THE UNIVERSITY *of* EDINBURGH

Restoring the balance between stuff and things in scene understanding

Holger Caesar

Doctor of Philosophy
Institute of Perception, Action and Behaviour
School of Informatics
University of Edinburgh
2018

Abstract

Scene understanding is a central field in computer vision that attempts to detect objects in a scene and reason about their spatial, functional and semantic relations. While many works focus on things (objects with a well-defined shape), less attention has been given to stuff classes (amorphous background regions). However, stuff classes are important as they allow to explain many aspects of an image, including the scene type, thing classes likely to be present and physical attributes of all objects in the scene. The goal of this thesis is to restore the balance between stuff and things in scene understanding. In particular, we investigate how the recognition of stuff differs from things and develop methods that are suitable to deal with both. We use stuff to find things and annotate a large-scale dataset to study stuff and things in context.

First, we present two methods for semantic segmentation of stuff and things. Most methods require manual class weighting to counter imbalanced class frequency distributions, particularly on datasets with stuff and thing classes. We develop a novel joint calibration technique that takes into account class imbalance, class competition and overlapping regions by calibrating for the pixel-level evaluation criterion. The second method shows how to unify the advantages of region-based approaches (accurately delineated object boundaries) and fully convolutional approaches (end-to-end training). Both are combined in a universal framework that is equally suitable to deal with stuff and things.

Second, we propose to help weakly supervised object localization for classes where location annotations are not available, by transferring things and stuff knowledge from a source set with available annotations. This is particularly important if we want to scale scene understanding to real-world applications with thousands of classes, without having to exhaustively annotate millions of images.

Finally, we present COCO-Stuff – the largest existing dataset with dense stuff and thing annotations. Existing datasets are much smaller and were made with expensive polygon-based annotation. We use a very efficient stuff annotation protocol to densely annotate 164K images. Using this new dataset, we provide a detailed analysis of the dataset and visualize how stuff and things co-occur spatially in an image. We revisit the question whether stuff or things are easier to detect and which is more important based on visual and linguistic analysis.

Acknowledgements

First and foremost, I would like to thank my PhD supervisor Vittorio Ferrari. His vigor and strive for perfection are truly infectious and he taught me that a paper is a fully connected graph, where each letter depends on each other. I would also like to thank Jasper Uijlings. Jasper's calm and systematic approach taught me that productivity is about more than just hard work: the right tools, the right language and the right cultural mindset. I would like to express my sincere gratitude to my committee of examiners, Cristian Sminchisescu and Hakan Bilen. Furthermore, I am thankful to Bob Fisher and Frank Keller for valuable support and feedback during the yearly reviews.

It has been said that we stand on the shoulders of giants and as such, my work has also been heavily influenced by some of the great minds in the literature. In particular I would like to thank Joseph Tighe, Ross Girshick and Derek Hoiem for their invaluable contributions to the field. I would also like to thank the entire Common Visual Data Foundation (CVDF) - particularly Tsung-Yi Lin, Michael Maire and Piotr Dollár - for giving me the opportunity to work with them. This thesis has been made possible through financial contributions from the CVDF, Mighty AI, a European Research Council Starting Grant, the German government and my parents.

On a more personal note, I would like to thank my CALVIN family: Vittorio Ferrari, Jasper Uijlings, Luca del Pero, Davide Modolo, Vicky Kalogeiton, Dim Papadopoulos, Abel Gonzalez-Garcia, Michele Volpi, Paul Henderson, Miaojing Shi and Buyu Liu. The concepts of CAF and la penúltima have made a lasting impression on my life. Finally, I am very grateful to my family and friends, without whom none of this would have been possible. I am thankful for the time we have shared and looking forward to the next $\backslash$*chapter*.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Some material from this thesis is included in the following publications:

- Holger Caesar, Jasper Uijlings and Vittorio Ferrari, "Joint calibration for semantic segmentation", In *British Machine Vision Conference* (**BMVC**), 2015.

- Holger Caesar, Jasper Uijlings and Vittorio Ferrari, "Region-based semantic segmentation with end-to-end training". In *European Conference on Computer Vision* (**ECCV**), 2016.

- Miaojing Shi, Holger Caesar and Vittorio Ferrari, "Weakly Supervised Object Localization Using Things and Stuff Transfer". In *International Conference on Computer Vision* (**ICCV**), 2017.

- Holger Caesar, Jasper Uijlings and Vittorio Ferrari, "COCO-Stuff: Thing and Stuff Classes in Context". In *Conference on Computer Vision and Pattern Recognition* (**CVPR**), 2018.

While Miaojing Shi is the first author of the ICCV 2017 paper, I have made significant contributions to it. In the thesis we provide details of how the workload was divided.

(Holger Caesar)

To the Alps.

(and those that I share them with)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Evolution has endowed human beings with outstanding capabilities for perceiving and understanding their visual surroundings. These capabilities are seemingly trivial and effortless: We do not require conscious effort to see and understand our environment. Scene understanding as a part of computer vision attempts to equip computers with similar vision capabilities. Modeling and training a vision system turns out to be extremely challenging: Infinitely many 3D scenes can project to the same 2D image. And since each pixel in an image might depend on each other pixel, we face a combinatorial explosion that poses challenges given limited computational resources and training data. Fortunately we can use priors to constrain the number of likely scenarios. This idea is closely related to the concept of *unconscious inference* (von Helmholtz, 1867), that humans unconsciously base their visual understanding of incomplete data on assumptions made from prior experience.

As humans we tend to think of the world as a set of entities (Adelson, 2001; Feldman, 2003) (*person*, *car*, *chair*) with a position, orientation and attributes such as shape and color. These countable *things* often form a composition of parts and we know how to grasp and manipulate them. We rarely focus on the amorphous background materials (*ceiling*, *grass*, *water*) that lack a characteristic size and shape. These uncountable *stuff* regions have no parts and it is therefore not clear how to grasp and manipulate them. However, stuff matters[1]. We use the presence of *walls*, *floors* and *ceilings* to create a mental 3D representation of a room. We use texture and lighting cues to perceive scale and orientation of surfaces. Stuff classes like *road*, *path* and *water* are used for navigation by humans and robots alike. Other stuff classes like *wall* and *bush* may limit the navigable free space. Material properties like friction and viscosity are

[1] "What would childhood be without mud, snow, or peanut butter?" (Adelson, 2001)

important attributes in path planning that affect speed, energy consumption and safety. Optical, mechanical, and chemical descriptors are abundant in advertising and convincing materials are essential to create a realistic impression in computer graphics and art (Adelson, 2001). We also prime our understanding based on contextual cues: *Cows* are likely to be found on *grass*, *giraffes* are not likely to be found on *water*. One might even argue that our focus on things is an artefact of the industrialized world, as most things are man-made (Cheng et al., 2012). For early humans stuff classes may have been more important than things. Stuff and things are closely related to perceptual grouping and *Gestalt psychology* (Wertheimer, 1923; Kanizsa, 1979). In linguistics, a similar concept is the *mass and count* distinction (Cheng, 1973; Fieder et al., 2014).

In computer vision things have received a lot of attention: Early research in the 1960s-70s focused on primitive shape detection (Duda and Hart, 1972) and template matching (Vanderbrug and Rosenfeld, 1977). Such methods are typically unable to generalize to real-world images. In the 2000s, researchers achieved significant breakthroughs in face (Viola and Jones, 2001) and pedestrian detection (Dalal and Triggs, 2005). For the detection of these thing classes, the spatial composition of parts plays an important role. A few years later, researchers were able to generalize to datasets with realistic images and train object detectors for dozens of thing classes and benchmark their results against the community (Everingham et al., 2015). The advent of large-scale datasets and deep learning methods in the 2010s pushed the performance to previously unexpected levels for hundreds of classes (Krizhevsky et al., 2012; Russakovsky et al., 2015).

Meanwhile stuff classes have received less attention: Early work focused on texture classification (Brodatz, 1966). Recent works look at materials in real-world images (Bell et al., 2015), but without consideration for their semantics. The fairly young task of semantic segmentation (He et al., 2004; Long et al., 2015) fills this gap by classifying each pixel in an image using stuff *and* thing classes. Early semantic segmentation dataset had stuff and thing annotations (Shotton et al., 2006; Liu et al., 2011). Unfortunately this changed over time and currently the most popular semantic segmentation dataset (PASCAL VOC 2012 (Everingham et al., 2015), according to Appendix A) only covers thing classes. Therefore semantic segmentation is recently sometimes perceived as a minor refinement of object (thing) detection.

This observed gap between stuff and things in computer vision has a profound impact. We see multiple reasons for this gap: 1) Differences in human perception of stuff and things as described above. 2) The difficulty in defining, distinguishing and

delineating different stuff classes. 3) A lack of datasets and sophisticated models to learn stuff and things and their interactions. In this work we attempt to restore the balance between stuff and things in scene understanding. We present guidelines how to distinguish stuff and things and describe their characteristics. We devise methods for semantic segmentation of stuff and things. We show how stuff can be used to find things and vice versa. Finally, we create the largest existing dataset to study stuff and things in context.

1.1 Stuff and things

Before we present the problem statement and general approach of this thesis, we discuss the nature of stuff and things. The literature provides several aspects of stuff and things, including: (1) Shape: Things have characteristic shapes (*car*, *cat*, *phone*), whereas stuff is amorphous (*sky*, *grass*, *water*) (Forsyth et al., 1996; Xiao et al., 2010; Ion et al., 2011; Tighe and Lazebnik, 2013a; Uijlings et al., 2013; Mottaghi et al., 2013; Endres and Hoiem, 2010; Dai et al., 2015b). (2) Size: Things occur at characteristic sizes with little variance, whereas stuff regions are highly variable in size (Forsyth et al., 1996; Adelson, 2001; Heitz and Koller, 2008). (3) Parts: Thing classes have identifiable parts (Wang and Yuille, 2015; Felzenszwalb et al., 2010), whereas stuff classes do not (a *piece of grass* is still *grass*, but a *wheel* is not a *car*). (4) Instances: Stuff classes are typically not countable (Adelson, 2001) and have no clearly defined instances (Dai et al., 2015b; Hariharan et al., 2014; Tighe et al., 2014). (5) Texture: Stuff classes are typically highly textured (Forsyth et al., 1996; Heitz and Koller, 2008; Tighe and Lazebnik, 2013a; Dai et al., 2015b).

