Large-Scale Image Segmentation with Convolutional Networks

THÈSE N^{O} 7571 (2017)

PRÉSENTÉE LE 17 FÉVRIER 2017

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

LABORATOIRE DE L'IDIAP

PROGRAMME DOCTORAL EN GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Pedro Henrique OLIVEIRA PINHEIRO

acceptée sur proposition du jury:

Prof. J.-Ph. Thiran, président du jury
Prof. H. Bourlard, Dr R. Collobert, directeurs de thèse
Dr C. Schmid, rapporteuse
Prof. R. Fergus, rapporteur
Dr M. Salzmann, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2017

"It irritated him that the "dog" of 3:14 in the
afternoon, seen in profile, should be indicated by
the same noun as the dog of 3:15, seen frontally..."
"My memory, sir, is like a garbage heap"
— from *Funes the Memorious*, Jorge Luis Borges

To my parents...

Abstract

Object recognition is one of the most important problems in computer vision. However, visual recognition poses many challenges when tried to be reproduced by artificial systems. A main challenge is the problem of variability: objects can appear across huge variations in pose, appearance, illumination and occlusion, and a visual system need to be robust to all these changes. In the present thesis, we are interested in pixel-level recognition problems, *i.e.*, problems in which the objective is to partition a given image into multiple regions (overlapping or not) that are considered meaningful according to some criterion.

Our interests are in algorithms that require the least amount of feature engineering and are easy to scale. Deep learning methods fit very well with this objective: these models alleviate the need of engineered features by discriminatively training a system from raw data (pixels). More precisely, we propose different convolutional neural network (CNN) based algorithms to deal with three important segmentation problems: semantic segmentation, object proposal generation and object detection with segments.

The objective of semantic segmentation is to generate a categorical label to each pixel present in a scene. We first study the problem of fully supervised semantic segmentation. We propose a recurrent CNN that is able to consider a large input context (while limiting its capacity), which is essential to model long range pixel label dependencies. This approach achieves state-of-the-art performance without relying on any post-processing smoothing step. However, having densely labeled images to train a model can be expensive and require a lot of human labor. We also propose a CNN-based model that is able to infer object semantic segmentation by leveraging only the object category information from images. This is achieved by casting the problem into a multiple instance learning framework. This approach beats previous state of the art in weakly supervised semantic segmentation by a large margin.

Object proposal algorithms generate a set of regions (segments) that are likely to contain objects, independent of their semantic category. Contrary to most approaches (which rely on low-level vision cues), we propose a CNN-based discriminative approach that is able to learn segmentation proposals from raw pixels. This approach is proven to be quite effective in this setting, achieving substantially higher recall using fewer proposals than other methods. The state of the art is pushed further with the introduction of a new top-down network augmentation. The resulting bottom-up/top-down network combines low-level rich spatial information with high-level object semantic information to improve segmentation, while remaining fast at test time.

Finally, we show that the proposals generated by our approach, when coupled with a standard state-of-the-art object detection pipeline, achieve considerably better performance than previous proposals methods.

Key words: object recognition, artificial neural networks, deep learning, semantic segmentation, object proposals, object detection, image segmentation.

Résumé

La reconnaissance d'objets est l'un des problèmes les plus importants de la vision par ordinateur. Cependant, la reconnaissance visuelle pose de nombreux défis lorsqu'on essaie de la reproduire par des systèmes artificiels. Un défi important est le problème de la variabilité : les objets peuvent apparaître à travers d'énormes variations de pose, d'apparence, d'illumination et d'occlusion, et un système visuel doit être robuste à tous ces changements. Dans la présente thèse, nous nous intéressons à des problèmes de reconnaissance au niveau des pixels, c'est-à-dire des problèmes dans lesquels l'objectif est de partitionner une image donnée en plusieurs régions (chevauchantes ou non) considérées comme significatives selon un certain critère.

Nos intérêts sont dans des algorithmes qui nécessitent le moins d'ingénierie de fonctionnalité et sont faciles à mettre en échelle. Les méthodes d'apprentissage profond s'accordent très bien avec cet objectif : ces modèles allègent le besoin de fonctionnalités techniques en formant discriminativement un système à partir de données brutes (pixels). Plus précisément, nous proposons différents algorithmes basés sur le réseau neuronal convolutif (CNN) pour traiter trois problèmes de segmentation importants : la segmentation sémantique, la génération de propositions d'objets et la détection d'objets avec des segments.

L'objectif de la segmentation sémantique est de générer un label catégorique pour chaque pixel présent dans une scène. Nous étudions d'abord le problème de la segmentation sémantique entièrement supervisée. Nous proposons un CNN récurrent capable de considérer un contexte d'entrée important (tout en limitant sa capacité), ce qui est essentiel pour modéliser les dépendances d'étiquettes de pixel à longue portée. Cette approche permet d'obtenir des performances de pointe sans compter sur une étape de lissage après-traitement. Cependant, avoir des images fortement marquées pour entrainer un modèle peut être coûteux. Nous proposons également un modèle basé sur CNN qui est capable d'inférer la segmentation sémantique d'objets en utilisant uniquement les informations de catégorie d'objet à partir d'images. Ceci est réalisé en jetant le problème dans le cadre d'apprentissage à instance multiple. Cette approche dépasse largement l'état de l'art dans le cadre de segmentation sémantique faiblement supervisée.

Les algorithmes de proposition d'objet génèrent un ensemble de régions (segments) susceptibles de contenir des objets, indépendamment de leur catégorie sémantique. Contrairement à la plupart des approches (qui s'appuient sur des indices de vision de bas niveau), nous proposons une approche discriminative basée sur CNN capable d'apprendre des propositions de segmentation à partir de pixels bruts. Cette approche s'est révélée très efficace dans ce contexte, ce qui a permis d'obtenir un rappel beaucoup plus important en utilisant moins de propositions que d'autres méthodes. L'état de l'art est poussé plus loin avec l'introduction d'une nouvelle augmentation de réseau de haut en bas. Le réseau ascendant / descendant résultant combine des informations spatiales riches de faible niveau avec des informations sémantiques d'objet de haut niveau pour améliorer la segmentation tout en restant rapide en inference.

Enfin, nous montrons que les propositions générées par notre approche, associées à un pipeline de

détection d'objets de pointe, atteignent des performances nettement meilleures que les méthodes de propositions précédentes.

Mots clefs : reconnaissance d'objet, réseaux de neurone artificielle, apprentissage profond, segmentation sémantique, proposition d'objets, détection d'objet, segmentation d'image.

Acknowledgements

First of all, I would like to thanks my two mentors throughout my PhD: Ronan Collobert and Piotr Dollár. I specially thank Ronan for introducing me to the research world (by first offering me an internship then the doctoral studies) and all the teaching in machine (and deep) learning. I specially thank Piotr for the invaluable help and support and all the computer vision insights I learned. I also would like to thank Prof. Hervé Bourlard for making Idiap such a nice place to work and providing support when needed, Nadine and Sylvie for always being very helpful in any bureaucratic need. I am also very grateful for NCCR-IM2, Idiap and Facebook for financial support.

I am also very grateful for my thesis committee for their time and dedication: Jean-Philippe Thiran, Mathieu Salzmann, Cordelia Schmid and Rob Fergus. Thank you very much for the insightful comments and discussion.

A special thanks goes to my friends of the AML group at Idiap: Dimitri, Rémi and Joel. Thank you so much for all the neural net tricks, coffee breaks, geeky discussions and help in improving my French skills. I would also like to give a special thank to Özgün, for being around during the thesis, providing good laughs, help and support whenever needed.

Finally, I would like to thank my family for all the love and inspiration throughout so many years. Thank you for supporting my decision of moving to Europe to pursue my dream when I was so young and naive. Without you, I would never have been here now.

Lausanne, 20 January 2017 Pedro O. Pinheiro

Contents

List of Figures

List of Tables

List of Abbreviations

ANN Artificial Neural Network.

AP Average Precision.

AR Average Recall.

AUC Area Under Curve.

BPTT Backpropagation Through Time.

CNN Convolutional Neural Network.

CRF Conditional Random Fields.

DPM Deformable Part Model.

GMIM Generalized Multi-Image Model.

HOG Histogram of Oriented Gradients.

ILP Image-Level Prior.

IoU Intersection over Union.

LSE Log-Sum-Exp.

MCG Multiscale Combinatorial Grouping.

MIL Multiple Instance Learning.

MIM Multi-Image Model.

MLP Multilayer Perceptron.

MPN MultiPath Network.

MRF Markov Random Fields.

PGC Probabilistic Graphlet Cut.

R-CNN Regions with Convolutional Neural Networks.

ReLU Rectified Linear Unit.

ResNet Residual Network.

RNN Recurrent Neural Network.

RoI Region of Interest.

RPN Region Proposal Networks.

SDS Simultaneous Detection and Segmentation.

SGD Stochastic Gradient Descent.

SIFT Scale-Invariant Feature Transformation.

SP Smoothing Prior.

SURF Speeded-Up Robust Features.

SVM Support Vector Machine.

1 Introduction

1.1 Overview

Computer vision is the science of endowing computing machines with visual perception, that is, the ability to see. Palmer (1999) defines *visual perception* as being the process of acquiring knowledge about environmental objects and events by extracting information from the light they emit or reflect (through any optical device such as an eye or a camera). By this definition, vision can be interpreted as the ability to model the perceivable world and achieve high-level understanding of it. Understanding in this context means transforming visual information into descriptions of the world.

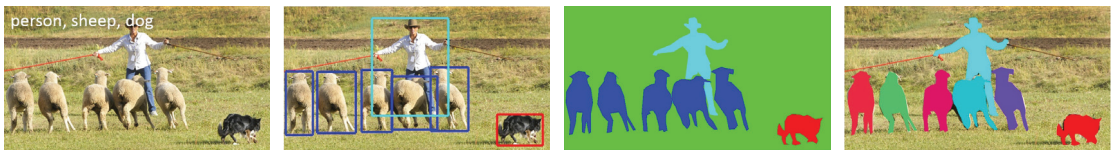
A central problem in computer vision is that of *object recognition.* Humans possess a remarkable ability to parse an image (or many images) simply by looking at them. In a blink of an eye, we are able to fully analyze an image and separate all the components present on it. Furthermore, humans can easily generalize from observing a set of objects to recognizing objects that have never been seen before. Nevertheless, it has been proved particularly difficult to build computing machines that can do this task effortlessly.

The main computational difficulty in visual recognition is the problem of variability (Riesenhuber and Poggio, 2000). A vision system is required to generalize objects across huge variations in pose, appearance, viewpoint, illumination and occlusion. In its general form, visual recognition is a very difficult computational problem, which is likely to be significantly involved in eventually making intelligent machines.

Modern object recognition can be roughly divided in four different subproblems, according to its level of complexity (see Figure 1.1):

(a) *Image classification* deals with giving a label to all the objects present in a scene, independent of its location.

(b) *Object detection with boxes* is interested in not only generating a label for each class but also define its location with a bounding box surrounding each object present in an image.



(a) image classification (b) detection with boxes (c) semantic segmentation (d) detection with segments

Figure 1.1 – The four subproblems of object recognition: (a) image classification, (b) object detection with boxes, (c) semantic segmentation and (d) object detection with segments. See text for detail. Image taken from (Lin et al., 2014).

(c) *Semantic segmentation* consists of giving a label to a possible semantic category to each pixel present in an image.

(d) *Object detection with segments* consists of localizing all objects on a scene in a pixelwise manner, that is, generate a segmentation mask to every object.

Note the difference between the third and the fourth subproblem. The objective of the former is to give a label to every pixel in an image, independent of which instance the object belongs to. The latter has to, at the same time, label all the pixels that belong to objects and assign each pixel to a given object instance. Subproblem (d), object detection with segments, is arguably the most challenging problem in object recognition and can be seen as a generalization of all previous subproblems.

In the early days of computer vision (1970s and 1980s), researchers believed that vision could be broken up into three different stages (Malik et al., 2016): (i) low level vision, related to processes such as edge detection, (ii) mid level vision, leading to representations of surfaces and (iii) high level features, corresponding to object recognition (Marr, 1982). In the 1990s, however, this idea slightly disappeared (with the exception of approaches based on multiple view geometry) to give place to *feature-based learning* approaches.

Feature-based learning approaches basically consist of extracting strong descriptive features (usually with a strong domain-specific knowledge) from images and to train a simple classifier to discriminate between different categories (Figure 1.2, top). Features used in vision problems evolved from edges and corners in the 1970s and 1980s to the use of linear filters such as Gaussian derivatives, Gabor and Haar wavelets in the 1990s (Malik et al., 2016). An important development in feature-based learning was the development of histogram-based features such as Scale-Invariant Feature Transformation (SIFT) (Lowe, 2004), Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005) and Speeded-Up Robust Features (SURF) (Bay et al., 2008).

During the 1990s, another machine learning-based paradigm for vision problems also emerged: that of *feature representation learning*. In this framework, instead of using hand-crafted discriminative features from pixel values in an image, this class of algorithms allow a machine to directly discover the features (representations) needed for recognition (Figure 1.2, bottom).

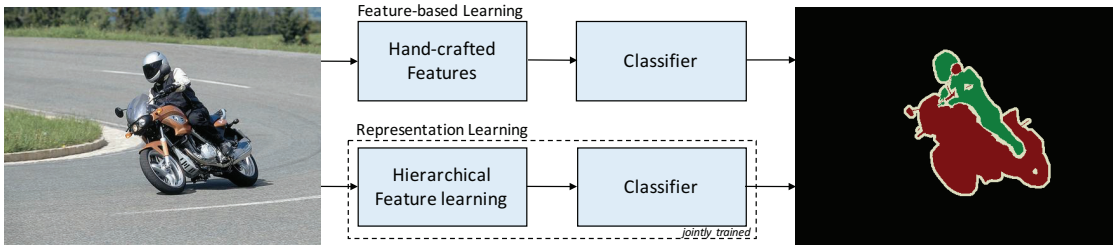


Figure 1.2 – Schematic representation of feature-based and representation learning for the problem of semantic segmentation. The former first extract engineered domain-specific features and train a simple classifier while the latter jointly learn an hierarchy of features and the classifier at the same time.

An important class of representation learning methods are the ones called *deep learning* (LeCun et al., 2015). This approach consists of a multiple level of representation, obtained by composing simple non linear modules that each transform the representation at one level into a representation at a higher, more abstract level. Deep learning methods trade *simple* machine learning models trained with *task-specific features* for *generic* (possibly more complex) machine learning algorithms trained with *simple* (raw pixels), hierarchical features.

Albeit its success in computer vision in 1990s (*e.g.* (LeCun et al., 1990, 1998)), deep learning methods, and in particular Convolutional Neural Networks (CNNs), were not considered as important as feature-based learning in object recognition. In the last few years, however, this scenario drastically changed, mainly due to increasing amounts of available data, more powerful computing machines and some new algorithmic developments. After Krizhevsky et al. (2012) achieved impressive results in the challenging ImageNet (Deng et al., 2009) classification problem, deep learning approaches became the *de facto* paradigm for learning based methods in computer vision.

1.2 Objectives

Object recognition is a diverse and complex problem which underlie many different mechanisms. The focus of this thesis is to develop different algorithms to tackle *large-scale segmentation* problems (*i.e.* subproblems (c) and (d) on Figure 1.1), that is, problems that deal with pixel-level information. In the context of the "three Rs of computer vision" proposed by Malik et al. (2016), the work of this thesis encompasses "recognition" and "re-organization".

The main objective of this thesis is to produce algorithms for different image segmentation problems that can scale nicely with data and require the least amount of feature engineering to achieve its goal. Deep learning methods fit particularly well with this motivation and have proven to be very efficient in different domains, such as natural language processing (Collobert et al., 2011), speech recognition (Hinton et al., 2012) and computer vision (He et al., 2016; Krizhevsky et al., 2012). In particular, CNN (LeCun et al., 1998), a specialized neural network

architecture, performs extremely well on image-based applications.

Throughout this thesis, we develop different algorithms to tackle three different large-scale segmentation problems:

- **Semantic Segmentation.** Semantic segmentation (subproblem (c) of Figure 1.1), also known as scene labeling, is the task of labeling each pixel of an image with the category it belongs to. This is a challenging task as it consists of solving both segmentation and multi-label recognition at once. Another challenge is the large-scale nature of the task: simply labeling one thousand 320×240 images with a computer algorithm already corresponds to producing over 76 million pixel labels. Paradoxically, in a database of 1000 images, most object classes occur only few times. In addition, the per-class pixel distribution is often quite unbalanced: some objects like 'sky' tend to cover much more pixels than other objects like 'moon'. To add a level of difficulty, hand-labeling images are very costly (as it requires segmenting objects at pixel level).
- **Segment Object Proposal Generation.** Object proposal algorithms aim to find diverse regions in an image which are likely to contain objects (independent of its category). As in semantic segmentation, these algorithms output a set of masks from an image. Unlike semantic segmentation, instead of generating one label for each pixel of an image, the interest of object proposal is to generate a set of regions that are likely to fully contain objects. An ideal proposal method should possess three key characteristics: (i) high recall (*i.e.* proposed regions should contain the maximum number of possible objects), (ii) high recall should be achieved with a minimum number of regions as possible and (iii) the proposal regions should match the object as accurately as possible. Object proposals have many applications in computer vision, *e.g.*, object detection, weakly supervised learning, class-agnostic detection.
- **Object Detection with Segments.** This problem aims at finding regions in an image that fully delineates an object as well as giving a label to each region. Object proposals play a key role in modern object detection problems. State-of-the-art object detection methods consist of two-phases: (i) a rich set of object proposals is generated and (ii) a powerful classifier (usually a CNN) is applied to each proposal. Using this pipeline, strong segment object proposal can be coupled with a classifier to deal with object detection with boxes (subproblem (b)) and with segments (subproblem (d)).

1.3 Thesis Contributions

This thesis contains different contributions for pixel-level object recognition tasks, namely semantic segmentation, segment object proposals generation and object detection with segments. The thesis contributions are the following:

- **Fully Supervised Semantic Segmentation.** The goal of the semantic segmentation is

to assign a class label to each pixel in an image. To ensure a good visual coherence and a high class accuracy, it is essential for a model to capture long range (pixel) label dependencies in images. In a feedforward architecture, this can be achieved simply by considering a sufficiently large input context patch, around each pixel to be labeled. The proposed approach consists of a recurrent convolutional neural network which considers a large input context while limiting the capacity of the model. Contrary to most standard approaches, our method does not rely on any segmentation technique nor any task-specific features. The system is trained in an end-to-end manner over raw pixels, and models complex spatial dependencies with low inference cost. As the context size increases with the built-in recurrence, the model identifies and corrects its own errors. Our approach yields good results in different scene labeling datasets.
This work has been first presented at the Deep Learning NIPS Workshop (Pinheiro and Collobert, 2013), before being published at ICML (Pinheiro and Collobert, 2014).

- **Weakly Supervised Semantic Segmentation.** Training large scale semantic segmentation model requires a large amount of pixelwise labeled data, and this require a lot of human labor. On the other hand, simply having the information of the presence (or not) of a given object category requires much less effort. It is useful to develop a model that can infer object segmentation by leveraging only object class information. This problem can be viewed as a kind of weakly supervised segmentation task, and naturally fits the multiple instance learning framework: every training image is known to have (or not) at least one pixel corresponding to the image class label, and the segmentation task can be rewritten as inferring the pixels belonging to the class of the object (given one image, and its object class). We propose a CNN-based model, which is constrained during training to put more weight on pixels which are important for classifying the image. At test time, the model has learned to discriminate the right pixels well enough, such that it performs very well on an existing segmentation benchmark, by adding only few smoothing priors. This algorithm achieves state-of-the-art results in the weakly supervised object segmentation task.
This work has been published at CVPR (Pinheiro and Collobert, 2015).

- **Generating Segment Object Proposal.** Another contribution of this thesis, is the development of a new algorithm to generate object proposals, *i.e.* a set of regions in an image that are likely to contain an object. We introduce an approach to learn object proposals based on a discriminative CNN. Such model is trained jointly with two objectives: given an image patch, the first part of the system outputs a class-agnostic segmentation mask, while the second part of the system outputs the likelihood of the patch being centered on a full object. At test time, the model is efficiently applied on the whole test image and generates a set of segmentation masks, each of them being assigned with a corresponding object likelihood score. Compared to previous approaches, our model obtains substantially higher object recall using fewer proposals. The proposed model is also able to generalize to unseen categories it has not seen during training. Unlike all previous approaches for generating object masks, this model does not rely on edges,

superpixels, or any other form of low-level segmentation.
This work has been published at NIPS (Pinheiro et al., 2015) as a "spotlight" paper.

- **Refining Object Segments.** Object segmentation requires both object-level information and low-level pixel data. This presents a challenge for feedforward networks: lower layers in convolutional networks capture rich spatial information, while upper layers encode object-level knowledge but are invariant to factors such as pose and appearance. Therefore, we propose to augment feedforward networks for object segmentation with a novel top-down refinement approach. The resulting bottom-up/top-down architecture is capable of efficiently generating high-fidelity object masks. Similarly to skip connections, this approach leverages features at all layers of the network. Unlike skip connections, our approach does not attempt to output independent predictions at each layer. Instead, the algorithm first output a coarse "mask encoding" in a feedforward pass, then refines this mask encoding in a top-down pass using features at successively lower layers. This approach is simple, fast, and effective. We demonstrate the efficiency of this approach in segment object proposal generation.
This work has been published at ECCV (Pinheiro et al., 2016) as a "spotlight" paper.

- **Object Detection.** Among many applications in computer vision, object proposal algorithms were found to be particularly important in object detection. Recent object detection systems rely on two critical steps: (i) a set of object proposals is predicted as efficiently as possible, and (ii) this set of candidate proposals is then passed to an object classifier. Such approaches have been shown they can be fast, while achieving the state of the art in detection performance. We show that the proposals generated by our approach, when coupled in standard state-of-the-art detection pipeline, achieve considerably better performance than previous proposal methods.
Part of this work has been published at BMVC (Zagoruyko et al., 2016).

1.4 Thesis Outline

The rest of the thesis is organized as follows:

- **Chapter 2: Background.** In this chapter, we introduce the large-scale image segmentation problems in the context of this thesis and present other approaches to deal with these problems. Then, we introduce the basics of convolution neural networks, a model vastly used throughout the thesis.

- **Chapter 3: Learning to Segment a Scene with Recurrent Convolutional Networks.** This chapter introduces a recurrent convolutional neural network model to deal with the problem of scene labeling. This architecture allows us to consider a large input context (while limiting its capacity), which is essential for a model to capture long range (pixel) label dependencies. The proposed model is evaluated in two standard semantic segmentation datasets.

- **Chapter 4: Learning to Segment with Image-Level Labeling.** In this chapter, an algorithm that is able to infer object segmentation by leveraging only object class information is proposed. The problem can be seen as a kind of weakly supervised segmentation task. The model is constrained during training to put more weight on pixels which are important for classifying the image. At test time, the model has learned to discriminate the right pixels and achieves good results in semantic segmentation benchmarks, by adding only a few smoothing priors.

- **Chapter 5: Learning to Generate Object Proposals.** In this chapter, we present a way to learn object proposals based on discriminative convolutional neural network. The model is trained to generate a class-agnostic segmentation mask and a score of how likely an input is to fully contains an object. At test time, the algorithm is efficiently applied on an image and generate a set of segmentation masks, each of them assigned with a likelihood score. Compared to previous approaches, our model obtains substantially higher object recall using fewer proposals.

- **Chapter 6: Learning to Refine Object Segments.** This chapter proposes a network augmentation to feedforward convolutional networks with a top-down refinement approach. This augmentation leverages features at all layers of the network to improve object segmentation. The improvement of this algorithm is demonstrated on the task of object proposal generation, although it could be applied in other pixel-level labeling task. The proposed model achieves a new state of the art performance in segment object proposal algorithm.

- **Chapter 7: Application of Proposals: Learning to Detect Objects.** In this chapter, we study an important application of object proposals: object detection. We show that the state of the art proposals generated with the method of the previous chapter can achieve important results on the challenging task of object detection with segments.

- **Chapter 8: Conclusion.** The last chapter concludes the work developed in this thesis and propose further directions for research in object segmentation problems.

2 Background

This thesis mainly focuses on deep learning methods for large-scale image segmentation. In this chapter, we provide a background on the segmentation problems treated in this thesis (Section 2.1). Then, we give a brief overview of visual representation learning with deep networks, and in particular convolutional neural networks, this thesis' method of choice to deal with such problems (Section 2.2).

2.1 Large-Scale Segmentation

The objective of image segmentation is to partition a given image into multiple regions (overlapping or not) that are considered meaningful according to some objective criterion or homogeneity in some feature space. This definition relies on selecting an objective function and, depending on this criterion, the purpose of segmentation may change.

A common criterion would be to segment images into different *objects*. However, the definition of objects is itself ambiguous. For example, an object may refer to a *thing* (*e.g.* an airplane, a horse, a person, etc.) or to a *stuff* (objects of amorfous spatial extent, *e.g.* sky, road, grass, etc.) (Forsyth et al., 1996).

The absence of a universal criterion has led to different definitions of segmentation in computer vision. In the following, we briefly describe three different segmentation problems studied in this thesis: semantic segmentation, object proposals generation and object detection with segments.

2.1.1 Semantic Segmentation

In semantic segmentation, also known as scene labeling (we use both terms indistinguishably), we are interested in labeling every pixel in an image (Figure 2.1). Each pixel should receive a semantic label (*e.g.* 'sky', 'car', 'person') based on its surrounding information (context). The two most common approaches for this problem are the *grammar-based* methods, dated since

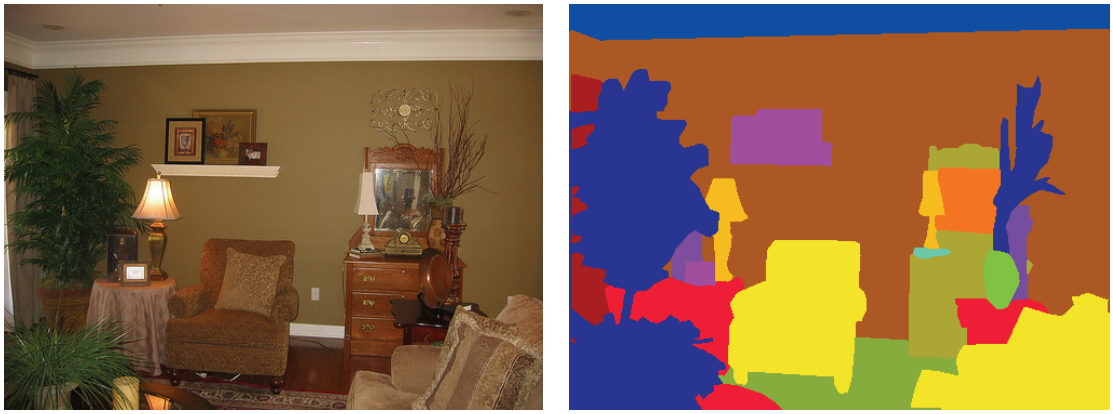


Figure 2.1 – Semantic Segmentation illustration. The picture (left) is segmented into object regions (right) represented here by different colors. Image from PASCAL-Context dataset (Mottaghi et al., 2014).

the origins of computer vision and *graphical models-based* methods, which rely on *Markov Random Fields (MRF)* (Li, 2009) or *Conditional Random Fields (CRF)* (Lafferty et al., 2001).

Grammar-based methods

The first approach to semantic segmentation was grammar-based methods. This line of work, known as *syntactic pattern recognition*, was an active area of research in the 1970s and 1980s (ichi Ohta et al., 1978; Hanson and Riseman, 1978; Fu and Albus, 1982; Ohta, 1985). Segmenting, in this scenario, consisted of breaking the image into regions and relate these regions to each other using formal grammar. Generating robust label predictions proven to be extremely difficult without powerful techniques such as image feature descriptors or statistical machine learning techniques. For this reason, this line of work almost stopped in the mid 1980s.

In the early 2000s, with the computer vision field more mature and with the help of tools like powerful image descriptors and statistical machine learning, some researchers started looking at grammar-based semantic segmentation once again (Zhu and Mumford, 2006; Zhao and chun Zhu, 2011; Socher et al., 2011). Much of the grammar-based segmentation involves learning and then enforcing the grammatical relationships, which can be difficult if the grammatical structure is unknown or the categories are not evenly sampled.

Graphical Models-based methods

In the 2000s, another framework emerged for semantic segmentation based on graphical models (mainly MRF and CRF). Most systems, followed approximately the same recipe (Tighe, 2013):

1. Extract a set of features for each pixel (or set of pixels, known as *superpixels*). These

features can be basic image statistics or powerful hand-crafted features (*e.g.* SIFT, HOG, SURF).

2. Train a local model to produce compatibility between the set of features and the ground-truth class annotation (local classifier).
3. Use the trained classifier output as the unary term of an MRF or CRF.
4. Define a binary term (smoothing prior to assume consistency of the labelling) of the MRF/CRF (usually a graph defined over the pixels/superpixels) and train the parameters of the random field (global classifier).
5. Perform inference on the random field.

Following this pipeline, He et al. (2004) expand the classifier to include both local and global image classification. Shotton et al. (2009) propose a parsing system that uses random forest classifiers to represent local spatial layout of texture and uses a CRF as a smoothing prior. Gould et al. (2009) introduced the idea of using two different label types (semantic and geometric) to improve labeling performance (by ensuring consistency of both labels on the same region).

Context can also be used to improve semantic segmentation. Contextual relationships between different categories (*e.g.* 'car' is usually supported by 'road', but not 'sky') are usually learned from the labeled dataset. The contextual relationships can be learned by simple co-occurence statistics (Rabinovich et al., 2007; Tighe and Lazebnik, 2010) or multiple forms of context, such as co-occurrences, location and appearance (Galleguillos et al., 2008, 2010). Lazebnik and Raginsky (2009) learn contextual smoothing directly from the images. These relationships are then incorporated into a CRF as a penalty term.

