

4 Scene Labeling Strategies

The simplest strategy for labeling the scene is to use the linear classifier described in Section 3.2, and assign each pixel with the *argmax* of the prediction at its location. More specifically, for each pixel i

$$l_i = \arg\max_{a \in \text{classes}} \hat{\mathbf{c}}_{i,a} \tag{7}$$

The resulting labeling l, although fairly accurate, is not satisfying visually, as it lacks spatial consistency, and precise delineation of objects. In this section, we explore three strategies to produce spatially more appealing labelings.

4.1 Superpixels

Predicting the class of each pixel independently from its neighbors yields noisy predictions. A simple cleanup can be obtained by forcing local regions of same color intensities to be assigned a single label.

As in [13], [16], we compute superpixels, following the method proposed by [11], to produce an over-segmentation of the image. We then classify each location of the image densely, and aggregate these predictions in each superpixel, by computing the average class distribution within the superpixel.

For this method, the pixelwise distributions $\hat{\mathbf{d}}_k$ at superpixel k are predicted from the feature vectors $\mathbf{F}_i$ using a two-layer neural network:

$$\begin{aligned}
\mathbf{y}_i &= \mathbf{W}_2 \tanh(\mathbf{W}_1 \mathbf{F}_i + \mathbf{b}_1), & (8)\\
\hat{\mathbf{d}}_{i,a} &= \frac{e^{\mathbf{y}_{i,a}}}{\sum_{b \in \text{classes}} e^{\mathbf{y}_{i,b}}}, & (9)\\
L_{\text{cat}} &= -\sum_{i \in \text{pixels}} \sum_{a \in \text{classes}} \mathbf{d}_{i,a} \ln(\hat{\mathbf{d}}_{i,a}), & (10)\\
\hat{\mathbf{d}}_{k,a} &= \frac{1}{\mathrm{s}(k)} \sum_{i \in k} \hat{\mathbf{d}}_{i,a}, & (11)
\end{aligned}$$

with $\mathbf{d}_i$ the groundtruth distribution at location i, and $\mathrm{s}(k)$ the surface of component k. Matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ are the trainable parameters of the classifier. Using a two-layer neural network, as opposed to the simple linear classifier used in Section 3.2, allows the system to capture non-linear relationships between the features at different scales. In this case, the final labeling for each component k is given by

$$l_k = \arg\max_{a \in \text{classes}} \hat{\mathbf{d}}_{k,a}. \tag{12}$$

The pipeline is depicted in Figure 2.

4.2 Conditional Random Fields

The local assignment obtained using superpixels does not involve a global understanding of the scene. In this section, we implement a classical CRF model, constructed on the superpixels. This is a quite standard approach for image labeling. Our multi-scale convolutional network already has the capability of modeling global relationships within a scene, but might still be prone to errors, and can benefit from a CRF, to impose consistency and coherency between labels, at test time.

A common strategy for labeling a scene consists in associating the image to a graph and define an energy function whose optimal solution corresponds to the desired segmentation [41], [13].

For this purpose, we define a graph $G = (V, E)$ with vertices $v \in V$ and edges $e \in E \subseteq V \times V$. Each pixel in the image is associated to a vertex, and edges are added between every neighboring nodes. An edge, e, spanning two vertices, v_i and v_j , is denoted by e_{ij}. The Conditional Random Field (CRF) energy function is typically composed of a unary term enforcing the variable l to take values close to the predictions $\hat{\mathbf{d}}$ and a pairwise term enforcing regularity or local consistency of l. The CRF energy to minimize is given by

$$E(l) = \sum_{i \in V} \Phi(\hat{\mathbf{d}}_i, l_i) + \gamma \sum_{e_{ij} \in E} \Psi(l_i, l_j) \tag{13}$$

We considered as unary terms

$$\Phi(\hat{\mathbf{d}}_{i,a}, l_i) = \exp\left(-\alpha \hat{\mathbf{d}}_{i,a}\right) \mathbf{1}(l_i \neq a), \tag{14}$$

where $\hat{\mathbf{d}}_{i,a}$ corresponds to the probability of class a to be present at a pixel i computed as in Section 4.1, and $\mathbf{1}(\cdot)$ is an indicator function that equals one if the input is true, and zero otherwise.

The pairwise term consists in

$$\Psi(l_i, l_j) = \exp\left(-\beta \|\nabla I\|_i\right) \mathbf{1}(l_i \neq l_j) \tag{15}$$

where $\|\nabla I\|_i$ is the ℓ_2 norm of the gradient of the image I at a pixel i. Details on the parameters used are given in the experimental section.

The CRF energy (13) is minimized using alpha-expansions [4], [3]. An illustration of the procedure appears in Figure 3.

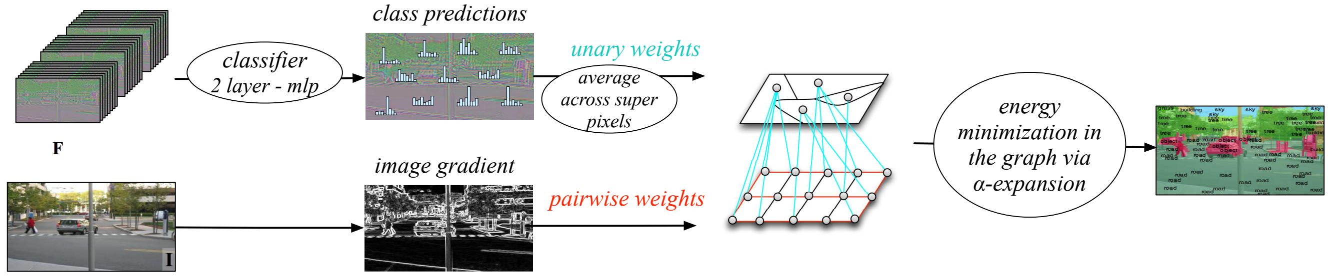

Fig. 3. Second labeling strategy from the features: using a CRF, described in Section 4.2.

4.3 Parameter-free multilevel parsing

One problem subsists with the two methods presented above: the observation level problem. An object, or object part, can be easily classified once it is segmented at the right level. The two methods above are based on an arbitrary segmentation of the image, which typically decomposes it into segments that are too small, or, more rarely, too large.

In this section, we propose a method to analyze a family of segmentations and automatically discover the best observation level for each pixel in the image. One special case of such families is the segmentation tree, in which components are hierarchically organized. Our method is not restricted to such trees, and can be used for arbitrary sets of neighborhoods.

In Section 4.3.1 we formulate the search for the most adapted neighborhood of a pixel as an optimization problem. The construction of the cost function that is minimized is then described in Section 4.3.2.

4.3.1 Optimal purity cover

We define the neighborhood of a pixel as a connected component that contains this pixel. Let C_k, $\forall k \in \{1,\ldots,K\}$ be the set of all possible connected components of the lattice defined on image I, and let S_k be a cost associated to each of these components. For each pixel i, we wish to find the index $k^*(i)$ of the component that best explains this pixel, that is, the component with the minimal cost $S_{k^*(i)}$:

$$k^*(i) = \underset{k \mid i \in C_k}{\operatorname{argmin}} S_k \tag{16}$$

Note that components $C_{k^*(i)}$ are non-disjoint sets that form a cover of the lattice. Note also that the overall cost $S^* = \sum_i S_{k^*(i)}$ is minimal.