The above aspects of stuff and things form guidelines rather than an accurate definition. Feldman (2003) takes a similar approach and concludes that "although each of these properties contributes to the perception of [things], none is, in and of itself, essential". The same author even argues that things are not primarily an objective aspect of how the world is structured, but rather a product of "how our subjective perceptual interpretations are organized". Hence some classes can be interpreted as both stuff and things: A large number of *people* is sometimes considered a *crowd* (Hoiem et al., 2005a; Everingham et al., 2010; Hariharan et al., 2014). The foliage of a *tree* looks like stuff, but some people consider *tree* a thing, as it has a characteristic branch skeleton and various parts (e.g. *root*, *trunk*, *branches*, *leaves*). The class *table* has a characteristic size and shape and forms a composition of parts, which would make it a thing.

However, in some photographs *tables* are pictured as a surface from above, so that materials and texture become more important, similar to the stuff class *floor*. Ultimately the categorization of a semantic category as stuff or thing is a useful simplification to analyze certain characteristics of these semantic categories.

The term *object* is often used synonymously with thing (Alexe et al., 2010; Feldman, 2003; Dai et al., 2015b) (cf. objectness (Alexe et al., 2010), object detection (Girshick et al., 2014)). Other scholars use the term object as an instance of a semantic class, which may be a stuff or thing class (Liu et al., 2011; Sun et al., 2013; Brahmbhatt et al., 2017). In this work we use the latter and more general definition of object, except for established concepts (like object detection).

1.2 Problem statement and general approach

As pointed out above, stuff has received less attention than things due to actual and perceived differences between both. We identify several areas where stuff has achieved less attention. In particular, we investigate how the recognition of stuff differs from things and develop methods that are suitable to deal with both. We present methods to efficiently scale the learning of stuff and things and annotate a large-scale dataset to study stuff and things in context. We use stuff to find things and analyze how stuff and things co-occur spatially in an image. We revisit the question whether stuff or things are easier to detect and which is more important based on visual and linguistic analysis. Our goal is to rectify the imbalance and to restore the balance between stuff and things.

Scene understanding is a central field in computer vision that attempts to detect objects in a scene and reason about their spatial, functional and semantic relations. We use *semantic segmentation* as a representative scene understanding task to learn stuff and thing classes. Semantic segmentation is the task of classifying each pixel in an image into a set of predefined semantic classes. It is an extremely challenging task because the label of a pixel typically depends on many other pixels, and possibly the entire image. Multiple images of the same object can differ widely in appearance due to modifications in position, orientation, pose, scale, lighting, and texture. The appearance of an object may also differ significantly from another instance of the same class (intra-class variation). Another problem in semantic segmentation is *class imbalance* in the image or pixel frequency distribution. The learning algorithm needs to take care of class imbalance, e.g. by loss weighting or subsampling. Some datasets only have

sparse annotations for thing classes (Everingham et al., 2015; Lin et al., 2014). These classes are sampled to have a similar distribution, thereby limiting the class imbalance problem. Other datasets cover the entire image with dense stuff and thing annotations, but typically contain only a few canonical stuff classes and many rare thing classes (Liu et al., 2011; Mottaghi et al., 2014). In this context the problem of class imbalance is particularly relevant as rare thing classes would otherwise be ignored. Furthermore, stuff and things often occur in a contained-in relationship (*cow* standing on *grass*, *airplane* flying in the *sky*). Instead of looking at smaller patches, a region-based approach allows us to classify both regions in their entirety. With proper calibration, the overlap between container and contained region can then be resolved. In this thesis we present semantic segmentation methods that automatically take into account class imbalance, class competition and overlapping regions (Chapter 3).

Semantic segmentation methods can be roughly categorized in two types of approaches: Fully convolutional methods (Long et al., 2015) operate directly on the pixel-level. Region-based methods (Girshick et al., 2014) operate on regions. Fully convolutional methods are conceptually simple and efficient and allow for straightforward *end-to-end training*. Region-based methods produce crisp object boundaries. We try to combine the advantages of both by enabling to train a region-based method end-to-end. Traditionally, region-based approaches are particularly used for things, as they produce crisp object boundaries and enforce helpful priors on the object location. Using multi-scale regions, these approaches are able to capture a thing instance at its canonical scale. On the contrary, stuff has less clearly defined boundaries and no specific part configuration. It has neither instances nor a canonical scale and a piece of stuff is still stuff (see Sec. 1.1). Our approach allows for a unified treatment of stuff and things by capturing both at their most discriminative scale, which may correspond to the canonical scale for things. Furthermore, optimizing for instances or regions is suboptimal for semantic segmentation. To address this issue, we present a new technique that allows us to train directly for the pixel level, which is particularly helpful for stuff as it does not have instances. To summarize, we present a unified region-based approach for both stuff and thing classes that enables end-to-end training (Chapter 4).

Most modern methods for semantic segmentation require a lot of data for training. Annotating a single image with pixel-level labels can take up to an hour (Cordts et al., 2016). Therefore we need new paradigms both for annotating datasets and learning more efficiently. Instead of carefully annotating each object by drawing a tightly enclosing polygon around it, we can use weaker levels of supervision, such as image-

level labels (*weakly supervised learning*). These types of annotations are cheaper to annotate and we can retrieve huge datasets from social media sites where users provide tags and captions. When annotations are scarce, we can furthermore use *domain transfer* from one task to another. This is particularly interesting in the context of stuff and things. Existing datasets typically have only a few canonical stuff classes that cover the majority of the surface and many rare thing classes that cover only small portions of the image. Hence we suggest to focus on the frequent stuff classes and transfer their knowledge across tasks and datasets. Using contextual relations, stuff can then be used to find thing classes (Heitz and Koller, 2008). This can also be seen in a context of curriculum learning (Bengio et al., 2009) and lifelong learning (Thrun, 1996), where we introduce "different concepts at different times, exploiting previously learned concepts to ease the learning of new abstractions" (Bengio et al., 2009). In this thesis we present multiple techniques for efficient annotation and learning under different levels of supervision (Chapter 3, 5 and 6).

To be able to study stuff and things in context, we need a large scale dataset with a suitable type of annotation. Existing datasets are too small in terms of number of classes and number of images (Shotton et al., 2006; Brostow et al., 2009; Liu et al., 2011; Silberman et al., 2012) or have an unfavorable frequency distribution with a few canonical stuff classes and many rare thing classes (Liu et al., 2011; Mottaghi et al., 2014; Zhou et al., 2017b). As part of this thesis we present a new large-scale dataset for stuff and things, called COCO-Stuff. The efficient annotation protocol exploits the characteristics of stuff – particularly the difficulty in accurately outlining stuff regions like trees and bushes. Furthermore it makes use of existing very accurate thing annotations. Contrary to most existing datasets with stuff and thing classes, COCO-Stuff has a similar granularity and frequency distribution for both stuff and things. This alleviates the previously mentioned problem of class imbalance between stuff and things. Once we have such a dataset, we can analyze the spatial relations between stuff and things, their relative importance and their role in semantic segmentation (Chapter 6). This will bring us closer to our ultimate goal of restoring the balance of stuff and things in scene understanding.

1.3 Thesis plan

This section gives a plan of the thesis and summarizes its contributions:

- We describe the motivation for this thesis, a definition of the problems we address, as well as the characteristics of stuff and things (Chapter 1).
- We give an overview of semantic segmentation and related tasks, deep learning, evaluation criteria as well as problems and approaches in semantic segmentation with respect to stuff and things. We also discuss how to efficiently scale up existing annotation and learning approaches and give an overview of how stuff and things were previously used in the literature (Chapter 2).
- We present two semantic segmentation methods which outperformed the state-of-the-art in class-accuracy at the time of publication on the SIFT Flow dataset (Liu et al., 2011). Both methods are developed with a focus on class imbalance, competition between classes, overlapping regions and training for the pixel-level evaluation criterion. We provide an in-depth analysis of the strengths and weaknesses of these methods. To promote further research, the code for these and other methods has been published at `https://github.com/nightrome/matconvnet-calvin` (Chapter 3 and 4).
- We show how to transfer stuff and things across domains and improve results in weakly supervised object localization on classes where location annotations are not available (Chapter 5).
- We build the largest existing dataset of stuff and things and use it to analyze stuff and things in context. We also co-organized the COCO Stuff Segmentation and COCO Panoptic Segmentation challenges and workshops in 2017 and 2018 to help promote the COCO-Stuff dataset and the mission of this thesis to a broader community (Chapter 6).
- We present additional unpublished works (Chapter 7), as well as ideas for future work (Chapter 8). We conclude this thesis by summarizing our findings (Chapter 9). As an appendix to the thesis, we present statistics of the largest existing collection of semantic segmentation papers (Appendix A).

Chapter 2

Background

This chapter presents the background knowledge and related work of this thesis. Related work specific to only certain chapters can be found in those chapters. We give a broad overview over semantic segmentation and deep learning and review efficient annotation and learning schemes. Furthermore we present datasets and methods for stuff and things and describe how they are used in previous works.

2.1 Semantic segmentation

Semantic segmentation is the task of classifying each pixel in an image into a set of predefined semantic classes, which can be stuff or things. An alternative view is that semantic segmentation consists of two subtasks: localization and classification. In the localization subtask we segment the semantic regions of an image. In the classification subtask we assign a label to each segmented region. The two alternative views are particularly linked to the different types of approaches: fully convolutional and region-based (Sec. 2.1.3). Semantic segmentation is a supervised learning task that typically requires pixel-level class labels at training time. Synonyms for semantic segmentation include semantic image segmentation, semantic (image) labeling, semantic object parsing, scene parsing, scene labeling, scene segmentation and scene understanding (although it generally has a broader meaning). In Appendix A we show that the term semantic segmentation is the most frequently used of these synonyms.