Some authors also considered nonparametric, data-driven approaches for open-universe dataset. Instead of learning a classifier for the unary terms, these approaches try to retrieve the most similar images from the training set and transfer the information to a test image. Liu et al. (2011) propose a nonparametric label transferring based on estimating 'SIFT Flow' between images. Tighe and Lazebnik (2010) transfer labels over different superpixels of the test image and all training images using several engineered features. More recently, Najafi et al. (2016) propose to sample labeled superpixels according to an image similarity score and formulate label transfer as an efficient filtering procedure. This approach achieves state-of-the-art results for nonparametric approaches.

Graphical models are powerful methods for modeling global spatial consistency in images. However, they possess certain limitations: (i) they require engineered, domain-specific features, (ii) inference is in general slow, as they rely on a huge label space search (and in practice only approximate inference is computationally possible), (iii) they fail to address parsing problems with more than few dozen classes, rare classes being challenging to model without overfitting.

Evaluation Metrics

In the computer vision literature, there are three important accuracy metrics to evaluate the performance on semantic segmentation: per-pixel accuracy, per-class accuracy and average precision.

Per-pixel accuracy is defined simply by the ratio of correctly classified pixels and the total number of pixels in the test set:

$$\text{acc}_{\text{pxl}} = \frac{N^p}{N}, \tag{2.1}$$

where N^p is the number of correctly classified pixels and N is the total number of pixels.

However, in many densely labeled semantic segmentation datasets, the per-category pixel distribution is very unbalanced: some categories like 'sky' tend to cover many more pixels than other objects, like 'moon'. This fact poses an extra difficulty to semantic segmentation problem. As an alternative evaluation metric, it is also common to evaluate the performance of a model by computing the *per-class accuracy*, defined as follows:

$$\text{acc}_{\text{class}} = \frac{1}{C} \sum_{c \in C} \frac{N_c^p}{N_c}, \tag{2.2}$$

where C is the total number of categories in the dataset, N_c^p is the total number of correctly classified pixels of class $c \in \{1, \ldots, C\}$ and N_c is the total number of pixels of class c.

In case of sparsely labeled semantic segmentation datasets, such as Pascal VOC (Everingham et al., 2010), the most common metric is the *Average Precision (AP)*. The average precision for a class is assessed using the intersection over union metric, defined as the number of correctly labeled pixels of that class, divided by the number of pixels labeled with that class in either the ground truth labeling or the inferred labeling. That is, for a given category c:

$$\text{AP}_c = \frac{\text{true positives}}{\text{true positives} + \text{false positives} + \text{false negatives}}. \tag{2.3}$$

It is also common in the literature to assess the mean average precision (mAP) over all categories present in the dataset:

$$\text{mAP} = \frac{1}{C} \sum_{c \in C} \text{AP}_c. \tag{2.4}$$

2.1.2 Object Proposals

The objective of object proposals algorithms is to extract meaningful regions of an image that are likely to *fully* contain an object, independent of its category (see Figure 2.2). This

Figure 2.2 – Qualitative illustration of segment object proposal generation. Given an image, the method needs to output regions that are likely to contain a class-agnostic object.

set of regions can be used in many different computer vision tasks such as object detection, segmentation, weakly supervised setting, or any object-based image parsing task.

Most object proposal approaches leverage low-level grouping and saliency cues. These models vary in terms of the type of proposal generated (bounding boxes or segmentation masks) and if the proposals are ranked or not. These methods usually fall in three categories: objectness scoring, seed segmentation and superpixel merging (See Figure 2.4).

Objectness Scoring

In this family of algorithms, the proposals are extracted by measuring the objectness score of bounding boxes. In Alexe et al. (2012) and Rahtu et al. (2011), boxes are scored based on multiple visual cues, such as saliency, color contrast and edge density. Once the boxes are scored, the algorithm outputs the ones with the highest scores. Zitnick and Dollár (2014) assume that the density of edges (obtained via structured decision forests (Dollár and Zitnick, 2013; Dollár and Zitnick, 2015)) fully enclosed in a box is a good indicator of the presence of an object. Kuo et al. (2015) propose Deepbox, a method based on CNN that learns to rerank proposals generated by (Zitnick and Dollár, 2014).

Seed Segmentation

Seed segmentation proposal methods start with a set of seed regions and generate separate foreground-background segmentation for each seed. Carreira and Sminchisescu (2012) (CPMC) compute graph cuts with several different seeds and unaries directly on pixels. The segments are then ranked using a large pool of features. (Humayun et al., 2014) improves over CPMC by re-using computation across multiple graph-cuts problems and using fast edge detectors.

In (Krähenbühl and Koltun, 2014), the seeds are placed with a classifier trained to discover objects. The authors then generate a foreground/background mask for each seed with a geodesic distance transform and proposals are computed by identifying critical level sets in each foreground. In (Krähenbühl and Koltun, 2015), the same authors trained an ensemble of

figure-ground segmentation models operated on simple image features.

Superpixel Merging

Superpixel merging proposals generate a set of over-segmentation from superpixels and merge them according to certain heuristics. *Selective Search* (Uijlings et al., 2013) relies on multiple hand-crafted features and similarity functions for merging superpixels. *Multiscale Combinatorial Grouping (MCG)* (Arbeláez et al., 2014; Pont-Tuset et al., 2015), introduces a fast algorithm for computing multi-scale hierarchical segmentation. Segments are merged based on edge strength and the proposals are ranked according to cues such as size, shape, location and edge strength.

Evaluation Metrics

A good object proposal algorithm needs to have a good coverage of the objects of interest in a test image. A common practice to evaluate the quality of proposals is, therefore, based on the recall of the ground truth annotations, in a class-agnostic manner.

Recall is usually measured in terms of segment *Intersection over Union (IoU)* for each ground truth annotation. IoU is the intersection of a candidate proposal M and ground truth annotation G divided by the area of their union, that is:

$$\text{IoU} = \frac{area(M \cap G)}{area(M \cup G)}. \tag{2.5}$$

This metric can also be applied in the bounding box detection scheme, simply by considering the regions M and G as being a rectangle box of the dimensions of the bounding boxes.

IoU can vary from 0 (no intersection at all) to 1 (perfect matching). Figure 2.3, from Krähenbühl and Koltun (2014), illustrates what segment IoU means. The first proposal is able to localize the object correctly, but predicts the shape very poorly (IoU of 0.55). As we move to the right, the IoU improves up to IoU 0.91, approaching human accuracy.

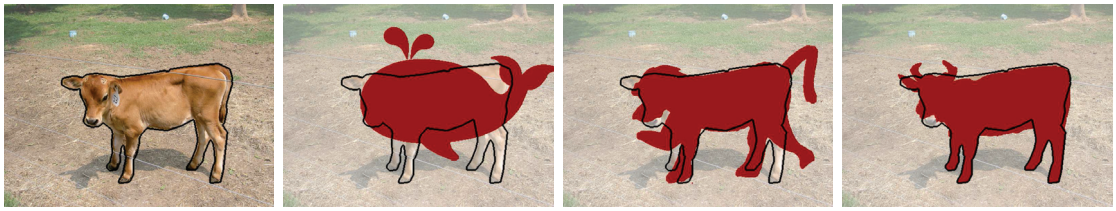
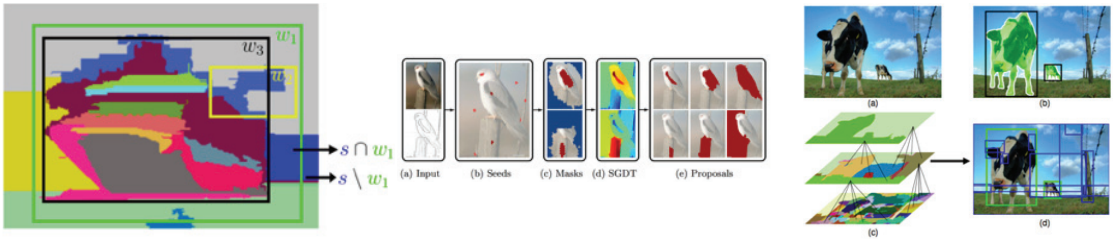


Figure 2.3 – Three segment proposals (in red) overlapping a given object (leftmost image). The segment IoUs are: 0.55, 0.70, 0.91. A coefficient of 0.9 demands very tight fit. Figure taken from (Krähenbühl and Koltun, 2014).

A common metric for evaluating proposals is, for a fixed number of proposals (*e.g.* 100), the



(a) Objectness scoring (b) Seed generation (b) Superpixel merging

Figure 2.4 – Classical approaches for object proposal generation: (a) object scoring (figure from Alexe et al. (2012)), (b) seed generation (figure from Krähenbühl and Koltun (2014)) and (c) superpixel merging (figure from Uijlings et al. (2013)). See text for details.

fraction of ground truth annotations covered as the IoU threshold is varied. A complementary metric is, for a fixed IoU threshold, to measure the proposal recall as the number of proposals considered varies. Hosang et al. (2016) propose an efficient metric, *Average Recall (AR)*, that measures the IoU between 0.5 and 1 for a fixed number of proposals. AR has been shown to correlate extremely well with detector performance (recall at a single IoU threshold is far less predictive (Hosang et al., 2016)).

2.1.3 Object Detection

The objective of object detection is to find all instances of different objects (assuming a definition of object) on a given image. The instances can be either in the form of bounding box comprising the object instance or a segmentation mask delineating the object. Until recently, the dominant paradigm in object detection was the sliding window framework: a classifier is applied at every object location and scale. A detector, in this case, can be seen as a classifier which takes as input an image, a location and a scale and determines whether or not there is an instance of the target category at the given position and scale.

Viola and Jones (2004) propose a cascading algorithm for face detection based on the AdaBoost classifier and Haar-based features. Dalal and Triggs (2005) propose a similar approach, in which they use a single filter on HOG features to represent an object category. A Support Vector Machine (SVM) classifier is trained on the top of these features to determine if an object is present or not in a bounding box at a given location and scale. At test time, the classifier is applied densely in each location and scale. This approach was first validated on pedestrian detection and then extended to multi-categorical object detection problem.

Felzenszwalb et al. (2010) propose a Deformable Part Model (DPM) for object detection, which extends this approach by integrating part-based models (Fischler and Elschlager, 1973; Felzenszwalb and Huttenlocher, 2000) into (Dalal and Triggs, 2005) system. In part-based models, objects are described as a collection of parts arranged in a deformable configuration. Each object part is then considered as a latent variable in a classification problem. The authors use a generalization of SVM which considers the object parts as a latent variable

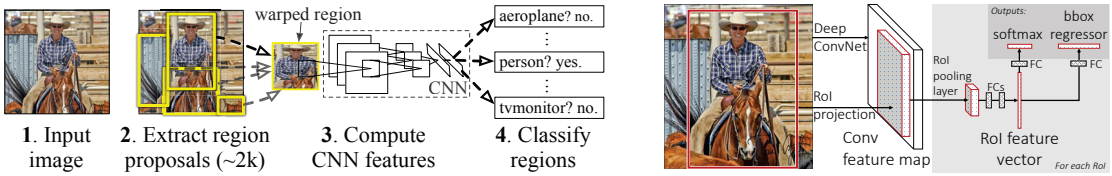


Figure 2.5 – Modern object detectors are based on two steps: a set of proposals are extracted from an image, and a classifier is applied to each proposal. In (a) RCNN apply a CNN classifier to each proposal extracted from the image. In (b) Fast RCNN extract the proposals in the final CNN feature map. Figures are taken from Girshick et al. (2014); Girshick (2015).

in the optimization problem. Training is done discriminatively using an iterative algorithm that alternates between estimating latent variables for positive examples and solving an large convex optimization problem.

Until 2013, DPMs models have dominated the state of the art in object detection. In 2014, however, this trend changed: Girshick et al. (2014) propose a two-phase approach for object detection, *R-CNN*. In this model, first, a rich set of object proposals is generated using a fast (but possibly imprecise) algorithm. Second, a convolutional neural network classifier is applied on each of the proposals. This approach provides a notable gain in object detection accuracy compared to classic sliding window approaches. See Figure 2.5a for a schematic representation of the method. The next year, Girshick (2015) proposes a variant of RCNN, dubbed *Fast R-CNN*, in which the proposals are extracted on the final spatial feature map of the classifier, speeding up the inference by a non-trivial amount (Figure 2.5b). In both methods, the authors use the Selective Search (Uijlings et al., 2013) algorithm to extract a set of proposals.

Currently, most state-of-the-art object detectors rely on object proposals as a first preprocessing step (Girshick et al., 2014; Girshick, 2015; Ren et al., 2015; Bell et al., 2016; He et al., 2016). Moreover, Hosang et al. (2016) empirically show that the quality of proposals used in this pipeline can substantially affect the performance of the detection system.

Evaluation Metrics

The most popular way to evaluate the quality of detection is through a precision/recall curve (Everingham et al., 2010; Lin et al., 2014) for each category. The quantitative measure in this case is *Average Precision (AP)* (see the next paragraph). It is important to note that average precision is defined differently for detection and for semantic segmentation.

A detection is considered true or false positive based on the IoU (as described above), for a fixed threshold θ. Note that this works for both bounding box and segmentation outputs. For each category and for a fixed IoU threshold, AP is computed as follows (Everingham et al., 2010):

1. Compute a version of the measured precision/recall curve with precision monotonically decreasing, by setting the precision for recall r to the maximum precision obtained for any recall $r^{'} \geq r$.
2. Compute the AP as the area under this curve by numerical integration.

The AP metric is commonly measured in two different ways: (i) the PASCAL VOC (Everingham et al., 2010) metric, which considers a fixed threshold $\theta = 0.5$ and (ii) the COCO (Lin et al., 2014) metric, which considers an average over different θ between 0.5 and 1. The latter metric takes into account more accurate localization, and thus can be seen as a more realistic metric for detection. In both cases, the AP is also usually averaged among all categories to give a final numerical value.

2.2 Representation Learning with Deep Learning

In this section, we present an overview of visual feature representation learning with deep learning methods. Contrary to standard approaches for vision problems based on hand-crafted features, deep learning methods aim at learning the features required to the task at hand.

This section starts with a basic description of the most common Artificial Neural Network (ANN) model, the *Multilayer Perceptron (MLP)*. We then introduce the *Convolutional Neural Network (CNNs)* models, a specialized ANN architecture particularly useful for computer vision problems. Then, we briefly describe how learning is achieved in such models.

2.2.1 Multilayer Perceptron

The first deep learning model was the multi-layer perceptron. These models were initially inspired by neuronal systems (Mcculloch and Pitts, 1943) (although today not much more than the name has relations to biological systems). MLPs are a type of machine learning models which apply a sequence of non-linear transformations to the input data.

Mathematically, a MLP with L layers can be described with the following equations (we use a similar notation as Farabet (2014)):

$$\begin{aligned} \mathbf{y} = f(\mathbf{x},\theta) &= \mathbf{h}_L \\ \mathbf{h}_l &= \sigma_l(\mathbf{W}_l\mathbf{h}_{l-1} + \mathbf{b}_l),\ \forall l \in \{1,2,\ldots,L\} \\ \mathbf{h}_0 &= \mathbf{x}, \end{aligned} \tag{2.6}$$

where $\theta = \{\mathbf{W_l}, \mathbf{b}_l\}, \forall l \in \{1,\ldots,L\}$ is the set of trainable parameters (consisting of bias parameters $\mathbf{b}_l$ and weight parameters $\mathbf{W}_l$) for each layer l, $\mathbf{x} \in \mathbb{R}^{d_{in}}$ is the input vector (*e.g.* a vectorized image), $\mathbf{y} \in \mathbb{R}^{d_{out}}$ is the output of the network (this output can be interpreted in different ways depending of the task of interest) and σ_l is the point-wise non-linear activation function at

layer l.

Common non-linear activation functions for hidden units ($\sigma_l, l \in \{1,\ldots,L-1\}$) are the *hyperbolic tangent*

$$\text{Tanh(x)} = \frac{e^{2x-1}}{e^{2x+1}}, \tag{2.7}$$

and the *Rectified Linear Unit (ReLU)*

$$\text{ReLU(x)} = \max(0, x). \tag{2.8}$$

Note the importance of the activation function: without them, the whole system would be a stack of linear operations (matrix multiplications) and could be equivalently written as a single matrix. The choice of activation function is a completely empirical question. If the network is very deep (*i.e.* possesses many layers), ReLU activations are popular as they reduce the likelihood of the gradient to vanish (Krizhevsky et al., 2012).

The output activation function (σ_L) depends on the problem at hand. For example, if we are interested in a regression problem, the output activation can be a simple linear or log-linear function. If we are interested in a classification problem, the output activation is designed so that the network models the likelihood of the data.

2.2.2 Convolutional Neural Networks

Consider, for example, an image with dimensions $3 \times 200 \times 200$ (3 color channels, 200 pixels of height and 200 pixels of width). A single fully-connected neuron on the first layer of a MLP would require $3 * 200 * 200 = 120000$ parameters. Moreover, to learn the complexity of the world, such a network would require multiple layers, each of them with multiple neurons. The number of parameters in such a model would quickly increase to an unbearable number and lead to overfitting and computational issues.

Convolutional Neural Networks (CNNs) (LeCun et al., 1990, 1998) are a natural extension of MLPs for processing data that has a known, grid-like topology (*e.g.* images). CNNs use the spatial correlation of the signal to constrain the architecture in a more sensible way. Their architecture, somewhat inspired by the biological visual system (Hubel and Wiesel, 1962; Fukushima, 1980; LeCun et al., 1998), possesses two key properties that make them extremely useful for image applications: spatially shared weights and spatial pooling. These kind of networks learn features that are shift-invariant, *i.e.*, filters that are useful across the entire image (due to the fact that image statistics are stationary). The pooling layers are responsible for reducing the sensitivity of the output to slight input shift and distortions.

A typical convolutional network is composed of multiple stages, as shown in Figure 2.6. The output of each stage is made of a set of 2D arrays called *feature maps*. Each feature map is the outcome of one convolutional (and an optional pooling) filter applied over the full image. A

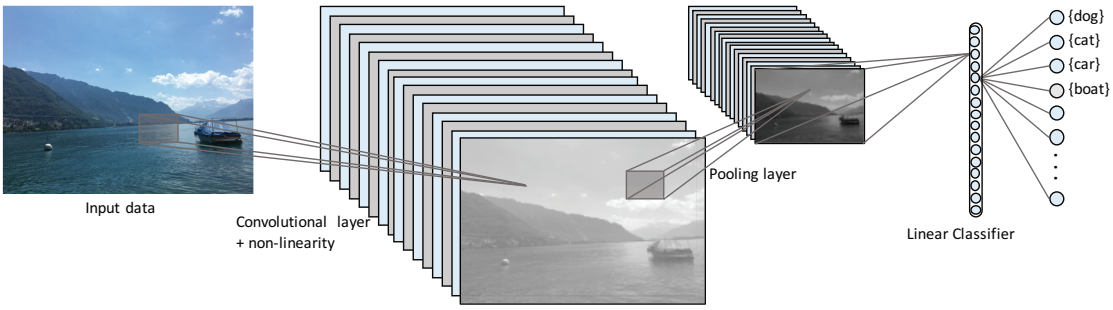


Figure 2.6 – Architecture of a typical convolutional network for object recognition. We represent one simple block (consisting of convolutional, non-linearity and pooling layer) and a final linear multi-class classifier.

point-wise non-linear activation function always follows a convolution layer.

In its more general form, a convolutional network can be written as:

$$\begin{aligned} \mathbf{Y} = f(\mathbf{X},\theta) &= \mathbf{H}_L \\ \mathbf{H}_l &= \text{pool}_l(\sigma_l(\mathbf{W}_l\mathbf{H}_{l-1} + \mathbf{b}_l)), \forall l \in \{1,\ldots,L\} \\ \mathbf{H}_0 &= \mathbf{X}, \end{aligned} \tag{2.9}$$

with $\theta = \{\mathbf{W}_l, \mathbf{b}_l\}, \forall l \in \{1,\ldots,L\}$ is the set of trainable parameters, as in MLP, $\mathbf{X} \in \mathbb{R}^{c\times h\times w}$ is the input image (with c color channels, height of h pixels and width of w pixels), $\mathbf{Y} \in \mathbb{R}^{n\times h^{'}\times w^{'}}$ is an array (with dimension $h^{'} \times w^{'}$) of output vectors of dimension n (each vector is an non-linear encoding of a sub-window of the input), σ_l is a point-wise non-linearity at layer l and pool_l is a (optional) pooling function at layer l.

The main difference between MLPs and CNNs lies in the parameter matrices $\mathbf{W}_l$: in MLPs, the matrices can take any general form, while in CNNs these matrices are constraints to be *Toeplitz* matrices (Gray, 2005). That is, the matrices have several entries constrained to be equal to each other (moreover, these matrices are very sparse since the kernel is usually much smaller than the input image). Therefore, each hidden unit array $\mathbf{H}_l$ can be expressed as a discrete-time convolution between kernels from $\mathbf{W}_l$ and the previous hidden unit $\mathbf{H}_{l-1}$ (transformed through a point-wise non-linearity and possibly pooled). More specifically,

$$\mathbf{H}_{lp} = \text{pool}_l\left(\sigma_l\left(b_{lp} + \sum_{q\in\text{parents(p)}} \mathbf{w}_{lq} * \mathbf{H}_{l-1,q}\right)\right), \tag{2.10}$$

where $\mathbf{H}_{lp}$ is the p^{th} component of the l^{th} feature map $\mathbf{H}_l$.

In general, the output of a CNN is usually coupled with a MLP classifier. As before, this output is connected to a final activation function that depends on the problem considered.

From the mathematical description above, a basic convolutional neural network can thus be seen as a stack of three distinct components (see Figure 2.6):

Convolutional Layer. The input of every layer is a 3D array with c_i 2D feature maps of size $h_i \times w_i$. Each component is denoted by x_{ijk} and each feature map denoted as x_i. The output is also a 3D array, y, corresponding to c_o feature maps of dimension $h_o \times w_o$. A trainable filter w_{ij} (and bias parameter b_j) is a trainable kernel of size $k_1 \times k_2$ and connect input feature map x_i with output feature map y_j. The convolutional module computes:

$$y_j = b_j + \sum_i w_{ij} * x_i \,, \tag{2.11}$$

where $*$ is the 2D discrete-time convolutional operator. Each filter w_{ij} learns a particular feature at *every* location on the input (hence, forcing spatial invariance). This convolution can have stride larger than 1. In recent convolutional layer implementation, it became popular to pad the input layer so that the output of a convolution layer possesses same dimension as the input.

Activation Layer. Similar to the MLP case, the point-wise non-linearity is applied to each location (ijk) of the feature maps.

Pooling Layer. This layer is responsible to reduce the spatial dimension of each feature map of its input. This property is important in image models for three reasons: (i) it makes the model robust to small variations in the location of features in previous layers, (ii) it increases the receptive field of the network and (iii) it controls the capacity of the model. A pooling operation can be applied (optionally) after each activation layer. Throughout this thesis, we always consider *max pooling* layers, which consist of reporting the maximum output within a rectangular neighborhood. Other popular pooling functions include the average of a rectangular neighborhood, the L^2 norm of a rectangular neighborhood, or a weighted average based on the distance from the central pixel. It is common practice to choose the stride of the pooling layer to be equal to its kernel size (*e.g.* a 2×2 pooling layer takes the maximum value at each 2×2 window and strided by 2, therefore, reducing the spatial dimension of the feature maps by a factor 2).

2.2.3 Learning

In this section, we briefly explain how parameter estimation (learning) is carried in the context of neural networks (the same principle is applied to MLP, CNN or any other architecture type). The different segmentation problems addressed in this thesis are all defined as *discriminative* tasks. Therefore, we will only consider learning for discriminative tasks, in which the network is designed to model conditional probabilities.

Loss Function

Once a model is defined, its structure can be abstracted and the network can be seen as a function approximator. In classification problems, neural networks are modeled in a way such

that its output can be seen as a conditional distribution of the target y, given the input $\mathbf{x}$ and the parameters θ. This probability can be modeled by transforming the output of the network $f_c(\mathbf{x},\theta)$ (for each class $c \in \{1,\dots,C\}$) with a *softmax* function (Bridle, 1990):

$$p(y^i|\mathbf{x}^i,\theta) = \prod_{c=1}^{C} \left(\frac{e^{f_c(\mathbf{x}^i,\theta)}}{\sum_j e^{f_j(\mathbf{x}^i,\theta)}} \right)^{\mathbb{1}_c(y^i)}, \tag{2.12}$$

where $\mathbb{1}_a(x)$ is the indicator function

$$\mathbb{1}_a(x) = \begin{cases} 1, \text{if } x = a \\ 0, \text{otherwise}. \end{cases} \tag{2.13}$$

In the special case where $C = 2$, the softmax function is reduced to logistic regression. Therefore, softmax can be seen as a multiclass generalization of the logistic regression.

In the following, we consider a dataset $\mathcal{D} = \{\mathbf{X},\mathbf{Y}\} = \{\mathbf{x}^i, y^i\}$, $i \in \{1,\dots,N\}$, where $\mathbf{x}^i$ is an input image and y^i is its associated target (label). We consider a multi-categorical problem in which each label y^i belongs to one of C categories, $y^i \in \{1,\dots,C\}$. The parameters of the model are found simply by maximizing the likelihood over the training data $\mathcal{D}$ with respect to the parameters θ:

$$\begin{aligned} \theta^* &= \underset{\theta}{\operatorname{argmax}}\, p(\mathbf{Y}|\mathbf{X},\theta) \\ &= \underset{\theta}{\operatorname{argmax}}\, p(y^1, y^2, \dots, y^N|\mathbf{x}^1,\mathbf{x}^2,...,\mathbf{x}^N,\theta) \\ &\overset{\text{i.i.d.}}{=} \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{N} p(y^i|\mathbf{x}^i,\theta), \end{aligned} \tag{2.14}$$

where the last line assumes that the training set is sampled from an unknown independent, identically distributed (i.i.d.) distribution. Equivalently, this optimization problem can be seen as minimizing the negative log-likelihood of the training data (since the logarithm function is monotonic):

$$\theta^* = \underset{\theta}{\operatorname{argmin}} -\sum_i \sum_c \mathbb{1}_c(y^i) \left[f_c(\mathbf{x}^i,\theta) - \log\left(\sum_j e^{f_j(x^i,\theta)} \right) \right]. \tag{2.15}$$

Note that this minimization problem is equivalent to minimizing the *cross-entropy* of the real target distribution and the distribution generated by the neural network. In classification problems, it is common to consider *one-hot encoding* on the target distribution. In this case, the minimization problem can be rewritten as:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} -\sum_i \left[f_{c_i^*}(\mathbf{x}^i,\theta) - \log\left(\sum_j e^{f_j(x^i,\theta)} \right) \right], \tag{2.16}$$

where c_i^* is the one-hot label encoding of training example i. This problem does not have a closed solution and we apply an iterative approach to find a minimum. That is, the parameters θ of the network are estimated by minimizing the following *loss function* (also know as cost function, objective function or criterion):

$$\mathcal{L}(\theta, \mathbf{X}, \mathbf{Y}) = -\sum_i \left[f_{c_i^*}(\mathbf{x}^i, \theta) - \log\left(\sum_j e^{f_j(x^i, \theta)} \right) \right]. \tag{2.17}$$

Optimization

The most common optimization method used in machine learning is the *gradient descent method*, which, with a randomly initialized set of parameters θ_1, is defined as:

$$\theta_{t+1} \longleftarrow \theta_t - \eta \nabla_\theta \mathcal{L}(\theta, \mathbf{X}, \mathbf{Y}). \tag{2.18}$$

This method is also know as *batch* gradient descent method. In large-scale machine learning problems (*e.g.*, the segmentation problems addressed in this thesis), this method can become inefficient in practice as it requires a full pass on the data at each iteration. Moreover, batch gradient descent can be redundant if the dataset contains similar examples.

A common way to address this issue is to consider a stochastic approximation of the gradient, commonly referred to as *Stochastic Gradient Descent (SGD)* (Robbins and Monro, 1951; Bottou, 1991). In this case, a random training sample $\{\mathbf{x}^i, y^i\}$ is used to estimate the gradient and the parameters are updated as:

$$\theta_{t+1} \longleftarrow \theta_t - \eta \nabla_\theta \mathcal{L}(\theta, \mathbf{x}^i, y^i). \tag{2.19}$$

More often, we optimize the loss function using a midterm between stochastic and batch method called *mini-batch* gradient descent. This approach, which uses a subset of $n < |\mathcal{D}|$ training samples to perform each parameter update is defined as:

$$\theta_{t+1} \longleftarrow \theta_t - \eta \nabla_\theta \mathcal{L}(\theta, \mathbf{x}^{(i,\ldots,i+n)}, y^{(i,\ldots,i+n)}). \tag{2.20}$$

There are many different variants of this learning rule used to reduce the noise in stochastic directions, for example the Momentum method (Rumelhart et al., 1986), averaged stochastic gradient descent (Polyak and Juditsky, 1992), Adagrad (Duchi et al., 2011), Rmsprop (Tieleman and Hinton, 2012) and Adam (Kingma and Ba, 2014). Another important class of algorithms are the second-order methods, which use second derivative information of the loss function to improve the optimization. However, these methods are infeasible to compute in the large-scale problems addressed in this thesis.

The gradients of the loss function with respect to the trainable parameters θ used in the gradient descent methods introduced above are computed using the *backpropagation* algorithm (Bryson et al., 1963; Werbos, 1974; Rumelhart et al., 1986).

Due to the non-linear nature of neural networks, this problem is non-convex and the optimization methods applied in practice find local minima instead of a global minimum. In practice, this has been show to not pose a big problem in deep nets (Dauphin et al., 2014; Choromanska et al., 2015).