In practice, the set of components C_k is too large, and only a subset of it can be considered. A classical technique to reduce the set of components is to consider a hierarchy of segmentations [33], [1], that can be represented as a tree T. This was previously explored in [7]. Solving Eq 16 on T consists in the following procedure: for each pixel (leaf) i, the optimal component $C_{k^*(i)}$ is the one along the path between the leaf and the root with minimal cost $S_{k^*(i)}$. The optimal cover is the union of all these components. For efficiency purposes, it can be done simply by exploring the tree in a depth-first search manner, and finding the component with minimal weight along each branch. The complexity of the optimal cover procedure is then linear in the

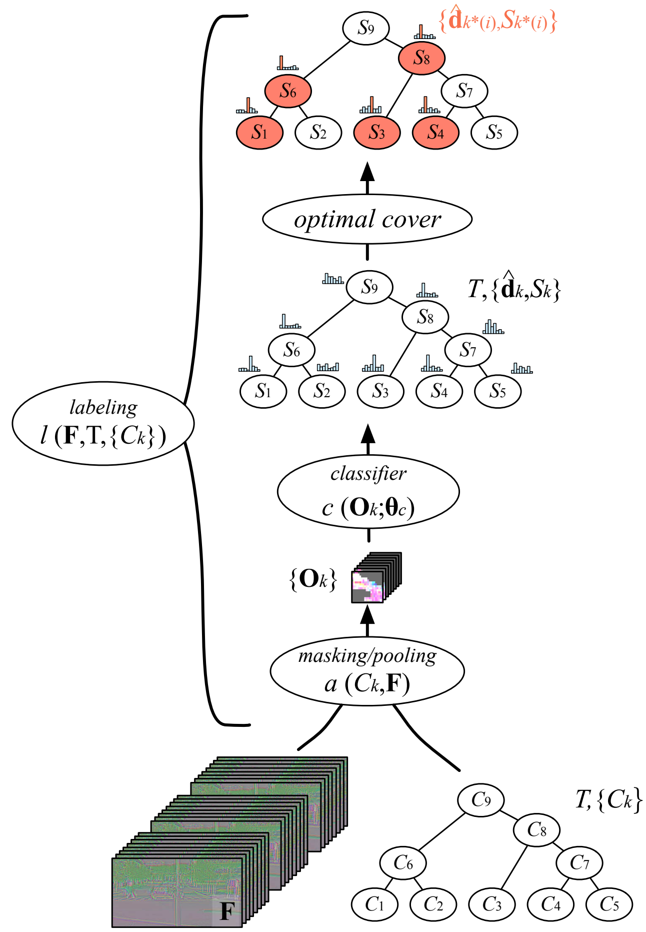

Fig. 4. Third labeling strategy from the features: using a family of of segmentations, as described in Section 4.3. On this figure, the family of segmentations is a segmentation tree. The segment associated with each node in the tree is encoded by a spatial grid of feature vectors pooled in the segment's region. A classifier is then applied to all the aggregated feature grids to produce a histogram of categories, the entropy of which measures the "impurity" of the segment. Each pixel is then labeled by the minimally-impure node above it, which is the segment that best "explains" the pixel.

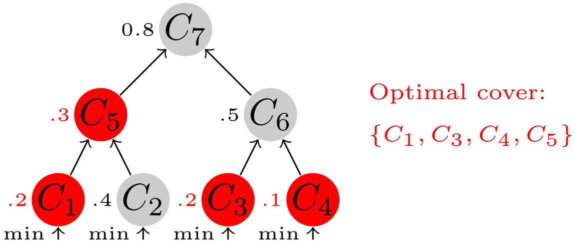

Fig. 5. Finding the optimal cover on a tree. The numbers next to the components correspond to the entropy scores S_i. For each pixel (leaf) i, the optimal component $C_{k^*(i)}$ is the one along the path between the leaf and the root with minimal cost $S_{k^*(i)}$. The optimal cover is the union of all these components. In this example, the optimal cover $\{C_1, C_3, C_4, C_5\}$ will result in a segmentation in disjoint sets $\{C_1, C_2, C_3, C_4\}$, with the subtle difference that component C_2 will be labelled with the class of C_5, as C_5 is the best observation level for C_2. The generalization to a family of segmentations is straightforward (see text).

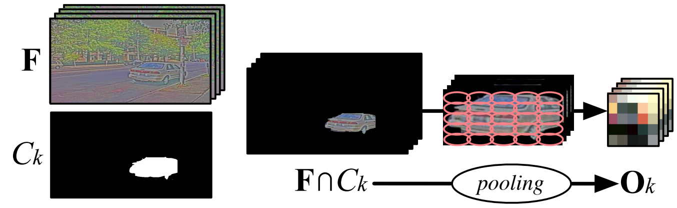

Fig. 6. The shape-invariant attention function a. For each component C_k in the family of segmentations $\mathcal{T}$, the corresponding image segment is encoded by a spatial grid of feature vectors that fall into this segment. The aggregated feature vector of each grid cell is computed by a component-wise max pooling of the feature vectors centered on all the pixels that fall into the grid cell; this produces a scale-invariant representation of the segment and its surroundings. The result, $\mathbf{O}_k$, is a descriptor that encodes spatial relations between the underlying object's parts. The grid size was set to 3×3 for all our experiments.

number of components in the tree. Figure 5 illustrates the procedure.

Another technique to reduce the set of components considered is to compute a set of segmentations using different merging thresholds. In Section 5, we use such an approach, by computing multiple levels of the Felzenszwalb algorithm [11]. The Felzenszwalb algorithm is not strictly monotonic, so the structure obtained cannot be cast into a tree: rather, it has a general graph form, in which each pixel belongs to as many superpixels as levels explored. Solving Eq 16 in this case consists in the following procedure: for each pixel i, the optimal component $C_{k^*(i)}$ is the one among all the segmentations with minimal cost $S_{k^*(i)}$. Thus the complexity to produce a cover on the family of components is linear on the number of pixels, but with a constant that is proportional to the number of levels explored.

4.3.2 Producing the confidence costs

Given a set of components C_k, we explain how to produce all the confidence costs S_k. These costs represent the class purity of the associated components. Given the groundtruth segmentation, we can compute the cost as being the entropy of the distribution of classes present in the component. At test time, when no groundtruth is available, we need to define a function that can predict this cost by simply looking at the component. We now describe a way of achieving this, as illustrated in Figure 6.

Given the scale-invariant features $\mathbf{F}$, we define a compact representation to describe objects as an elastic spatial arrangement of such features. In other terms, an object, or category in general, can be best described as a spatial arrangement of features, or parts. We define a simple attention function a used to mask the feature vector map with each component C_k, producing a set of K masked feature vector patterns $\{\mathbf{F} \bigcap C_k\}$, $\forall k \in \{1, \ldots, K\}$. The function a is called an attention function because it suppresses the background around the component being analyzed. The patterns $\{\mathbf{F} \bigcap C_k\}$ are resampled to produce fixed-size representations. In our model the sampling is done using an elastic max-pooling function, which remaps input patterns of arbitrary size into a fixed $G \times G$ grid. This grid can be seen as a highly invariant representation that encodes spatial relations between an object's attributes/parts. This representation is denoted $\mathbf{O}_k$. Some nice properties of this encoding are: (1) elongated, or in general ill-shaped objects, are nicely handled, (2) the dominant features are used to represent the object, combined with background subtraction, the features pooled represent solid basis functions to recognize the underlying object.