Semantic segmentation is closely related to other tasks that combine localization and classification. Object detection is the task of localizing each thing in an image and drawing a tightly enclosing bounding box around it. Instance segmentation combines aspects of semantic segmentation and object detection: The goal is to predict a

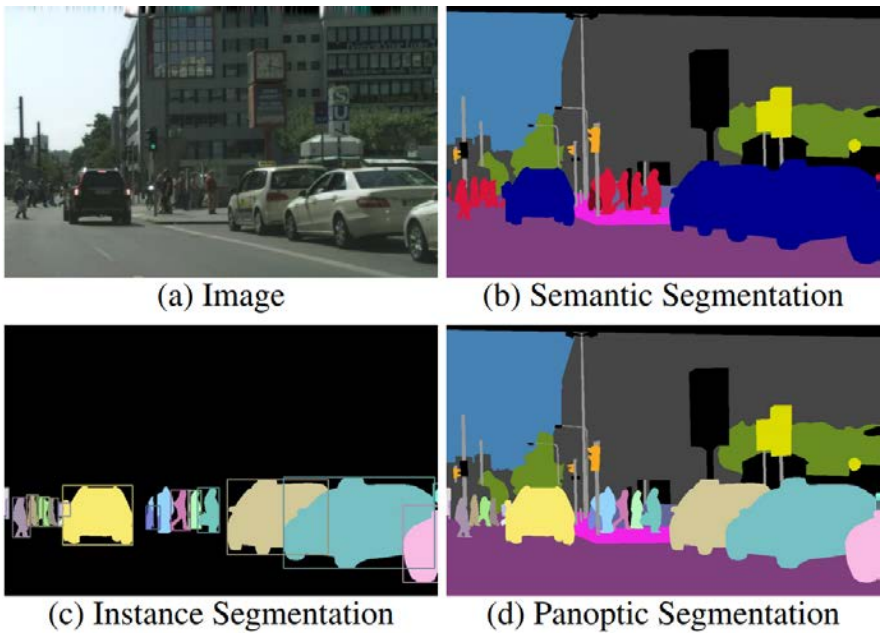


Figure 2.1: Comparison of the different segmentation tasks. Image taken from Kirillov et al. (2018).

pixel-level mask for each thing instance. This task was first made popular under the name *Simultaneous Detection and Segmentation* (Hariharan et al., 2014). Semantic segmentation may cover stuff and thing classes, whereas object detection and instance segmentation only cover things. A recent task called panoptic segmentation (Kirillov et al., 2018) unites semantic segmentation and instance segmentation. Each pixel is assigned both a semantic label and an instance id. For stuff classes, which do not have instances (see Sec. 1.1), the instance id is set to a dummy value. This holistic stuff and thing task is aligned with the goal of this thesis – to restore the balance between stuff and things. Fig. 2.1 shows an image and the desired outputs for the semantic segmentation, instance segmentation and panoptic segmentation tasks.

2.1.1 Evaluation criteria

In Table 2.1 we present the evaluation criteria commonly used in semantic segmentation. Early works use pixel accuracy to measure the percentage of correctly classified pixels in a dataset (He et al., 2004; Gould et al., 2009). Synonyms for pixel accuracy are global accuracy and (per-pixel) classification rate. This metric is simple and intuitive and can be directly trained for using a standard cross-entropy log-loss. However it is not ideal for datasets with class imbalance. The SIFT Flow (Liu et al., 2011) dataset has 33 classes, but the class *sky* covers 26.7% of the pixels. This means that a

dummy classifier that always outputs *sky* will achieve a pixel accuracy of 26.7%, while ignoring 97.0% of the classes.

Later works use class accuracy – the average of the per-class pixel accuracies – as an additional metric (Ladicky et al., 2009; Tighe and Lazebnik, 2010; Farabet et al., 2013). Synonyms for class accuracy are mean accuracy, class-average pixel accuracy, average per-class accuracy and average per-class rate. By giving the same importance to each class, this metric penalizes ignoring classes. Therefore a dummy classifier tht outputs only one label can only achieve a class accuracy of $^1/_C$ for a dataset with C classes, e.g. 3.0% on SIFT Flow. A neural network can be trained to optimize for class accuracy by using an inverse class frequency-weighted cross-entropy log-loss. In practice class accuracy tends to saturate less quickly than pixel accuracy, making it easier to compare different methods, e.g. on SIFT Flow the state-of-the-art has advanced by +78% relative on class accuracy and +12% relative on pixel accuracy from 2010 (Tighe and Lazebnik, 2010) to 2015 (Long et al., 2015). One problem of the class accuracy metric is that it does not directly take into account false negatives. This means that a classifier will assign disproportionately high scores to rare classes to increase their true positive rate, while only very slightly decreasing the true positive rate of frequent classes. We call this behavior "overshooting". As an example, on SIFT Flow it is beneficial to predict entire *sky* regions as *bird*, as *bird* is rare and small in size, but likely to be surrounded by *sky*.

Everingham et al. (2015) suggest to use the Intersection-over-Union (IOU) criterion for semantic segmentation, also known as the Jaccard index. Compared to the IOU criterion used in object detection, in semantic segmentation IOU is computed over all pixels belonging to a certain class in an image instead of boxes. Furthermore they ignore object instances by treating all pixels of a class in an image as a single mask. Hence this measure is useful for semantic segmentation of stuff and things, as stuff does not have instances (see Sec. 1.1). The formula for mean IOU in Table 2.1 shows that – compared to pixel accuracy and class accuracy – it includes false negatives, thereby explicitly penalizing the overshooting behavior described above. Optimizing directly for mean IOU is hard as it is a non-decomposable performance measure (Ranjbar, 2013), which means that it cannot be decomposed into a sum of per-pixel measurements. However several authors have attempted to approximate the mean IOU measure: Cogswell et al. (2014); Rahman and Wang (2016) use probabilistic approximations of IOU. Instead of using the predicted and ground-truth labels, they compute the intersection and union of the pixel probabilities for the respective

classes. This loss function can be optimized using standard Stochastic Gradient Descent (SGD). Other works use Conditional Random Fields (CRF) (Nowozin, 2014) or Markov Random Fields (MRF) (Ranjbar, 2013) to optimize IOU. For a more detailed overview we refer the reader to Rahman and Wang (2016). Some authors regard the frequency of a class in a dataset as a measure of its importance (e.g. *person* is frequent in PASCAL VOC (Everingham et al., 2015) and COCO (Lin et al., 2014)). Frequency weighted (FW) IOU (Long et al., 2015) takes this into account by weighting the per-class IOUs by their frequency.

Cordts et al. (2016) point out that the (global) mean IOU measure is "biased toward object instances that cover a large image area". They therefore also use an instance-based IOU metric, which is "weighted by the ratio of the class' average instance size to the size of the respective ground-truth instance". This is justified as they focus particularly on individual traffic participants in an autonomous driving scenario (e.g. *car*, *person*). However the metric requires instance annotations and therefore excludes stuff, which does not have instances. Other scholars have argued that a suitable evaluation metric should be normalized per image. This takes into account images with highly varying numbers of instances (e.g. in ADE20K there are up to 279 instances per image (Zhou et al., 2017b)). It also gives the same importance to images with varying image resolution. Furthermore plotting the histogram of a per-image metric allows us to understand whether one method has advantages over another method on all images or only on a specific subset (Csurka et al., 2013). Other types of evaluation metrics focus on the performance on instance or region boundaries. Kohli et al. (2009) present an evaluation metric on trimaps, which are the band-like regions surrounding an object boundary. They compute the metric for different widths. To collapse the boundary measures for different widths into a single value, Csurka et al. (2013) propose an F1-measure on binarized segmentation maps with a given distance error tolerance to decide "whether a boundary point has a match or not". While contour-based evaluation criteria are useful for things, they are less applicable for stuff. In fact, due to the difficulty of accurately delineating a stuff region (e.g. *bush*), the boundary pixels may be the least reliable in the ground-truth annotation.

To summarize, we presented and discussed the evaluation criteria used in semantic segmentation. The most suitable evaluation criterion is highly dependent on the application and no single criterion is able to satisfy all requirements. In Sec. 3.6 and Chapter 8 we will discuss other options for suitable evaluation criteria.

Metric	Formula
Pixel accuracy	$\frac{\sum_c TP_c}{\sum_c TP_c + FP_c}$
Class accuracy	$1/C \sum_c \frac{TP_c}{TP_c + FP_c}$
Mean IOU	$1/C \sum_c \frac{TP_c}{TP_c + FP_c + FN_c}$
FW IOU	$\sum_c \frac{P_c}{P} \frac{TP_c}{TP_c + FP_c + FN_c}$

Table 2.1: *Semantic segmentation evaluation criteria. TP_c indicates true positives, FP_c false positives, P_c all positives and FN_c false negatives of a class $c \in [1 \dots C]$. P indicates the total number of pixels.*

2.1.2 Popular deep learning architectures in computer vision

The term *deep learning* describes a family of machine learning methods that involve learning data representations. Deep learning has had an immense impact on semantic segmentation and as such we describe the fundamentals here, before presenting leading methods for semantic segmentation in Sec. 2.1.3. Deep learning replaces the traditional two-step pipeline of feature extraction and classification (or regression) with a single step. The depth refers to the number of trainable layers chained together in a neural network. Deep learning achieved its breakthrough in computer vision in the early 2010s due to several enabling factors:

- Convolutional Neural Networks (CNN) (LeCun et al., 1990) and a collection of technical elements that reduce overfitting and allow to train particularly deep networks: rectified linear units (ReLU) (Nair and Hinton, 2010), dropout (Hinton et al., 2012), batch normalization (Ioffe and Szegedy, 2015) and residual connections (He et al., 2016).

- Dedicated hardware for highly parallelizable operations such as Graphics Processing Units (GPU) and Tensor Processing Units (TPU). Application Programming Interfaces such as CUDA (Nickolls et al., 2008) and OpenCL (Stone et al., 2010) allow the user to directly interface with this dedicated hardware.

- Public availability of large-scale datasets with semantic labels for hundreds of classes and millions of images (e.g. ImageNet by Russakovsky et al. (2015)).