Regularization

Neural networks contain a large number of free parameters that need to be learned. These models can describe a huge range of phenomena, but require a lot of data to avoid *overfitting*, that is, when the model is able to achieve good performance on training data but performs poorly on the test data.

A very simple way to reduce overfitting in neural networks is to increase the training data to improve its generalization (assuming the capacity remains constant). However high-quality labeled data can be expensive to acquire. Fortunately, other techniques exist in the literature that can reduce overfitting, assuming a fixed network and a fixed training data size. These methods are called *regularization* techniques. In this section, we describe the most commonly applied regularization techniques found in the literature.

The simplest regularization technique is called *early stopping*. It consists, as the name implies, to stop the training once the validation error (computed in a hold-off set of training data not used during training) achieves a minimum error.

Weight decay is another very common regularization technique. It is used to penalize large weights using certain constraints on their values. These techniques are usually implemented by adding extra terms to the network cost function. In L^2 regularization, an extra term is added to the cost function that penalizes the square magnitude of all parameters. That is, for every weight w_i in the network, we add a term $\frac{1}{2}\lambda w_i^2$ to the loss function, where λ is the regularization strength. L^2 regularization has the intuitive interpretation of heavily penalizing peaky weight vectors and preferring diffuse weight vectors. Another common weight decay regularization is the L^1 regularization, which poses sparse constraints on the weight. Similarly to L^2, L^1 regularization includes an extra term to the cost function, $\lambda|w_i|$. Neurons with L^1 regularization end up using only a sparse subset of their most important inputs and become nearly invariant to the "noisy" inputs.

An extremely effective regularization technique for neural networks is *Dropout* (Srivastava et al., 2014). During training, Dropout is implemented by only keeping a neuron active with a certain probability p (a hyperparameter), or setting it to zero otherwise. It can be interpreted as sampling a neural network within the full neural network, and only updating the parameters of the sampled network based on the input data. At test time, we would ideally like to find a sample average of all possible 2^n dropped-out networks. Unfortunately this is unfeasible for large values of n. However, we can find an approximation by using the full network with each node's output weighted by a factor of p, so the expected value of the output of any node is the

same as in the training stages.

Batch normalization (Ioffe and Szegedy, 2015) is another popular regularization technique for deep convolutional neural networks. This method partially alleviates the problem of internal covariate shift in deep networks, that is, the fact that training deep networks is difficult because the distribution of each layer's input changes during training, as the parameters of the previous layers change. During SGD training, each activation of the mini-batch is centered to zero-mean and unit variance. The mean and variance are measured over the whole mini-batch, independently for each activation. Batch normalization allows us to use much higher learning rates and be less careful about initialization.

2.3 Summary

In this chapter we briefly introduced the different aspects of the image segmentation problems that will be treated in this thesis. We then presented a basic introduction to the deep learning methods most commonly used to work with vision problems.

3 Learning to Segment a Scene with Recurrent Convolutional Networks

In this chapter, we address the problem of *fully supervised* semantic segmentation (or scene labeling). That is, given a densely labeled dataset (in which all pixels are labeled), the objective is to predict the class label of each pixel in a scene.

Semantic segmentation is most commonly addressed with some kind of *local* classifier constrained in its predictions with a graphical model (*e.g.* Conditional Random Fields (CRF), Markov Random Fields (MRF)), in which *global* decisions are made. These approaches usually consist of segmenting the image into superpixels or segment regions to assure a visible consistency of the labeling and also to take into account similarities between neighbor segments, giving a high level understanding of the overall structure of the image. Each segment contains a series of input features describing it and contextual features describing spatial relation between the label of neighbor segments. These models are then trained to maximize the likelihood of correct classification given the features (Verbeek and Triggs, 2008; Gould et al., 2009; Munoz et al., 2010; Liu et al., 2011; Kumar and Koller, 2010; Socher et al., 2011; Lempitsky et al., 2011; Tighe and Lazebnik, 2010). The main limitation of scene labeling approaches based on graphical models is the computational cost at test time, which limits the model to simple contextual features.

In this chapter, we consider a neural network approach which can take into account long range label dependencies in the scenes while controlling the capacity of the network. We achieve state-of-the-art accuracy while keeping the computational cost low at test time, thanks to the complete feedforward design. Our method relies on a recurrent architecture for convolutional neural networks: a sequential series of networks sharing the same set of parameters. Each instance takes as input both an RGB image and the classification predictions of the previous instance of the network. The network automatically learns to smooth its own predicted labels. As a result, the overall network performance is increased as the number of instances increases.

Compared to graphical model approaches relying on image segmentation, our method has several advantages: (i) it does not require any engineered features, since deep learning architectures train (hopefully) adequate discriminative filters in an end-to-end manner, (ii) the

Method	Task-specific features
Gould et al. (2009)	17-dimensional color and texture features, 9 grid locations around the pixel and the image row, region segmentation.
Munoz et al. (2010)	Gist, pyramid histogram of oriented gradients, color Histogram CIELab, relative relocation, hierarchical region representation.
Kumar and Koller (2010)	Color, texture, shape, percentage pixels above horizontal, region-based segmentation.
Socher et al. (2011)	Same as Gould et al. (2009).
Lempitsky et al. (2011)	Histogram of visual SIFT, histogram of RGB, histogram of locations, contour shape descriptor.
Tighe and Lazebnik (2010)	Global, shape, location, texture/SIFT, color, appearance, MRF.
Farabet et al. (2013)	Laplacian pyramid, superpixels/CRF/tree segmentation, data augmentation.
Our Recurrent CNN	Raw pixels.

Table 3.1 – Comparison between different methods for full scene labeling. The advantage of our proposed method is the simplicity of inference, not relying on any task-specific feature extraction nor segmentation method.

prediction phase does not rely on any label space searching, since it requires only the forward evaluation of a function.

This chapter is organized as follows. Section 3.1 briefly presents related works. Section 3.2 describes the proposed strategy. Section 3.3 presents the results of our experiments in two standard datasets: the Stanford Background Dataset (8 classes) and the SIFT Flow Dataset (33 classes), and compares the performance with other methods. Finally, Section 3.4 provides a discussion followed by a conclusion.

3.1 Related Work

Recurrent Neural Networks (RNNs) date back from the late 1980s. Already in (Jordan, 1986), the network was fed (in a time series framework) with the input of the current time step, plus the output of the previous one. Several variants have been later introduced, such as in (Elman, 1990). RNNs have been successfully applied to a wide variety of tasks, including in natural language processing (Stoianov et al., 1997; Cho et al.), speech processing (Robinson, 1994; Graves et al., 2013) and image processing (Graves and Schmidhuber, 2008). Our approach can be viewed as a particular instance of Jordan's recurrent network adapted to image processing (we use a convolutional neural network). Providing feedback from the output into the input allows the network to model label dependencies, and correct its own previous predictions.

In a preliminary work, Grangier et al. (2009) proposed an innovative approach to scene labeling without the use of any graphical model. The authors proposed a solution based on deep convolutional networks relying on a supervised greedy learning strategy. These network architectures when fed with raw pixels are able to capture texture, shape and contextual

information.

Socher et al. (2011) also considered the use of deep learning techniques to deal with scene labeling, where off-the-shelf features of segments are recursively merged to assign a semantic category label. In contrast, our approach uses the recurrent architecture to parse the scene with a smoother class annotation.

In (Socher et al., 2012), the authors proposed an approach which combines convolutional and recursive networks for classifying RGB-D images. The approach first extracts features using a convolutional network which is then fed to a standard recurrent net. In that respect, our approach is more end-to-end.

More recently, Farabet et al. (2013) investigated the use of convolutional networks to extract features from a multiscale pyramid of images. This solution yields satisfactory results for the categorization of the pixels, but poor visual coherence. In order to improve visual coherence, three different over-segmentation approaches were proposed: (i) the scene is segmented in superpixels and a single class is assigned to each of the superpixels, (ii) a conditional random field is defined over a set of superpixels to model joint probabilities between them and correct aberrant pixel classification (such as 'road' pixel surrounded by 'sky'), and (iii) the selection of a subset of tree nodes that maximize the average "purity" of the class distribution, hence maximizing the overall likelihood that each segment will contain a single object. In contrast, our approach is simpler and completely feedforward, as it does not require any image segmentation technique, nor the handling of a multiscale pyramid of input images. Similarly to (Farabet et al., 2013), Schulz and Behnke (2012) proposed a multiscale convolutional architecture. In their approach, the authors smooth out the predicted labels with pairwise class filters.

Compared to existing approaches, our method does not rely on any task-specific feature (see Table 3.1). Furthermore, our scene labeling system is able to extract relevant contextual information from raw pixels.

On the years following the work presented in this chapter, CNN had become extremely popular for semantic segmentation, and many different works flourished (Chen et al., 2015; Long et al., 2015; Sharma et al., 2015; Noh et al., 2015; Caesar et al.; Zheng et al., 2015). These models, which in general use very deep networks and are pretrained on ImageNet (Deng et al., 2009), pushed even further the state of the art in fully supervised semantic segmentation.

3.2 Method Description

3.2.1 Convolutional Neural Networks for Scene Labeling

A typical convolutional network is composed of multiple stages, as shown in Figure 3.1. The output of each stage is made of a set of 2D arrays called feature maps. Each feature map is the outcome of one convolutional layer (followed by a non-linear activation function) or pooling

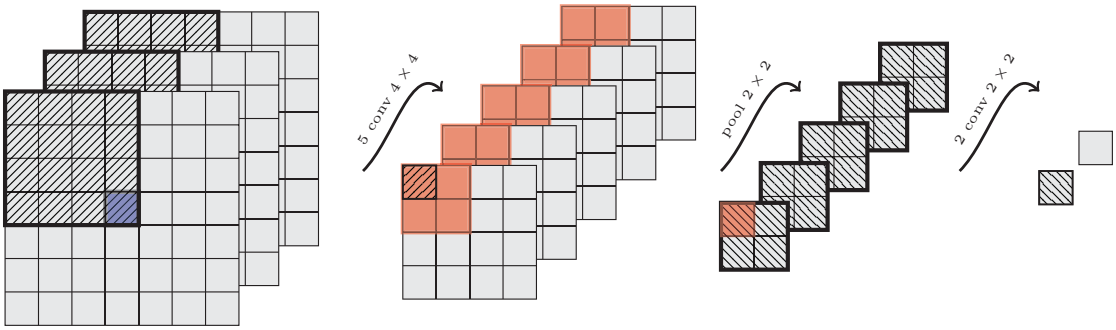


Figure 3.1 – A simple convolutional network. Given an image patch providing a context around a pixel to classify (here blue), a series of convolutions and pooling operations (filters slid through input planes) are applied (here, five 4×4 convolutions, followed by one 2×2 pooling, followed by two 2×2 convolutions. Each 1×1 output plane is interpreted as a score for a given class.

filter applied over the full image.

In the context of scene labeling, given an image I_k we are interested in finding the label of each pixel at location (i, j) in the image. More precisely, the network is fed with a squared *context patch* $I_{i,j,k}$ surrounding the pixel at location (i, j) in the k^{th} image. It can be shown (see Figure 3.1) that the output plane size sz_l of the l^{th} convolution or pooling layer is computed as:

$$sz_l = \frac{sz_{l-1} - kW_l}{dW_l} + 1\,, \tag{3.1}$$

where sz_0 is the input patch size, kW_l is the size of the convolution (or pooling) kernels in the l^{th} layer, and dW_l is the pixel step size used to slide the convolution (or pooling) kernels over the input planes.[1] Given a network architecture and an input image, one can compute the output image size by successively applying (3.1) on each layer of the network. During the training phase, the size of the input patch $I_{i,j,k}$ is chosen carefully such that the output layer produces 1×1 planes, which are then interpreted as scores for each class of interest.

The output of a network f with L stages and trainable parameters $\theta = \{\mathbf{W}_l, \mathbf{b}_l\}, \forall l \in \{1, \ldots, L\}$, for a given input patch $I_{i,j,k}$ can be formally written as:

$$f(I_{i,j,k}; \theta) = \mathbf{W}_L \mathbf{H}_{L-1} + \mathbf{b}_L\,, \tag{3.2}$$

with the output of the l^{th} hidden layer computed as:

$$\mathbf{H}_l = \text{pool}(\tanh(\mathbf{W}_l \mathbf{H}_{l-1} + \mathbf{b}_l))\,, \tag{3.3}$$

for $l = \{1, \ldots, L\}$ and denoting $\mathbf{H}_0 = I_{i,j,k}$. $\mathbf{b}_l$ is the bias vector of layer l and $\mathbf{W}_l$ is the Toeplitz matrix of connection between layer $l-1$ and layer l. The pool$(\cdot)$ function is the max-pooling

[1] Most people use $dW = 1$ for convolutional layers, and $dW = kW$ for pooling layers.

operator. In this chapter, we use the point-wise hyperbolic tangent as the activation function.

The network is trained by transforming the scores $f_c(I_{i,j,k};\theta)$ (for each class of interest $c \in \{1,\ldots,C\}$) into conditional probabilities, by applying a softmax function (Bridle, 1990):

$$p(c|I_{i,j,k};\theta) = \frac{e^{f_c(I_{i,j,k};\theta)}}{\sum_{d\in\{1,\ldots,C\}} e^{f_d(I_{i,j,k};\theta)}}, \tag{3.4}$$

and maximizing the likelihood of the training data. More specifically, the parameters θ of the network $f(\cdot)$ are learned in an end-to-end supervised way, by minimizing the negative log-likelihood over the training set:

$$\mathcal{L}_f(\theta) = -\sum_{I_{(i,j,k)}} \ln p(l_{i,j,k}|I_{i,j,k};\theta), \tag{3.5}$$

where $l_{i,j,k}$ is the correct pixel label class at position (i,j) in image I_k. The minimization is achieved with the Stochastic Gradient Descent (SGD) algorithm with a fixed learning rate η:

$$\theta \longleftarrow \theta - \eta\frac{\partial\mathcal{L}_f}{\partial\theta}. \tag{3.6}$$

3.2.2 Long Range Label Dependencies

Scene labeling methods leverage long range label dependencies in some way. The most common approach is to add some kind of graphical model (*e.g.* CRF) over local decisions, such that a certain global coherence is maintained. In the case of convolutional networks, an obvious way to efficiently capture long range dependencies would be to consider large input patches when labeling a pixel. However, this approach might face generalization issues, as considering larger context often implies considering larger models (*i.e.* higher capacity).

In Table 3.2, we review possible ways to control the capacity of a convolutional neural network by assuming a large input context. The easiest way is probably to increase the filter sizes in pooling layers, reducing the overall number of parameters in the network. However, performing large poolings decreases the network label output resolution (*e.g.*, if one performs a $1/8$ pooling, the label output plane size will be about $1/8^{th}$ of the input image size). As shown later in Section 3.2.4, this problem could be overcome at the cost of a slow inference process.

Yet another approach would be the use of a multiscale convolutional network (Farabet et al., 2013). Large contexts are integrated into local decisions while making the model still manageable in terms of parameters/dimensionality. Label coherence can then be increased by leveraging, for instance, superpixels.

Another way to consider a large input context size while controlling the capacity of the model is to make the network recurrent. In this case, the architecture might be very deep (with many convolution layers), but parameters between several layers at various depths are shared. We

	Capacity control	Speed
graphical model	–	slow
multiscale	scale down input image	fast
large input patches	increase pooling	slow
	recurrent architecture	fast

Table 3.2 – Long range pixel label dependencies integration in CNN-based scene labeling models. Methods to control capacity and speed of each architecture is reported.

will now detail our recurrent network approach.

3.2.3 Recurrent Network Approach

The recurrent architecture (see Figure 3.2) consists of the composition of P instances of the "plain" convolutional network $f(\cdot)$ introduced in Section 3.2.1. Each instance has identical (shared) trainable parameters θ. For clarity, we drop the θ notation in subsequent paragraphs. The p^{th} instance of the network ($1 \leq p \leq P$) is fed with an input "image" $\mathbf{F}^p$ of $N+3$ feature maps:

$$\begin{aligned} \mathbf{F}^p &= [f(\mathbf{F}^{p-1}), I^p_{i,j,k}], \\ \mathbf{F}^1 &= [\mathbf{0}, I_{i,j,k}], \end{aligned} \tag{3.7}$$

which are the output label planes of the previous instance, and the scaled[2] version of the raw RGB squared patch surrounding the pixel at location (i, j) of the training image k. Note that the first network instance takes 0 label maps as previous label predictions.

As shown in Figure 3.2, the size of the input patch $I_{i,j,k}$ needed to label one pixel increases with the number of compositions of f. However, the capacity of the system remains constant, since the parameters of each network instance are shared.

The system is trained by maximizing the likelihood

$$\mathcal{L}(f) + \mathcal{L}(f \circ f) + \ldots + \mathcal{L}(f \circ^P f), \tag{3.8}$$

where $\mathcal{L}(f)$ is a shorthand for the likelihood introduced in (3.5) in the case of the plain CNN, and $\circ^p$ denotes the composition operation performed p times. This way, we ensure that each network instance is trained to output the correct label at location (i, j). In that respect, the system is able to learn to *correct its own mistakes* (made by earlier instances). It can also learn *label dependencies*, as an instance receives as input the label predictions made by the previous instance around location (i, j) (see Figure 3.2). Note that maximizing (3.8) is equivalent to

[2] $I^p_{i,j,k}$ is $I_{i,j,k}$ scaled to the size of $f(F^{p-1})$.

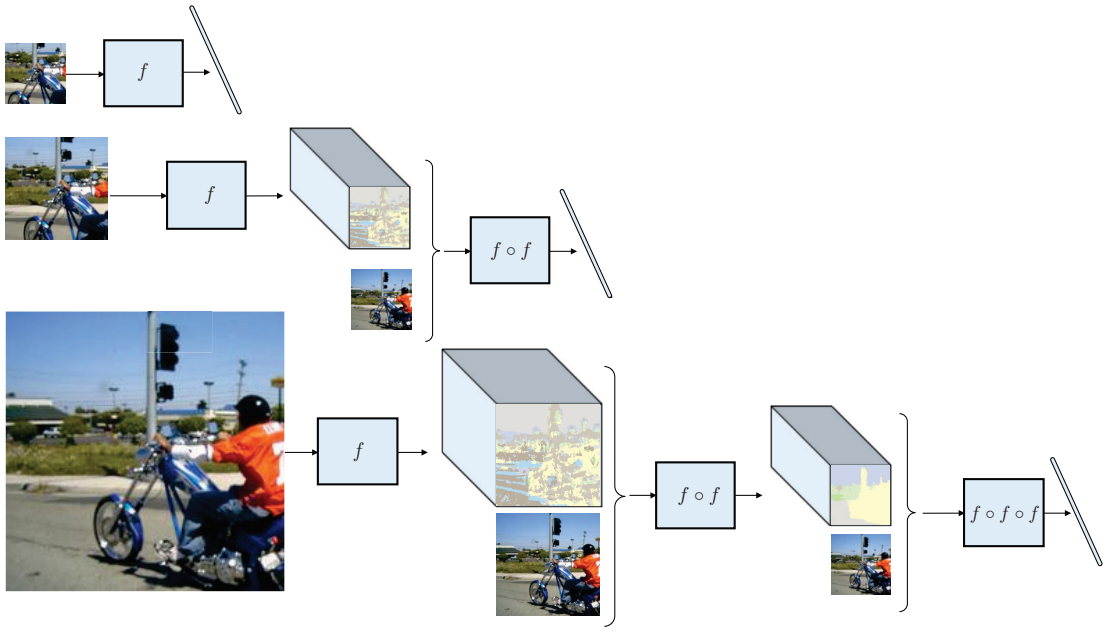


Figure 3.2 – Representation of the model considering one (f), two ($f \circ f$) and three ($f \circ f \circ f$) instances of the network. In all three cases, the architecture produces labels (1×1 output planes corresponding to the pixel at the center of the input patch. Each network instance is fed with the previous label predictions, as well as a RGB patch surrounding the pixel of interest. For space constraints, we do not show the label maps of the first instances, as they are zero maps. Adding network instances increases the context patch size seen by the architecture (both RGB pixels and previous predicted labels).

randomly alternating (with equal weight) the maximization of each likelihood $\mathcal{L}(f \circ^p f)$ (for $1 \leq p \leq P$). We chose this approach for simplicity of implementation.

The learning procedure is the same as for a standard CNN (stochastic gradient descent), where gradients are computed with the *Backpropagation Through Time (BPTT)* algorithm – the network is first unfolded as shown in Figure 3.2 and then the standard backpropagation algorithm is applied.

3.2.4 Scene Inference

Given a test image I_k, for each pixel at location (i, j) the network predicts a label as:

$$\hat{l}_{i,j,k} = \underset{c \in \{1,\ldots,C\}}{\operatorname{argmax}} \; p(c|I_{i,j,k}; \theta)\,, \tag{3.9}$$

considering the context patch $I_{i,j,k}$. Note that this implies padding the input image when inferring label of pixels close to the image border. In practice, simply extracting patches $I_{i,j,k}$ and then feeding them through the network for all pixels of a test image is computationally very inefficient. Instead, it is better to feed the full test image (also properly padded) to the

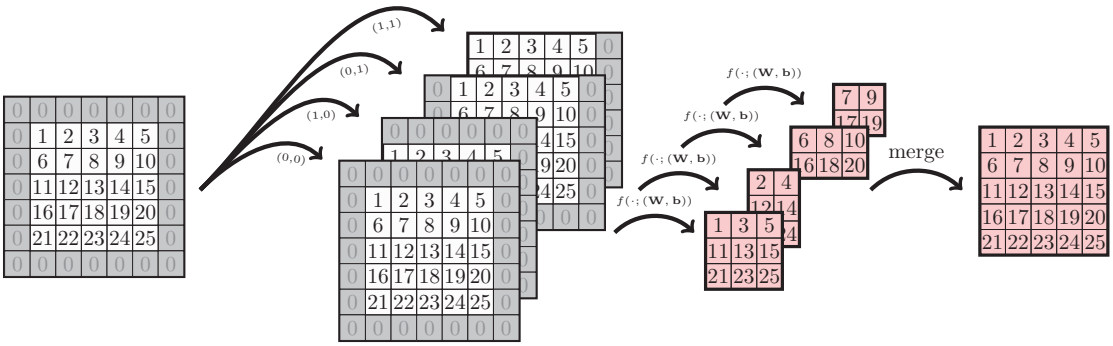


Figure 3.3 – Example of interleaving for efficient scene inference. Convolutional neural networks output downscaled label planes (compared to the input image) due to pooling layers. To alleviate this problem, one can feed several shifted version of the input image (here represented by pixels $1,\ldots,25$) in the X and Y axis. In this example the network is assumed to have a single 2×2 pooling layer. Downscaled predicted label planes (here in red) are then merged to get back the full resolution label plane in an efficient manner. Note that pixels represented by 0 are adequate padding.

convolutional network: applying one convolution to a large image is much faster than applying the same convolution many times to small patches. When fed with the full input image, the network will output a plane of label scores. However, following (3.1), the plane size is smaller than the input image size: this is mainly due to pooling layers, but also due to border effects when applying the convolution. For example, if the network includes two 2×2 pooling layers, only 1 every 4 pixels of the input image will be labeled. Most convolutional network users (*e.g.* (Farabet et al., 2013)) upscale the label plane to the input image size.

In fact, it is possible to compute efficiently the label plane with a fine resolution by interleaving, that is, feeding to the network several versions of the input image, shifted on the X and Y axis. Figure 3.3 shows an example for a network which would have only one 2×2 pooling layer, and one output plane: low resolution label planes (coming out of the network for the input image shifted by $(0,0)$, $(0,1)$, $(1,0)$ and $(1,1)$ pixels) are "merged" to form the high resolution label plane. Merging is a simple copy operation which matches a pixel in a low resolution label plane with the location of the corresponding original pixel to label in the (high resolution) input plane. The number of forwards is proportional to the number of pooling layers. However, this would be still much faster than forwarding patches at each location of the test image. We will see in Section 3.3.3 that having a finer label resolution can increase the classification performance.

3.3 Experimental Results

We tested our proposed method on two different fully-labeled datasets: the Stanford Background Dataset (Gould et al., 2009) and the SIFT Flow Dataset (Liu et al., 2011). The Stanford

dataset has 715 images from rural and urban scenes composed of 8 classes. The scenes have approximately 320 × 240 pixels. As in (Gould et al., 2009), we performed a 5-fold cross-validation with the dataset randomly split into 572 training images and 143 test images in each fold. The SIFT Flow is a larger dataset composed of 2688 images of 256 × 256 pixels and 33 semantic labels.

Each image of the training set was properly padded and normalized such that they have zero mean and unit variance. All networks were trained by sampling patches surrounding a randomly chosen pixel from a randomly chosen image from the training set. Contrary to (Farabet et al., 2013) (i) we did not consider addition of any distortion on the images[3], (ii) we did not use contrastive normalization and (iii) we did not sample training patches according to balanced class frequencies.

We considered two different accuracy measures to compare the performance of the our method with other approaches (see Section 2.1.1). The first one is the accuracy per pixel of test images. This measure is simply the ratio of correctly classified pixels of all images in the test set. However, in scene labeling (especially in datasets with large number of classes), classes which are much more frequent than others (*e.g.* the class 'sky' is much more frequent than class 'moon') have more impact on this measure. We also consider the averaged per class accuracy on the test set (all classes have the same weight in this measure). Note that as mentioned above, we did not train with balanced class frequencies, which would have optimized this second measure.

We consider three CNNs architectures. A "plain CNN_1" was designed to take large input patches. CNN_2 and CNN_3 architectures were designed such that their recurrent versions (with respectively two or three compositions) would still lead to a reasonable input patch size. We denote rCNN_i for the recurrent version of the regular convolutional network CNN_i. For rCNN_3, we show results considering both half resolution and full-resolution inference (see Section 3.2.4), in which we are able to achieve better results (at the cost of a higher computing time). Table 3.3 compares the performance of our architectures with related works on the Stanford Background dataset and Table 3.4 compares the performance on the SIFT Flow dataset. Note that the inference time in the second dataset does not change, since we exclude the need of any segmentation method. In the following, we provide additional technical details for each architecture used.

3.3.1 Plain Network

CNN_1 was trained with 133 × 133 input patches. The network was composed of a 6 × 6 convolution with nhu_1 output planes, followed by an 8 × 8 pooling layer, a $\tanh(\cdot)$ non-linearity, another 3 × 3 convolutional layer with nhu_2 output planes, a 2 × 2 pooling layer, a $\tanh(\cdot)$ non-linearity, and a final 7 × 7 convolution to produce label scores. The hidden units were chosen to be $nhu_1 = 25$ and $nhu_2 = 50$ for the Stanford dataset, and $nhu_1 = 50$ and $nhu_2 = 50$

[3] Which is known to improve the generalization accuracy by few extra percents.

	Pixel Accuracy (%)	Class Accuracy (%)	Computing Time (s)
Gould et al. (2009)	76.4	–	10 to 600
Tighe and Lazebnik (2010)	77.5	–	10 to 300
Munoz et al. (2010)	76.9	66.2	12
Kumar and Koller (2010)	79.4	–	< 600
Socher et al. (2011)	78.1	–	?
Lempitsky et al. (2011)	81.9	72.4	> 60
Farabet et al. (2013)*	78.8	72.4	0.6
Farabet et al. (2013)†	81.4	76.0	60.5
Plain CNN_1	79.4	69.5	15
CNN_2 ($\circ^1$)	67.9	58.0	0.2
$rCNN_2$ ($\circ^2$)	79.5	69.5	2.6
CNN_3 ($\circ^1$)	15.3	14.7	0.06
$rCNN_3$ ($\circ^2$)	76.2	67.2	1.1
$rCNN_3$ 1/2 resolution ($\circ^3$)	79.8	69.3	2.15
$rCNN_3$ 1/1 resolution ($\circ^3$)	80.2	69.9	10.7

* Multiscale CNN + superpixels
† Multiscale CNN + CRF

Table 3.3 – Pixel and averaged per class accuracy and computing time of other methods and our proposed approaches on the Stanford Background dataset. For recurrent networks, $\circ^n$ indicates the number of compositions.

for the SIFT Flow dataset.

3.3.2 Recurrent Architectures

We consider two different recurrent convolutional network architectures.

The first architecture, $rCNN_2$, is composed of two consecutive instances of the convolutional network CNN_2 with shared parameters (system in the center of Figure 3.2). CNN_2 is composed of an 8×8 convolution with 25 output planes, followed by a 2×2 pooling layer, a $\tanh(\cdot)$ non-linearity, another 8×8 convolutional layer with 50 output planes, a 2×2 pooling layer, a $\tanh(\cdot)$ non-linearity, and a final 1×1 convolution to produce N label scores. As described in Section 3.2.3, $rCNN_2$ is trained by maximizing the likelihood given in (3.8). As shown in Figure 3.2, the input context patch size depends directly on the number of network instances in the recurrent architecture. In the case of $rCNN_2$, the input patch size is 25×25 when considering one instance (f) and 121×121 when considering two network instances ($f \circ f$).

The second recurrent convolutional neural network $rCNN_3$ is composed of a maximum of three instances of the convolutional network CNN_3 with shared parameters. Each instance of

	Pixel Accuracy (%)	Class Accuracy (%)
Liu et al. (2011)	76.67	–
Tighe and Lazebnik (2010)	77.0	30.1
Farabet et al. (2013)	78.5	29.6
Plain CNN_1	76.5	30.0
CNN_2 ($\circ^1$)	51.8	17.4
$rCNN_2$ ($\circ^2$)	76.2	29.2
$rCNN_3$ ($\circ^2$)	65.5	20.8
$rCNN_3$ ($\circ^3$)	77.7	29.8

Table 3.4 – Pixel and averaged per class accuracy of other methods and our proposed approaches on the SIFT Flow dataset. For recurrent networks, $\circ^n$ indicates the number of compositions.