Once we have the set of object descriptors $\mathbf{O}_k$, we define a function $c : \mathbf{O}_k \rightarrow [0,1]^{N_c}$ (where N_c is the number of classes) as predicting the distribution of classes present in component C_k. We associate a cost S_k to this distribution. In this paper, c is implemented as a simple 2-layer neural network, and S_k is the entropy of the predicted distribution. More formally, let $\mathbf{O}_k$ be the feature vector associated with component C_k, $\hat{\mathbf{d}}_k$ the predicted class distribution, and S_k the cost associated to this distribution. We have

$$\mathbf{y}_k = \mathbf{W}_2 \tanh(\mathbf{W}_1 \mathbf{O}_k + \mathbf{b}_1), \quad (17)$$

$$\hat{\mathbf{d}}_{k,a} = \frac{e^{\mathbf{y}_{k,a}}}{\sum_{b \in \text{classes}} e^{\mathbf{y}_{k,b}}}, \quad (18)$$

$$S_k = -\sum_{a \in \text{classes}} \mathbf{d}_{k,a} \ln(\hat{\mathbf{d}}_{k,a}), \quad (19)$$

with $\mathbf{d}_k$ the groundtruth distribution for component k. Matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ are noted θ_c, and represent the trainable parameters of c. These parameters need to be learned over the complete set of segmentation families, computed on the entire training set available. The training procedure is described in Section 4.3.3.

For each component C_k chosen by the optimal purity cover (Section 4.3.1) the label is produced by:

$$l_k = \underset{a \in \text{classes}}{\arg\max}\, \hat{\mathbf{d}}_{k,a} \quad C_k \in \text{cut}. \quad (20)$$

4.3.3 Training procedure

Let $\mathcal{F}$ be the set of all feature maps in the training set, and $\mathcal{T}$ the set of all families of segmentations. We construct the segmentation collections $(T)_{T\in\mathcal{T}}$ on the entire training set, and, for all $T \in \mathcal{T}$ train the classifier c to predict the distribution of classes in component $C_k \in T$, as well as the costs S_k.

Given the trained parameters θ_s, we build $\mathcal{F}$ and $\mathcal{T}$, *i.e.* we compute all vector maps $\mathbf{F}$ and segmentation collections T on all the training data available, so as to produce a new training set of descriptors $\mathbf{O}_k$. This time, the parameters θ_c of the classifier c are trained to minimize the KL-divergence between the true (known) distributions of labels $\mathbf{d}_k$ in each component, and the prediction from the classifier $\hat{\mathbf{d}}_k$ (Eq 18):

$$l_{div} = \sum_{a\in\text{classes}} \hat{\mathbf{d}}_{k,a} \ln(\frac{\hat{\mathbf{d}}_{k,a}}{\mathbf{d}_{k,a}}). \tag{21}$$

In this setting, the groundtruth distributions $\mathbf{d}_k$ are not hard target vectors, but normalized histograms of the labels present in component C_k. Once the parameters θ_c are trained, $\hat{\mathbf{d}}_k$ accurately predicts the distribution of labels, and Eq 19 is used to assign a purity cost to the component.

5 Experiments

We report our semantic scene understanding results on three different datasets: "Stanford Background" on which related state-of-the-art methods report classification errors, and two more challenging datasets with a larger number of classes: "SIFT Flow" and "Barcelona". The Stanford Background dataset [15] contains 715 images of outdoor scenes composed of 8 classes, chosen from other existing public datasets so that all the images are outdoor scenes, have approximately 320×240 pixels, where each image contains at least one foreground object. We use the evaluation procedure introduced in [15], 5-fold cross validation: 572 images used for training, and 143 for testing. The SIFT Flow dataset [31] is composed of $2,688$ images, that have been thoroughly labeled by LabelMe users, and split in $2,488$ training images and 200 test images. The authors used synonym correction to obtain 33 semantic labels. The Barcelona dataset, as described in [44], is derived from the LabelMe subset used in [38]. It has $14,871$ training and 279 test images. The test set consists of street scenes from Barcelona, while the training set ranges in scene types but has no street scenes from Barcelona. Synonyms were manually consolidated by [44] to produce 170 unique labels.

To evaluate the representation from our multiscale convolutional network, we report results from several experiments on the Stanford Background dataset: (1) a system based on a plain convolutional network alone; (2) the multiscale convolutional network presented in Section 3.1, with raw pixelwise prediction; (3) superpixel-based predictions, as presented in Section 4.1; (4) CRF-based predictions, as presented in Section 4.2; (5) cover-based predictions, as presented in Section 4.3.

Results are reported in Table 1, and compared with related works. Our model achieves very good results in comparison with previous approaches. Methods of [25], [30] achieve similar or better performances on this particular dataset but to the price of several minutes to parse one image.

	Pixel Acc.	Class Acc.	CT (sec.)
Gould *et al.* 2009 [15]	76.4%	-	10 to 600s
Munoz *et al.* 2010 [32]	76.9%	66.2%	12s
Tighe *et al.* 2010 [44]	77.5%	-	10 to 300s
Socher *et al.* 2011 [43]	78.1%	-	?
Kumar *et al.* 2010 [25]	79.4%	-	$< 600s$
Lempitzky *et al.* 2011 [30]	**81.9%**	72.4%	$> 60s$
singlescale convnet	66.0 %	56.5 %	0.35s
multiscale convnet	78.8 %	72.4%	0.6s
multiscale net + superpixels	80.4%	74.56%	**0.7s**
multiscale net + gPb + cover	80.4%	75.24%	61s
multiscale net + CRF on gPb	81.4%	**76.0**%	60.5s

TABLE 1
Performance of our system on the Stanford Background dataset [15]: per-pixel / average per-class accuracy. The third column reports compute times, as reported by the authors. Our algorithms were computed using a 4-core Intel i7.

	Pixel Acc.	Class Acc.
Liu *et al.* 2009 [31]	74.75%	-
Tighe *et al.* 2010 [44]	76.9%	29.4%
raw multiscale net[1]	67.9%	45.9%
multiscale net + superpixels[1]	71.9%	**50.8%**
multiscale net + cover[1]	72.3%	**50.8%**
multiscale net + cover[2]	**78.5%**	29.6%

TABLE 2
Performance of our system on the SIFT Flow dataset [31]: per-pixel / average per-class accuracy. Our multiscale network is trained using two sampling methods: [1]balanced frequencies, [2]natural frequencies. We compare the results of our multiscale network with the raw (pixelwise) classifier, Felzenszwalb superpixels [11] (one level), and our optimal cover applied to a stack of 10 levels of Felzenszwalb superpixels. Note: the threshold for the single level was picked to yield the best results; the cover automatically finds the best combination of superpixels.

We then demonstrate that our system scales nicely when augmenting the number of classes on two other datasets, in Tables 2 and 3. Results on these datasets were obtained using our cover-based method, from Section 4.3. Example parses on the SIFT Flow dataset are shown on Figure 9.