Many works have shown that *pretraining* for image classification on ImageNet results in a network whose internal feature representation is able to generalize to other tasks and datasets (Razavian et al., 2014; Girshick et al., 2014; Donahue et al., 2014; Long et al., 2015). A common procedure is then to use the learned network weights as an initialization of the lower layers and then finetune the network for the target task. Saito et al. (2017) show that pretraining on ImageNet even helps for the stuff class *road*, despite ImageNet containing mostly things. This ability to generalize and transfer knowledge brings us closer to curriculum learning (Bengio et al., 2009) and lifelong learning (Thrun, 1996).

Network architectures. Deep neural networks are often described as black boxes due to their high complexity with millions of parameters (Girshick et al., 2015; Chen et al., 2016a). However certain design choices have proven successful for certain tasks and requirements (such as low memory footprint and energy consumption). Generally speaking a backbone architecture that achieves better results in image classification is also very likely to achieve better results in semantic segmentation. Here we present the most popular backbone architectures used in the literature:

- LeNet by LeCun et al. (1990) is considered the first CNN architecture and was used for digit classification. It uses normalized images of size 16×16 as inputs and outputs 10 values, one per class. The network has 5 layers, which typically consist of a convolutional filter, a squashing function and a non-overlapping local averaging and subsampling operator.

- AlexNet by Krizhevsky et al. (2012) is considered the first deep network architecture to be successful at general image classification. It won the ILSVRC 2012 image classification challenge (Russakovsky et al., 2015). Fig. 2.2 shows an overview of the architecture. It takes an RGB image of size 224×224 as input and outputs 1000 values, one per class. The network has 60 million parameters and is therefore vastly bigger than previous networks. It consists of 5 convolutional layers and 3 fully connected layers. The final outputs are produced via a 1000-way softmax, which produces a distribution over the class labels. Some convolutional layers are followed by overlapping max pooling layers. The network uses ReLUs (Nair and Hinton, 2010) to speedup training and Local Response Normalization to improve generalization. It also makes use of dropout (Hinton et al., 2012) to randomly drop activations, which serves as a model averaging step and reduces the test error.

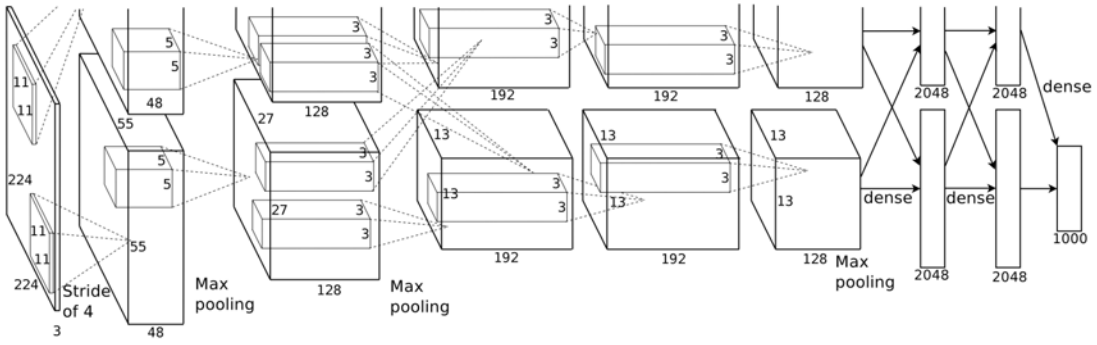


Figure 2.2: AlexNet deep learning network architecture. Image taken from Krizhevsky et al. (2012).

- VGG by Simonyan and Zisserman (2015) refers to several network architectures with 11 to 19 layers, the most popular being VGG-16 with 138 million parameters. It won the first and second place in the ILSVRC 2014 localization and classification challenges (Russakovsky et al., 2015). VGG is based on AlexNet, but increases the depth and replaces all convolutions (up to size 11×11) with 3×3 convolutions. It thus shows that even very small filters can achieve expressiveness, if they are stacked sufficiently deep. This reduces the number of parameters and allows for deeper networks overall. VGG also does not make use of Local Response Normalization, as for their network it results in more parameters without any improvements in performance.

- GoogLeNet by Szegedy et al. (2015) is a 22-layer incarnation of the Inception architecture. It won the ILSVRC 2014 classification and detection challenges (Russakovsky et al., 2015). The idea is to increase the width and depth of a network, while keeping the computational budget constant. It is based on the Hebbian principle and the idea of multi-scale processing. The network is a stacked cascade of identical building blocks – so called Inception modules. Fig. 2.3 shows the layout of one Inception module. Each Inception module concatenates the output of a 1×1, a 3×3 and a 5×5 convolution block and a 3×3 max pooling layer. To avoid a steady increase in the feature size, 1×1 convolutions are used before each convolutional block to reduce the feature dimensionality. Occasional max pooling layers are further used to downsample the feature maps. This results in a state-of-the art network with 12x fewer parameters than AlexNet.

- Residual Network (ResNet) by He et al. (2016) addresses the degradation prob-

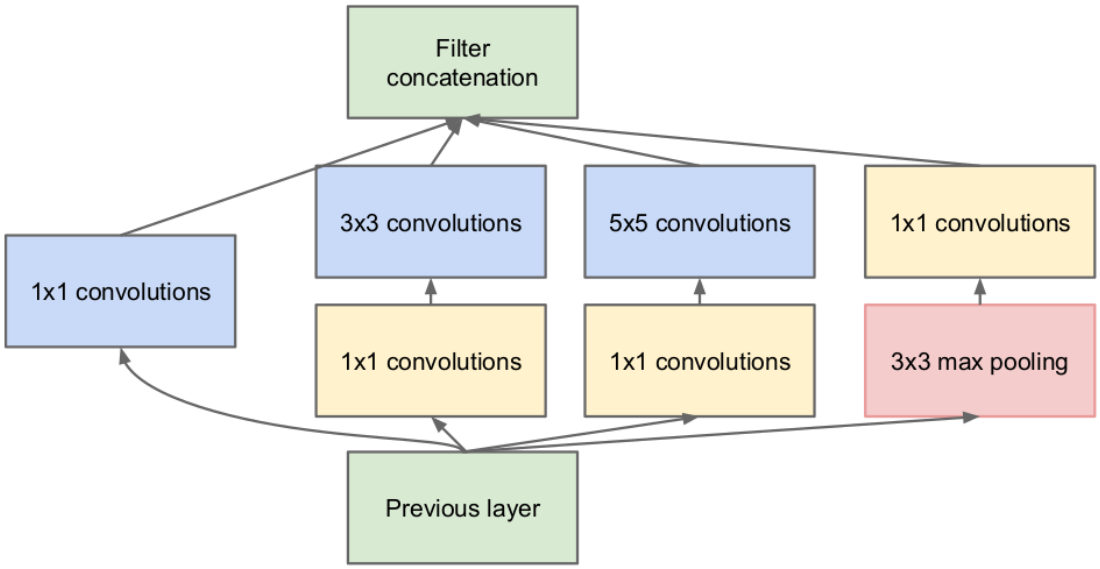


Figure 2.3: Inception block used in GoogLeNet. Image taken from Szegedy et al. (2015).

lem, that "with the network depth increasing, accuracy gets saturated [...] and then degrades rapidly." It won the first place in the ILSVRC 2015 classification, detection and localization challenges (Russakovsky et al., 2015). The idea is that for each layer (i.e. a sequence of convolution, ReLU and another convolution), they learn the residual function with reference to the layer input. This achieves much deeper networks of 101 and 152 layers, while having a lower complexity. Fig. 2.4 compares the VGG-19 architecture to a 34 layer architecture with and without residual connections. For image classification on the 32×32 images of CIFAR-10 (Krizhevsky and Hinton, 2009) they even present a network with 1000 layers.

2.1.3 Semantic segmentation approaches

Modern semantic segmentation methods can be categorized in two types of approaches: *Fully convolutional methods* (Long et al., 2015) operate directly on the pixel or (rectangular) patch level. *Region-based methods* (Girshick et al., 2014) operate on regions. These regions are designed to cover whole objects and are often created before the semantic segmentation model operates. Some works combine both, e.g. by pooling fully convolutional features over regions in a post-processing step (Farabet et al., 2013). Earlier works in semantic segmentation often use image retrieval techniques instead of directly learning a classifier (Tighe and Lazebnik, 2010, 2013a; George, 2015). How-

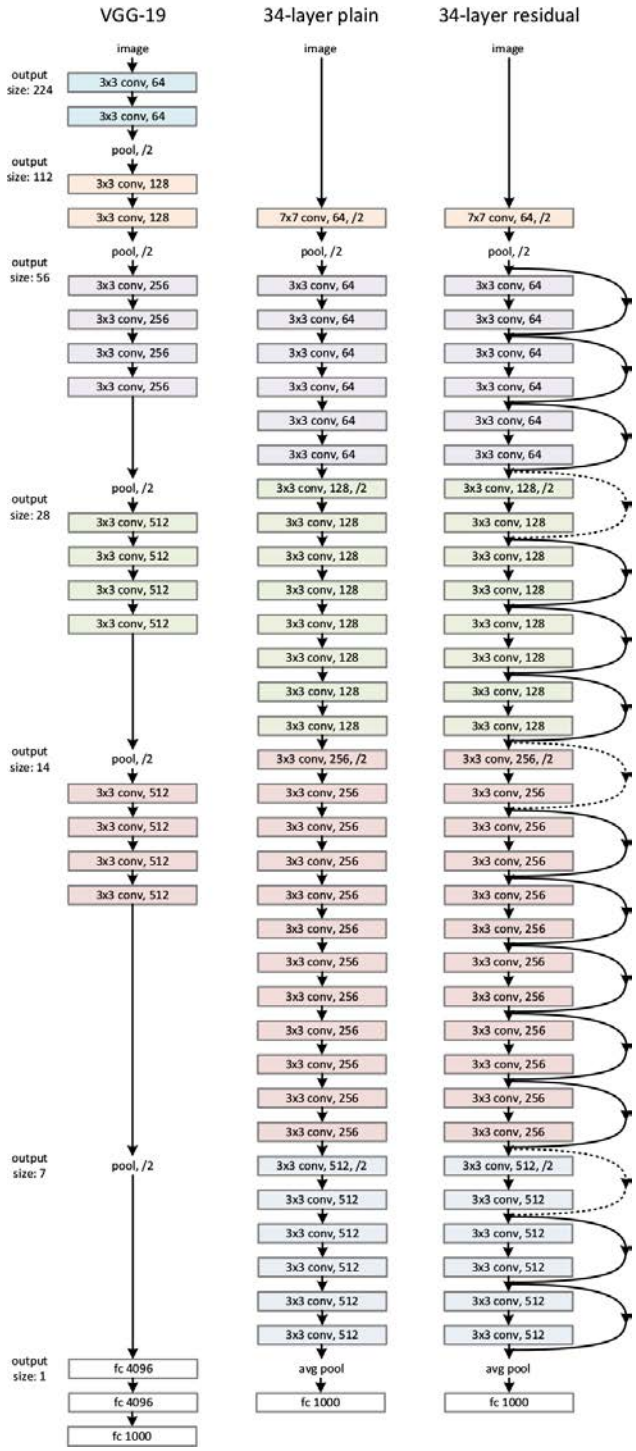


Figure 2.4: Comparison of the VGG-19 architecture to a 34 layer architecture with and without residual connections. Image taken from He et al. (2016).