CNN_3 is composed of a 8×8 convolution with 25 output planes, followed by a 2×2 pooling layer, a $\tanh(\cdot)$ non-linearity, another 8×8 convolution with 50 planes and a final 1×1 convolution which outputs the N label planes. Following 3.8, we aim at maximizing

$$\mathcal{L}(f)+\mathcal{L}(f\circ f)+\mathcal{L}(f\circ f\circ f)\,. \tag{3.10}$$

This appeared too slow to train on a single computer in the case of $rCNN_3$. Instead, we initialized the system by first starting training with two network instances (maximizing $\mathcal{L}(f\circ f)$). We then switched to the training of the full cost function (3.10). The input patch size is 23×23, 67×67 and 155×155 when considering one, two or three instances of the network (f, $f\circ f$ and $f\circ f\circ f$), respectively. In all cases, the learning rate in (3.6) was equal to 10^{-4}. All hyper-parameters were tuned with a 10% held-out validation data.

Figure 3.4 and Figure 3.5 illustrate inference of the recurrent network $rCNN_2$ with one and two instances for images form Stanford and SIFT Flow dataset, respectively. It can be seen that the network learns by itself how to correct its own label prediction.

3.3.3 Inference Time and Performance

In Table 3.5, we analyze the trade off between inference time and test accuracy by running several experiments with different output resolutions for recurrent network $rCNN_3$ (see Section 3.2.4 and Figure 3.3). Labeling about $1/4^{th}$ of the pixels seems to be enough to lead to near state-of-the-art performance, while keeping a very fast inference time.

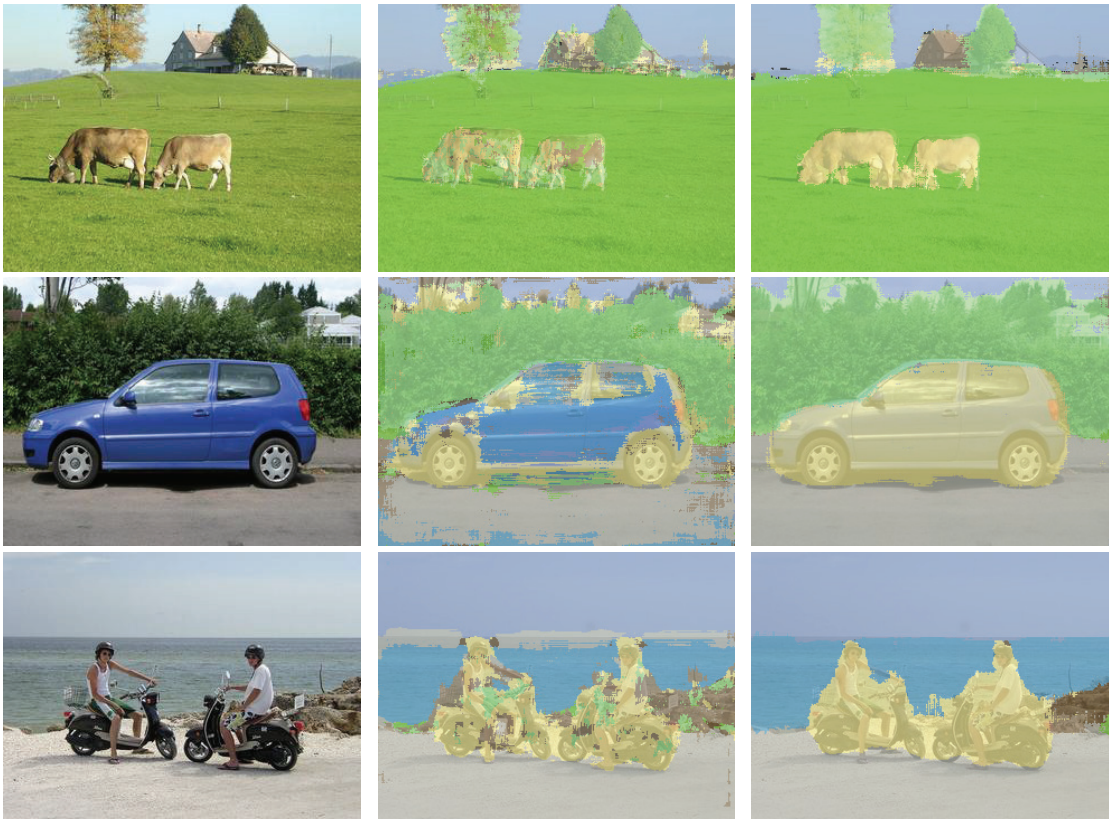


Figure 3.4 – Qualitative results on Stanford dataset. The test image is shown on the first column. The two next columns illustrates the output of $rCNN_2$ with one and two instances, respectively. Most mistakes of first instance are corrected on the second one.

Output Resolution	Computing Time Per Image	Pixel Accuracy
1/8	0.20s	78.4%
1/4	0.70s	79.3%
1/2	2.15s	79.8%
1/1	10.68s	80.2%

Table 3.5 – Inference time and performance in per-pixel accuracy for the recurrent convolutional network $rCNN_3$ with different label resolution on the Stanford dataset. Our algorithms were run on a 4-core Intel i7.

3.4 Summary

In this chapter, we presented a novel approach for full scene labeling based on supervised deep learning strategies which model in a rather simple way non-local class dependencies in a scene from raw pixels. We demonstrated that the problem of scene labeling can be effectively achieved without the need of any expensive graphical model or segmentation technique to

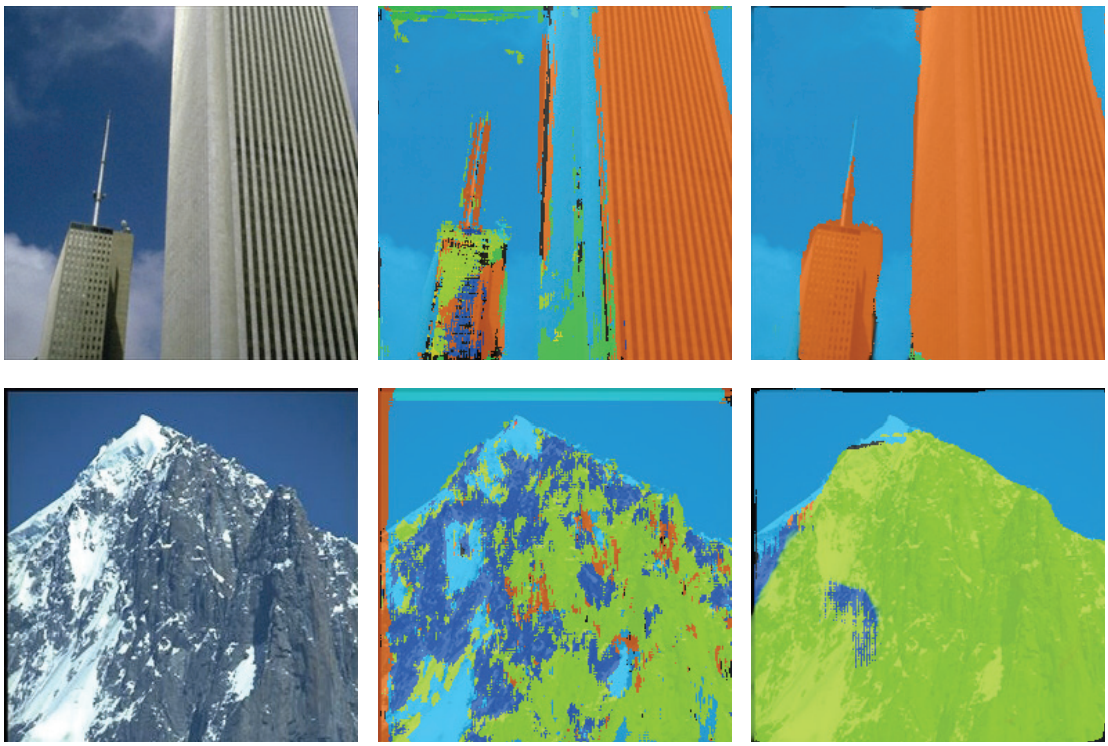


Figure 3.5 – Qualitative results on SIFT Flow dataset. The test image is shown on the first column. The two next columns illustrates the output of $rCNN_2$ with one and two instances, respectively. Most mistakes of first instance are corrected on the second one.

ensure labeling. The scene labeling is inferred simply by forward evaluation of a function applied to a RGB image. In terms of accuracy, our system achieves state-of-the-art results on both Stanford Background and SIFT Flow datasets, while keeping a fast inference time.

4 Learning to Segment with Image-Level Label

Segmenting objects is extremely challenging. Each object in the world generates an infinite number of images with variations in position, pose, lightning, texture, geometrical form and background. Natural image segmentation systems have to cope with these variations, while being limited in the amount of available training data. Increasing computing power, and recent releases of reasonably large segmentation datasets such as PASCAL VOC (Everingham et al., 2010) and MS COCO (Lin et al., 2014) have nevertheless made the segmentation task a reality.

Convolutional neural networks (LeCun et al., 1990, 1998) achieve state-of-the-art results on large object recognition tasks (Krizhevsky et al., 2012; Szegedy et al., 2015; Farabet et al., 2013). A big advantage of CNNs is that they learn sufficiently general features, and therefore they can excel in transfer learning: *e.g.* CNN models trained on the ImageNet classification dataset (Deng et al., 2009) could be exploited for different vision tasks (Girshick et al., 2014; Oquab et al., 2014; Hariharan et al., 2014). Their main disadvantage, however, is the need of a large number of fully-labeled dataset for training. Given that classification labels are much more abundant than segmentation labels, it is natural to find a bridge between classification and segmentation, which would transfer efficiently learned features from one task to the other one.

In the previous chapter, we developed a method to learn how to give a label to every pixel in a scene in a fully supervised way. In this chapter, the proposed CNN-based model is not trained with segmentation labels, nor bounding box annotations. Instead, we only consider a single object class label for a given image, and the model is constrained to put more weight on important pixels for classification. This approach can be seen as an instance of *Multiple Instance Learning (MIL)* (Maron and Lozano-Pérez, 1998). In this context, every image is known to have (or not) – through the image class label – one or several pixels matching the class label. However, the positions of these pixels are unknown, and have to be inferred. This learning paradigm is called *weakly supervised* learning.

Because of computing power limitations, we built our model over the *Overfeat* feature extractor,

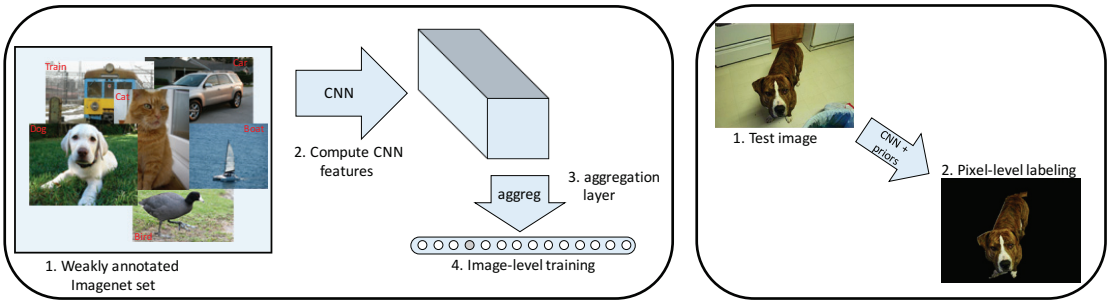


Figure 4.1 – A schematic illustration of our method. **(Left)**: (1) The model is trained using weakly annotated data (only image-level class information) from ImageNet. (2) The CNN generates feature planes. (3) These planes pass through an aggregation layer to constrain the model to put more weight on the right pixels. (4) The system is trained by classifying the correct image-level label. **(Right)**: During test time, the aggregation layer is removed and the CNN densely classifies every pixel of the image (considering only few segmentation priors).

developed by Sermanet et al. (2014). This feature extractor correspond to the first layers of a CNN, well-trained over ImageNet. Features are fed into few extra convolutional layers, which forms our "segmentation network".

Training is achieved by maximizing the classification likelihood over the classification training set (we consider a subset of ImageNet), by adding an extra layer to our network, which constrains the model to put more weight on pixels which are important for the classification decision. At test time, the constraining layer is removed, and the label of each image pixel is efficiently inferred. Figure 4.1 shows a general illustration of our approach.

4.1 Related Work

Weakly supervised semantic segmentation Labeling data for segmentation task is difficult if compared to labeling data for classification. For this reason, several weakly supervised semantic segmentation systems have been proposed in the past few years. For instance, Vezhnevets and Buhmann (2010) proposed an approach based on Semantic Texton Forest (Shotton et al., 2008), derived in the context of MIL. However, the method fails to model relationship between superpixels. In order to model these relationships, Vezhnevets et al. (2011) introduced a graphical model – named Multi-Image Model (MIM) – to connect superpixels from all training images, based on their appearance similarity. The unary potentials of the MIM are initialized with the output of Vezhnevets and Buhmann (2010).

In (Vezhnevets et al., 2012), the authors define a parametric family of structured models, where each model carries visual cues in a different way. A maximum expected agreement model selection principle evaluates the quality of a model from a family. An algorithm based on Gaussian processes is proposed to efficiently search the best model for different visual cues.

Zhang et al. (2014) proposed an algorithm that learns the distribution of spatially structural superpixel sets from image-level labels. This is achieved by first extracting *graphlets* (small graphs consisting of superpixels and encapsulating their spatial structure) from a given image. Labels from the training images are transfered into graphlets throughout a proposed manifold embedding algorithm. A Gaussian mixture model is then used to learn the distribution of the post-embedding graphlets, *i.e.* vectors output from the graphlet embedding. The inference is done by leveraging the learned GMM prior to measure the structure homogeneity of a test image.

In contrast with previous approaches for weakly supervised segmentation, we avoid designing task-specific features for segmentation. Instead, a CNN learns the features: the model is trained through a cost function which casts the problem of segmentation into the problem of finding pixel-level labels from image-level labels. As we will see in Section 4.3, learning the right features for segmentation leads to better performance compared to existing weakly supervised segmentation systems. Another difference from our approach is that we train our model in a different dataset (ImageNet) from the one we validate the results (PASCAL VOC).

Transfer Learning and CNNs In the last few years, convolutional networks have been widely used in the context of object recognition. A notable system is the one from (Krizhevsky et al., 2012), which performs very well on ImageNet. Oquab et al. (2014) built upon Krizhevsky's approach and showed that a model trained for classification on the ImageNet dataset can be used for classification in a different dataset (namely PASCAL VOC) by taking into account the bounding box information. Oquab et al. (2015) adapt an ImageNet-trained CNN to the PASCAL VOC classification task. The network is fine-tuned on PASCAL VOC, by modifying the cost function to include a final max-pooling layer. Similar to our aggregation layer, the max-pooling outputs a single image-level score for each of the classes. In contrast, (1) we do not limit ourselves to the PASCAL VOC classification problem, but tackle the more challenging problem of segmentation and (2) our model is not fine-tuned on PASCAL VOC.

In the same spirit, Girshick et al. (2014) showed that a model trained for classification on ImageNet can be adapted for object detection on PASCAL VOC. The authors proposed to combine bottom-up techniques for generating detection region candidates with pre-trained CNNs. The authors achieved state-of-the-art performance in object detection. Based upon this work, Hariharan et al. (2014) derived a model that detects all instances of a category in an image and, for each instance, marks the pixels that belong to it. Their model, entitled Simultaneous Detection and Segmentation (SDS), uses category-specific, top-down figure-ground predictions to refine bottom-up detection candidates.

As for these existing state-of-the-art approaches, our system leverages features learned over the ImageNet classification dataset. However, our approach differs from theirs in some important aspects. Compared to (Girshick et al., 2014; Oquab et al., 2014), we consider the more challenging problem of object segmentation and do not use any information other than

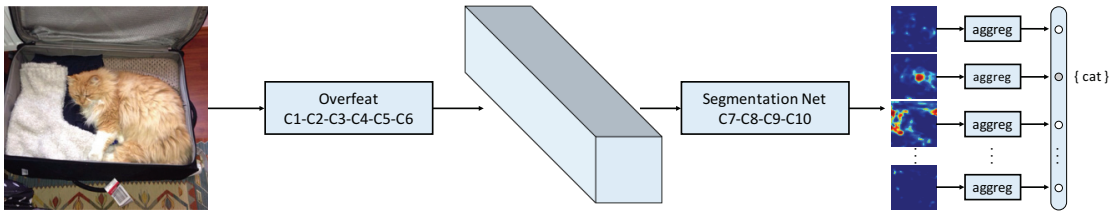


Figure 4.2 – Outline of the proposed architecture. The full RGB image is forwarded through the network (composed of Overfeat and four extra convolutional features), generating output planes of dimension $(C+1) \times h^o \times w^o$. These output planes can be seen as pixel-level labels of a sub-sampled version of the input image. The output then passes through a *aggrerg* layer to aggregate pixel-level labels into image-level ones. The error is backpropagated through layers C10-C7.

the image-level annotation. Oquab et al. (2015) consider a weakly supervised scenario, but only deal with the classification problem. Compared to (Hariharan et al., 2014), we consider only the image-level annotation to infer the pixel-level one. In that respect, we do not use any segmentation information (our model is not refined over the segmentation data either), nor bounding box annotation during the training period. One could argue that a classification dataset like ImageNet has somewhat already cropped properly objects. While this might be true for certain objects, it is not the case for many images, and in any case the bounding box remains quite loose.

After the publication of this work in 2015, many different weakly supervised semantic segmentation methods appeared on the literature. For example Pathak et al. (2015) considers to use each image-level tag as a constraint in the loss function of the CNN. Papandreou et al. (2015) also proposes a CNN-based approach that constraint the loss using an expectation-maximization algorithm. Saleh et al. (2016) propose to extract a foreground/background mask by directly exploiting the unit activations of some of the hidden layers in the network. Tokmakov et al. (2016) propose to leverage segmentation information from weakly annotated videos, using motion segment as soft constraints. Kolesnikov and Lampert (2016) achieve surprising results using a three step pipeline model: first, they use a pre-trained CNN to find weak localization cues, second they expand objects based on the information of which object are present in an image, then they constraint these segmentation using a CRF.

4.2 From Image-level to Pixel-level labeling

As we pointed out, CNNs are very flexible models which can be applied on various image processing tasks, as they alleviate the need for task-specific features. CNNs learn a hierarchy of filters, which extract higher level of representations as one goes “deeper” in the hierarchy (Zeiler and Fergus, 2014). The type of features they learn is also sufficiently general that CNNs make transfer learning (to another task) quite easy. The main drawback of these models, however, is that a large amount of data is necessary during training.

Since the number of image-level object labels is much bigger than pixel-level segmentation labels, it is thus natural to leverage image classification datasets for performing segmentation. In the following, we consider a problem of segmentation with a set of classes $\mathcal{C} = \{1, \ldots, C\}$. We assume the classification dataset contains at least the same classes. Extra classes available at classification time, but which are not in the segmentation dataset are mapped to a "background" class. This background class is essential to limit the number of false positive during segmentation.

Our architecture is a CNN, which is trained over a subset of ImageNet, to produce pixel-level labels from image-level labels. As shown in Figure 4.2, our CNN is quite standard, with 10 levels of convolutions and (optional) pooling. It takes as input a 400×400 RGB patch I, and outputs $C+1$ planes (one per class, plus the background class) corresponding to the score of the 12-times downsampled image pixels labels. During training, an extra layer, described in Section 4.2.1, aggregates pixel-level labels into an image-level label. For computational reasons, we "froze" the first layers of our CNN, to the ones of some already well-trained (over ImageNet classification data) CNN model.

We pick Overfeat (Sermanet et al., 2014), trained to perform object classification on the ILSVRC13 challenge (a subset of ImageNet). The Overfeat model generates feature maps of dimensions $1024 \times h^i \times w^i$, where h^i and w^i are functions of the size of the RGB input image, the convolution kernel sizes, convolution strides and max-pooling sizes. Keeping only the first 6 convolution layers and 2 pooling layers of Overfeat, our RGB 400×400 image patch I is transformed into a $1024 \times 29 \times 29$ feature representation.

We add four extra convolutional layers (we denote $\mathbf{H}^6$ for feature planes coming out from OverFeat. Each of them (but the last one $\mathbf{Y}$) is followed by a pointwise rectification non-linearity (ReLU):

$$\begin{aligned} \mathbf{H}^p &= \max(0, \mathbf{W}^p \mathbf{H}^{p-1} + \mathbf{b}^p), \ p \in \{7,8,9\}, \\ \mathbf{Y} &= \mathbf{W}^{10} \mathbf{H}^9 + \mathbf{b}^{10}. \end{aligned} \tag{4.1}$$

The parameters of the p^{th} layer are denoted with $(\mathbf{W}^p, \mathbf{b}^p)$. On this step, we do not use any max-pooling. A dropout regularization strategy (Srivastava et al., 2014) is applied on all layers. The network outputs $C+1$ feature planes of dimensions $h^o \times w^o$, one for each class considered on training, plus background.

4.2.1 Multiple Instance Learning

The network produces one score $s^k_{i,j} = Y^k_{i,j}$ for each pixel location (i, j) from the subsampled image I, and for each class $k \in \mathcal{C}$. Given that at training time we have only access to image classification labels, we need a way to aggregate these pixel-level scores into a single image-level classification score $s^k = aggreg_{i,j}(s^k_{i,j})$, that will then be maximized for the right class label $k^\star$. Assuming an aggregation procedure $aggreg(\cdot)$ is chosen, we interpret image-level

class scores as class conditional probabilities by applying a softmax function (Bridle, 1990):

$$p(k|I,\theta) = \frac{e^{s^k}}{\sum_{c\in\mathcal{C}} e^{s^c}}, \tag{4.2}$$

where $\theta = \{\mathbf{W}^p, \mathbf{b}^p\}\,\forall\, p$ represents all the trainable parameters of our architecture. We then maximize the log-likelihood (with respect to θ), over all the training dataset pairs $(I, k^\star)$:

$$\mathcal{L}(\theta) = \sum_{(k^\star,I)} \left[s^{k^\star} - \log \sum_{c\in\mathcal{C}} e^{s^c} \right]. \tag{4.3}$$

Training is achieved with stochastic gradient, backpropagating through the softmax, the aggregation procedure, and up to the first non-frozen layers of our network.

Aggregation

The aggregation should drive the network towards correct pixel-level assignments, such that it could perform decently on segmentation tasks. An obvious aggregation would be to take the sum over all pixel positions:

$$s^k = \sum_{i,j} s^k_{i,j} \quad \forall\, k \in \mathcal{C}. \tag{4.4}$$

This would however assign the same weight on all pixels of the image during the training procedure, even to the ones which do not belong to the class label assigned to the image. Note that this aggregation method is equivalent to applying a traditional fully-connected classification CNN with a mini-batch. Indeed, each value in the $h^o \times w^o$ output plane corresponds to the output of the CNN fed with a sub-patch centered around the correspond pixel in the input plane. At the other end, one could apply a max pooling aggregation:

$$s^k = \max_{i,j} s^k_{i,j} \quad \forall\, k \in \mathcal{C}. \tag{4.5}$$

This would encourage the model to increase the score of the pixel which is considered as the most important for image-level classification. In our experience, this type of approach does not train very well. Note that at the beginning of training all pixels might have the same (wrong) score, but only one (selected by the max) will have its score increased at each step of the training procedure. It is thus not surprising it takes an enormous amount of time to the model to converge.

We chose instead a smooth version and convex approximation of the *max* function, called *Log-Sum-Exp (LSE)* (Boyd and Vandenberghe, 2004):

$$s^k = \frac{1}{r} \log \left[\frac{1}{h^o\, w^o} \sum_{i,j} \exp(r\, s^k_{i,j}) \right]. \tag{4.6}$$

The hyper-parameter r controls how smooth one wants the approximation to be: high r values imply having an effect similar to the max, very low values will have an effect similar to the score averaging. The advantage of this aggregation is that pixels having similar scores will have a similar weight in the training procedure, r controlling this notion of "similarity".

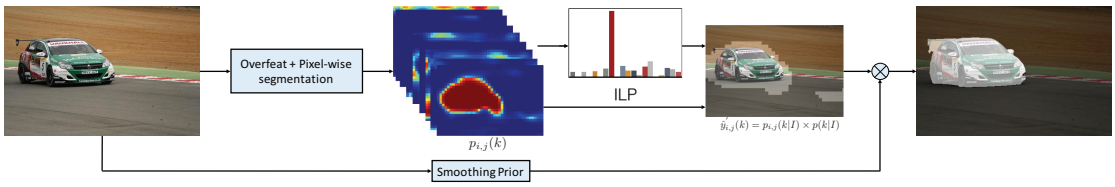


Figure 4.3 – Inference Pipeline. The test image is forwarded through the segmentation network to generate a $(C+1) \times h \times w$ output, one plane for each class. The image-level prior is extracted from these planes and the class of each pixel is selected by taking the maximum probability for each pixel. A smoothing prior is also considered to generate a smoother segmentation output.

4.2.2 Inference

At test time, we feed the padded and normalized RGB test image I (of dimension $3 \times h \times w$) to our network, where the aggregation layer has been removed. We thus obtain $C+1$ planes of pixel-level scores $s^k_{i,j}$ ($1 \leq i \leq h^o$, $1 \leq j \leq w^o$). For convenience, we transform these scores into conditional probabilities $p_{i,j}(k|I)$ using a softmax over each location (i, j).

Due to the pooling layers in the CNN, the output planes labels correspond to a sub-sampled version of the input test image. As shown in previous chapter, one can efficiently retrieve the label of all pixels of the image using a CNN model, by simply shifting the input image in both spatial directions, and forwarding it again through the network.

Adding Segmentation Priors

Given we do not fine-tune our model on segmentation data, we observed our approach is subject to false positive. To circumvent this issue, we consider simple post-processing techniques, namely Image-Level Prior (ILP) and three different Smoothing Prior (SP), with increasing amount of information. Figure 4.3 summarizes the pipeline of our approach during inference time.

Image-Level Prior The model makes inference using local context based on the patch surrounding a pixel. In order to improve the overall per-pixel accuracy, we add the global context information of the scene into play. We propose the use of an ILP (Shotton et al., 2008; Vezhnevets and Buhmann, 2010) based on the output feature planes. This prior, which is extracted from the trained network, is important to reduce the number of false positives generated by the model. As at training time, the probability $p(k|I)$ of each class $k \in \mathcal{C}$ to be present in the scene can be computed by applying the softmax in the LSE score of each label plane. This

probability is used as the image-level prior to encourage the likely categories and discourage the unlikely ones.

The ILP is integrated into the system by multiplying each conditional probability $p_{i,j}(k|I)$ by its class ILP, that is:

$$\hat{y}'_{i,j}(k) = p_{i,j}(k|I) \times p(k|I)\,, \tag{4.7}$$

for each location (i, j) and class $k \in \mathcal{C}$.

Smoothing Prior Predicting the class of each pixel independently from its neighbors yields noisy predictions. In general, objects have smooth boundaries and well defined shapes, different from the background which tends to be amorphous regions. At test time we considered three different approaches (of increasing prior knowledge) to impose local regions with strong boundaries to be assigned to the same label:

(i) *SP-sppxl* smooths the output using standard superpixels. We followed the method proposed by (Felzenszwalb and Huttenlocher, 2004), which largely over-segments a given image into a set of disjoint components. Prediction smoothing is achieved by simply picking the label that appears the most in each superpixel.

(ii) *SP-bb* leverages *bounding box proposals* to improve the smoothing. We picked the BING algorithm (Cheng et al., 2014) to generate a set of 10^4 (possibly overlapping) bounding box proposals given an image, each bounding box having a score. These scores are normalized to fit the $[0,1]$ interval. Each pixel (i, j) in the image is assigned a score (of belonging to an object) by summing the score of all bounding box proposals that contains the pixel. The score at each pixel is then converted into a probability $p((i, j) \in \text{Obj})$ by normalizing the sum by the number of boxes containing the pixel. Label smoothing for each pixel (i, j) is then achieved with:

$$\hat{y}_{i,j} = \begin{cases} k, & \text{if } \max\limits_{k \in \mathcal{C}} \quad \hat{y}'_{i,j}(k) \times p((i, j) \in \text{Obj}) > \delta_k \\ 0, & \text{otherwise} \end{cases}, \tag{4.8}$$

where δ_k $(0 \leq \delta_k < 1)$ is a per-class confidence threshold and $\hat{y}_{i,j} = 0$ means that the background class is assigned to the pixel.

(iii) *SP-seg* is a smoothing prior which has been trained with *class-independent* segmentation labels. We consider the Multiscale Combinatorial Grouping (MCG) algorithm (Arbeláez et al., 2014), which generates a series of overlapping object candidates with a corresponding score. Pixel label smoothing is then achieved in the same way as in *SP-bb*.

The smoothing prior improves our algorithm in two ways: (i) it forces pixels with low proba-

bility of being part of an object to be labeled as background and (ii) it guarantees local label consistency. While the former reduces the number of false positives, the latter increases the number of true positives. We will see in Section 4.3 that (as it can be expected) more complex smoothing priors improve performance accuracy.

4.3 Experiments

Given that our model uses only weak supervision labels (class labels), and is never trained with segmentation data, we compare our approach with current state-of-the-art *weakly supervised* segmentation systems. We also compare it against state-of-the-art *fully supervised* segmentation systems, to demonstrate that weakly supervised segmentation is a promising and viable solution.

4.3.1 Datasets

We considered the PASCAL VOC dataset (Everingham et al., 2010) as a benchmark for segmentation. This dataset includes 20 different classes, and represents a particular challenge as an object segmentation task. The objects from these classes can appear in many different poses, possibly highly occluded, and also possess a very large intra-class variation. The dataset was only used for testing purposes, not for training.