For the SIFT Flow and Barcelona datasets, we experimented with two sampling methods when learning the multiscale features: respecting natural frequencies of classes, and balancing them so that an equal amount of each class is shown to the network. Balancing class occurrences is essential to model the conditional likelihood of each class (*i.e.* ignore their prior distribution). Both results are reported in Table 2. Training with balanced frequencies allows better discrimination of small objects, and although it decreases the overall pixelwise accuracy, it is more correct from a recognition point of view. Frequency balancing is used on the Stanford Background dataset, as it consistently gives better results. For the Barcelona dataset, both sampling methods are used as well, but frequency balancing worked rather poorly in that case. This can be explained by the fact that this dataset has a large amount of classes with very few training examples. These classes are therefore extremely

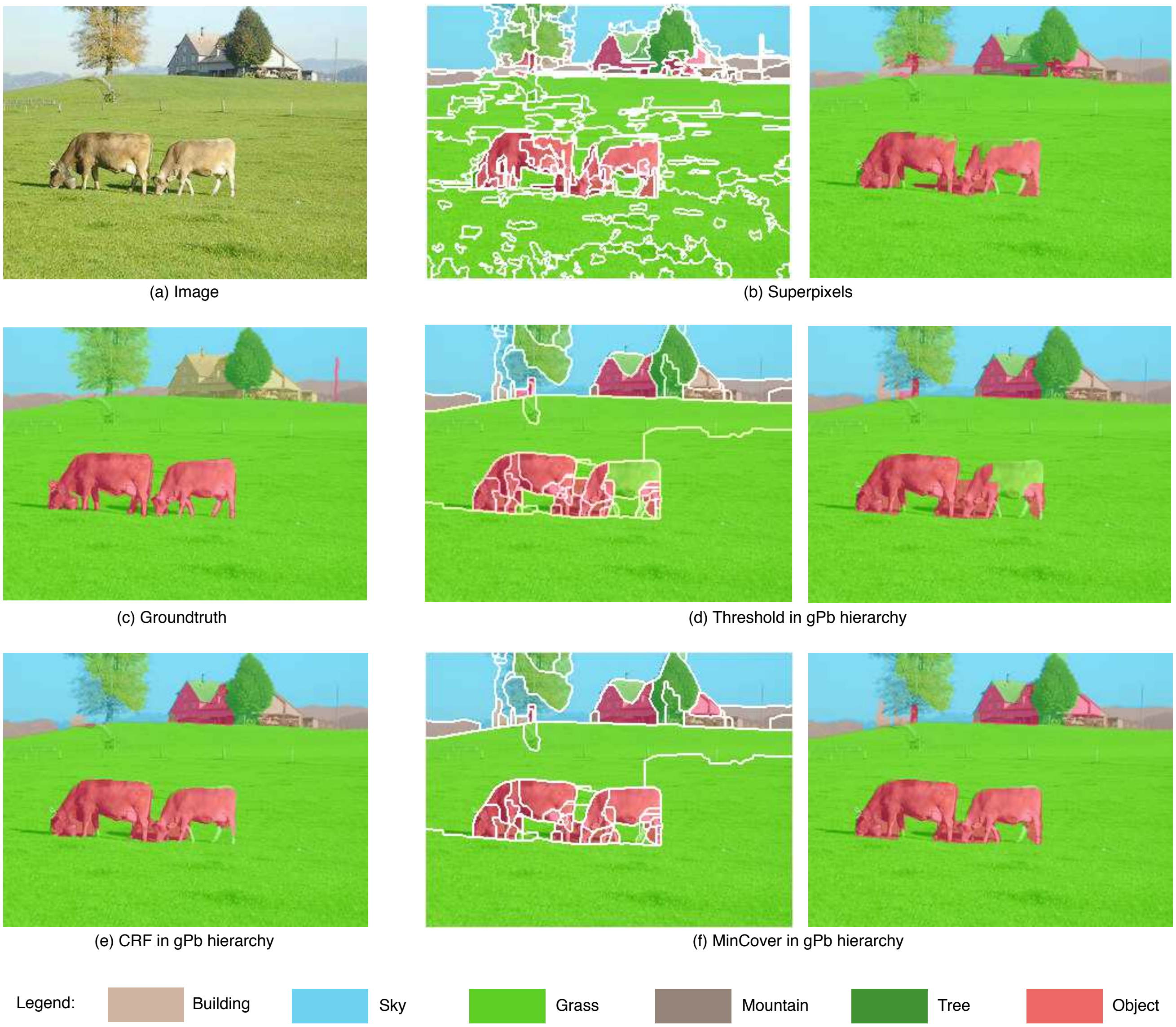

Fig. 7. Example of results on the Stanford background dataset. (b),(d) and (f) show results with different labeling strategies, overlaid with superpixels (cf Section 4.1), segments results of a threshold in the gPb hierarchy [1], and segments recovered by the maximum purity approach with an optimal cover (cf 4.3). The result (c) is obtained with a CRF on the superpixels shown in (d), as described in Section 4.2.

hard to model, and overfitting occurs much faster than for the SIFT Flow dataset. Results are shown on Table 3.

Results in Table 1 demonstrate the impressive computational advantage of convolutional networks over competing algorithms. Exploiting the parallel structure of this special network, by computing convolutions in parallel, allows us to parse an image of size 320×240 in less than one second on a 4-core Intel i7 laptop. Using GPUs or other types of dedicated hardware, our scene parsing model can be run in real-time (*i.e.* at more than 10fps).

5.1 Multiscale feature extraction

For all experiments, we use a 3-stage convolutional network. The first two layers of the network are composed of a bank of filters of size 7×7 followed by $\tanh$ units and 2×2 max-pooling operations. The last layer is a simple filter bank. The filters and pooling dimensions were chosen by a grid search. The input image is transformed into YUV space, and a Laplacian pyramid is constructed from it. The Y, U and V channels of each scale in the pyramid are then independently locally normalized, such that each local 15×15 patch has zero-mean and unit variance. For these experiments, the pyramid consists of 3 rescaled versions of the input ($N = 3$), in octaves: 320×240,160×120, 80×60.

The network is then applied to each 3-dimension input map $\mathbf{X}_s$. This input is transformed into a 16-dimension feature map, using a bank of 16 filters, 10 connected to the Y channel, the 6 others connected to the U and V channels. The second layer transforms this 16-dimension feature map into a 64-dimension feature map, each map being produced by a combination of 8 randomly selected feature maps from the previous layer. Finally the 64-dimension feature map is transformed into