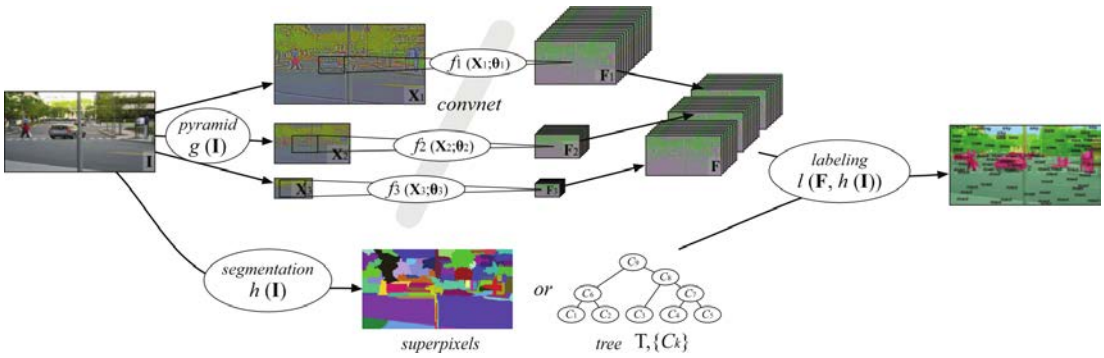


Figure 2.5: Overview of the semantic segmentation architecture of Farabet et al. Image taken from Farabet et al. (2013).

ever the unit of retrieval is still a patch or region.

2.1.3.1 Fully convolutional approaches

For the fully convolutional approach the idea is to directly learn a mapping from image pixels to class labels. The output is an image which assigns a class label to each pixel in the input image. This results in a single model, usually directly optimized end-to-end for the task at hand.

The seminal work of Farabet et al. (2013) foresaw many of the later trends in semantic segmentation, as shown in Fig. 2.5. They present a fully convolutional network that is slid over an image at multiple scales of a Laplacian pyramid. It uses various post-processing techniques such as superpixel pooling, a segmentation tree over superpixels and a Conditional Random Field (CRF). This is the first work that successfully applies CNNs to semantic segmentation, instead of using hand-crafted features. However, it is still missing many components of modern works: The three-stage CNN used by the authors is very shallow and it does not use most of the bells and whistles of modern architectures (see Sec. 2.1.2), such as ReLU and dropout. The network is not pretrained for image classification, which is usually a good initialization of the network. The patchwise training procedure is arguably less efficient due to redundant computations. Furthermore, while the CNN is trained end-to-end, the post-processing steps are not.

As the first work of its kind, Long et al. (2015) define the Fully Convolutional Network (FCN), a network where all layers only depend on *relative* spatial coordinates:

$$y_{ij} = f_{ks}\left(\{x_{s+\delta i, sj+\delta j}\}_{0\leq\delta i,\delta j<k}\right), \tag{2.1}$$

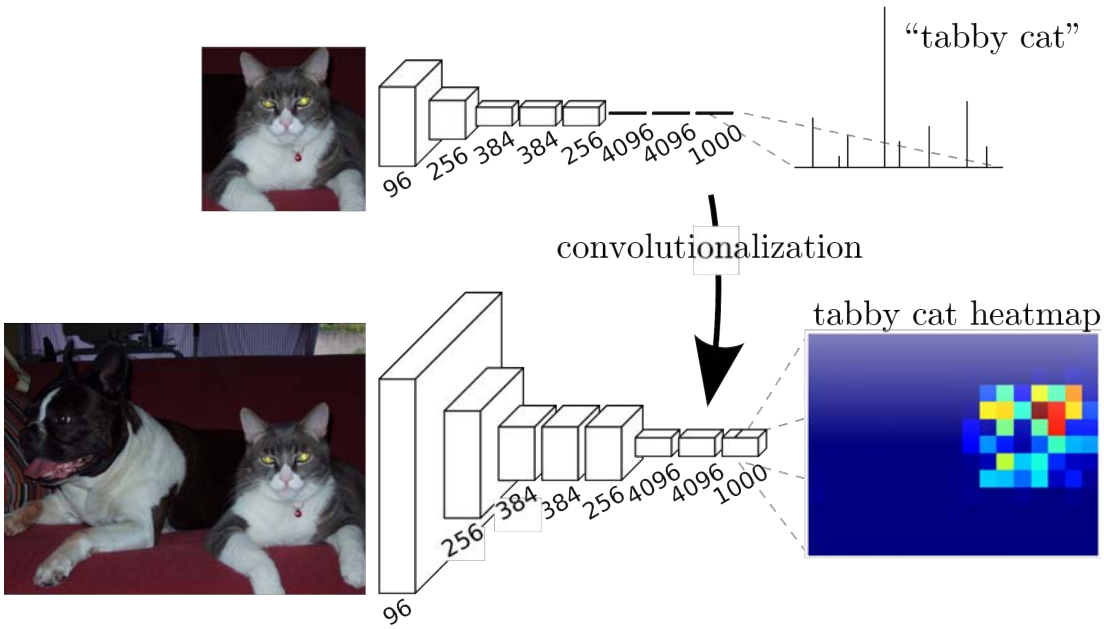


Figure 2.6: Replacing fully connected layers with 1×1 convolutional layers as done in the Fully Convolutional Network (Long et al., 2015) architecture. Image taken from Long et al. (2015).

where y_{ij} is the layer output, f_{ks} is the function that determines the layer type (convolution, pooling, activation function, etc.), s is the stride and x_{ij} determines the data vector at location (i, j). As visualized in Fig. 2.6, the authors give a recipe how to convert a classification network into a network for dense prediction, by replacing the fully connected layers with 1×1 convolutions, which satisfy Eq. 2.1. This is much more efficient and it allows the network to process arbitrary input sizes, contrary to most classification networks which require a fixed input size (see Sec. 2.1.2). Reusing state-of-the-art networks also allows them to profit from pretraining for image classification. After converting the network, the authors present a series of improvements for semantic segmentation. The network outputs are upsampled to the input size using a deconvolution layer. The deconvolution layer is initialized to bilinear interpolation, but refined during training of the network. They also use skip connections to fuse outputs of deep and shallow layers with different strides. This lets the network "make local predictions that respect global structure". As such the skip connections are a learned nonlinear alternative to the multi-scale processing in Laplacian pyramids.

Deeplab by Chen et al. (2015a) is based on FCN, but uses atrous convolutions to avoid the reduction in spatial resolution incurred by max-pooling. Atrous convolutions (Holschneider et al., 1990; Sermanet et al., 2014) (also known as dilated convo-

lutions (Yu and Koltun, 2016)) are a generalization of regular convolutions. The *rate* parameter r defines the upsampling factor of a convolutional filter, with zeros between filter values. For $r = 1$ we have regular convolutions. Setting $r \neq 1$ allows us to control the spatial resolution of the feature maps. Fig. 2.7 shows an example of a standard convolution with downsampling and upsampling (top) and an atrous convolution with $r = 2$ (bottom). While standard convolutions perform sparse feature extraction on a low resolution feature map, atrous convolutions perform dense feature extraction on a high resolution feature map. Although the effective filter size increases, atrous convolutions have the same number of parameters as regular convolutions. In practice the authors use a hybrid approach that combines atrous convolutions and fixed bilinear interpolation to achieve a good efficiency/accuracy trade-off. Furthermore, to make the network invariant to scale, the authors present a technique called Atrous Spatial Pyramid Pooling (ASPP). ASPP avoids the computational overhead of rerunning the entire network for each scale, by computing atrous convolutions with different rates and fusing the score maps by taking the maximum response across scales. Using the fully connected CRF by (Krähenbühl and Koltun, 2011), they are able to recover the detailed local structure of the segmentation. While Deeplab applies the CRF as a post-processing step, recent works train the CNN and CRF jointly and end-to-end (Lin et al., 2016b; Zheng et al., 2015; Liu et al., 2016; Chen et al., 2015b; Schwing and Urtasun, 2015). In particular, Zheng et al. (2015); Schwing and Urtasun (2015) propose unrolling the CRF mean-field iterations to enable end-to-end training.