We created a large classification training set from the ImageNet dataset containing images of each of the twenty classes and also an extra class labeled as background – set of images in which none of the classes appear. We consider all the sub-classes located below each of the twenty classes in the full ImageNet tree, for a total of around 700,000 samples. For the background, we chose a subset of ImageNet consisting of a total of around 60,000 images not containing any of the twenty classes[1]. To increase the size of the training set, jitter (horizontal flip, rotation, scaling, brightness and contrast modification) was randomly added to each occurrence of an image during the training procedure. Each image was then normalized for each RGB channel. No other preprocessing was done during training.

4.3.2 Experimental Setup

Each training sample consists of a central patch of size 400×400 randomly extracted from a deformed image in the training set. If the image dimensions are smaller than 400×400, it is rescaled such that its smaller dimension is of size 400.

The first layers of our network are extracted (and "frozen") from the public available Overfeat[2] model. In all our experiments, we use the *slow* Overfeat model, as described in (Sermanet

[1] 60K background images might look surprisingly not large, but we found not easy to pick images where none of the 20 PASCAL VOC classes were not present.

[2] http://cilvr.nyu.edu/doku.php?id=software:overfeat:start

Conv. Layer	1	2	3	4
# channels	1024	768	512	21
Filter Size	3×3	3×3	3×3	3×3
Input Size	29×29	27×27	25×25	23×23

Table 4.1 – Architecture of the segmenter network used in our experiments.

Model	VOC2008	VOC2009	VOC2010
MIM	8.11%	38.27%	28.43%
GMIM	9.24%	39.16%	29.71%
PGC	30.12%	43.37%	32.14%
aggreg-max	44.31%	45.46%	45.88%
aggreg-sum	47.54%	50.01%	50.11%
aggreg-LSE	**56.25**%	**57.01**%	**56.12**%

Table 4.2 – Averaged per-class accuracy of weakly supervised models and ours for different PASCAL VOC datasets. We consider three different aggregation layers.

et al., 2014). With the 400×400 RGB input image, the Overfeat feature extractor outputs 1024 feature maps of dimension 29×29. As detailed in Section 4.2, these feature maps are then fed into 4 additional convolutional layers followed by ReLU non-linearity. A dropout procedure with a rate of 0.5 is applied on each layer. The whole network has a total of around 20 million parameters. Table 4.1 details the architecture used in our experiments.

The final convolution layer outputs a 21 feature maps of dimension 21×21. These feature maps are passed through the aggregation layer (in the case of LSE, we consider $r = 5$), which outputs 21 scores, one for each class (plus background). These scores are then transformed into posterior probabilities through a *softmax* layer.

Design architecture and hyper-parameters were chosen considering the validation data of the PASCAL VOC 2012 segmentation dataset. We considered a learning rate $\lambda = 0.001$ which decreases by a factor of 0.8 for every 5 million examples seen by the model. We trained our model using stochastic gradient descent with a batch size of 16 examples, momentum 0.9 and weight decay of 0.00005.

The optimal class confidence thresholds δ_k for smoothing priors (see Section 4.2.2) were chosen through a grid search. The AP changes in function of the confidence threshold for each class. The different values for the threshold is due to the variability of each class in the training data and how their statistics approach the PASCAL VOC images statistics.

Our network takes about a week to train on a Nvidia GeForce Titan GPU with 6GB of memory.

4.3.3 Experimental Results

Compared to weakly supervised models

We compare the proposed algorithm with three state-of-the-art approaches in weakly supervised segmentation scenario: (i) Multi-Image Model (MIM) (Vezhnevets et al., 2011), (ii) a variant, Generalized Multi-Image Model (GMIM) (Vezhnevets et al., 2012) and (iii) the most recent Probabilistic Graphlet Cut (PGC) (Zhang et al., 2014, 2013). Note that there are variations in the experimental setup on the experiments. The compared models use PASCAL VOC for weak supervision while we use ImageNet. Also, (iii) considers additional labels on the data. In our training framework, the PASCAL VOC dataset was used only for selecting the thresholds on the class priors. Our system learns features that are independent of the PASCAL VOC data distribution and would a priori yields similar results in other datasets.

Table 4.2 reports the results of the three compared models and our approach. In our experiments, we consider the *ILP* and the *SP-sppxl* smoothing prior, which does not take into account any segmentation or bounding box information. We consider the three aggregation layers described in Section 4.2.1. This result empirically demonstrates our choice of the *Log-Sum-Exp* layer.

The results for the compared models reported on this table are from (Zhang et al., 2014). We use the same metric and evaluate on the same datasets (PASCAL VOC 2008, 2009 and 2010) as the authors. The metric used, average per-class accuracy, is defined by the ratio of correct classified pixels of each class. We show that our model achieves significantly better results than the previous state-of-the-art weakly supervised algorithms, with an increase from 30% to 90% in average per-class accuracy.

Compared to fully supervised models

In table 4.3, we compare the performance of our model against the best performers in PASCAL VOC 2012 segmentation competition[3]: Second Order Pooling (O_2P) (Carreira et al., 2012), DivMBest (Yadollahpour et al., 2013) and Simultaneous Detection and Segmentation (SDS) (Hariharan et al., 2014). Average precision metric[4], as defined by the PASCAL VOC competition, is reported. We show results using the image-level prior and all three smoothing priors (as described in 4.2.2). The performance of our model increases as we consider more complex priors.

We reach near state-of-the-art performance for several classes (even with the simplest smoothing prior *SP-sppxl*, which is object and segmentation agnostic) while some other classes perform worse. This is not really surprising, given that the statistics of the images for some

[3] These were the leading methods on PASCAL VOC official evaluation server at the time this work was done. The fully supervised state of the art has now been vastly improved with CNN-based methods (*e.g.* (Long et al., 2015; Chen et al., 2015))

[4] $\mathrm{AP} = \frac{TruePositive}{TruePositive+FalsePositive+FalseNegative}$

	bgnd	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
Fully Sup.																						
O_2P	86.1	64.0	27.3	54.1	39.2	48.7	56.6	57.7	52.5	14.2	54.8	29.6	42.2	58.0	54.8	50.2	36.6	58.6	31.6	48.4	38.6	47.8
DivMBest	85.7	62.7	25.6	46.9	43.0	54.8	58.4	58.6	55.6	14.6	47.5	31.2	44.7	51.0	60.9	53.5	36.6	50.9	30.1	50.2	46.8	48.1
SDS	86.3	63.3	25.7	63.0	39.8	59.2	70.9	61.4	54.9	16.8	45.0	48.2	50.5	51.0	57.7	63.3	31.8	58.7	31.2	55.7	48.5	51.6
Weak. Sup.																						
Ours-sppxl	74.7	38.8	19.8	27.5	21.7	32.8	40.0	50.1	47.1	7.2	44.8	15.8	49.4	47.3	36.6	36.4	24.3	44.5	21.0	31.5	41.3	35.8
Ours-bb	76.2	42.8	20.9	29.6	25.9	38.5	40.6	51.7	49.0	9.1	43.5	16.2	50.1	46.0	35.8	38.0	22.1	44.5	22.4	30.8	43.0	37.0
Ours-seg	78.7	48.0	21.2	31.1	28.4	35.1	51.4	55.5	52.8	7.8	56.2	19.9	53.8	50.3	40.0	38.6	27.8	51.8	24.7	33.3	46.3	40.6

Table 4.3 – Per class average precision and mean average precision (mAP) on PASCAL VOC 2012 segmentation challenge *test set*. We consider different smoothing priors in our model.

classes (*e.g.* 'dog', 'cat', 'cow') are closer in the two different datasets than for some other classes (*e.g.* 'bird', 'person'). The results on the specific PASCAL VOC challenge could be improved by "cheating" and considering training images that are more similar to those represented on the test data (*e.g.* instead of choosing all bird images from ImageNet, we could have chosen the bird breeds that are similar to the ones presented on PASCAL VOC).

	bgnd	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
base	37.0	10.4	12.4	10.8	5.3	5.7	25.2	21.1	25.15	4.8	21.5	8.6	29.1	25.1	23.6	25.5	12.0	28.4	8.9	22.0	11.6	17.8
base+ILP	73.2	25.4	18.2	22.7	21.5	28.6	39.5	44.7	46.6	11.9	40.4	11.8	45.6	40.1	35.5	35.2	20.8	41.7	17.0	34.7	30.4	32.6
base+ILP+SP-sppxl	77.2	37.3	18.4	25.4	28.2	31.9	41.6	48.1	50.7	12.7	45.7	14.6	50.9	44.1	39.2	37.9	28.3	44.0	19.6	37.6	35.0	36.6
base+ILP+SP-bb	78.6	46.9	18.6	27.9	30.7	38.4	44.0	49.6	49.8	11.6	44.7	14.6	50.4	44.7	40.8	38.5	26.0	45.0	20.5	36.9	34.8	37.8
base+ILP+SP-seg	79.6	50.2	21.6	40.6	34.9	40.5	45.9	51.5	60.6	12.6	51.2	11.6	56.8	52.9	44.8	42.7	31.2	55.4	21.5	38.8	36.9	42.0

Table 4.4 – Effect of image-level and smoothing priors on segmentation results. Per class average precision on PASCAL VOC 2012 *validation set*. We consider the inference with no priors (base), with image-level prior (base+ILP) and different smoothing priors (base+ILP+SP-sppxl, base+ILP+SP-bb, base+ILP+SP-seg).

Effect of Priors

Table 4.4 shows the average precision of each class on the PASCAL VOC 2012 validation set considering the inference assuming no prior was used (base), only the image-level prior (base+ILP) and the image-level together with different smoothing priors (base+ILP+SP-sppxl, base+ILP+SP-bb, base+ILP+SP-seg). Figure 4.4 illustrates inference in PASCAL VOC images assuming different steps of inference. Priors have a huge importance to reduce false positives, and smooth predictions.

4.4 Summary

In this chapter, we proposed a way to segment objects with weakly supervision only. Our model is built on the top of a CNN pre-trained on ImageNet which is constrained during training to put more weight on pixels which are important for classifying images. Our algorithm is able to distinguish, at a pixel level, the differences between distinct classes, assuming only few simple

Figure 4.4 – Qualitative results. For each test image (left), we show the output assuming the image-level prior (center) and image-level and *SP-seg* smoothing prior (right).

prior knowledge about segmentation. This is an interesting result as one might circumvent the necessity of using the very costly segmentation datasets and use only image-level annotations. Our approach outperforms previously proposed models for weakly supervised segmentation, with an increase form 30% to 90% in average per-class accuracy, on PASCAL VOC dataset. We also achieve competitive performance (at least for several classes) compared to state-of-the-art fully supervised segmentation systems.

5 Learning to Generate Object Segments

Object detection is one of the most foundational tasks in computer vision. Until recently, the dominant paradigm in object detection was the sliding window framework: a classifier is applied at every object location and scale (Dalal and Triggs, 2005; Felzenszwalb et al., 2010; Viola and Jones, 2004). More recently, Girshick et al. (2014) proposed a two-phase approach. First, a rich set of object proposals (*i.e.*, a set of image regions which are likely to contain an object) is generated using a fast (but possibly imprecise) algorithm. Second, a convolutional neural network classifier is applied on each of the proposals. This approach provides a notable gain in object detection accuracy compared to classic sliding window approaches. Since then, most state-of-the-art object detectors for the PASCAL VOC (Everingham et al., 2010) ImageNet (Deng et al., 2009) and COCO (Lin et al., 2014) datasets rely on object proposals as a first preprocessing step (Girshick et al., 2014; Girshick, 2015; Ren et al., 2015; Bell et al., 2016; He et al., 2016).

In this chapter, we present an object proposal algorithm based on CNN (LeCun et al., 1998) that satisfies these constraints better than existing approaches. CNNs are an important class of algorithms which have been shown to be state of the art in many large scale object recognition tasks. They can be seen as a hierarchy of trainable filters, interleaved with non-linearities and pooling. Moreover, these models learn sufficiently general image features, which can be transferred to many different tasks (Girshick et al., 2014; Hariharan et al., 2015; Chen et al., 2015; Oquab et al., 2015).

Given an input image patch, our algorithm generates a class-agnostic mask and an associated score which estimates the likelihood of the patch fully containing a centered object (without any notion of an object category). The core of our model is a CNN which jointly predicts the mask and the object score. A large part of the network is shared between those two tasks: only the last few network layers are specialized for separately outputting a mask and score prediction. The model is trained by optimizing a cost function that targets both tasks simultaneously. We train on COCO and evaluate the model on two object detection datasets, PASCAL VOC and COCO.

By leveraging powerful CNN feature representations trained on ImageNet and adapted on the large amount of segmented training data available in COCO, we are able to beat the state of the art in object proposals generation under multiple scenarios. Our most notable achievement is that our approach beats other methods by a large margin while considering a smaller number of proposals. Moreover, we demonstrate the generalization capabilities of our model by testing it on object categories not seen during training. Finally, unlike all previous approaches for generating segmentation proposals, we do not rely on edges, superpixels, or any other form of low-level segmentation. This approach is the first to learn to generate segmentation proposals directly from raw image data.

5.1 Related Work

In recent years, CNNs have been widely used in the context of object recognition. Notable systems are AlexNet (Krizhevsky et al., 2012) and more recently GoogLeNet (Szegedy et al., 2015), VGG (Simonyan and Zisserman, 2015) and ResNet (He et al., 2016), which perform exceptionally well on ImageNet. In the setting of object detection, Girshick et al. (2014) proposed R-CNN, a CNN-based model that beats by a large margin models relying on hand-designed features. Their approach can be divided into two steps: selection of a set of salient object proposals (Uijlings et al., 2013), followed by a CNN classifier (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015). Currently, most state-of-the-art object detection approaches (Szegedy et al., 2014; He et al., 2014; Girshick, 2015; Ren et al., 2015) rely on this pipeline. Although they are slightly different in the classification step, they all share the first step, which consists of choosing a rich set of object proposals.

Most object proposal approaches leverage low-level grouping and saliency cues. These approaches usually fall into three categories: (1) objectness scoring (Alexe et al., 2012; Zitnick and Dollár, 2014), in which proposals are extracted by measuring the objectness score of bounding boxes, (2) seed segmentation (Humayun et al., 2014; Krähenbühl and Koltun, 2014, 2015), where models start with multiple seed regions and generate separate foreground-background segmentation for each seed, and (3) superpixel merging (Uijlings et al., 2013; Pont-Tuset et al., 2015), where multiple over-segmentations are merged according to various heuristics. These models vary in terms of the type of proposal generated (bounding boxes or segmentation masks) and if the proposals are ranked or not. For a more complete survey of object proposal methods, we recommend the recent survey from (Hosang et al., 2016).

Although our model shares high level similarities with these approaches (we generate a set of ranked segmentation proposals), these results are achieved quite differently. All previous approaches for generating segmentation masks, including (Krähenbühl and Koltun, 2015) which has a learning component, rely on low-level segmentations such as superpixels or edges. Instead, we propose a data-driven discriminative approach based on a deep-network architecture to obtain our segmentation proposals.

Most closely related to our approach, *Multibox* (Erhan et al., 2014; Szegedy et al., 2014) pro-

posed to train a CNN model to generate bounding box object proposals. Their approach, similarly to ours, generates a set of ranked class-agnostic proposals. However, our model generates segmentation proposals instead of the less informative bounding box proposals. Moreover, the model architectures, training scheme, etc., are quite different between our approach and (Szegedy et al., 2014). *Deepbox* (Kuo et al., 2015) proposed a CNN model that learns to rerank proposals generated by *EdgeBox* (Zitnick and Dollár, 2014), a bottom-up method for bounding box proposals. This system shares some similarities to our scoring network. Our model, however, is able to generate the proposals and rank them in one shot from the test image, directly from the pixel space. Concurrently with the work in this chapter, (Ren et al., 2015) proposed "region proposal networks" for generating box proposals that shares similarities with our work. We emphasize, however, that unlike all these approaches our method generates segmentation masks instead of bounding boxes. Finally, Hayder et al. (2016) present an approach to co-generate object proposals in multiple images. They use a deep structure network that jointly predicts the objectness scores and the bounding box locations of multiple object candidates. Contrary to our work, this method relies in a previously generated set of object proposals (*e.g.*, EdgeBox) and generates box proposals only.

5.2 DeepMask Proposals

Our object proposal method predicts a segmentation mask given an input patch, and assigns a score corresponding to how likely the patch is to contain an object.

Both mask and score predictions are achieved with a single convolutional network. CNNs are flexible models which can be applied to various computer vision tasks and they alleviate the need for manually designed features. Their flexible nature allows us to design a model in which the two tasks (mask and score predictions) can share most of the layers of the network. Only the last layers are task-specific (see Figure 5.1). During training, the two tasks are learned jointly. Compared to a model which would have two distinct networks for the two tasks, this architecture choice reduces the capacity of the model and increases the speed of full scene inference at test time.

Each sample k in the training set is a triplet containing (1) the RGB input patch x_k, (2) the binary mask corresponding to the input patch m_k (with $m_k^{ij} \in \{\pm 1\}$, where (i, j) corresponds to a pixel location on the input patch) and (3) a label $y_k \in \{\pm 1\}$ which specifies whether the patch contains an object. Specifically, a patch x_k is given label $y_k = 1$ if it satisfies the following constraints:

(i) the patch contains an object roughly centered in the input patch,

(ii) the object is fully contained in the patch and in a given scale range.

Otherwise, $y_k = -1$, even if an object is partially present. The positional and scale tolerance used in our experiments are given shortly. Assuming $y_k = 1$, the ground truth mask m_k has

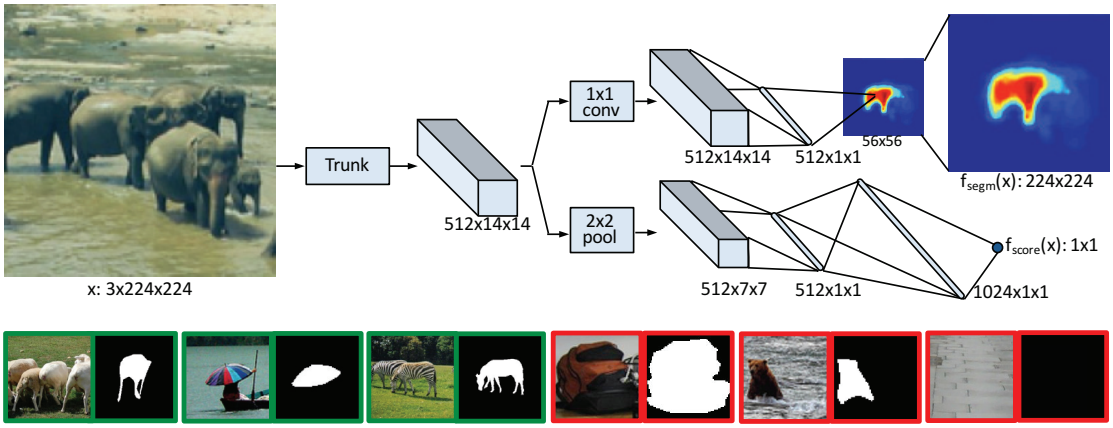


Figure 5.1 – **(Top)** Model architecture: the network is split into two branches after the shared feature extraction layers. The top branch predicts a segmentation mask for the the object located at the center while the bottom branch predicts an object score for the input patch. **(Bottom)** Examples of training triplets: input patch x, mask m and label y. Green patches contain objects that satisfy the specified constraints and therefore are assigned the label $y = 1$. Note that masks for negative examples (shown in red) are not used and are shown for illustrative purposes only.

positive values only for the pixels that are part of the *single* object located in the center of the patch. If $y_k = -1$ the mask is not used. Figure 5.1, bottom, shows examples of training triplets.

Figure 5.1, top, illustrates an overall view of our model, which we call *DeepMask*. The top branch is responsible for predicting an object segmentation mask and the bottom branch predicts the likelihood that an object is present and satisfies the above two constraints.

We next describe in detail each part of the basic architecture (the first version of DeepMask, introduced in (Pinheiro et al., 2015)), the training procedure, and the fast inference procedure. In Section 5.3, we describe some efficient variants of this basic architecture that improve speed without decreasing the performance (introduced in (Pinheiro et al., 2016)).

5.2.1 Network Architecture

In this section we describe the three components of the basic DeepMask architecture: the common shared trunk, the segmentation branch and the scoring branch.

Common Trunk

The parameters for the layers shared between the mask prediction and the object score prediction are initialized with a network that was pre-trained to perform classification on the ImageNet dataset (Deng et al., 2009). This model is then fine-tuned for generating object proposals during training. For the basic architecture, we choose the VGG-A (Simonyan and

Zisserman, 2015) which consists of eight 3×3 convolutional layers (followed by ReLU nonlinearities) with 0-padding and five 2×2 max-pooling layers and has shown good performance.

As we are interested in inferring segmentation masks, the spatial information provided in the convolutional feature maps is important. We therefore remove all the final fully connected layers of the VGG-A model. Additionally we also discard the last max-pooling layer. The output of the shared layers has a downsampling factor of 16 due to the remaining four 2×2 max-pooling layers; given an input image of dimension $3 \times h \times w$, the output is a feature map of dimensions $512 \times \frac{h}{16} \times \frac{w}{16}$.

Segmentation Head

The branch of the network dedicated to segmentation is composed of a single 1×1 convolution layer (and ReLU non-linearity) followed by a classification layer. The classification layer consists of $h \times w$ pixel classifiers, each responsible for indicating whether a given pixel belongs to the object in the center of the patch. Note that each pixel classifier in the output plane must be able to utilize information contained in the *entire* feature map, and thus have a complete view of the object. This is critical because unlike in semantic segmentation, our network must output a mask for a single object even when multiple objects are present (*e.g.*, see the elephants in Figure 5.1).

For the classification layer one could use either locally or fully connected pixel classifiers. Both options have drawbacks: in the former each classifier has only a partial view of the object while in the latter the classifiers have a massive number of redundant parameters. Instead, we opt to decompose the classification layer into two linear layers with no non-linearity in between. This can be viewed as a "low-rank" variant of using fully connected linear classifiers. Such an approach massively reduces the number of network parameters while allowing each pixel classifier to leverage information from the entire feature map. Its effectiveness is shown in the experiments. Finally, to further reduce model capacity, we set the output of the classification layer to be $h^o \times w^o$ with $h^o < h$ and $w^o < w$ and upsample the output to $h \times w$ to match the input dimensions.

Scoring Head

The second branch of the network is dedicated to predicting if an image patch satisfies constraints (i) and (ii): that is, if an object is centered in the patch and at the appropriate scale. In its basic form, the scoring head is composed of a 2×2 max-pooling layer, followed by two fully connected (plus ReLU non-linearity) layers. The final output is a single "objectness" score indicating the presence of an object in the center of the input patch (and at the appropriate scale).

5.2.2 Joint Learning

Given an input patch $x_k \in \mathcal{I}$, the model is trained to jointly infer a pixel-wise segmentation mask and an object score. The loss function is a sum of binary logistic regression losses, one for each location of the segmentation network and one for the object score, over all training triplets (x_k, m_k, y_k):

$$\mathcal{L}(\theta) = \sum_k \left(\frac{1+y_k}{2w^o h^o} \sum_{ij} \log(1 + e^{-m_k^{ij} f_{segm}^{ij}(x_k)}) + \lambda \log(1 + e^{-y_k f_{score}(x_k)}) \right). \tag{5.1}$$

Here θ is the set of parameters, $f_{segm}^{ij}(x_k)$ is the prediction of the segmentation network at location (i, j), and $f_{score}(x_k)$ is the predicted object score. We alternate between backpropagating through the segmentation branch and scoring branch (and set $\lambda = \frac{1}{32}$). For the scoring branch, the data is sampled such that the model is trained with an equal number of positive and negative samples.

Note that the factor multiplying the first term of (5.1) implies that we only backpropagate the error over the segmentation branch if $y_k = 1$. An alternative would be to train the segmentation branch using negatives as well (setting $m_k^{ij} = 0$ for all pixels if $y_k = -1$). However, we found that training with positives only was critical for generalizing beyond the object categories seen during training and for achieving high object recall. This way, during inference the network attempts to generate a segmentation mask at *every* patch, even if no known object is present.

5.2.3 Full Scene Inference

During full image inference, we apply the model densely at multiple locations and scales. This is necessary so that for each object in the image we test at least one patch that fully contains the object (roughly centered and at the appropriate scale), satisfying the two assumptions made during training. This procedure gives a segmentation mask and object score at each image location. Figure 5.2 illustrates the segmentation output when the model is applied densely to an image at a single scale.

The full image inference procedure is efficient since all computations can be computed convolutionally. The CNN features can be computed densely in a fraction of a second given a typical input image. For the segmentation branch, the last fully connected layer can be computed via convolutions applied to the CNN features. The scores are likewise computed by convolutions on the CNN features followed by two 1×1 convolutional layers.

Finally, note that the scoring branch of the network has a downsampling factor $2\times$ larger than the segmentation branch due to the additional max-pooling layer. Given an input test image of size $h^t \times w^t$, the segmentation and object network generate outputs of dimension $\frac{h^t}{16} \times \frac{w^t}{16}$ and $\frac{h^t}{32} \times \frac{w^t}{32}$, respectively. In order to achieve a one-to-one mapping between the mask prediction and object score, we apply the interleaving trick right before the last max-pooling layer for the scoring branch to double its output resolution (we use exactly the implementation described

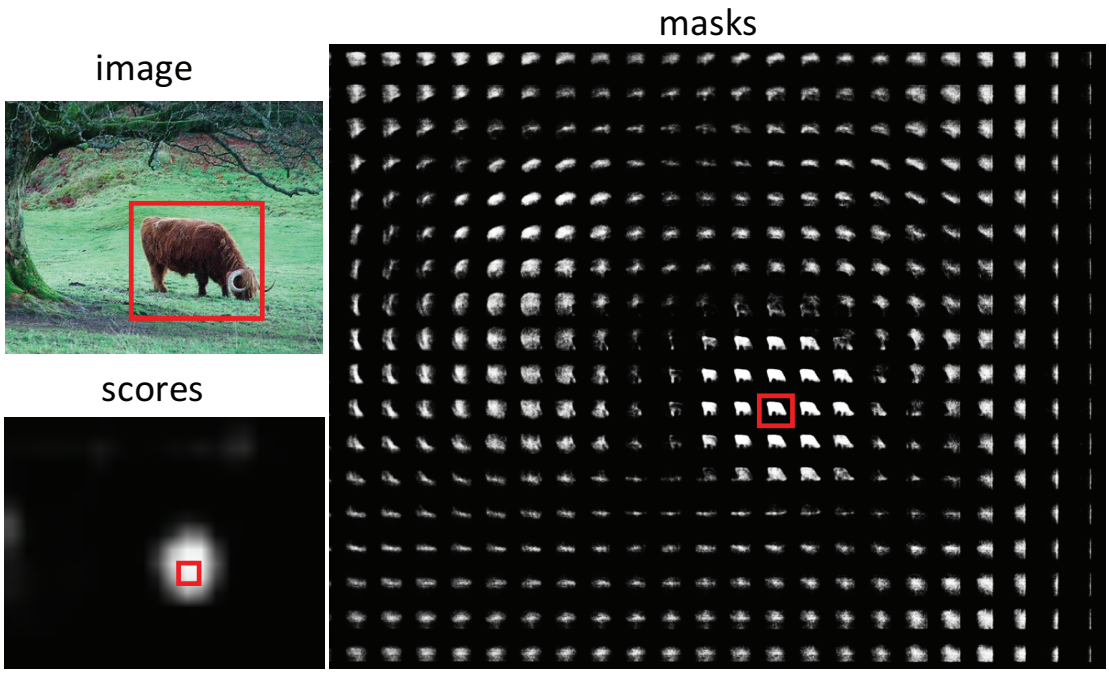


Figure 5.2 – Output of segmentation masks and object scores generated by the proposed model on a given test image, with a 16 pixels stride (at a single scale). The model outputs high quality mask and and high objectness score on object locations.

in (Sermanet et al., 2014)).

5.2.4 Implementation Details

During training, an input patch x_k is considered to contain a "canonical" positive example if an object is precisely centered in the patch and has maximal dimension equal to exactly 128 pixels. However, having some tolerance in the position of an object within a patch is critical as during full image inference most objects will be observed slightly offset from their canonical position. Therefore, during training, we randomly jitter each "canonical" positive example to increase the robustness of our model. Specifically, we consider translation shift (of ± 16 pixels), scale deformation (of $2^{\pm 1/4}$), and also horizontal flip. In all cases we apply the same transformation to both the image patch x_k and the ground truth mask m_k and assign the example a positive label $y_k = 1$. Negative examples ($y_k = -1$) are any patches at least ± 32 pixels or $2^{\pm 1}$ in scale from any canonical positive example.

During full image inference we apply the model densely at multiple locations (with a stride of 16 pixels) and scales (scales 2^{-2} to 2^1 with a step of $2^{1/2}$). This ensures that there is at least one tested image patch that fully contains each object in the image (within the tolerances used during training).

In the first implementation, the model is fed with RGB input patches of dimension $3 \times 224 \times$

224. Since we removed the fifth pooling layer, the common branch outputs a feature map of dimensions $512 \times 14 \times 14$. The score branch of our network is composed of 2×2 max pooling followed by two fully connected layers (with 512 and 1024 hidden units, respectively). Both of these layers are followed by ReLU non-linearity and a dropout (Srivastava et al., 2014) procedure with a rate of 0.5. A final linear layer then generates the object score (see Section 5.4.1 for more efficient implementation).

The segmentation branch begins with a single 1×1 convolutional layer with 512 units. This feature map is then fully connected to a low dimensional output of size 512, which is further fully connected to each pixel classifier to generate an output of dimension 56×56. As discussed, there is no non-linearity between these two layers. In total, this implementation of the model contains around 75M parameters.

A final bilinear upsampling layer is added to transform the 56×56 output prediction to the full 224×224 resolution of the ground-truth (directly predicting the full resolution output would have been much slower). We opted for a non-trainable layer as we observed that a trainable one simply learned to bilinearly upsample. Alternatively, we tried downsampling the ground-truth instead of upsampling the network output; however, we found that doing so slightly reduced accuracy.