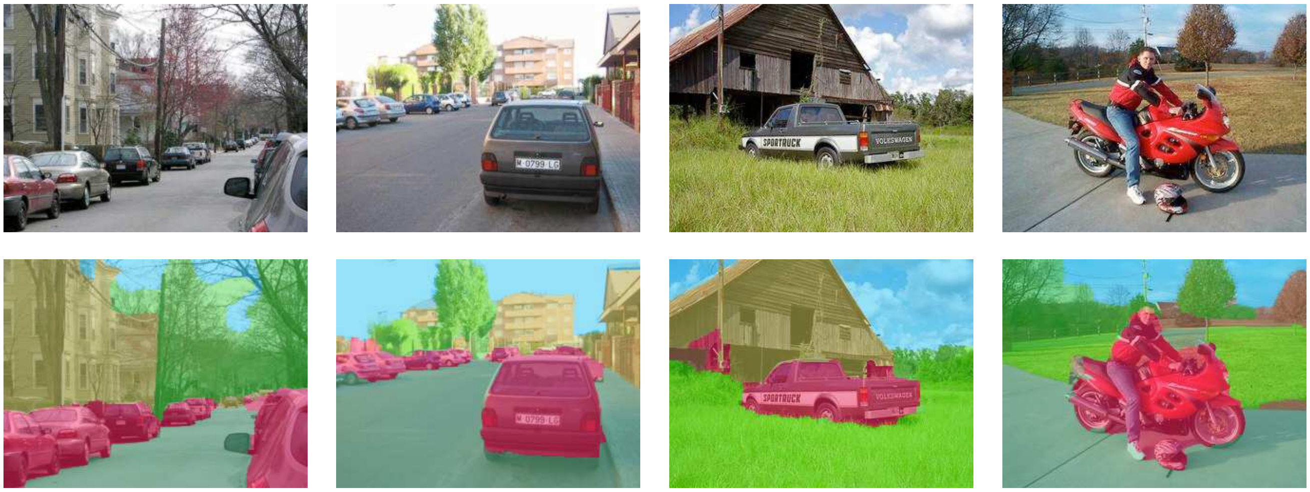

Fig. 8. More results using our multiscale convolutional network and a flat CRF on the Stanford Background Dataset.

	Pixel Acc.	Class Acc.
Tighe *et al.* 2010 [44]	66.9%	7.6%
raw multiscale net[1]	37.8%	**12.1%**
multiscale net + superpixels[1]	44.1%	**12.4%**
multiscale net + cover[1]	46.4%	**12.5%**
multiscale net + cover[2]	**67.8%**	**9.5%**

TABLE 3
Performance of our system on the Barcelona dataset [44]: per-pixel / average per-class accuracy. Our multiscale network is trained using two sampling methods: [1]balanced frequencies, [2]natural frequencies. We compare the results of our multiscale network with the raw (pixelwise) classifier, Felzenszwalb superpixels [11] (one level), and our optimal cover applied to a stack of 10 levels of Felzenszwalb superpixels. Note: the threshold for the single level was picked to yield the best results; the cover automatically finds the best combination of superpixels.

a 256-dimension feature map, each map being produced by a combination of 32 randomly selected feature maps from the previous layer.

The outputs of each of the 3 networks are then upsampled and concatenated, so as to produce a $256 \times 3 = 768$-dimension feature vector map $\mathbf{F}$. Given the filter sizes, the network has a field of view of 46×46, at each scale, which means that a feature vector in $\mathbf{F}$ is influenced by a 46×46 neighborhood at full resolution, a 92×92 neighborhood at half resolution, and a 184×184 neighborhood at quarter resolution. These neighborhoods are shown in Figure 1.

The network is trained on all 3 scales in parallel, using stochastic gradient descent with no second-order information, and mini-batches of size 1. Simple grid-search was performed to find the best learning rate (10^{-3}) and regularization parameters (L2 coefficient: 10^{-5}), using a holdout of 10% of the training data for validation. The holdout is also used to select the best network, *i.e.* the network that generalizes the most on the holdout.

Convergence, that is, maximum generalization performance, is typically attained after between 10 to 50 million patches have been seen during stochastic gradient descent. This typically represents between two to five days of training. No special hardware (GPUs) was used for training.

The convolutional network has roughly 0.5 million trainable parameters. To ensure that features do not overfit some irrelevant biases present in the data, jitter – horizontal flipping of all images, rotations between -8 and 8 degrees, and rescaling between 90 and 110% – was used to artificially expand the size of the training data. These additional distortions are applied during training, before loading a new training point, and are sampled from uniform distributions. Jitter was shown to be crucial for low-level feature learning in the works of [42] and [6].

For our baseline, we trained a single-scale network and a three-scale network as raw site predictors, for each location i, using the classification loss L_{cat} defined in Eq 10, with the two-layer neural network defined in Eq 9. Table 1 shows the clear advantage of the multi-scale representation, which captures scene-level dependencies, and can classify more pixels accurately. Without an explicit segmentation model, the visual aspect of the predictions still suffers from inaccurate object delineation.

5.2 Parsing with superpixels

The results obtained with the strategy presented in section 4.1 demonstrate the quality of our multiscale features, by reaching a very high classification accuracy on all three datasets. This simple strategy is also a real fit for real time applications, taking only an additional 0.2 second to label a 320×240 image on Intel i7 CPU. An example of result is given in Figure 7.

The $2-$layer neural network used for this method (Eq 9) has 768 input units, 1024 hidden units; and as many output units as classes in each dataset. This neural network is trained with no regularization.

5.3 Multilevel parsing

Although the simple strategy of the previous section seems appealing, the results can be further improved using the multilevel approach of Section 4.3.

The family of segmentations used to find the optimal cover could be a simple segmentation tree constructed on the raw image gradient. For the Stanford Background dataset experiments, we used a more sophisticated tree based on a semantic image gradient. We used the gPb hierarchies of Arbelaez *et al.* , which are computed using spectral clustering to produce semantically consistent contours of objects. Their computation requires one minute per image.

For the SIFT Flow and Barcelona datasets, we used a cheaper technique, which does not rely on a tree: we ran the superpixel method proposed by Felzenszwalb in [11] at 10 different levels. The Felzenszwalb algorithm is not strictly monotonic, so the structure obtained cannot be cast into a tree: rather, it has a general graph form, in which each pixel belongs to 10 different superpixels. Our optimal cover algorithm can be readily applied to arbitrary structures of this type. The 10 levels were chosen such that they are linearly distributed and span a large range.

Classically, segmentation methods find a partition of the segments rather than a cover. Partitioning the segments consists in finding an optimal cut in a tree (so that each terminal node in the pruned tree corresponds to a segment). We experimented with graph-cuts to do so [12], [2], but the results were less accurate than with our optimal cover method (Stanford Background dataset only).

The $2-$layer neural network c from Eq 17 has $3 \times 3 \times 768 = 6912$ input units (using a 3×3 grid of feature vectors from $\mathbf{F}$), 1024 hidden units; and as many output units as classes in each dataset. This rather large neural network is trained with L2 regularization (coefficient: 10^{-2}), to minimize overfitting.

Results are better than the superpixel method, in particular, better delineation is achieved (see Fig. 7).

5.4 Conditional random field

We demonstrate the state-of-the-art quality of our features by employing a CRF on the superpixels given by thresholding the gPb hierarchy, on the Stanford Background dataset. A similar test is performed in Lempitsky *et al.* [30], where the authors also use a CRF on the same superpixels (at the threshold 20 in the gPb hierarchy), but employ different features. Histograms of densely sampled SIFT words, colors, locations, and contour shape descriptors. They report a ratio of correctly classified pixels of 81.1% on the Stanford Background dataset. We recall that this accuracy is the best one has achieved at the present day on this dataset with a flat CRF.

In our CRF energy, we performed a grid search to set the parameters of (13) ($\beta = 20$, $\alpha = 0.1$ $\gamma = 200$), and used a grey level gradient. The accuracy of the resulting system is 81.4, as reported in Table 1. Our features are thus outperforming the best publicly available combination of handcrafted features.