Many semantic segmentation works can be described as encoder-decoder networks (Long et al., 2015; Noh et al., 2015; Ronneberger et al., 2015; Hong et al., 2015, 2016; Paszke et al., 2017; Badrinarayanan et al., 2017). Encoder-decoder networks are related to the information bottleneck method in machine learning (Tishby et al., 2000). Fig. 2.8 shows the U-Net (Ronneberger et al., 2015) architecture as an example of an encoder-decoder architecture. The encoder progressively reduces image resolution at each layer. By doing so it learns to abstract from the image and ignore noise, keeping only features that preserve the maximum relevant information about the pixel labels. The decoder learns to upsample and localize the segmentation. As pointed out in relation to FCNs, this view is particularly appealing in the context of pretraining. The encoder weights can be copied from the pretrained network. The decoder weights can be initialized to perform bilinear upsampling. Many architectures have symmetric encoder and decoder networks (Noh et al., 2015; Hong et al., 2015; Badrinarayanan et al., 2017). In Badrinarayanan et al. (2017) the decoder uses pooling indices computed in

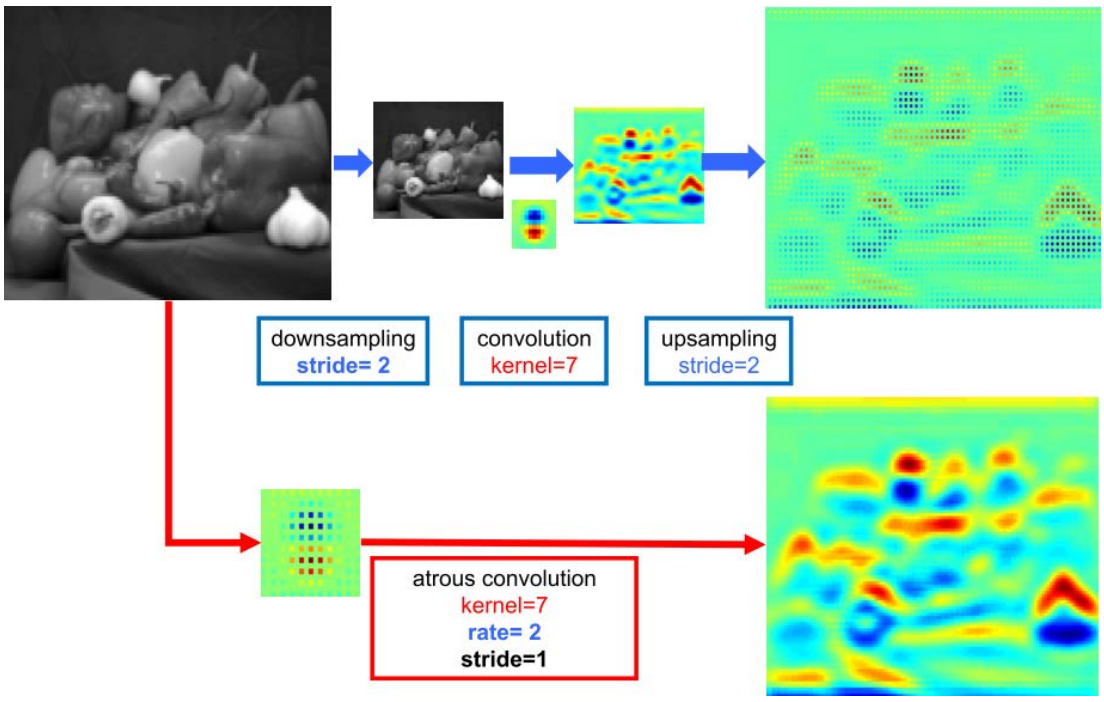


Figure 2.7: Comparison of standard convolution with downsizing and atrous convolution. Image taken from Chen et al. (2015a).

the max-pooling steps of the encoder to route the semantic information to its accurate location in the image. Hong et al. (2015) take this one step further and completely decouple the classification (encoder) and segmentation (decoder) networks. This allows them to perform semi-supervised learning as the segmentation network only requires very few annotated images for training. However, we would like to stress that this encoder-decoder view has its limits. When all layers of the network are finetuned, there is no guarantee of this separation of labor between encoder and decoder and both parts may be involved in classification and segmentation.

Fully convolutional methods are also related to the sliding-window concept in object detection (Heitz and Koller, 2008; Chen et al., 2015a; Gadde et al., 2016).

2.1.3.2 Region-based approaches

Region-based approaches classify image regions: In a preprocessing step, a large number of region proposals are generated per image. These region proposals are designed to tightly cover all objects in the image. For each region proposal local features are extracted. These region features are classified. Each pixel is assigned the label with the highest score of all regions that include the pixel. Region-based methods enforce strong location priors on a classifier. Contrary to the dense sliding-window paradigm

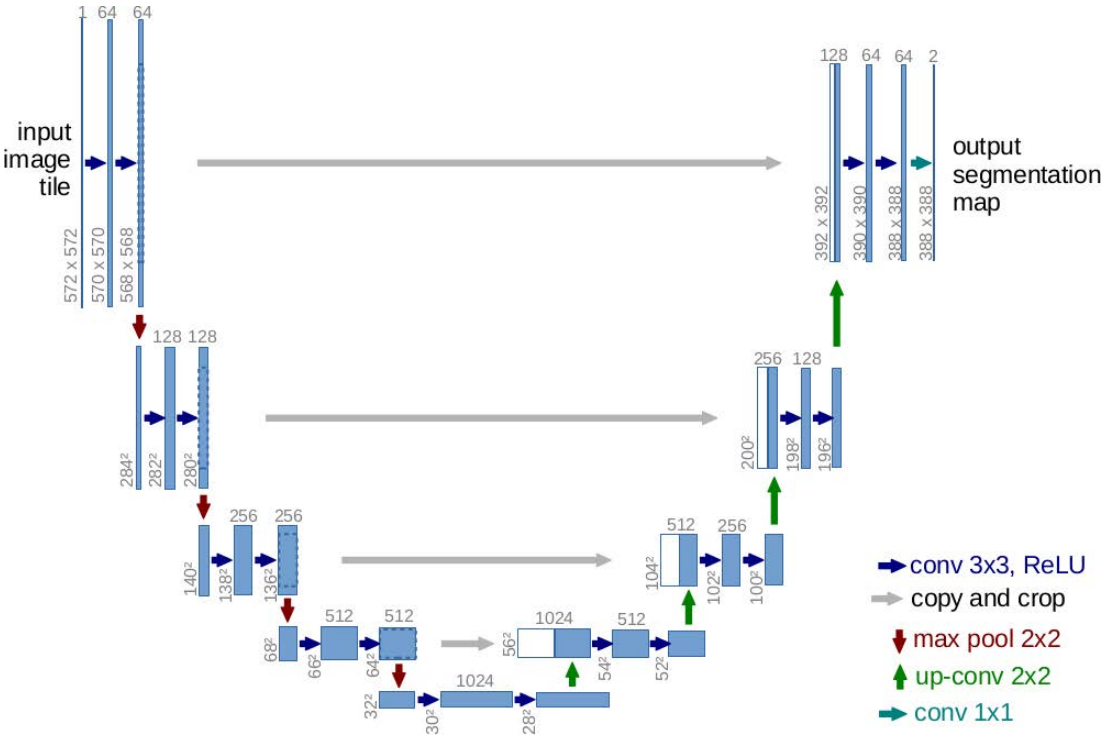


Figure 2.8: The U-Net architecture – an example of an encoder-decoder architecture. Image taken from Ronneberger et al. (2015).

of fully convolutional approaches, region-based approaches sparsely sample the possible image locations. Therefore methods that use dense rectangular patches are not considered region-based.

Early region proposal methods create non-overlapping oversegmentations (so-called *superpixels*) of an image, e.g. by treating image segmentation as a graph partitioning problem – the so-called normalized cuts (Shi and Malik, 2000). However, the resulting superpixels are non-overlapping and all have the same scale. Typically each superpixel does not cover the whole object, but rather only a piece of it. Alexe et al. (2010) present a generic objectness measure that quantifies how likely it is for a rectangular region to cover *any* thing class. They show that, for most objects, there is at least one sampled box that covers it wholly. They also sample bounding boxes from an image according to the objectness distribution. While some methods only produce bounding boxes (Alexe et al., 2010; Rahtu et al., 2011; Manen et al., 2013; Zitnick and Dollár, 2014; Cheng et al., 2014), *free-form* region proposal methods produce masks that can more precisely characterize the spatial extent of an object (Carreira and Sminchisescu, 2010; Endres and Hoiem, 2010; Uijlings et al., 2013; Arbeláez et al., 2014; Krähenbühl and Koltun, 2014; Rantalankila et al., 2014). If required, bounding boxes

can then be computed that tightly surround these free-form proposals. The first work in this line, Carreira and Sminchisescu (2010), provides a framework to generate figure-ground object hypotheses, by solving a sequence of constrained parametric min-cut problems (CPMC) on the pixels. The segments are then ranked according to graph, region and Gestalt properties. The use of Gestalt properties like continuity and convexity can be seen as an extension of the objectness (Alexe et al., 2010) measure from bounding boxes to free-form regions. While presented primarily for things, the authors of CPMC stress that their method is general and can rank stuff and things. Endres and Hoiem (2010) propose a related approach, but employ a learned affinity measure between superpixels instead of pixels. They use a structured learning approach on a maximum marginal relevance measure. As a region representation they combine several cues related to object and occlusion boundaries. They furthermore learn a simple pixel-level model of the "stuff-like" background and use geometry classes (*vertical*, *porous*, *solid* and *sky*), which are noted to "often correspond to object and background classes". Then they encode the differences in color and texture between object and local background or global background. The resulting region proposals are only suitable for things. Fig. 2.9 shows the Selective Search algorithm by Uijlings et al. (2013), a hierarchical bottom-up grouping of an initial oversegmentation that is particularly diverse and captures all scales.

The next step is to extract features for each region. In the pre-CNN era features were typically handcrafted (Ojala et al., 2002; Lowe, 2004; Shotton et al., 2009; Carreira et al., 2012). Local Binary Patterns (Ojala et al., 2002) are a multiresolution and rotation invariant texture classification filter that compares a pixel value to its neighbors and computes a normalized histogram. Scale Invariant Feature Transform (SIFT) (Lowe, 2004) is a keypoint descriptor that is invariant to scale, translation, rotation and (partially) to illumination changes. SIFT computes a normalized orientation histogram of the gradients of a local neighborhood of a keypoint. To use SIFT in semantic segmentation, keypoint locations can be sampled densely across the image region. Carreira et al. (2012) present a second-order pooling of SIFT and LBP features over CPMC region proposals, by computing the average or maximum of the outer products of the feature descriptors. After feature enrichment and descriptor normalization, they classify the descriptors using a Support Vector Machine (Cortes and Vapnik, 1995). Later Ionescu et al. (2015) showed how to peform matrix backpropagation in a CNN, which enables global structured matrix computation layers, such as second-order pooling and normalized cuts.