Design architecture and hyper-parameters were chosen using a subset of the COCO validation data (Lin et al., 2014) (non-overlapping with the data we used for evaluation). We considered a learning rate of .001. We trained our model using stochastic gradient descent with a batch size of 32 examples, momentum of .9, and weight decay of .00005. Aside from the pre-trained VGG features, weights are initialized randomly from a uniform distribution. Our model takes around 5 days to train on a Nvidia Tesla K40m. To binarize predicted masks we simply threshold the continuous output (using a threshold of .1 for PASCAL and .2 for COCO).

5.3 Architecture Optimization

The DeepMask architecture described above achieves state of the art in object proposal over different datasets. We now describe a series of architecture optimization that further improves its performance in terms of accuracy and inference speed. In the next two subsections we carefully examine the design of the network "trunk" and "head".

5.3.1 Trunk Architecture

We begin by identifying model bottlenecks. The architecture defined in the previous section spends 40% of its time for feature extraction, 40% for mask prediction, and 20% for score prediction. Given the time of feature extraction, increasing model depth or breadth can incur a non-trivial computational cost. Simply upgrading the 11-layer VGG-A model (Simonyan and Zisserman, 2015) to the 16-layer VGG-D model can double run time. Recently, He et al. (2016)

introduced Residual Networks (ResNet) and showed excellent results. In these architecture variants, we use the 50-layer ResNet model pre-trained on ImageNet, which achieves the accuracy of VGG-D but with the inference time of VGG-A.

We explore models with varying input size **W**, number of pooling layers **P**, stride density **S**, model depth **D**, and final number of feature channels **F**. These factors are intertwined but we can achieve significant insight by a targeted study.

Input size W Given a minimum object size O, the input image needs to be upsampled by W/O to detect small objects. Hence, reducing W improves speed of both mask prediction and inference for small objects. However, a smaller W reduces the input resolution which in turn lowers the accuracy of mask prediction. Moreover, reducing W decreases stride density S which further harms accuracy.

Pooling layers P Assuming 2×2 pooling, the final kernel width is $W/2^P$. During inference, this necessitates convolving with a large $W/2^P$ kernel in order to aggregate information (*e.g.*, 14×14 for DeepMask). However, while more pooling P results in faster computation, it also results in loss of feature resolution.

Stride density S We define the stride density to be S=W/stride (where typically stride is 2^P). The smaller the stride, the denser the overlap with ground truth locations. We found that the stride density is key for mask prediction. Doubling the stride while keeping W constant greatly reduces performance as the model must be more spatially invariant relative to a fixed object size.

Depth D For typical networks (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; Szegedy et al., 2015; He et al., 2016), spatial resolution decreases with increasing D while the number of feature channels F increases. In the context of instance segmentation, reducing spatial resolution hurts performance. One possible direction is to start with lower layers that have less pooling and increase the depth of the model without reducing spatial resolution or increasing F. This would require training networks from scratch which we leave to future work.

Feature channels F The high dimensional features at the top layer introduce a bottleneck for feature aggregation. An efficient approach is to first apply dimensionality reduction before feature aggregation. We adopt 1×1 convolution to reduce F and show that we can achieve large speedups in this manner.

In Section 5.4.1 and Table 5.1 we examine various choices for W, P, S, D, and F.

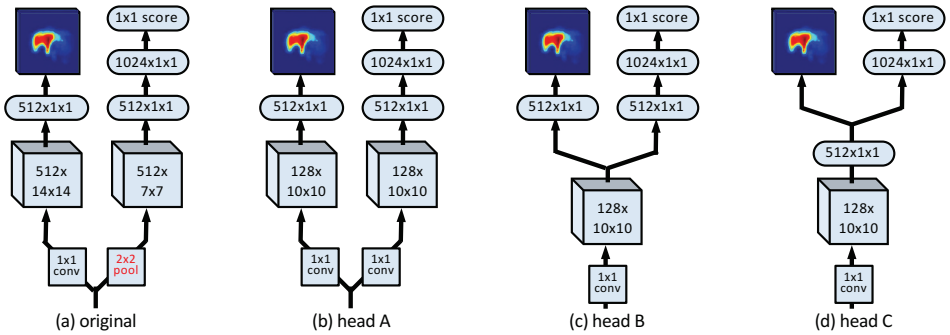


Figure 5.3 – Network head architecture. (a) The original DeepMask head. (b-d) Various head options with increasing simplicity and speed. The heads share identical pathways for mask prediction but have progressively simplified score branches.

5.3.2 Head Architecture

We also examine the "head" of the DeepMask model, focusing on score prediction. Our goal is to simplify the head and further improve inference speed.

In the base DeepMask architecture described in the previous section, the mask and scoring heads branch after the final $512 \times 14 \times 14$ feature map (see Figure 5.1 and Figure 5.3a). Both mask and score prediction require a large convolution, and in addition, the score branch requires an extra pooling step and hence interleaving to match the stride of the mask network during inference. Overall, this leads to a slow inference procedure.

We propose a sequence of simplified network structures that have identical mask branches but that share progressively more computation. A series of model heads A-C is detailed in Figure 5.3. Head A removes the need for interleaving in DeepMask by removing max pooling and replacing the $512 \times 7 \times 7$ convolutions by $128 \times 10 \times 10$ convolutions; overall this network is much faster. Head B simplifies this by having the $128 \times 10 \times 10$ features shared by both the mask and score branch. Finally, model C further reduces computation by having the score prediction utilize the same low rank $512 \times 1 \times 1$ features used for the mask.

In Section 5.4.1 we evaluate these variants in terms of performance and speed.

5.4 Experimental Results

In this section, we evaluate the performance of our approach on the PASCAL VOC 2007 test set (Everingham et al., 2010) and on the first 5000 images of the COCO 2014 validation set (Lin et al., 2014). Our model is trained on the COCO training set which contains about 80,000 images and a total of nearly 500,000 segmented objects. Although our model is trained to generate segmentation proposals, it can also be used to provide box proposals by taking the bounding boxes enclosing the segmentation masks. Figure 5.4 and Figure 5.5 show examples of generated proposals with highest IoU to the ground truth on COCO.

Figure 5.4 – DeepMask proposals with highest IoU to the ground truth on selected images from COCO. Missed objects (no matching proposals with IoU > 0.5) are marked with a red outline.

The accuracy is measured using the common IoU metric. IoU is the intersection of a candidate proposal and ground-truth annotation divided by the area of their union. This metric can be applied to both segmentation and box proposals. Following Hosang et al. (2016), we evaluate the performance of the proposal methods considering the AR between IoU 0.5 and 1.0 for a fixed number of proposals (see Section 2.1.2 for further details). AR has been shown to correlate extremely well with detector performance (recall at a single IoU threshold is far less predictive (Hosang et al., 2016)).

The results are measured in terms of AR at 10, 100, and 1000 proposals and averaged across all counts (Area Under Curve (AUC)). As the COCO dataset contains objects in a wide range of scales, it is also common practice to divide objects into roughly equally sized sets according to object pixel area a: small ($a < 32^2$), medium ($32^2 \leq a \leq 96^2$), and large ($a > 96^2$) objects, and report accuracy at each scale.

	W	P	D	S	kernel	F	AR	AR^S	AR^M	AR^L	time
DeepMaskBase	224	4	8	14	512x14x14	512	36.6	18.2	48.7	50.6	1.32s
W160-P4-D8-VGG	160	4	8	10	1024x10x10	512	35.5	15.1	47.5	53.2	.58s
W160-P4-D39	160	4	39	10	1024x10x10	512	37.0	15.9	50.5	53.9	.58s
W160-P4-D39-F128	160	4	39	10	1024x10x10	128	36.9	15.6	49.9	54.8	.45s
W112-P4-D39	112	4	39	7	1024x7x7	512	30.8	11.2	42.3	47.8	.31s
W112-P3-D21	112	3	21	14	512x14x14	512	36.7	16.7	49.1	53.1	.75s
W112-P3-D21-F128	112	3	21	14	512x14x14	128	36.1	16.3	48.4	52.2	.33s

Table 5.1 – Model performance (upper bound on AR) for varying input size W, number of pooling layers P, stride density S, depth D, and features channels F. See Section 5.3.1 and Section 5.4.1 for details. Timing is for multiscale inference excluding the time for score prediction.

5.4.1 Architecture Variants

We begin by reporting the performance of different variations of DeepMask architecture presented in previous sections. For our initial results, we measure AR for densely computed masks (~10^4 proposals per image). This allows us to factor out the effect of objectness score prediction and focus exclusively on evaluating mask quality. In our experiments, AR across all proposals is highly correlated, hence this upper bound on AR is predictive of performance at more realistic settings (*e.g.* at AR^{100}).

Trunk Architecture We begin by investigating the effect of the network trunk parameters described in Section 5.3.1 with the goal of optimizing both speed and accuracy. Performance of a number of representative models is shown in Table 5.1. First, replacing the 224×224 DeepMask VGG-A model with a 160×160 version is much faster (over two times). Surprisingly, accuracy loss for this model, W160-P4-D8-VGG, is only minor, partially due to an improved learning schedule. Upgrading to a Residual Network (ResNet) trunk, W160-P4-D39, restores accuracy and keeps speed identical. We found that reducing the feature dimension to 128 (-F128) shows almost no loss, but improves speed. Finally, as input size is a bottleneck, we also tested a number of W112 models. Nevertheless, overall, W160-P4-D39-F128 gave the best tradeoff between speed and accuracy.

Head Architecture In Table 5.2 we evaluate the performance of the various network heads in Figure 5.3 (using standard AR, not upper-bound AR as in Table 5.1). Head A is already substantially faster than base DeepMask. All heads achieve similar accuracy with a decreasing inference time as the score branch shares progressively more computation with the mask. Interestingly, head C is able to predict both the score and mask from a single compact 512 dimensional vector. We chose this variant due to its simplicity and speed.

Based on these experiments, we combine the W160-P4-D39-F128 trunk with the C head. On

	AR^{10}	AR^{100}	AR^{1K}	AUC^{S}	AUC^{M}	AUC^{L}	AUC	mask	score	total
DeepMaskBase	12.6	24.5	33.1	2.3	26.6	33.6	18.3	1.32s	.27s	1.59s
head A	14.0	25.8	33.4	2.2	27.3	36.6	19.3	.45s	.06s	.51s
head B	14.0	25.4	33.0	2.0	27.0	36.9	19.1	.45s	.05s	.50s
head C	14.4	25.8	33.1	2.2	27.3	37.4	19.4	.45s	.01s	.46s

Table 5.2 – All model variants of the head have similar performance. Head C is a win in terms of both simplicity and speed. See Figure 5.3 for head definitions.

the following experiments, we refer to this model as *DeepMask*, while the base DeepMask architecture (as defined in Section 5.2) is referred to as *DeepMaskBase*. DeepMask is over 3× faster than DeepMaskBase (average of .46s versus 1.59s per image on COCO val set). Moreover, model parameter count is reduced from ~75M to ~17M.

5.4.2 Comparison with State of the Art

Methods We compare to the current top publicly-available proposal methods including: EdgeBoxes (Zitnick and Dollár, 2014), SelectiveSearch (Uijlings et al., 2013), Geodesic (Krähenbühl and Koltun, 2014), Rigor (Humayun et al., 2014), MCG (Pont-Tuset et al., 2015) and Region Proposal Networks (RPN) (Ren et al., 2015). Among these results, only the most recent, RPN, is based on CNNs (we obtain improved RPN proposals from the authors of (Bell et al., 2016)). These methods achieve top results on object detection (when coupled with R-CNNs (Girshick et al., 2014)) and also obtain the best AR (Hosang et al., 2016).

Results Tables 5.3 and 5.4 compare the performance of our approach, DeepMask, to existing proposal methods on COCO (using both boxes and segmentations) and PASCAL (using boxes), respectively. The tables show the AR at selected proposal counts and averaged across all counts (AUC). Under all scenarios DeepMask (and its variants) achieves substantially better AR for all numbers of proposals considered. Notably, DeepMask achieves an *order of magnitude reduction* in the number of proposals necessary to reach a given AR under most scenarios. For example, with 100 segmentation proposals DeepMask achieves an AR of 25.8 on COCO while competing methods require nearly 1000 segmentation proposals to achieve similar AR.

Generalization To see if our approach can generalize to unseen classes, we train two additional versions of our model, DeepMask20 and DeepMask20*. DeepMask20 is trained *only* with objects belonging to one of the 20 PASCAL categories (subset of the full 80 COCO categories). DeepMask20* is similar, except we use the scoring network from the original DeepMask. Results for the two models when evaluated on all 80 COCO categories (as in all other experiments) are shown in Table 5.3. Compared to DeepMask, DeepMask20 exhibits a drop in AR (but still outperforms all previous methods). DeepMask20*, however, matches the performance of DeepMask. This surprising result demonstrates that the drop in accuracy is

	Box Proposals				Segmentation Proposals						
	AR^{10}	AR^{100}	AR^{1K}	AUC	AR^{10}	AR^{100}	AR^{1K}	AUC^{S}	AUC^{M}	AUC^{L}	AUC
EdgeBoxes (Zitnick and Dollár, 2014)	07.4	17.8	33.8	13.9	–	–	–	–	–	–	–
Geodesic (Krähenbühl and Koltun, 2014)	04.0	18.0	35.9	12.6	02.3	12.3	25.3	01.3	08.6	20.5	08.5
Rigor (Humayun et al., 2014)	–	13.3	33.7	10.1	–	09.4	25.3	02.2	06.0	17.8	07.4
SelectiveSearch (Uijlings et al., 2013)	05.2	16.3	35.7	12.6	02.5	09.5	23.0	00.6	05.5	21.4	07.4
MCG (Pont-Tuset et al., 2015)	10.1	24.6	39.8	18.0	07.7	18.6	29.9	03.1	12.9	32.4	13.7
RPN (Ren et al., 2015; Bell et al., 2016)	12.8	29.2	42.6	21.4	–	–	–	–	–	–	–
DeepMaskBase	15.3	31.3	44.6	23.3	12.6	24.5	33.1	02.3	26.6	33.6	18.3
DeepMask20	17.0	31.8	44.9	24.4	12.4	22.6	31.4	01.9	23.3	35.3	17.4
DeepMask20*	18.5	34.1	45.0	25.6	14.0	24.5	31.4	01.9	26.4	35.7	18.6
DeepMask	18.7	34.9	46.5	26.2	14.4	25.8	33.1	02.2	27.3	37.4	19.4

Table 5.3 – Results on the COCO dataset for both bounding box and segmentation proposals. We report AR at different number of proposals (10, 100 and 1000) and also AUC (AR averaged across all proposal counts). For segmentation proposals we report overall AUC and also AUC at different scales (small/medium/large objects indicated by superscripts S/M/L). See text for details.

PASCAL VOC07	**AR^{10}**	**AR^{100}**	**AR^{1K}**	**AUC**
EdgeBoxes (Zitnick and Dollár, 2014)	20.3	40.7	60.1	30.9
Geodesic (Krähenbühl and Koltun, 2014)	12.1	36.4	59.6	23.0
Rigor (Humayun et al., 2014)	16.4	32.1	58.9	23.9
SelectiveSearch (Uijlings et al., 2013)	08.5	34.7	61.8	24.1
MCG (Pont-Tuset et al., 2015)	23.2	46.2	63.4	34.4
DeepMask	41.2	62.6	71.9	48.8

Table 5.4 – Quantitative results on PASCAL VOC 2007 test.

due to the discriminatively trained scoring branch (DeepMask20 is inadvertently trained to assign low scores to the other 60 categories). The segmentation branch generalizes extremely well even when trained on a reduced set of categories.

Speed Inference takes an average of .46s per image in the COCO dataset (.34s on the smaller PASCAL images). Our method is much faster than other state of the art segmentation proposal methods: Geodesic (Krähenbühl and Koltun, 2015) runs at ~1s per PASCAL image and MCG (Pont-Tuset et al., 2015) takes ~30s. Inference time can further be dropped by ~30% by parallelizing all scales in a single batch (eliminating GPU overhead). We do, however, require use of a GPU for efficient inference.

5.5 Summary

In this chapter, we proposed a new way to generate segmentation object proposals directly from image pixels. Our model consists of a discriminative convolutional neural network that is

capable to, at same time, generate a class-agnostic segmentation mask and an object score for a given input patch. At test time, the model is applied densely over the entire image at multiple scales and generates a set of ranked segmentation proposals. We show that learning features for object proposal generation is not only feasible but effective. Our approach surpasses the previous state of the art by a large margin (an increase of ~40%-50% in AUC) in box and segmentation proposal generations, and in both PASCAL and COCO datasets. Moreover, our method performs faster than previous methods (although we require GPU for a fast inference).

Figure 5.5 – Additional DeepMask proposals with highest IoU to the ground truth on selected images from COCO. Missed objects (no matching proposals with IoU > 0.5) are marked with a red outline.

6 Learning to Refine Object Segments

As object detection (Felzenszwalb et al., 2010; Sermanet et al., 2013; Szegedy et al., 2014; He et al., 2014; Girshick et al., 2014; Girshick, 2015; Ren et al., 2015; Bell et al., 2016) has rapidly progressed, there has been a renewed interest in object instance segmentation (Lin et al., 2014). As the name implies, the goal is to both detect and segment each individual object in an image. The task is related to both object detection with bounding boxes (Lin et al., 2014; Everingham et al., 2010; Deng et al., 2009) and semantic segmentation (Shotton et al., 2008; Everingham et al., 2010; Farabet et al., 2013; Pinheiro and Collobert, 2014; Eigen and Fergus, 2015; Zheng et al., 2015; Chen et al., 2015; Schwing and Urtasun, 2015; Noh et al., 2015). It involves challenges from both domains, requiring accurate pixel-level object segmentation coupled with identification of each individual object instance.

A number of recent papers have explored the use of CNNs (LeCun et al., 1998) for object instance segmentation (Hariharan et al., 2014; Dai et al., 2016; Hariharan et al., 2015). Standard feedforward CNNs (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; Szegedy et al., 2015; He et al., 2016) interleave convolutional layers (with pointwise nonlinearities) and pooling layers. Pooling controls model capacity and increases receptive field size, resulting in a coarse, highly-semantic feature representation. While effective and necessary for extracting object-level information, this general architecture results in low resolution features that are invariant to pixel-level variations. This is beneficial for classification and identifying object instances but poses challenge for pixel-labeling tasks. Hence, CNNs that utilize only upper network layers for object instance segmentation (Hariharan et al., 2014; Dai et al., 2016), as in Figure 6.1a, can effectively generate coarse object masks but have difficulty generating pixel-accurate segmentations.

For pixel-labeling tasks such as semantic segmentation and edge detection, "skip" connections (Sermanet et al., 2013; Long et al., 2015; Hariharan et al., 2015; Xie and Tu, 2015), as shown in Figure 6.1b, are popular. In practice, common skip architectures are equivalent to making independent predictions from each network layer and upsampling and averaging the results (see Fig. 2 in (Hariharan et al., 2015), Fig. 3 in (Long et al., 2015), and Fig. 3 in (Xie and Tu, 2015)). This is effective for semantic segmentation as local receptive fields in early

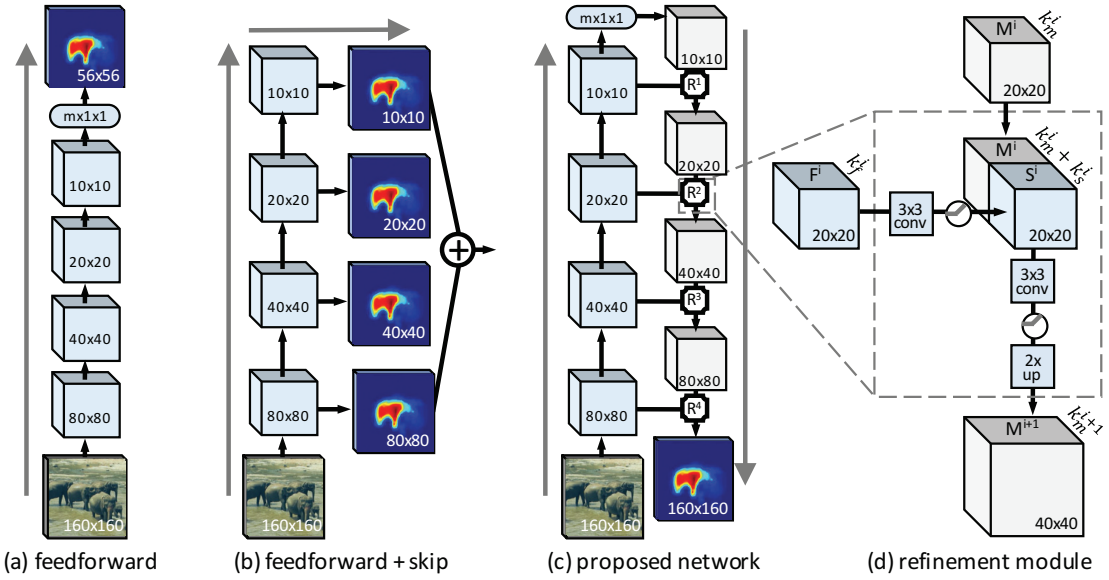


Figure 6.1 – Architectures for object instance segmentation. (a) Feedforward nets, such as DeepMask (Pinheiro et al., 2015), predict masks using only upper-layer CNN features, resulting in coarse pixel masks. (b) Common "skip" architectures are equivalent to making independent predictions from each layer and averaging the results (Long et al., 2015; Hariharan et al., 2015; Xie and Tu, 2015), such an approach is not well suited for object instance segmentation. (c,d) In this work we propose to augment feedforward nets with a novel top-down refinement approach. The resulting bottom-up/top-down architecture is capable of efficiently generating high-fidelity object masks.

layers can provide sufficient data for pixel labeling. For object segmentation, however, it is necessary to differentiate between object instances, for which local receptive fields are insufficient (*e.g.* local patches of sheep fur can be labeled as such but without object-level information it can be difficult to determine if they belong to the same animal).

In this chapter, we propose a novel CNN which efficiently merges the spatially rich information from low-level features with the high-level object knowledge encoded in upper network layers. Rather than generating independent outputs from multiple network layers, our approach first generates a coarse *mask encoding* in a feedforward manner, which is simply a semantically meaningful feature map with multiple channels, then refines it by successively integrating information from earlier layers. Specifically, we introduce a *refinement module* and stack successive such modules together into a top-down refinement process. See Figures 6.1c and 6.1d. Each refinement module is responsible for "inverting" the effect of pooling by taking a mask encoding generated in the top-down pass, along with the matching features from the bottom-up pass, and merging the information in both to generate a new mask encoding with double spatial resolution. The process continues until full resolution is restored and the final output encodes the object mask. The refinement module is efficient and fully backpropable.

We apply our approach in the context of object proposal generation (although it could be

applied to other pixel-labeling tasks). As shown in Chapter 5, training a discriminative CNN can largely improve the performance of object proposals if compared to classic approaches that utilize low-level grouping and saliency clues (Hosang et al., 2016). However, DeepMask was designed to use a standard feedforward CNN which, as mentioned above, poses problems to pixel-labeling tasks.

In this chapter we utilize the DeepMask architecture (described in the previous chapter) as our starting point for object instance segmentation due to its simplicity and effectiveness. We augment the basic DeepMask architecture with our refinement module (see Figure 6.1) and refer to the resulting approach as *SharpMask* to emphasize its ability to produce sharper, higher-fidelity object segmentation masks.

SharpMask improves the segmentation mask quality relative to DeepMask. For object proposal generation, this improvement boosts average recall of on the COCO dataset (Lin et al., 2014) by 10-20%, establishing the new state of the art on this task.

This chapter is organized as follows: Section 6.1 presents related work, Section 6.2 introduces our novel top-down refinement network and Section 6.3 validates our approach experimentally on the context object proposal generation.

6.1 Related Work

Following their success in image classification (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; Szegedy et al., 2015; He et al., 2016), CNNs have been adopted with great effect to pixel-labeling tasks such as depth estimation (Eigen and Fergus, 2015), optical flow (Dosovitskiy et al., 2015), and semantic segmentation (Farabet et al., 2013). Below we describe architectural innovations for such tasks, and discuss how they relate to our approach. Aside from skip connections (Sermanet et al., 2013; Hariharan et al., 2015; Long et al., 2015; Xie and Tu, 2015), these techniques can be roughly classified as multiscale architectures, deconvolutional networks, and graphical model networks. We discuss each in turn next. We emphasize, however, that most of these approaches are not applicable to our domain due to severe computational constraints: we must refine hundreds of proposals per image implying the marginal time per proposal must be minimal.

Multiscale architectures (Farabet et al., 2013; Eigen and Fergus, 2015) compute features over multiple rescaled versions of an image. Features can be computed independently at each scale (Farabet et al., 2013), or the output from one scale can be used as additional input to the next finer scale (Eigen and Fergus, 2015). Our approach relies on a similar intuition but does not require recomputing features at each image scale. This allows us to apply refinement efficiently to hundreds of locations per image as necessary for object proposal generation.

Deconvolutional networks (Zeiler and Fergus, 2014) proposed to invert the pooling process in a CNN to generate progressively higher resolution input images by storing the "switch" variables from the pooling operation. Deconvolutional networks have recently been applied successfully to semantic segmentation (Noh et al., 2015). Deconvolutional layers share similarities with our refinement module, however, "switches" are communicated instead of the feature values, which limits the information that can be transferred. Finally, Dosovitskiy et al. (2015) proposed to progressively increase the resolution of an optical flow map. This can be seen as a special case of our refinement approach where: (1) the "features" for refinement are set to be the flow field itself, (2) no feature transform is applied to the bottom-up features, and (3) the approach is applied monolithically to the entire image. Restricting our method in any of these ways would cause it to fail in our setting as discussed in Section 6.3.

Graphical model networks a number of recent papers have proposed integrating graphical models into CNNs by demonstrating they can be formulated as recurrent nets (Zheng et al., 2015; Chen et al., 2015; Schwing and Urtasun, 2015). Good results were demonstrated on semantic segmentation. While too slow to apply to multiple proposals per image, these approaches likewise attempt to sharpen a coarse segmentation mask.

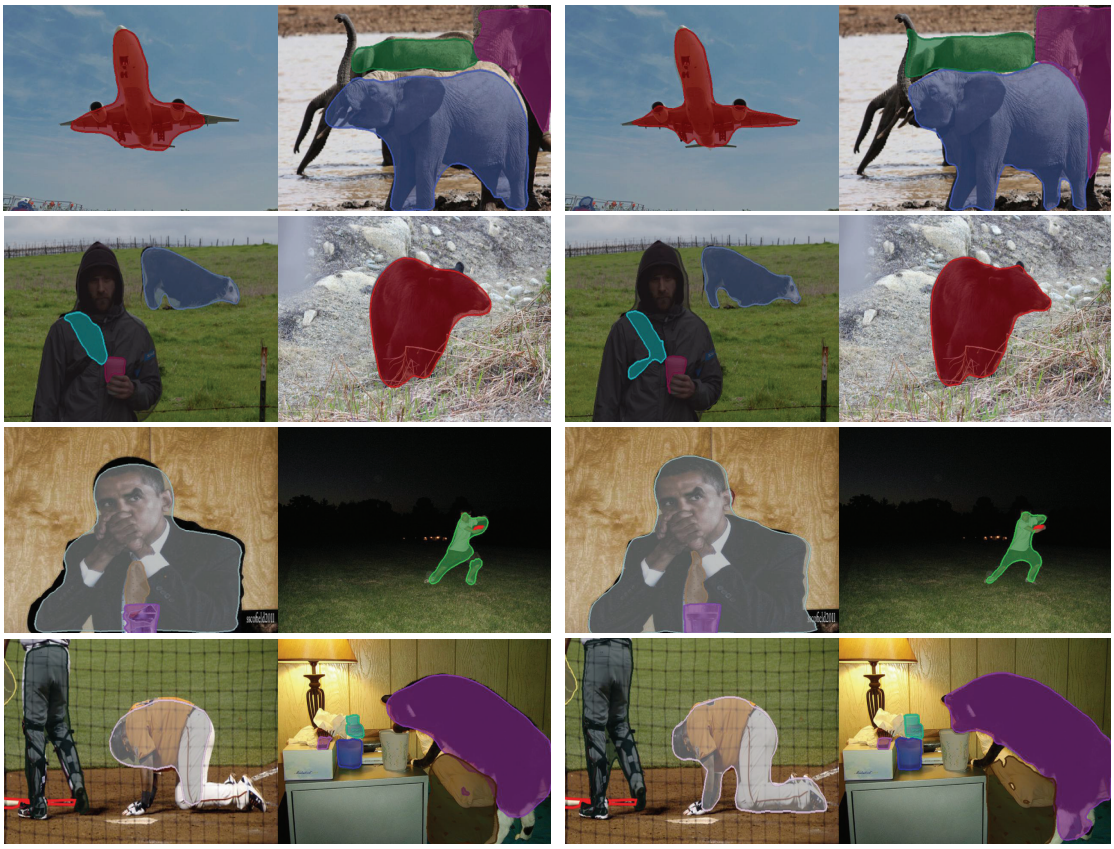
6.2 Learning Mask Refinement

We apply our proposed bottom-up/top-down refinement architecture to object instance segmentation. Specifically, we focus on object proposal generation, which forms the cornerstone of modern object detection (Girshick et al., 2014). We note that although we test the proposed refinement architecture on the task of object segmentation, it could potentially be applied to other pixel-labeling tasks.

As seen in the previous chapter, object proposal algorithms aim to find diverse regions in an image which are likely to contain objects. Both proposal recall and quality correlate strongly with detector performance (Hosang et al., 2016). We adopt the DeepMask network as the starting point for proposal generation. DeepMask is trained to jointly generate a class-agnostic object mask and an associated "objectness" score for each input image patch. At inference time, the model is run convolutionally to generate a dense set of scored segmentation proposals. We refer readers to Chapter 5 for full details.