5.5 Some comments on the learned features

With recent advances in unsupervised (deep) learning, learned features have become easier to analyze and understand. In this work, the entire stack of features is learned in a purely supervised manner, and yet we found that the features obtained are rather meaningful. We believe that the reason for this is the type of loss function we use, which enforces a large invariance: the system is forced to produce an invariant representation for all the locations of a given object. This type of invariance is very similar to what can be achieved using semi-supervised techniques such as Dr-LIM [18], where the loss enforces pairs of similar patches to yield a same encoding. Figure 10 shows an example of the features learned on the SIFT Flow dataset.

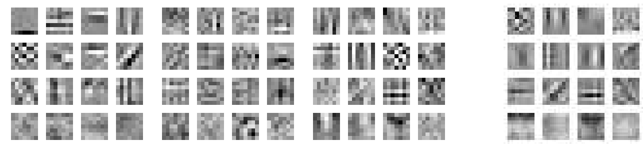

Fig. 10. Typical first layer features, learned on the SIFT Flow dataset. (a) to (c) show the 16 filters learned at each scale, when no weight sharing is used (networks at each scale are independent). (d) show the 16 filters obtained when sharing weights across all 3 scales. All the filters are 7×7. We observe typical oriented edges, and high-frequency filters. Filters at higher layers are more difficult to analyze.

5.6 Some comments on real-world generalization

Now that we have compared and discussed several strategies for scene parsing based on our multiscale features, we consider taking our system in the real-world, to evaluate its generalization properties. The work of [45], measuring dataset bias, raises the question of the *generalization* of a recognition system learned on specific, publicly available datasets.

We used our multiscale features combined with classification using superpixels as described in Section 4.1, trained on the SiftFlow dataset (2,688 images, most of them taken in non-urban environments, see Table 2 and Figure 9). We collected a 360 degree movie in our workplace environment, including a street and a park. introducing difficulties such as lighting conditions and image distortions: see Figure 11.

The movie was built from four videos that were stitched to form a 360 degree video stream of 1280×256 images, thus creating artifacts not seen during training. We processed each frame independently, without using any temporal consistency or smoothing.

Despite all these constraints, and the rather small size of the training dataset, we observe rather convincing generalization of our models on these previously unseen scenes. The two video sequences are available at http://www.clement.farabet.net/. Two snapshots are included in Figure 11. Our scene parsing system constitutes at the best of our knowledge the first approach achieving real time performance, one frame being processed in less than a second on a 4-core Intel i7. Feature extraction, which represent around 500ms on the i7 can be reduced to 60ms using dedicated FPGA hardware [9], [10].

6 Discussion

The main lessons from the experiments presented in this paper are as follows:

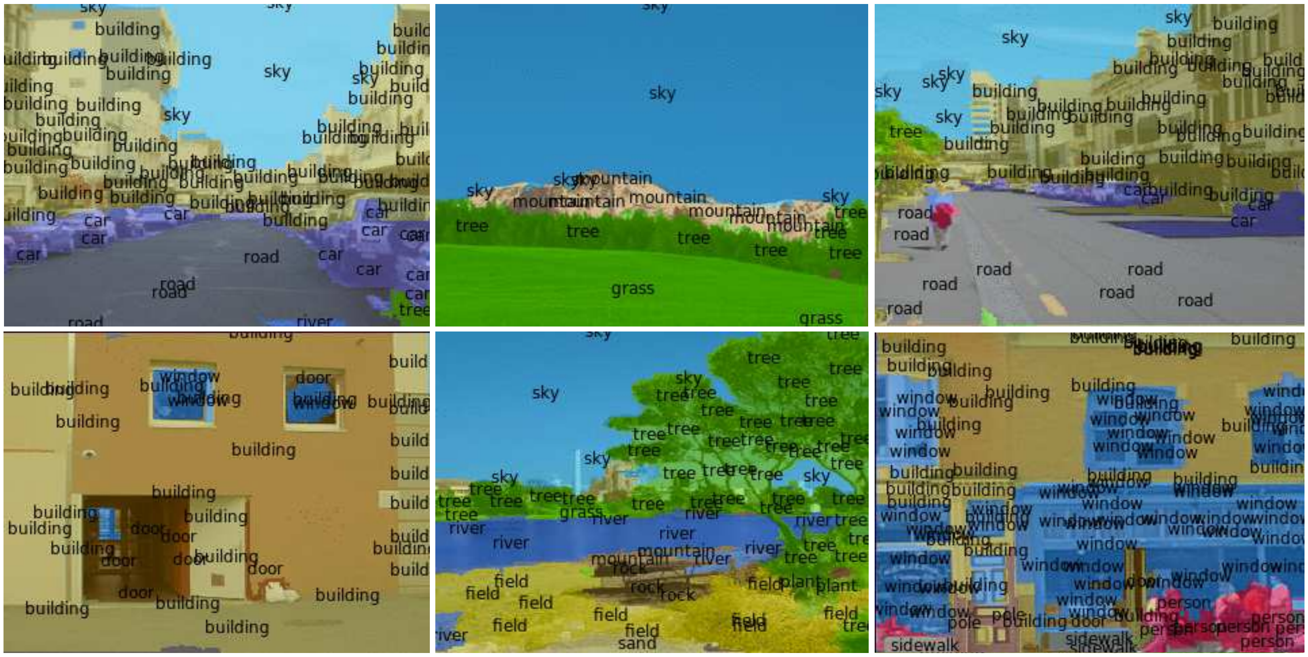

Fig. 9. Typical results achieved on the SIFT Flow dataset.

- Using a high-capacity feature-learning system fed with raw pixels yields excellent results, when compared with systems that use engineered features. The accuracy is similar or better than competing systems, even when the segmentation hypothesis generation and the post-processing module are absent or very simple.
- Feeding the system with a wide contextual window is critical to the quality of the results. The numbers in table 1 show a dramatic improvement of the performance of the multi-scale convolutional network over the single scale version.
- When a wide context is taken into account to produce each pixel label, the role of the post-processing is greatly reduced. In fact, a simple majority vote of the categories within a superpixel yields state-of-the-art accuracy. This seems to suggest that contextual information can be taken into account by a feed-forward trainable system with a wide contextual window, perhaps as well as an inference mechanism that propagates label constraints over a graphical model, but with a considerably lower computational cost.
- The use of highly sophisticated post-processing schemes, which seem so crucial to the success of other models, does not seem to improve the results significantly over simple schemes. This seems to suggest that the performance is limited by the quality of the labeling, or the quality of the segmentation hypotheses, rather than by the quality of the contextual consistency system or the inference algorithm.
- Relying heavily on a highly-accurate feed-forward pixel labeling system, while simplifying the post-processing module to its bare minimum cuts down the inference times considerably. The resulting system is dramatically faster than those that rely heavily on graphical model inference. Moreover, the bulk of the computation takes place in the convolutional network. This computation is algorithmically simple, easily parallelizable. Implementations on multi-core machines, general-purpose GPUs, Digital Signal Processors, or specialized architectures implemented on FPGAs is straightforward. This is demonstrated by the FPGA implementation [9], [10] of the feature extraction scheme presented in this paper that runs in 60ms for an image resolution of 320×240.

7 Conclusion and Future Work

This paper demonstrates that a feed-forward convolutional network, trained end-to-end in a supervised manner, and fed with raw pixels from large patches over multiple scales, can produce state of the art performance on standard scene parsing datasets. The model does not rely on engineered features, and uses purely supervised training from fully-labeled images to learn appropriate low-level and mid-level features.