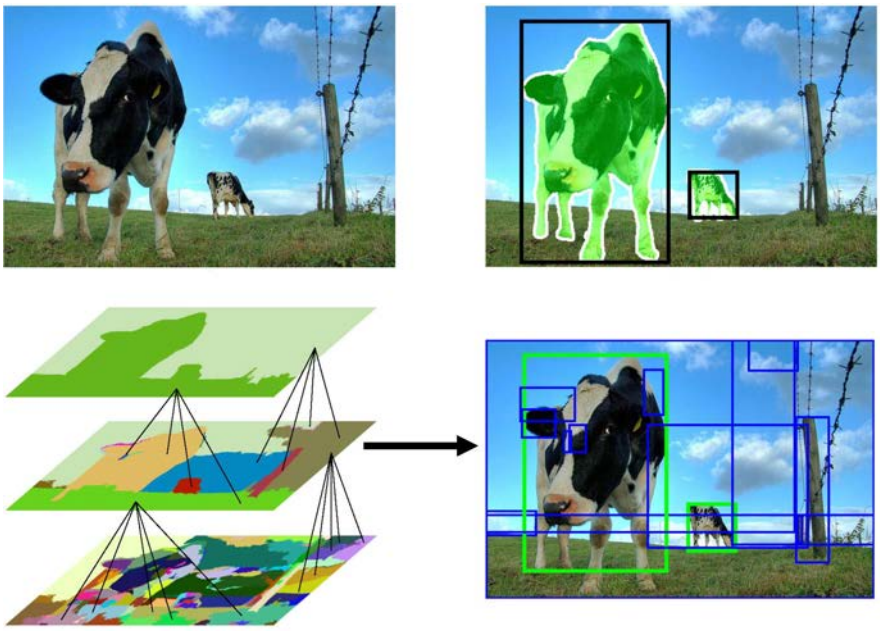


Figure 2.9: Selective Search bottom-up grouping of superpixels. We can see an image (top left), the ground-truth boxes (top right), the oversegmentation at multiple levels of the hierarchy (bottom left) and the final region proposals (bottom right). Image taken from Van de Sande et al. (2011).

Girshick et al. (2014) achieved a breakthrough in object detection by combining Regions with CNNs (R-CNN). Part of its success is attributable to harnessing pretraining for image classification on ImageNet (Krizhevsky et al., 2012) and then finetuning for object detection. They show how to crop and warp parts of an image that correspond to a rectangular region proposal to a fixed square size, as required by the network (Fig. 2.10, top). In the case of free-form proposals, one strategy is to compute the features on the region's foreground mask, by subtracting the mean image of the whole dataset from the background pixels. They use Selective Search proposals for object detection and CPMC proposals for semantic segmentation. After a non-maximum suppression step, the features are classified either via the classification layer of the network or via an additional Support Vector Machine (Cortes and Vapnik, 1995). Following standard practice, the authors use the network trained for object detection to classify regions and then assign to each pixel the label with the highest classification score, which achieved state-of-the-art performance on semantic segmentation at the time of publication.

Fast R-CNN (Girshick, 2015) yields a large speedup over R-CNN, by extracting convolutional features only once per image and applying a region-of-interest (ROI)

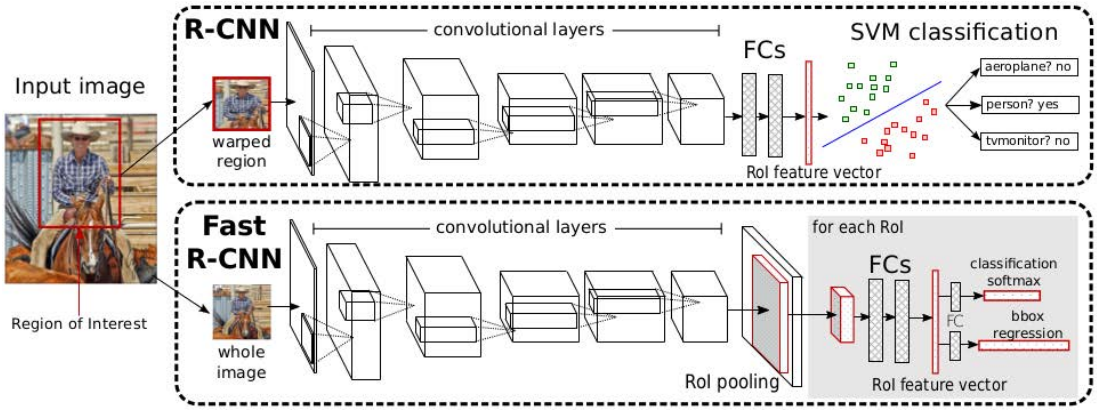


Figure 2.10: The Regions with CNN (R-CNN) method (top) and its successor Fast R-CNN. Image taken from Gonzalez-Garcia (2017).

pooling layer after the convolutional filters (Fig. 2.10, bottom). The ROI pooling layer takes an arbitrarily sized input, scales it to a 7×7 grid and applies max-pooling to each of the grid cells. Since the ROI pooling layer is partially differentiable with respect to the inputs, the whole network can be trained end-to-end. They also replace the SVM by the classification layer of the network, which yields equally good results and a simpler and faster end-to-end trainable pipeline. Besides the classification score output, the network also outputs a bounding box regression offset for each of the object classes. This bounding box regression step is used to correct the input bounding box, conditioned on the classification label output by the network. They show that the resulting multi-task loss further improves the result as both tasks influence each other through a shared representation in the earlier layers of the CNN. The authors of Fast R-CNN do not apply it to the problem of semantic segmentation, but we show in Sec. 4 that this is a promising approach.

Faster R-CNN (Ren et al., 2015) eliminates the remaining bottleneck of Fast R-CNN – the extraction of region proposals *outside* the network. It introduces a Region Proposal Network (RPN) that shares convolutional features with the classification network and therefore enables "nearly cost-free region proposals" generated *inside* the network. The RPN is technically a Fully Convolutional Network (FCN) (Long et al., 2015) with two convolutional layers that predicts an objectness score and the regressed bounds for a constant number of rectangular region proposals per image location. The proposals are then ranked according to their objectness score and only the highest scoring proposals are used in the Fast R-CNN part of the network. Using only 300 proposals, the authors are able to outperform Fast R-CNN with competing proposal

algorithms with 2,000 proposals. While the idea of Faster R-CNN is elegant, training it is not trivial and requires alternating between training the RPN and the Fast R-CNN component. Furthermore the region proposals generated by the RPN are rectangular, which is not suitable for semantic segmentation (see related experiments in Table 4.4).

Mask R-CNN (He et al., 2017a) is an extension of Faster R-CNN that adds a third branch that outputs the free-form object mask for each detection. This makes it particularly suited for the task of instance segmentation, but semantic segmentation and human pose estimation are also possible. The mask branch is itself an FCN that predicts an $m \times m$ mask for each ROI and each class. The actual mask of an ROI is then the one selected by the classification branch of the network. This is crucial, as it differs from the common practice when applying FCNs to semantic segmentation. Typically these approaches use a per-pixel softmax and a multinomial cross-entropy loss, which leads to competition between the masks of different classes. In Mask R-CNN they use a per-pixel sigmoid and a binary loss, which defers the competition until after the classification stage and allows both branches to execute in parallel. Another improvement over Fast(er) R-CNN is that Mask R-CNN uses ROIAlign instead of ROI pooling. While ROI pooling quantizes the convolutional feature map into bins, ROIAlign uses continuous bins and bilinear interpolation. This better preserves the spatial correspondence between inputs and outputs.

2.1.4 Key challenges in semantic segmentation

In this section we cover key topics related to semantic segmentation that are recurring throughout this thesis.

2.1.4.1 Competition

A key difference between object detection and semantic segmentation is the degree of *competition* between classes. In object detection there is relatively little competition between classes, as two heavily overlapping ground-truth boxes may tightly enclose two objects of different classes (e.g. a *person* sitting on a *chair*). However, a typical detection framework like R-CNN (Girshick et al., 2014) uses a softmax layer that normalizes the prediction scores of all classes to sum to one. Additionally the use of a set of one-vs-all linear SVMs also introduces some element of competition. But this form of competition is based on the idea that a box that looks like one class is unlikely (but not impossible) to contain another class. While this may be true in the majority of the

cases, there may also be attracting contextual forces between classes (such as a *person* wearing a *backpack* as shown in Fig. 6.8). Looking at the most common evaluation measure in detection – mean Average Precision (mAP) – we see that competition plays no role. It is merely the average of the per-class Average Precision values. For its computation a box is considered a correct detection if the ratio between the intersection and the union of the predicted and the ground-truth box is at least 0.5 (Everingham et al., 2015). Including an incorrect box for a particular class in the predictions does not reduce performance on any of the other classes. In constrast, in semantic segmentation each pixel is assigned a single class label. Hence there is competition between the classes for each pixel.

The problem of class competition is particularly pronounced on semantic segmentation datasets with *class imbalance*. If all classes occur with the same pixel-level frequency, their prediction scores can be naively compared without additional calibration. But when some classes are a factor of 10^4 less frequent than others (e.g. in SIFT Flow (Liu et al., 2011)), the classifier will simply ignore them. While this may be acceptable for some applications (e.g. in entertainment, product recognition), it may have disastrous consequences in others (e.g. cancer detection, autonomous driving).

Class competition is particularly relevant in the context of region-based semantic segmentation approaches. Here thousands of region proposals may partially overlap and cover objects with different class labels. For each pixel we need to figure out which regions contain it and which is the highest scoring label assigned to those regions. This label is then assigned to that pixel. In this thesis we present two methods that tackle the problems of class competition, class imbalance and overlapping regions, by training for the final evaluation criterion, instead of a proxy measure (see Sec. 3 and 4).