A simplified diagram of the segmentation branch of DeepMask is illustrated in Figure 6.1a. The network is trained to infer the mask for the object located in the center of the input patch. It contains a series of convolutional layers interleaved with pooling stages that reduce the spatial dimensions of the feature maps, followed by a fully connected layer to generate the object mask. Hence, each pixel prediction is based on a complete view of the object, however, its input feature resolution is low due to the multiple pooling stages.

As a result, DeepMask generates masks that are accurate on the object level but only coarsely



(a) DeepMask Proposals (b) SharpMask Proposals

Figure 6.2 – Qualitative comparison of DeepMask versus SharpMask segmentations. Proposals with highest IoU to the ground truth are shown for each method. Both DeepMask and SharpMask generate object masks that capture the general shape of the objects. However, SharpMask improves the masks near object boundaries.

align with object boundaries, see Figure 6.2a. In order to obtain higher-quality masks, we augment the basic DeepMask architecture with a refinement approach. We refer to the resulting method as *SharpMask* to emphasize its ability to produce sharper, pixel-accurate object masks, see Figure 6.2b. We begin with a high-level overview of our approach followed by further details.

6.2.1 Refinement Overview

Our goal is to efficiently merge the spatially rich information from low-level features with the high-level semantic information encoded in upper network layers. This approach is guided by three principles:

(i) object-level information is often necessary to segment an object,

(ii) given object-level information, segmentation should proceed in a top-down fashion,

successively integrating information from earlier layers,

(iii) the approach should invert the loss of resolution from pooling (with the final output matching the resolution of the input).

To satisfy these principles, we augment standard feedforward networks with a top-down refinement process. An overview of our approach is shown in Figure 6.1c. We introduce a "refinement module" R that is responsible for inverting the effect of pooling and doubling the resolution of the input mask encoding. Each module R^i takes as input a mask encoding M^i generated in the top-down pass, along with matching features F^i generated in the bottom-up pass, and learns to merge the information to generate a new upsampled object encoding M^{i+1}. In other words: $M^{i+1} = R^i(M^i, F^i)$, see Figure 6.1d. Multiple such modules are stacked (one module per pooling layer). The final output of our network is a pixel labeling of the same resolution as the input image. Full details are presented next.

6.2.2 Refinement Details

The feedforward pathway of our network outputs a "mask encoding" M^1, or simply, a low-resolution but semantically meaningful feature map with k_m^1 channels. M^1 serves as the input to the top-down refinement module, which is responsible for progressively increasing the mask encoding's resolution. Note that using $k_m^1 > 1$ allows the mask encoding to capture more information than a simple segmentation mask, which proves to be key for obtaining good accuracy.

Each refinement module R^i aggregates information from a coarse mask encoding M^i and features F^i from the corresponding layer of the bottom-up computation (we always use the last convolutional layer prior to pooling). By construction, M^i and F^i have the same spatial dimensions; the goal of R^i is to generate a new mask encoding M^{i+1} with double spatial resolution based on inputs M^i and F^i. We denote this via $M^{i+1} = R^i(M^i, F^i)$. This process is applied iteratively n times (where n is the number of pooling stages) until the feature map has the same dimensions as the input image patch. Each module R^i has separate parameters, allowing the network to learn stage-specific refinements.

The refinement module aims to enhance the mask encoding M^i using features F^i. As M^i and F^i have the same spatial dimensions, one option is to first simply concatenate M^i and F^i. However, directly concatenating F^i with M^i poses two challenges. Let k_m^i and k_f^i be the number of channels in M^i and F^i respectively. Typically, k_f^i can be quite large in modern CNNs, so using F^i directly would be computationally expensive. Second, typically $k_f^i \gg k_m^i$, so directly concatenating the features maps risks drowning out the signal in M^i.

Instead, we opt to first reduce the number of channels k_f^i (but preserving the spatial dimensions) of these features through a 3×3 convolutional module (plus ReLU), generating "skip" features S^i, with $k_s^i \ll k_f^i$ channels. This substantially reduces computational requirements,

moreover, it allows the network to transform F^i into a form S^i more suitable for use in refinement. An important but subtle point is that during full image inference, as with the features F^i, skip features are shared by overlapping image patches, making them highly efficient to compute. In contrast, the remaining computations of R^i are patch dependent as they depend on the local mask M^i and hence cannot be shared across locations.

The refinement module concatenates the mask encoding M^i with the skip features S^i resulting in a feature map with $k_m^i + k_s^i$ channels, and applies another 3×3 convolution (plus ReLU) to the result. Finally, the output is upsampled using bilinear upsampling by a factor of 2, resulting in a new mask encoding M^{i+1} with k_m^{i+1} channels (k_m^{i+1} is determined by the number of 3×3 kernels used for the convolution). As with the convolution for generating the skip features, this transformation is used to simultaneously learn a nonlinear mask encoding from the concatenated features and to control the capacity of the model. See Figure 6.1d for a complete overview of the refinement module R. Further optimizations to R are possible, for details see Figure 6.3.

Note that the refinement module uses only convolution, ReLU, bilinear upsampling, and concatenation, hence it is fully backpropable and highly efficient. In Section 6.3.1, we analyze different architecture choices for the refinement module in terms of performance and speed. As a general design principle, we aim to keep k_s^i and k_m^i large enough to capture rich information but small enough to keep computation low. In particular, we can start with a fairly large number of channels but as spatial resolution is increased the number of channels should decrease. This reverses the typical design of feedforward networks where spatial resolution decreases while the number of channels increases with increasing depth.

6.2.3 Training and Inference

We train SharpMask with an identical data definition and loss function as the original DeepMask model. Each training sample is a triplet containing an input patch, a label specifying if the input patch contains a centered object at the correct scale, and for positive samples a binary object mask. The network trunk parameters are initialized with a network that was pre-trained on ImageNet (Deng et al., 2009). All the other layers are initialized randomly from a uniform distribution.

Training proceeds in two stages: first, the model is trained to jointly infer a coarse pixel-wise segmentation mask and an object score, second, the feedforward path is "frozen" and the refinement modules trained. The first training stage is identical to the one described in Chapter 5. Once learning of the first stage converges, the final mask prediction layer of the feedforward network is removed and replaced with a linear layer that generates a mask encoding M^1 in place of the actual mask output. We then add the refinement modules to the network and train using standard stochastic gradient descent, backpropagating the error only on the horizontal and vertical convolution layers on each of the n refinement modules.

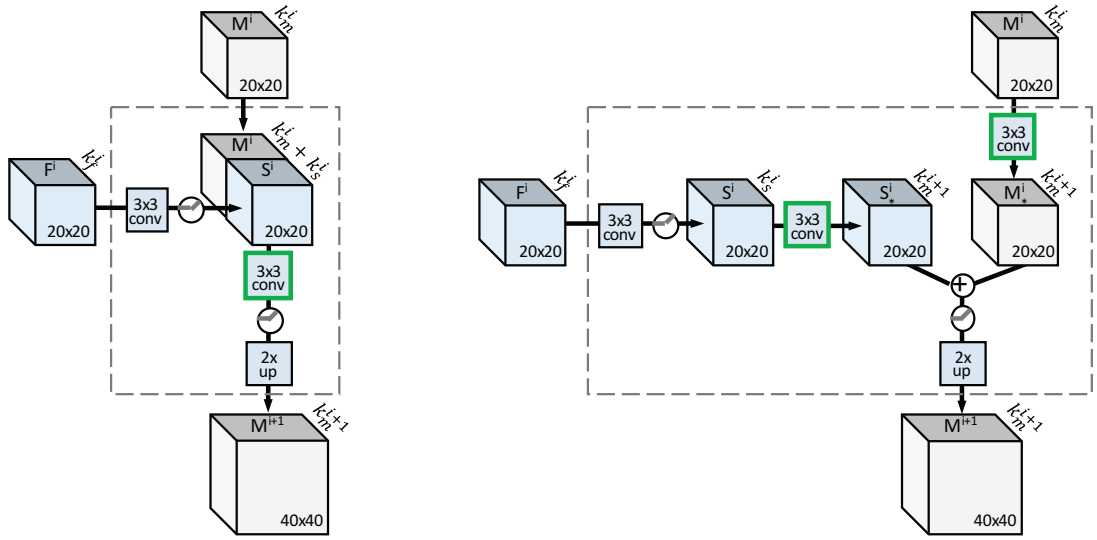


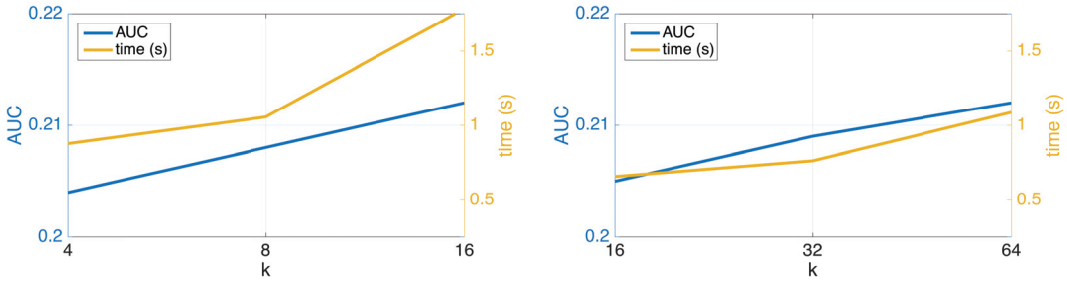
Figure 6.3 – (a) Original refinement model. (b) Refactored but *equivalent* model that leads to a more efficient implementation. The models are equivalent as concatenating along depth and convolving along the spatial dimensions can be rewritten as two separate spatial convolutions followed by addition. The green "conv" boxes denote the corresponding convolutions (note also the placement of the ReLUs). The refactored model is more efficient as skip features (both S^i and S^i_*) are shared by overlapping refinement windows (while M^i and M^i_* are not). Finally, observe that setting $k^i_m = 1$, $\forall\, i$, and removing the top-down convolution would transform our refactored model into a standard "skip" architecture (however, using $k^i_m = 1$ is not effective in our setting).

This two-stage training procedure was selected for three reasons. First, we found it led to faster convergence. Second, at inference time, a *single* network trained in this manner can be used to generate either a coarse mask using the forward path only or a sharp mask using our bottom-up/top-down approach. Third, we found the gains of fine-tuning through the entire network to be minimal once the forward branch had converged.

During full-image inference, most computation for neighboring windows is shared through the use of convolution, including for skip layers S^i. However, as discussed, the refinement modules receive a unique input M^1 at each spatial location, hence, computation proceeds independently at each location for this stage. Rather than refining every proposal, we simply refine only the most promising locations. Specifically, we select the top N scoring proposal windows and apply the refinement in a batch mode to these top N locations.

6.3 Experimental Results

We train our model on the training set of the COCO dataset (Lin et al., 2014), which contains 80k training images and 500k instance annotations. For most of our experiments, results are



(a) schedule a (b) schedule b

Figure 6.4 – Performance and inference time for multiple SharpMask variants with two different schedules, a and b. See text for detail.

reported on the first 5k COCO validation images. Mask accuracy is measured by Intersection over Union (IoU) which is the ratio of the intersection of the predicted mask and ground truth annotation to their union. A common method for summarizing object proposal accuracy is using the Average Recall (AR) between IoU 0.5 and .95 for a fixed number of proposals. Hosang et al. (2016) show that AR correlates well with object detector performance.

Our results are measured in terms of AR at 10, 100, and 1000 proposals and averaged across all counts (AUC) (as described Section 2.1.2). As the COCO dataset contains objects in a wide range of scales, it is also common practice to divide objects into roughly equally sized sets according to object pixel area a: small ($a < 32^2$), medium ($32^2 \leq a \leq 96^2$), and large ($a > 96^2$) objects, and report accuracy at each scale.

We use a different subset of the COCO validation set to decide architecture choices and hyper-parameter selection. We use a learning rate of 1e-3 for training the refinement stage, which takes about 2 days to train on an Nvidia Tesla K40m GPU. To mitigate the mismatch of per-patch training with convolutional inference, we found that training deeper model such as ResNet requires adding extra image content (32 pixels) surrounding the training patches and using reflective-padding instead of 0-padding at every convolutional layer. Finally, similar to DeepMask, we binarize our continuous mask prediction using a threshold of 0.2.

6.3.1 SharpMask Analysis

We begin by analyzing different parameter settings for the top-down refinement network. As described in Section 6.2, each of the four refinement modules R^i in SharpMask is controlled by two parameters k_m^i and k_s^i, which denote the size of the mask encoding M^i and skip encoding S^i, respectively. These parameters control network capacity and effect inference speed. We experiment with two different schedules for these parameters: (a) $k_m^i = k_s^i = k$ and (b) $k_m^i = k_s^i = \frac{k}{2^{i-1}}$ for each $i \leq 4$.

Figure 6.4 shows performance for the two schedules for different k both in terms of AUC and

	Box Proposals				Segmentation Proposals						
	AR^{10}	AR^{100}	AR^{1K}	AUC	AR^{10}	AR^{100}	AR^{1K}	AUC^{S}	AUC^{M}	AUC^{L}	AUC
EdgeBoxes (Zitnick and Dollár, 2014)	7.4	17.8	33.8	13.9	-	-	-	-	-	-	-
Geodesic (Krähenbühl and Koltun, 2014)	4.0	18.0	35.9	12.6	2.3	12.3	25.3	1.3	8.6	20.5	8.5
Rigor (Humayun et al., 2014)	-	13.3	33.7	10.1	-	9.4	25.3	2.2	6.0	17.8	7.4
SelectiveSearch (Uijlings et al., 2013)	5.2	16.3	35.7	12.6	2.5	9.5	23.0	0.6	5.5	21.4	7.4
MCG (Pont-Tuset et al., 2015)	10.1	24.6	39.8	18.0	7.7	18.6	29.9	3.1	12.9	32.4	13.7
RPN (Bell et al., 2016; Ren et al., 2015)	12.8	29.2	42.6	21.4	-	-	-	-	-	-	-
DeepMaskBase	15.3	31.3	44.6	23.3	12.6	24.5	33.1	2.3	26.6	33.6	18.3
DeepMask	18.7	34.9	46.5	26.2	14.4	25.8	33.1	2.2	27.3	37.4	19.4
DeepMaskZoom	18.3	36.3	50.3	27.2	14.5	27.4	36.6	6.5	27.0	34.3	20.5
SharpMask	19.7	36.4	48.2	27.4	15.6	27.6	35.5	2.5	29.1	40.4	20.9
SharpMaskZoom	20.1	39.4	52.8	29.1	16.1	30.3	39.2	6.9	29.7	38.4	22.4
$SharpMaskZoom^{2}$	19.2	39.9	55.0	29.2	15.4	30.7	40.8	10.6	27.3	36.0	22.5

Table 6.1 – Results on the COCO validation set on box and segmentation proposals. AR at different proposals counts is reported and also AUC (AR averaged across all proposal counts). For segmentation proposals, we also report AUC at multiple scales. SharpMask has largest for segmentation proposals and large objects.

inference time (measured when refining the top 500 proposals per image, at which point object detection performance saturates, see Figure 7.4). We consistently observe higher performance as we increase the capacity, with no sign of overfitting. Parameter schedule b, in particular with $k = 32$, has the best trade-off between performance and speed, so we chose this as our final model.

We note that we were unable to obtain good results with schedule a for $k \leq 2$, indicating the importance of using sufficiently large k. Also, we observed that a single 3×3 convolution encounters learning difficulties when ($k_s^i \ll k_f^i$). Therefore, in all experiments we used a sequence of two 3×3 convolutions (followed by ReLUs) to generate S^i from F^i, reducing F^i to 64 channels first followed by a further reduction to k_s^i channels.

Finally, we performed two additional ablation studies. First, we removed all downward convolutional layers, set $k_m^i = k_s^i = 1$, and averaged the output of all layers. Second, we kept the vertical convolutions but removed all horizontal convolutions. These two variants are related to "skip" and "deconvolutional" networks, respectively. Neither setup showed meaningful improvement over the baseline feedforward network. In short, we found that both horizontal and vertical connections were necessary for this task.

6.3.2 Comparison with State of the Art

We train SharpMask using DeepMask as the feedforward network. As the two networks have an identical score branch, we can disentangle the performance improvements achieved by our top-down refinement approach. We observe a considerable boost in performance on AR due to the top-down refinement. We note that improvement for segmentation predictions is bigger than box predictions, which is not surprising, as sharpening masks might not change

the tight box around the objects in many examples. Inference for SharpMask is .76s per image. Moreover, the refinement modules require fewer than 3M additional parameters.

Table 6.1 compares the performance of our model, SharpMask, to other existing methods on the COCO dataset and, in particular, to DeepMask. We compare results both on box and segmentation proposals (for box proposals we extract tight bounding boxes surrounding our segmentation masks). Figure 6.5 (a-b) compares the performance of SharpMask to existing proposal methods COCO (using both boxes and segmentations). Shown is the AR of each method as a function of the number of generated proposals. SharpMask achieves the state of the art in all metrics for both speed and accuracy by a large margin.

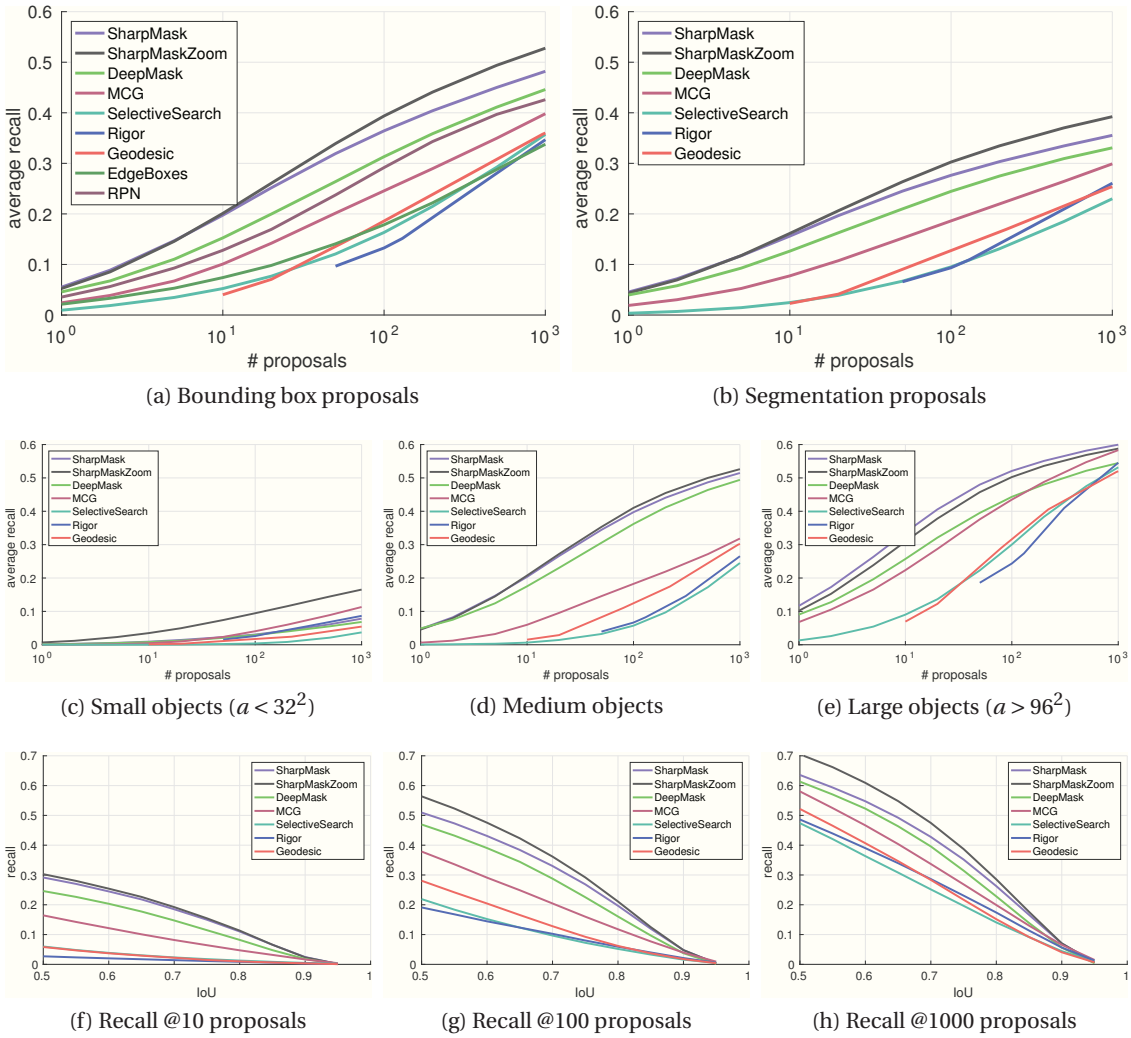
The COCO dataset contains objects in a wide range of scales. Figure 6.5 (c-e) shows performance at each scale (according to equally divided set of objects by its area a); all models perform poorly on small objects. To improve accuracy of DeepMask and SharpMask to small objects, we apply it at additional one or two smaller scales (*DeepMaskZoom* and *SharpMaskZoom*) and achieves a large boost in AR for small objects (at a cost of increased inference time).

Figure 6.5 (f-h) shows the recall each model achieves as the IoU varies, shown for different number of proposals per image. SharpMask achieves a higher recall in every scenario.

In Figure 6.2, we show direct comparison between SharpMask and DeepMask and we can see SharpMask generates higher-fidelity masks that more accurately delineate object boundaries. In Figure 6.6, we show more qualitative results.

6.4 Summary

In this chapter, we introduce a novel architecture for object instance segmentation, based on an augmentation of feedforward networks with top-down refinement modules. This network is able to merge the high-level semantic information with the low-level spatial information of the network. We show that this refinement improves the quality of object segmentations and, when applied in segmentation object proposal generation, it achieves a new state of the art in terms of performance. Building on the top of DeepMask for generating object proposals, we show accuracy improvements of 10-20% in average recall for various setups. Moreover, the proposed refinement approach is general and could be applied to other pixel-labeling tasks.



(a) Bounding box proposals

(b) Segmentation proposals

(c) Small objects ($a < 32^2$)

(d) Medium objects

(e) Large objects ($a > 96^2$)

(f) Recall @10 proposals

(g) Recall @100 proposals

(h) Recall @1000 proposals

Figure 6.5 – (a-b) Average recall versus number of box and segment proposals on COCO. (c-e) AR versus number of proposals for different object scales on segment proposals. (f-h) Recall versus IoU threshold for different number of segment proposals.

Figure 6.6 – SharpMask proposals with highest IoU to the ground truth on selected COCO images. Missed objects (no matching proposals with IoU > 0.5) are marked in red. The last row shows a number of failure cases.

7 Application of Proposals: Learning to Detect Objects

In the two previous chapters, we presented a novel set of discriminative trainable algorithms to generate state-of-the-art (both in terms of performance and inference time) segmentation proposals. It was also argued that one of the most important applications of object proposals is in object detection (object recognition subproblems (b) and (d)): the current state-of-the-art pipeline consists of applying a CNN classifier on the set of generated proposals (He et al., 2014; Girshick et al., 2014; Girshick, 2015; Ren et al., 2015; Bell et al., 2016).

In this final chapter, we show how the proposal algorithm presented on last chapter can achieve state-of-the-art performance on object detection with boxes and segments, utilizing the Fast R-CNN framework (Girshick, 2015) and the *MultiPath Network (MPN)* (Zagoruyko et al., 2016) as the classifier.

We start by giving a brief overview of the Fast R-CNN pipeline for object detection (Section 7.1). Then, we quickly describe the main components of MPN, the classifier of choice for our experiments (Section 7.2). Finally, in Section 7.3, we show how SharpMask proposals, when coupled with the MPN classifier, can achieve state-of-the-art results in object detection with both bounding boxes and segments on the challenging COCO (Lin et al., 2014) dataset.

7.1 Overview of Fast R-CNN

Fast R-CNN (Girshick, 2015) is a convolutional neural network that takes an image and a set of proposals as input and outputs, for each proposal, softmax probabilities and per-class bounding box regression offsets.

First the image is processed through multiple convolutional and pooling layers to produce a final feature map. Then, for each object proposal, a Region of Interest (RoI) pooling layer extracts a fixed-length feature vector from the feature map. Each feature vector passes through a sequence of fully connected layers and branches into two different outputs: one that produces the softmax probabilities over all K classes on the training set (plus the background) and another that outputs 4 real-valued numbers for each of the K object classes (see Figure 7.1).

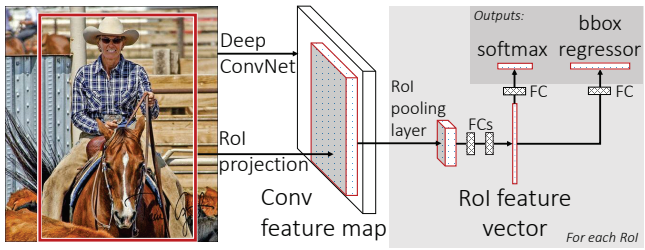


Figure 7.1 – Fast R-CNN architecture. The network has as input an image and a set of region of interests (RoI), stemmed from the proposals. Each proposals is pooled into a fixed-size feature map, and mapped into a vector by fully connected layers. The network has two outputs: one for the probability of the class present on the RoI and per-class bounding-box regression offsets. Image taken from (Girshick, 2015).

Fast R-CNN is trained to optimize two sibling output layers (for each RoI). The first outputs a discrete probability distribution per RoI, $p = (p_0, ..., p_K)$, over $K+1$ categories (which is computed with a softmax). The second layer outputs bounding-box regression offsets, $t^k = (t^k_x, t^k_y, t^k_w, t^k_h)$, for each K category. t^k is parametrized such that it specifies a scale-invariant translation and log-scale height/width shift relative to an object proposal.

Each RoI is labeled with a ground-truth class u and a ground truth bounding box regression target v. Training is achieved by jointly minimizing the classification and the box regression losses:

$$\mathcal{L}(p, u, t^u, v) = \mathcal{L}_{\text{cls}}(p, u) + \lambda \mathbb{1}(u > 0)\mathcal{L}_{\text{loc}}(t^u, v)\,. \tag{7.1}$$

The first term in the loss, $\mathcal{L}_{\text{cls}} = -\log p_u$, is the log loss of true class u. The second task loss, $\mathcal{L}_{\text{loc}}$, is defined over a tuple of true bounding box regression targets for class u, $v = (v_x, v_y, v_w, v_h)$, and the predicted tuple $t^u = (t^u_x, t^u_y, t^u_w, t^u_h)$, again for class u. By convention, the background class is labeled $u = 0$, so the localization loss only applies to object categories different of background (*i.e.* $u > 0$). For bounding box regression, the loss is:

$$\mathcal{L}_{\text{loc}}(t^u, v) = \sum_{i \in \{x,y,w,h\}} L(t^u_i - v_i)\,, \tag{7.2}$$

in which

$$L(x) = \begin{cases} 0.5x^2, \text{if} |x| < 1 \\ |x| - 0.5, \text{otherwise}\,, \end{cases} \tag{7.3}$$

is a robust L_1 loss.

For each image during training, a set of N RoIs are sampled from the image. As in the original implementation of Fast R-CNN, 25% of the RoIs are taken from the set of object proposals that have an intersection over union (IoU) overlap with ground truth bounding box of at least 0.5.

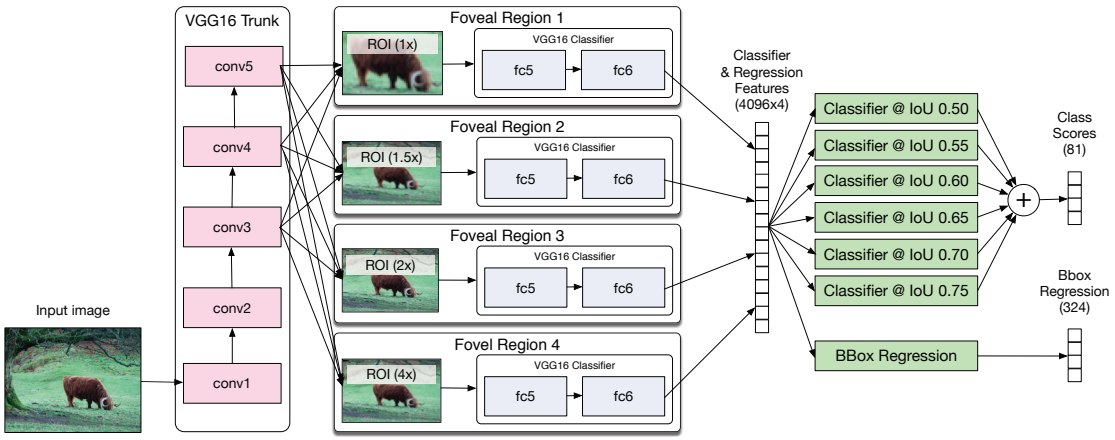


Figure 7.2 – MultiPath Network architecture contains three key modifications over standard CNN classifiers: skip connections, foveal regions, and and an integral loss function. Together these modifications allow information to flow along multiple paths through the network, enabling the classifier to operate at multiple scales, utilize context effectively, and perform more precise object localization. Image taken from (Zagoruyko et al., 2016).

These are the RoIs corresponding to object classes (*i.e.* $u > 0$). The remaining RoIs are taken from proposals which have a maximum IoU with the ground truth in the interval $[0.1, 0.5)$ (following (Girshick et al., 2014)). These are the background examples (*i.e.* $u = 0$).

Fast R-CNN is trained with stochastic gradient descent by backpropagating the error through the fully connected layers, the RoI layer and the convolutional layers. In the experiments in this chapter, we use the exact same hyper-parameters as in (Girshick, 2015).