Perhaps the most surprising results is that even in the absence of any post-processing, by simply labeling each pixel with the highest-scoring category produced by the convolutional net for that location, the system yields near state-of-the-art pixel-wise accuracy, and better per-class accuracy than all previously-published results. Feeding the features of the convolutional net to various sophisticated schemes that generate segmentation hypotheses, and that find consistent segmentations and labeling by taking local constraints into account improves the results slightly, but not considerably.

While the results on datasets with few categories are good, the accuracy of the best existing scene parsing systems, including ours, is still quite low when the number of categories is large. The problem of scene parsing is far from being solved. While the system presented here has a number of advantages and shortcomings, the framing of the scene parsing task itself is in need of refinement.

Fig. 11. Real-time scene parsing in natural conditions. Training on SiftFlow dataset. We display one label per component in the final prediction.

First of all, the pixel-wise accuracy is a somewhat inaccurate measure of the visual and practical quality of the result. Spotting rare objects is often more important than accurately labeling every boundary pixel of the sky (which are often in greater number). The average per-class accuracy is a step in the right direction, but not the ultimate solution: one would prefer a system that correctly spots every object or region, while giving an approximate boundary to a system that produces accurate boundaries for large regions (sky, road, grass), but fail to spot small objects. A reflection is needed on the best ways to measure the accuracy of scene labeling systems.

Scene parsing datasets also need better labels. One could imagine using scene parsing datasets with hierarchical labels, so that a window within a building would be labeled as "building" and "window". Using this kind of labeling in conjunction with graph structures on sets of labels that contain `is-part-of` relationships would likely produce more consistent interpretations of the whole scene.

The framework presented in this paper trains the convolutional net as a pixel labeling system in isolation from the post-processing module that ensures the consistency of the labeling and its proper registration with the image regions. This requires that the convolutional net be trained with images that are fully labeled at the pixel level. One would hope that jointly fine-tuning the convolutional net and the post-processor produces better overall interpretations. Gradients can be back-propagated through the post-processor to the convolutional nets. This is reminiscent of the Graph Transformer Network model, a kind of non-linear CRF in which an un-normalized graphical model based post-processing module was trained jointly with a convolutional network for handwriting recognition [27]. Unfortunately, preliminary experiments with such joint training yielded lower test-set accuracies due to overtraining.

A more importantly advantage of joint training would allow the use of weakly-labeled images in which only a list of objects present in the image would be given, perhaps tagged with approximate positions. This would be similar in spirit to sentence-level discriminative training methods used in speech recognition and handwriting recognition [27].

Another possible direction for improvement includes the use of objective functions that directly operates of the edge costs of neighborhood graphs in such as way that graph-cut segmentation and similar methods produce the best answer. One such objective function is is Turaga's Maximin Learning [46], which pushes up the lowest edge cost along the shortest path between two points in different segments, and pushes down the highest edge cost along a path between two points in the same segment.

Our system so far has been trained using purely supervised learning applied to a fairly classical convolutional network architecture. However, a number of recent works have shown the advantage of architectural elements such as rectifying non-linearities and local contrast normalization [21]. More importantly, several works have shown the advantage of using unsupervised pre-training to prime the convolutional net into a good starting point before supervised refinement [37], [22], [23], [29], [24]. These methods improve the performance in the low training set size regime, and would probably improve the performance of the present system.

Finally, code and data are available online at http://www.clement.farabet.net/.

Acknowledgment

We would like to thank Marco Scoffier for fruitful discussions and the 360 degree video collection. We are also grateful to Victor Lempitsky who kindly provided us with his results on the Stanford Database for comparison.

This work was funded in part by DARPA contract "Integrated deep learning for large scale multi-modal data representation", ONR MURI "Provably-stable vision-based control of high-speed flight", ONR grant "Learning Hierarchical Models for Information Integration".

References

[1] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour Detection and Hierarchical Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, 2011. 6, 9

[2] Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *Proceedings of International Conference of Computer Vision (ICCV)*, volume 1, pages 105–112, 2001. 11

[3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, 2004. 5

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001. 5

[5] J. Carreira and C. Sminchisescu. CPMC: Automatic Object Segmentation Using Constrained Parametric Min-Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012. 2

[6] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber. A committee of neural networks for traffic sign classification. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1918–1921. IEEE, 2011. 10

[7] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Scene parsing with multiscale feature learning, purity trees, and optimal covers. In *Proceedings of the International Conference on Machine Learning (ICML)*, June 2012. 2, 6

[8] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Scene parsing with multiscale feature learning, purity trees, and optimal covers. *CoRR*, abs/1202.2160, February 2012. 3

[9] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello. Hardware accelerated convolutional neural networks for synthetic vision systems. In *International Symposium on Circuits and Systems (ISCAS'10)*, Paris, May 2010. IEEE. 11, 12

[10] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun. Neuflow: A runtime reconfigurable dataflow processor for vision. In *Proceedings of the Fifth IEEE Workshop on Embedded Computer Vision*. IEEE, 2011. 11, 12

[11] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:167–181, 2004. 5, 7, 8, 10, 11

[12] L. R. Ford and D. R. Fulkerson. A simple algorithm for finding maximal network flows and an application to the hitchcock problem. Technical report, RAND Corp., Santa Monica, 1955. 11

[13] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *ICCV*, pages 670–677. IEEE, 2009. 5

[14] C. Garcia and M. Delakis. Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004. 3

[15] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. *IEEE International Conference on Computer Vision*, pages 1–8, Sept. 2009. 2, 8

[16] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller. Multi-class segmentation with relative location prior. *Int. J. Comput. Vision*, 80(3):300–316, Dec. 2008. 5

[17] D. Grangier, L. Bottou, and R. Collobert. Deep Convolutional Networks for Scene Parsing. In *ICML 2009 Deep Learning Workshop*, 2009. 3

[18] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'06)*. IEEE Press, 2006. 11

[19] X. He and R. Zemel. Learning hybrid models for image annotation with partially labeled data. *Advances in Neural Information Processing Systems*, 2008. 2

[20] V. Jain, J. F. Murray, F. Roth, S. Turaga, V. Zhigulin, K. Briggman, M. Helmstaedter, W. Denk, and S. H. Seung. Supervised learning of image restoration with convolutional networks. In *ICCV*, 2007. 3

[21] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *Proc. International Conference on Computer Vision (ICCV'09)*. IEEE, 2009. 3, 14

[22] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun. Learning invariant features through topographic filter maps. In *Proc. International Conference on Computer Vision and Pattern Recognition*. IEEE, 2009. 14

[23] K. Kavukcuoglu, M. Ranzato, and Y. LeCun. Fast inference in sparse coding algorithms with applications to object recognition. Technical report, Courant Institute of Mathematical Sciences, New York University, 2008. Tech Report CBLL-TR-2008-12-01. 14

[24] K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning convolutional feature hierachies for visual recognition. In *Advances in Neural Information Processing Systems (NIPS 2010)*, volume 23, 2010. 14

[25] M. Kumar and D. Koller. Efficiently selecting regions for scene understanding. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3217–3224. IEEE, 2010. 2, 8

[26] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS'89*, 1990. 4

[27] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. 1, 2, 4, 14

[28] Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In G. Orr and M. K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998. 4

[29] H. Lee, R. Grosse, R. Ranganath, and Y. Ng, Andrew. Convolu-

tional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proc. of International Conference on Machine Learning (ICML'09)*, 2009. 14

[30] V. Lempitsky, A. Vedaldi, and A. Zisserman. A pylon model for semantic segmentation. In *Advances in Neural Information Processing Systems*, 2011. 2, 8, 11

[31] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. *Artificial Intelligence*, 2009. 8

[32] D. Munoz, J. Bagnell, and M. Hebert. Stacked hierarchical labeling. *ECCV 2010*, Jan 2010. 2, 8

[33] L. Najman and M. Schmitt. Geodesic saliency of watershed contours and hierarchical segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(12):1163–1173, December 1996. 6

[34] F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P. Barbano. Toward automatic phenotyping of developing embryos from videos. *IEEE Trans. on Image Processing*, 2005. Special issue on Molecular & Cellular Bioimaging. 3

[35] M. Osadchy, Y. LeCun, and M. Miller. Synergistic face detection and pose estimation with energy-based models. *Journal of Machine Learning Research*, 8:1197–1215, 2007. 3

[36] C. Pantofaru, C. Schmid, and M. Hebert. Object recognition by integrating multiple image segmentations. In *ECCV 2008, 10th European Conference on Computer Vision, Marseille, France*, pages 481–494, 2008. 2

[37] M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. of Computer Vision and Pattern Recognition*. IEEE, 2007. 14

[38] B. Russell, A. Torralba, C. Liu, R. Fergus, and W. Freeman. Object recognition by scene alignment. In *Neural Advances in Neural Information*, 2007. 8

[39] C. Russell, P. H. S. Torr, and P. Kohli. Associative hierarchical CRFs for object class image segmentation. In *Proc. ICCV*, 2009. 2

[40] H. Schulz and S. Behnke. Learning object-class segmentation with convolutional neural networks. In *11th European Symposium on Artificial Neural Networks (ESANN)*, 2012. 3

[41] J. Shotton, J. M. Winn, C. Rother, and A. Criminisi. *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In A. Leonardis, H. Bischof, and A. Pinz, editors, *ECCV (1)*, volume 3951 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2006. 5

[42] P. Simard, D. Steinkraus, and J. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, volume 2, pages 958–962, 2003. 10

[43] R. Socher, C. C. Lin, A. Y. Ng, and C. D. Manning. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2011. 2, 8

[44] J. Tighe and S. Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. *ECCV*, pages 352–365, 2010. 2, 8, 10

[45] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR*, pages 1521–1528. IEEE, 2011. 11

[46] S. Turaga, K. Briggman, M. Helmstaedter, W. Denk, and H. Seung. Maximin affinity learning of image segmentation. *NIPS*, Jan 2009. 3, 14

[47] R. Vaillant, C. Monrocq, and Y. LeCun. Original approach for the localisation of objects in images. *IEE Proc on Vision, Image, and Signal Processing*, 141(4):245–250, August 1994. 3

Clément Farabet received a Master's Degree in Electrical Engineering with honors from Institut National des Sciences Appliques (INSA) de Lyon, France in 2008. His Master's thesis work was developed at the Courant Institute of Mathematical Sciences of New York University with Prof Yann LeCun. He then joined Prof Yann LeCun's laboratory in 2008, as a research scientist. In 2009, he started collaborating with Yale University's e-Lab, led by Prof Eugenio Culurciello. In 2010, he started the PhD program at Université Paris-Est, with Prof Laurent Najman, in parallel with his research work at Yale and NYU. His research interests include intelligent hardware, embedded super-computers, computer vision, machine learning, embedded robotics, and more broadly artificial intelligence. His current work aims at developing a massively-parallel yet low-power processor for general-purpose vision. Algorithmically, most of this work is based on Prof Yann LeCun's Convolutional Networks, while the hardware has its roots in dataflow computers and architectures as they first appeared in the 1960s.

Camille Couprie earned a PhD in computer science at the Université Paris Est, France in 2011 after an engineer degree at ESIEE Paris with the highest honors in 2008. Her PhD, advised by Professors Laurent Najman and Hugues Talbot, was supported by the French Direction Générale de l'Armement MRIS program and the Centre National de la Recherche Scientifique. Since Autumn 2011, she is a postdoctoral researcher at the Courant Institute of Mathematical Sciences, in the Computer Science department with Professor Yann Lecun. Her research focuses on image segmentation, optimization techniques, graph theory, PDE, mathematical morphology and machine learning. Other interests include stereo vision, image registration, medical imaging and topology.

Laurent Najman received the Habilitation à Diriger les Recherches in 2006 from University the University of Marne-la-Vallée, a Ph.D. of applied mathematics from Paris-Dauphine University in 1994 with the highest honor (Félicitations du Jury) and an "Ingénieur" degree from the Ecole des Mines de Paris in 1991. After earning his engineering degree, he worked in the central research laboratories of Thomson-CSF for three years, working on some problems of infrared image segmentation using mathematical morphology. He then joined a start-up company named Animation Science in 1995, as director of research and development. The technology of particle systems for computer graphics and scientific visualisation, developed by the company under his technical leadership received several awards, including the "European Information Technology Prize 1997" awarded by the European Commission (Esprit programme) and by the European Council for Applied Science and Engineering and the "Hottest Products of the Year 1996" awarded by the Computer Graphics World journal. In 1998, he joined OC Print Logic Technologies, as senior scientist. He worked there on various problem of image analysis dedicated to scanning and printing. In 2002, he joined the Informatics Department of ESIEE, Paris, where he is professor and a member of the Gaspard-Monge computer science research laboratory (LIGM), Université Paris-Est Marne-la-Vallée. His current research interest is discrete mathematical morphology.

Yann Lecun is Silver Professor of Computer Science and Neural Science at the Courant Institute of Mathematical Sciences and the Center for Neural Science of New York University. He received the Electrical Engineer Diploma from Ecole Supérieure d'Ingénieurs en Electrotechnique et Electronique (ESIEE), Paris in 1983, and a PhD in Computer Science from Université Pierre et Marie Curie (Paris) in 1987. After a postdoc with Geoffrey Hinton at the University of Toronto, he joined AT&T Bell Laboratories in Holmdel, NJ, in 1988, and became head of the Image Processing Research Department at AT&T Labs-Research in 1996. He joined NYU as a professor in 2003, after a brief period as Fellow at the NEC Research Institute in Princeton. His current interests include machine learning, computer perception and vision, mobile robotics, and computational neuroscience. He has published over 140 technical papers and book chapters on these topics as well as on neural networks, handwriting recognition, image processing and compression, and VLSI design. His handwriting recognition technology is used by several banks around the world to read checks. His image compression technology, called DjVu, is used by hundreds of web sites and publishers and millions of users to access scanned documents on the Web, and his image recognition methods are used in deployed systems by companies such as Google, Microsoft, NEC, France Telecom and several startup companies for document recognition, human-computer interaction, image indexing, and video analytics. He has been on the editorial board of IJCV, IEEE PAMI, IEEE Trans on Neural Networks, was program chair of CVPR'06, and is chair of the annual Learning Workshop. He is on the science advisory board of Institute for Pure and Applied Mathematics, and is the co-founder of MuseAmi, a music technology company.