7.2 The MultiPath Network

The MultiPath network (Zagoruyko et al., 2016) is a CNN-based image classifier based on the VGG-D network (Simonyan and Zisserman, 2015). VGG-D is a standard deep CNN containing a series of small filters interleaved by pooling layers. MPN extends VGG-D in three ways: foveal structure, skip connections and integral loss function.

Context plays an important role in object classification (Torralba, 2003). To incorporate more context in the model and improve its performance, MPN adds four region crops with different "foveal" views (see Figure 7.2). RoI pooling is used to generate feature maps of same spatial dimension given differently-sized foveal regions.

Effective localization of small objects requires higher-resolution features from earlier layers (Bell et al., 2016; Hariharan et al., 2015). To improve performance of small objects, MPN concatenates the RoI-pooled normalized features from different convolutions layers, and provides these as input to each foveal classifier (see Figure 7.2).

Figure 7.3 – Selected detection results on COCO. Only high-scoring detections are shown. While there are missed objects and false positives, many of the detections and segmentations are quite good.

In the original Fast R-CNN loss (Equation 7.1), the classification loss $\mathcal{L}_{\text{cls}}$ does not prefer object proposals with high IoU: all proposals with IoU greater than .5 are treated equally. Ideally, proposals with higher overlap to the ground truth should be scored more highly. MPN thus, utilizes a modified $\mathcal{L}_{\text{cls}}$ to explicitly measure integral loss over all IoU thresholds.

For more detail about the MPN architecture, we refer the reader to (Zagoruyko et al., 2016).

7.3 Experimental Results

In the following experiments, we coupled SharpMask proposal with a CNN classifier and use the Fast R-CNN pipeline to detect objects (Zagoruyko et al., 2016). Figure 7.3 and Figure 7.5 show selected detection results from this system.

The experiments are done in the COCO (Lin et al., 2014) object detection dataset, which was recently introduced to push object detection to more challenging settings. COCO introduces a number of new challenges compared to previous object detection datasets (Everingham et al., 2010; Deng et al., 2009): (i) objects appears in a broad range of scales, including a high percentage of small objects, (ii) objects are less iconic, often in non-standard configurations and occlusion and (iii) the evaluation metric encourages more accurate object localization.

In the experiments in this section, we report both the average precision (AP) averaged over multiple IoUs, COCO official evaluation metric, and the average precision at IoU threshold of .5 (AP^{50}), PASCAL VOC (Everingham et al., 2010) official metric.

Effectiveness of SharpMask Proposals

We first compare the the performance of Fast R-CNN with MPN using two different types of proposals: Selective Search (SelSearch) (Uijlings et al., 2013), the most common proposal algorithm for object detection (and the proposal of choice on the original implementation of Fast R-CNN Girshick (2015)) and SharpMask.

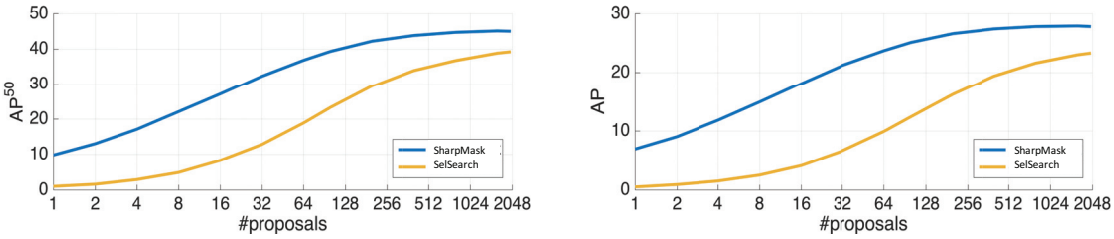


Figure 7.4 – AP^{50} and AP versus number and type of proposals. Accuracy saturates using 400 SharpMask proposals per image and using ~50 SharpMask proposals matches 2000 Selective Search proposals.

Figure 7.4 shows the comparison of bounding box detection results for SharpMask and SelSearch for AP^{50} and AP and for varying number of proposals. The SharpMask bounding box proposals are taken by extracting the smaller box fully containing each proposal. SharpMask achieves 28 AP, which is 5 AP higher than SelSearch. Not only accuracy is substantially higher using SharpMask, but fewer proposals are necessary to achieve top performance. SharpMask results saturates with ~400 SharpMask proposals per image and using only 50 SharpMask proposals matches accuracy with 2000 Selective Search proposals.

Next, we compare the result of the model with two important published baseline: Fast R-CNN (Girshick, 2015) and Faster R-CNN (Ren et al., 2015) (and considering the same classifier, VGG-D, in all of them). Table 7.1 shows results of these baselines without bells and whistles, trained on the train set only. SharpMask achieves top results with the VGG classifier, outperforming both RPN (Ren et al., 2015) and SelSearch (Uijlings et al., 2013).

	AP	**AP^{50}**
SelSearch + VGG (Girshick, 2015)	19.3	39.3
RPN + VGG (Ren et al., 2015)	21.9	42.7
SharpMask + VGG	25.2	43.4

Table 7.1 – COCO bounding box results of various baselines without bells and whistles, trained on the train set only, and reported on test-dev set (results for (Girshick, 2015; Ren et al., 2015) obtained from original papers). We denote methods using "proposal+classifier" notation for clarity. SharpMask achieves top results, outperforming both RPN and SelSearch proposals.

Comparison with other methods

Finally, Table 7.2 shows results from the 2015 COCO detection challenges[1]. The performance is reported with model ensembling and the MPN classifier. The ensemble model achieve 33.5 AP for boxes and 25.1 AP for segments, and achieved second place in the challenges. Note that for the challenges, both SharpMask and MPN used the VGG trunk (ResNets were concurrent work,

[1] http://mscoco.org/dataset/#detections-leaderboard

	AP	**AP**^{50}	**AP**^{75}	**AP**^{S}	**AP**^{M}	**AP**^{L}	**AR**^{1}	**AR**^{10}	**AR**^{100}	**AR**^{S}	**AR**^{M}	**AR**^{L}
ResNet++ (He et al., 2016)	28.2	51.5	27.9	9.3	30.6	45.2	25.7	37.4	38.2	16.8	43.9	57.6
SharpMask+MPN (Zagoruyko et al., 2016)	25.1	45.8	24.8	7.4	29.2	39.1	24.1	36.8	38.7	17.3	46.9	53.9
ResNet++ (He et al., 2016)	37.3	58.9	39.9	18.3	41.9	52.4	32.1	47.7	49.1	27.3	55.6	67.9
SharpMask+MPN (Zagoruyko et al., 2016)	33.5	52.6	36.6	13.9	37.8	47.7	30.2	46.2	48.5	24.1	56.1	66.4
ION (Bell et al., 2016)	31.0	53.3	31.8	12.3	33.2	44.7	27.9	43.1	45.7	23.8	50.4	62.8
CMU_A2 (Shrivastava et al., 2016)	25.7	46.0	26.1	5.90	28.7	41.7	24.8	35.5	36.5	10.5	43.0	58.2

Table 7.2 – **Top**: Winners of the 2015 COCO segmentation challenge. **Bottom**: Winners of the 2015 COCO bounding box challenge.

and won the competitions). We have not re-run our model with ensembling and additional bells and whistles after integrating ResNets into SharpMask.

7.4 Summary

In this chapter, we study how SharpMask performs in an important application of object proposals: object detection. We consider the Fast R-CNN framework, which consists of generating a set of proposals and classifying them with a classifier. We show that by simply changing the set of generated proposals to SharpMask, an object detection system can improve the performance in terms of AP by a big margin (around 5 points). Moreover, a much smaller number of SharpMask proposals per image is required to bypass the performance using Selective Search (the most common set of proposals used in detection): using only 50 SharpMask proposals, we are able to match the accuracy of Selective Search with 2000 proposals.

Figure 7.5 – Extra selected detection results on COCO. Only high-scoring detections are shown. While there are missed objects and false positives, many of the detections and segmentations are quite good.

8 Conclusion

8.1 Overview

In this thesis, we studied different large-scale image segmentation problems. The lack of a universal criterion for segmentation led to different definitions of segmentation in the context of computer vision. Therefore, we addressed three different important segmentation problems: semantic segmentation (Chapters 3 and 4), object proposals generation (Chapters 5 and 6) and object detection with segments (Chapters 7).

We advocate the use of algorithms that learn from raw data (*e.g.* pixels) and are easy to scale. Deep learning methods, and in particular CNNs, fit well with this objective. Throughout this thesis, we proposed different CNN-based algorithms to deal with the three segmentation problems mentioned above.

In Chapter 3, we studied the problem of fully supervised semantic segmentation. We proposed a recurrent convolutional neural network that allows us to consider a larger input context (while limiting its capacity). The proposed approach is able to model non local class dependencies in a scene directly from raw pixels in a rather simple way. This is essential for a model to capture long range (pixel) label dependencies. Our approach achieves competitive results (in two standard semantic segmentation datasets, Stanford Background and SIFT Flow dataset) without the need of any expensive graphical model or segmentation technique to ensure labeling.

Large-scale fully supervised semantic segmentation dataset, however, require a lot of human labor to be annotated. In Chapter 4, we proposed a model that is able to infer object segmentation by leveraging only object class information. The proposed model, based on a CNN architecture, is designed in a way it is constrained to put more weight on pixels which are important for classifying the image with its image-level label. We also proposed a number of different smoothing priors that are able to boost the performance further and achieve competitive results to fully supervised methods. The model is trained on a large corpus of image-level annotated images extracted from ImageNet. We are able to surpass previous

state-of-the-art models for weakly supervised semantic segmentation by a large margin on the challenging PASCAL VOC dataset (an increase form 30% to 90% in average per-class accuracy).

In Chapter 5, we studied the problem of object proposal generation. Most of the previous approaches rely on low level vision cues to generate object proposals. We proposed, instead, a CNN-based model that is able to discriminatively learn a set of segmentation proposals directly from pixels. Our method shows that learning features for object proposal generation is not only feasible but effective and efficient. This approach surpasses the previous state of the art, on both PASCAL VOC and COCO datasets, by a large margin on both performance (an increase of ~40%-50% in AUC) for box and segmentation proposals and for both PASCAL VOC and COCO datasets. Moreover, our method performs faster than previous methods (although we require GPU for a fast inference).

In Chapter 6, a new architecture for object instance segmentation was introduced. We proposed an augmentation of feedforward network with top-down refinement modules. The top-down augmentation uses the object-level information of the higher layers of the network with spatial information from the lower level features. Our new architecture is able to increase the quality of object instance segmentation masks by iteratively refining a mask encoding using lower level features of the bottom-up path. We used the new proposed architecture in the same object proposal problem studied in the previous chapter. We showed qualitatively and quantitatively that the top-down refinement augmentation improves the quality of the masks (an accuracy improvements of 10-20% in average recall for various setups). Although we applied the proposed architecture to the problem of proposal, this approach is suitable to other object instance segmentation problems.

Finally, in Chapter 7, we studied how the proposal algorithm described in previous chapter performs in an important application: object detection. We considered the Fast R-CNN framework, which consists of generating a set of proposals and classifying them with a classifier. We showed that by simply changing the set of generated proposals to SharpMask, an object detection system can improve the performance in terms of AP by a big margin (around 5 points). Moreover, a much smaller number of SharpMask proposals per image is required to bypass the performance when using Selective Search (the most common set of proposals used in detection).

8.2 Perspectives for Future Work

Full pipeline for object detection Current state-of-the-art detection systems are based on two steps: proposal generation and classification of these proposals. DeepMask/SharpMask models could be extended to deal with object detection directly and circumvent this two-step system. The simplest extension (although possibly not the optimal) would be to instead of detecting the presence or not of an object, detect directly one of the classes present in the training set. Some work would be necessary, however, to deal with the amount of false positives that would possibly be generated by such system.

Less supervision Deep learning methods are changing the landscape of computer vision. These methods, although quite powerful, require a large amount of data to be able to learn the necessary representation. At the same time, a huge amount of unlabeled/weakly-labeled data is being generated everyday. New learning algorithms are required to make use of so much data. Recently, many authors start to learn visual features in convolutional networks using only the structure of images and videos (Doersch et al., 2015; Wang and Gupta, 2015; Pathak et al., 2016; Isola et al., 2015), although there still exists much space to be explored. The temporal coherence in videos is also a strong signal that could be exploited in vision learning algorithms. For example, we could consider the flow of moving objects in videos as a weak supervision for class agnostic object masks.

Hierarchical semantic constraints Object segmentation can also be improved by leveraging high level semantic constraints. One approach would be to infer and integrate global scene aspects (such as 'outdoor' or 'indoor') into a deep learning system (such constraints would avoid predictions of labels in unlikely situations). Another direction would be to infer and leverage hierarchy of semantic labels, *e.g.*, if the model could learn constraints such that a bike is made of two 'wheels' and one 'seat' (which is more complex than only knowing that a 'wheel' is nearby a 'seat', as current systems infer), this would constrain its predictions in a much better way. (Gould et al., 2009; Socher et al., 2011) proposed systems which go along these lines. However, the fact they considered only datasets with few categories must have limited the kind of semantic they could reach.

Language Language is a very special form of regularity in the world. Babies have shown to use language to improve their perception of the world (Smith and Gasser, 2005). Another research direction to improve segmentation would be to constrain the hierarchy of semantic labels with additional semantic knowledge coming form language processing. These semantics could be exploited from knowledge bases such as WordNet[1], *e.g.*, relations such as 'door *has part* lock' could consolidate relations extracted from labeled images. Other types of relations such as 'auto *has instance* SUV' might also help when two labels in the database have some kind of meaning overlap. Semantic knowledge could also be exploited from large unlabeled corpora: class labels which co-occurs often in a same paragraph have possibly higher chances to occur in the same scene. Leveraging natural language information could also help in zero-shot or one-shot detection.

[1] http://wordnet.princeton.edu

Bibliography

Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2012.

Pablo Arbeláez, Jordi Pont-Tuset, Jonathan Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.

Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 2008.

Sean Bell, C. Lawrence Zitnick, Kavita Bala, and Ross B. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. 2016.

Léon Bottou. Stochastic gradient learning in neural networks. In *In Proceedings of Neuro-Nîmes.*, 1991.

Stephen P. Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

John S. Bridle. Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition. 1990.

Arthur E. Bryson, W. F. Denham, and S. E. Dreyfus. Optimal programming problems with inequality constraints I: necessary conditions for extremal solutions. *Journal of the American Institute of Aeronautics and Astronautics (AIAA)*, 1963.

Holger Caesar, Jasper R. R. Uijlings, and Vittorio Ferrari. Region-based semantic segmentation with end-to-end training. In *Proceedings European Conference on Computer Vision (ECCV)*.

Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic segmentation with second-order pooling. In *European Conference on Computer Vision (ECCV)*, 2012.

João Carreira and Cristian Sminchisescu. Cpmc: Automatic object segmentation using constrained parametric min-cuts. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2012.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *International Conference on Learning Representations (ICLR)*, 2015.

Ming-Ming Cheng, Ziming Zhang, Wen-Yan Lin, and Philip Torr. BING: Binarized normed gradients for objectness estimation at 300fps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio.

Anna Choromanska, Mikael Henaff, Michaël Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *International Conference on Artificial Intelligence and Statistics, (AISTATS)*, 2015.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research (JMLR)*, 2011.

Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *International Conference on Computer Vision, (ICCV)*, 2015.

Piotr Dollár and Lawrence Zitnick. Structured forests for fast edge detection. In *International Conference on Computer Vision, (ICCV)*, 2013.

Piotr Dollár and Lawrence Zitnick. Fast edge detection using structured forests. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2015.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philipp Hausser, Caner Hazirbas, Vladimir Golkov, Patrick v.d. Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *International Conference on Computer Vision, (ICCV)*, 2015.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research (JMLR)*, 2011.

David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *International Conference on Computer Vision, (ICCV)*, 2015.

Jeffrey L. Elman. Finding structure in time. In *Cognitive Sciences*, 1990.

Dumitru Erhan, Christian Szegedy, Alex Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.

Marc Everingham, Luc Van Gool, C. K. I. Williams, J. Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision (IJCV)*, 2010.

Clement Farabet. *Towards Real-Time Image Undersding with Convolutional Networks*. PhD thesis, Université Paris-Est, 2014.

Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2013.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient matching of pictorial structures. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 2004.

Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2010.

Martin A. Fischler and Robert A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 1973.

David A. Forsyth, Jitendra Malik, Margaret M. Fleck, Hayit Greenspan, Thomas Leung, Serge Belongie, Chad Carson, and Christoph Bregler. Finding pictures of objects in large collections of images. Technical report, EECS Department, University of California, Berkeley, 1996.

King-Sun Fu and John E. Albus. *Syntatic Pattern Recognition and Applications*. Prentice-Hall Englewood Cliffs, 1982.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 1980.

Carolina Galleguillos, Andrew Rabinovich, and Serge Belongie. Object categorization using co-occurrence, location and appearance. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.

Carolina Galleguillos, Brian McFee, Serge Belongie, and Gert Lanckriet. Multi-class object localization by combining local contextual interactions. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.

Ross B. Girshick. Fast R-CNN. In *International Conference on Computer Vision (ICCV)*, 2015.

Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a scene into geometric and semantically consistent regions. In *International Conference on Computer Vision, (ICCV)*, 2009.

David Grangier, Leon Bottou, and Ronan Collobert. Deep convolutional networks for scene parsing. In *International Conference on Machine Learning (ICML) Deep Learning Workshop*, 2009.

Alex Graves and Jurgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

Robert M. Gray. Toeplitz and circulant matrices: A review. *Communications and Information Theory*, 2005.

Allen R. Hanson and Edward M. Riseman. VISIONS: A computer system for interpreting scenes. In *Computer Vision Systems*. 1978.

Barath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision (ECCV)*, 2014.

Barath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

Zeeshan Hayder, Xuming He, and Mathieu Salzmann. Learning to co-generate object proposals with a deep structured network. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision (ECCV)*, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

Xuming He, Richard S. Zemel, and Miguel Á. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 2012.

Jan Hosang, Rodrigo Benenson, Piotr Dollár, and Bernt Schiele. What makes for effective detection proposals? *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2016.

David Hubel and Torsten Wiesel. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *Journal of Physiology*, 1962.

Ahmad Humayun, Fuxin Li, and James M. Rehg. RIGOR: Reusing Inference in Graph Cuts for generating Object Regions. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.

Yu ichi Ohta, Takeo Kanade, and Toshiyuki Sakai. An analysis system for scenes containing objects with substructures. In *International Joint Conference on Pattern Recognitions*, 1978.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML) Deep Learning Workshop*, 2015.

Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H. Adelson. Learning visual groups from co-occurrences in space and time. *CoRR*, 2015.

Michael I. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Conference of the Cognitive Science Society*, 1986.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2014.

Alexander Kolesnikov and Christoph H. Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. In *European Conference on Computer Vision (ECCV)*, 2016.

Philipp Krähenbühl and Vladen Koltun. Learning to propose objects. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

Philipp Krähenbühl and Vladlen Koltun. Geodesic object proposals. In *European Conference on Computer Vision (ECCV)*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

Pawan Kumar and Daphne Koller. Efficiently selecting regions for scene understanding. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

Weicheng Kuo, Bharath Hariharan, and Jitendra Malik. Deepbox: Learning objectness with convolutional networks. In *International Conference on Computer Vision (ICCV)*, 2015.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*, 2001.

Svetlana Lazebnik and Maxim Raginsky. An empirical bayes approach to contextual region classification. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

Yann LeCun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems (NIPS)*, 1990.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Pierre Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 2015.

Victor Lempitsky, Andrea Vedaldi, and Andrew Zisserman. A pylon model for semantic segmentation. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

Stan Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer Publishing Company, Incorporated, 2009.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

Ce Liu, Jenny Yuen, and Antonio Torralba. Nonparametric scene parsing via label transfer. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2011.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 2004.

Jitendra Malik, Pablo Arbeláez, João Carreira, Katerina Fragkiadaki, Ross Girshick, Georgia Gkioxari, Saurabh Gupta, Bharath Hariharan, Abhishek Kar, and Shubham Tulsiani. The three r's of computer vision. *Pattern Recognition Letters*, 2016.

Oded Maron and Tomas Lozano-Pérez. A framework for multiple instance learning. In *Advances in Neural Information Processing Systems (NIPS)*, 1998.

David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., 1982.

Warren Mcculloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 1943.

Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Daniel Munoz, Andrew Bagnell, and Martial Hebert. Stacked hierarchical labeling. In *Proceedings European Conference on Computer Vision (ECCV)*, 2010.

Mohammad Najafi, Sarah Taghavi Namin, Mathieu Salzmann, and Lars Petersson. Sample and filter: Nonparametric scene parsing via efficient filtering. 2016.

Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *International Conference on Computer Vision (ICCV)*, 2015.

Yuichi Ohta. *Knowledge-based Interpretation of Outdoor Natural Color Scenes*. Pitman Publishing, Inc., 1985.

Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free? - weakly-supervised learning with convolutional neural networks. In *Computer Vision and Pattern Recognition, CVPR*, 2015.

Stephen E. Palmer. *Vision science : photons to phenomenology*. MIT Press, 1999.

George Papandreou, Liang-Chieh Chen, Kevin P. Murphy, and Alan L. Yuille. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *International Conference on Computer Vision, (ICCV)*, 2015.

Deepak Pathak, Philipp Krähenbühl, and Trevor Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *International Conference on Computer Vision, (ICCV)*, 2015.

Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context encoders: Feature learning by inpainting. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Pedro O. Pinheiro and Ronan Collobert. Recurrent convolutional neural networks for scene parsing. In *Advances in Neural Information Processing Systems (NIPS), Deep Learning Workshop*, 2013.

Pedro O. Pinheiro and Ronan Collobert. Recurrent convolutional neural networks for scene labeling. In *International Conference on Machine Learning (ICML)*, 2014.

Pedro O. Pinheiro and Ronan Collobert. From image-level to pixel-level labeling with convolutional networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Pedro O. Pinheiro, Ronan Collobert, and Piotr Dollár. Learning to segment object candidates. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

Pedro O. Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *European Conference on Computer Vision (ECCV)*, 2016.

Boris T. Polyak and Anatoli Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optimization*, 1992.

Jordi Pont-Tuset, Pablo Arbeláez, Jonathan Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. 2015.

Andrew Rabinovich, Adrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge Belongie. Objects in context. In *International Conference on Computer Vision (ICCV)*, 2007.

Esa Rahtu, Juho Kannala, and Matthew B. Blaschko. Learning a category independent object detection cascade. In *International Conference on Computer Vision, (ICCV)*, 2011.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

Maximilian Riesenhuber and Tomaso Poggio. Models of object recognition. *Nature Neuroscience*, 2000.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 1951.

Tony Robinson. An application of recurrent nets to phone probability estimation. *Transactions on Neural Networks*, 1994.

David Rumelhart, Geoffrey Hinton, and Ricahrd J. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1986.

Fatemehsadat Saleh, Mohammad Sadegh Ali Akbarian, Mathieu Salzmann, Lars Petersson, Stephen Gould, and Jose M. Alvarez. Built-in foreground/background prior for weakly-supervised semantic segmentation. In *European Conference on Computer Vision (ECCV)*, 2016.

Hannes Schulz and Sven Behnke. Learning object-class segmentation with convolutional neural networks. In *European Symposium on Artificial Neural Networks (ESANN)*, 2012.

Alexander G Schwing and Raquel Urtasun. Fully connected deep structured networks. *arXiv:1503.02351*, 2015.

Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

Pierre Sermanet, David Eigen, X. Zhang, M. Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Abhishek Sharma, Oncel Tuzel, and David W. Jacobs. Deep hierarchical parsing for semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

Jamie Shotton, M. Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision (IJCV)*, 2009.

Abhinav Shrivastava, Abhinav Gupta, and Ross B. Girshick. Training region-based object detectors with online hard example mining. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 2015.

Linda B. Smith and Michael Gasser. The development of embodied cognition: Six lessons from babies. *Artificial Life*, 2005.

Ricahrd Socher, Brody Huval, Bharath Bhat, Christopher D. Manning, and Andrew Y. Ng. Convolutional-recursive deep learning for 3d object classification. In *Advances in Neural Information Processing Systems (NIPS)*. 2012.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *International Conference on Machine Learning (ICML)*, 2011.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 2014.

Ivelin Stoianov, John Nerbonne, and Huub Bouma. Modelling the phonotactic structure of natural language words with simple recurrent networks. In *Computational Linguistics in the Netherlands*, 1997.

Christian Szegedy, Scott Reed, Dumitru Erhan, and Dragomir Anguelov. Scalable, high-quality object detection. In *arXiv:1412.1441*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dimitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5. rmsprop: Divide the gradient by a running average of its recent magnitude, 2012.

Joseph Tighe and Svetlana Lazebnik. Superparsing: Scalable nonparametric image parsing with superpixels. In *European Conference on Computer Vision (ECCV)*, 2010.

Joseph P. Tighe. *Towards Open-Universe Image Parsing with Broad Coverage*. PhD thesis, University of North Carolina at Chapel Hill, 2013.

Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. Weakly-supervised semantic segmentation using motion cues. In *European Conference on Computer Vision (ECCV)*, 2016.

Antonio Torralba. Contextual priming for object detection. *International Journal of Computer Vision (IJCV)*, 2003.

Jasper Uijlings, Koen van de Sande, Theo Gevers, and Arnold W.M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision (IJCV)*, 2013.

Jakob Verbeek and Bill Triggs. Scene segmentation with crfs learned from partially labeled images. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

Alexander Vezhnevets and Joachim M. Buhmann. Towards weakly supervised semantic segmentation by means of multiple instance and multitask learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

Alexander Vezhnevets, Vittorio Ferrari, and Joachim Buhmann. Weakly supervised semantic segmentation with a multi-image model. In *International Conference on Computer Vision (ICCV)*, 2011.

Alexander Vezhnevets, Vittorio Ferrari, and Joachim M. Buhmann. Weakly supervised structured output learning for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision (IJCV)*, 2004.

Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *International Conference on Computer Vision, (ICCV)*, 2015.

Paul J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.

Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *International Conference on Computer Vision, (ICCV)*, 2015.

Payman Yadollahpour, Dhruv Batra, and Gregory Shakhnarovich. Discriminative re-ranking of diverse segmentations. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

Sergey Zagoruyko, Adam Lerer, Tsung-Yi Lin, Pedro O. Pinheiro, Sam Gross, Soumith Chintala, and Piotr Dollár. A multipath network for object detection. *British Machine Vision Conference (BMVC)*, 2016.

Matthew Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, 2014.

Luming Zhang, Mingli Song, Zicheng Liu, Xiao Liu, Jiajun Bu, and Chun Chen. Probabilistic graphlet cut: Exploiting spatial structure cue for weakly supervised image segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

Luming Zhang, Yue Gao, Yingjie Xia, Ke Lu, Jialie Shen, and Rongrong Ji. Representative discovery of structure cues for weakly-supervised image segmentation. *Transactions on Multimedia*, 2014.

Yibiao Zhao and Song chun Zhu. Image parsing with stochastic scene grammar. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

S. Zheng, S. Jayasumana, B. Romera-Paredes, B. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural nets. In *International Conference on Computer Vision, (ICCV)*, 2015.

Song-Chun Zhu and David Mumford. A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, 2006.

Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision (ECCV)*, 2014.

Pedro Oliveira Pinheiro

http://pedro.opinheiro.com
Rue du Centre, 66 - St-Sulpice, 1025 - Suisse
e-mail: pedro@opinheiro.com – Tel: +41 (0)79 579 43 84

EDUCATION

École Polytechnique Fédérale de Lausanne *since 2013*
EPFL, *Lausanne, Switzerland*
PhD Candidate in Machine Learning and Computer Vision.

Institut National des Sciences Appliquées *2007 - 2012*
INSA, *Lyon, France*
Master of Engineering, Electrical Engineering.
Final year's major in Image and Signal Processing.

EXPERIENCE

Idiap Research Institute, *Switzerland* *since February 2013*
Applied Machine Learning Group
Research Assistant

Facebook, Inc, *California, USA* *August 2015 - July 2016*
Facebook AI Research
Remote Part-time Research Collaborator

Facebook, Inc, *California, USA* *February 2015 - July 2015*
Facebook AI Research
Research Internship in Machine Learning

Idiap Research Institute, *Switzerland* *October 2012 - January 2013*
Applied Machine Learning Group
Research Internship in Machine Learning

PUBLICATIONS

- P.O. Pinheiro, T.-Y. Lin, R. Collobert, P. Dollár, *Learning to Refine Object Segments*. ECCV, 2016 (spotlight).
- S. Zagoruyko, A. Lerer, T-Y Lin, P. O. Pinheiro, S. Gross, S. Chintala, P. Dollár, *A MultiPath Network for Object Detection*. BMVC, 2016.
- P.O. Pinheiro, R. Collobert, P. Dollár, *Learning to Segment Object Candidates*. NIPS, 2015 (spotlight).
- R. Lebret, P.O. Pinheiro, R. Collobert, *Phrase-Based Image Captioning*. ICML, 2015.
- P.O. Pinheiro, R. Collobert, *From Image-level to Pixel-level Labeling with Convolutional Networks*. CVPR, 2015.
- R. Lebret, P. O. Pinheiro, R. Collobert, *Simple image description generator via a linear phrase-based approach*. ICLR Workshop, 2015.
- P.O. Pinheiro, R. Collobert, *Recurrent Convolutional Neural Networks for Scene Labeling*. ICML, 2014.
- P.O. Pinheiro, R. Collobert, *Recurrent Convolutional Networks for Scene Parsing*. NIPS Workshop on Deep Learning, 2013.
- L. A. Madureira, D. Q Madureira, P. O. Pinheiro, *A multiscale numerical method for the heterogeneous cable equation*. Neurocomputing, 2012.

REVIEWER CVPR, ICCV, ECCV

SKILLS

Languages: *Portuguese* Native, *English* Fluent, *French* Fluent, *Spanish* Proficient
Languages (CS): Lua/LuaJIT, Torch, C/C++, UNIX Bash/Shell, Matlab, Python.