One approach to deal with competition in object detection is to use *non-maximum suppression* (NMS). As an example, Girshick et al. (2014) greedily reject every box that has an IOU overlap larger than a threshold with a higher scoring box that has already been selected. Similar procedures can be traced back to at least 1994 (Burel and Carel, 1994; Hosang et al., 2017). Although this may in theory be applied jointly between classes, it is usually done within each class independently (Felzenszwalb et al., 2010; Girshick et al., 2014) or only on the positive class in binary classification (Dalal and Triggs, 2005; Malisiewicz et al., 2011). Hence it is a different form of competition. Henderson and Ferrari (2016) show how to train an object detection network with NMS, which enables end-to-end training (see also Sec. 2.1.4.2). Hosang et al. (2017) show that NMS itself can be learned by a CNN, by introducing a special loss

that penalizes multiple detections of the same object. This way they remove the traditional handcrafted procedure. Xiao et al. (2010) state that NMS is not important for stuff: "While it is reasonable to require that one chair should generate one detection, a forest can generate an arbitrary number of smaller forest scenes". The authors focus on scenes (e.g. beach, forest) instead of objects, but they state that scenes behave like stuff, as they have unspecified spatial extent.

2.1.4.2 End-to-end training

The term end-to-end training is very popular in deep learning (He et al., 2017a; Henderson and Ferrari, 2016; Ren and Zemel, 2017; Caesar et al., 2016b), although the exact definitions may vary. We define end-to-end training as being able to backpropagate the gradients of a loss function all the way through the network to update all *trainable* parameters. This definition excludes approaches like R-CNN (Girshick et al., 2014), which consist of a CNN and an SVM that are trained separately, to optimize different criteria. It also excludes post-processing steps like superpixel pooling and separately trained CRFs (Farabet et al., 2013). There are different opinions as to whether methods that rely on traditional region proposals (as opposed to Region Proposal Networks (RPN) (Ren et al., 2015)) are end-to-end trainable. We argue that a layer (i.e. the region proposal method) that has no trainable parameters (cf. ReLU, dropout) and does not block the flow of gradients (as there are no layers before the region proposal method) does not affect the ability for end-to-end training. Furthermore, a method like Faster R-CNN (Ren et al., 2015) is end-to-end trainable despite the RPN. The fact that the training alternates between training the RPN and the Fast R-CNN component is an artefact of the optimization technique and the initialization, it is not an architectural issue.

End-to-end training is generally considered to be highly desirable from a machine learning point of view. Instead of training consecutive parts of a network to optimize different losses, it makes more sense to define a single network that optimizes the final loss, which typically achieves a higher performance. If the network is hard to optimize in its entirety, auxiliary losses can be used (cf. Szegedy et al. (2015)). From an engineering point of view end-to-end training is also highly desirable. If we adapt a traditional multi-stage pipeline to a new dataset or task, all design choices need to be reevaluated. End-to-end trainable networks require much less human (re-)design effort. This is particularly the case due to strong generalization abilities of deep CNNs. Finetuning allows the network to adapt to new scenarios with very few new training

samples.

2.1.4.3 Context

An important topic in object detection and semantic segmentation is image context. Context helps to reduce ambiguities by exploiting spatial co-occurrence statistics. Context can be roughly divided into local object context and global image context.

A straightforward approach to local context is to increase the size of the box or mask of an object to include surrounding pixels. The effective receptive field size of a CNN typically covers significant portions of the image (e.g. 228×228 pixels for VGG in Ren et al. (2015)). In R-CNN (Girshick et al., 2014) the authors use two representations for semantic segmentation – features of the actual object mask and features extracted from the entire bounding box surrounding the mask. They show that both representations are complementary, which shows the advantage of local context. Several works cover an object at multiple scales, thus combining local and global context. Mostajabi et al. (2015) present zoom-out features for semantic segmentation. They extract features from a "sequence of nested regions of increasing extent" from SLIC (Achanta et al., 2012) superpixels to upscaled copies of the superpixels and eventually the entire image (Fig. 2.11). Lim et al. (2009) proceed in a similar bottom-up fashion by constructing a region tree from the image contours using the method of Arbeláez et al. (2009). However their concept of region ancestry does not just extract features over upscaled versions of a superpixel, but finds discriminative parts, objects and scenes. They then learn the importance of different ancestral regions and types of features for different classes. Some works explicitly model the context between different objects. Bilen et al. (2014a) use a latent SVM model to learn an object and its context in terms of spatial location and appearance. Heitz and Koller (2008) present a graphical model that explicitly uses stuff to find things. Other works use attention mechanisms. Chen et al. (2016a) and Fan et al. (2010) weight the importance of the predictions at different scales with an attention mechanism. Ren and Zemel (2017) use a spatial recurrent attention model to successively segment instances in a human-like counting process. Finally, Alexe et al. (2012b) and Gonzalez-Garcia et al. (2015) use active search strategies in object detection to sequentially choose the next window to evaluate, which results in a substantial reduction of the number of classifier evaluations.

On the other hand many works use global image context. Oliva and Torralba (2001) model the *gist* of a scene using a spatial envelope model that covers spectral and

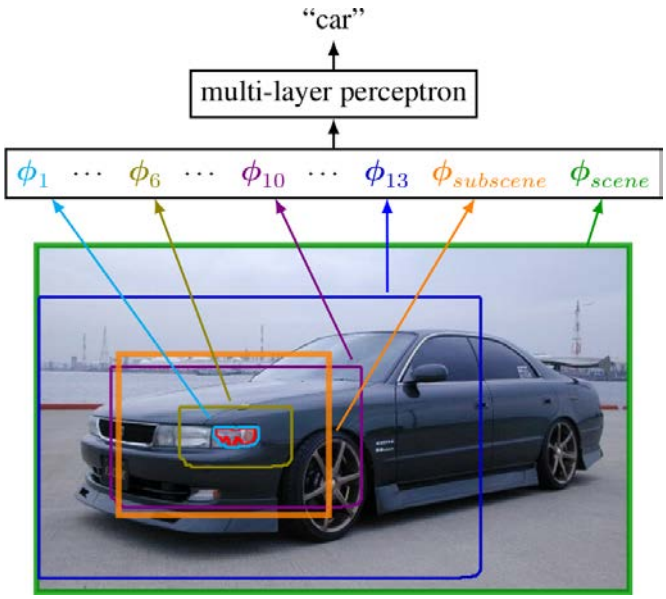


Figure 2.11: Zoom-out features. Image taken from Mostajabi et al. (2015).

coarsely localized information, but no specific information about objects in the image. Hoiem et al. (2008) model the interplay of objects in the scene and geometry of the scene. They then estimate the likely object size and location and manage to reduce the missed detection rate by up to 40% over the baseline while fixing the number of false positives. Perhaps the most important techniques to include context in semantic segmentation are Conditional Random Fields (CRF) and Markov Random Fields (MRF). For computational reasons most works only use unary and pairwise terms among pixels or superpixels (Shotton et al., 2006; Verbeek and Triggs, 2007; Ladicky et al., 2010a; Farabet et al., 2013; Chen et al., 2015a; Zheng et al., 2015). Some authors present efficient techniques for higher-order terms, defined over pixels (Komodakis and Paragios, 2009), superpixels (Kohli et al., 2009; Arnab et al., 2016), detections (Arnab et al., 2016) or segmentation instances (Arnab and Torr, 2016). These higher-order terms attempt to assign the same label to all pixels in a pattern, superpixel, detection or segmentation instance. Kohli et al. (2009) enforce superpixel label consistency as a soft constraint. Komodakis and Paragios (2009) present pattern-based potentials, which allow for efficient message passing for a sparse set of patterns. Pattern-based potentials can be seen as a generalization of P^n Potts potentials and exemplar-based priors. Without special consideration, CRFs often exert merely a spatial smoothing effect on the pixel labels, thereby accidentally removing small predictions. Rother et al. (2004) use contrast-sensitive potentials to improve localization. Deeplab (Chen et al., 2015a) uses the fully connected CRF of Krähenbühl and Koltun (2011), with enables efficient

inference on the pixel-level and captures fine-edge details. As an alternative to CRFs, Chen et al. (2016a) present domain transform, an edge-preserving filtering method. Gadde et al. (2016) propose a bilateral inception module that propagates edge-aware information and allows for long-range interaction between pixels. Finally, a number of works slide a spatial Recurrent Neural Network (RNN) over an image and aggregate context. This is particularly useful for long-range or even global context. Shuai et al. (2016) place a directed acyclic graph RNN for context aggregation between encoder and decoder. Bell et al. (2016) use an RNN only outside the region of interest, while using skip connections inside the region to extract features at multiple scales. Yan et al. (2016) develop ReNet Long-Short Term Memory (LSTM) layers to aggregate context before application of a Fully Convolutional Network, which results in "full-image receptive fields". Byeon et al. (2015) go one step further and carry out classification, segmentation and context integration with LSTM networks.

2.2 Efficient annotation and learning schemes

Before presenting efficient annotation and learning schemes, we begin by describing the status quo of dataset annotation in semantic segmentation and object detection. Semantic labels are either predefined (Cordts et al., 2016; Caesar et al., 2018) or they are free-form text labels (Tighe and Lazebnik, 2010; Mottaghi et al., 2014), which means that an annotator can add missing labels. In semantic segmentation annotators draw the outlines of an object mask with a polygon annotation tool (e.g. LabelMe (Russell et al., 2008)). This typically takes between a few minutes and one hour per image (Lin et al., 2014; Cordts et al., 2016). In object detection annotators draw only an axis-aligned bounding box per object. This typically takes less than a minute per box (Su et al., 2012). The method is then trained with the same level of supervision that it outputs (e.g. fully supervised object detection is trained with boxes to predict boxes).

2.2.1 Efficient annotation

Many works have attempted to make annotation more time or cost efficient. A straightforward approach to reduce annotation time is by labeling superpixels instead of pixels or polygons (Yamaguchi et al., 2012; Galasso et al., 2012; Pont-Tuset et al., 2015), especially in video segmentation. We annotate superpixels in Sec. 6, which leads to a significant speedup while achieving the same quality of annotations. An indispens-

able tool to create large-scale datasets is crowd-sourcing. While small datasets are often created by in-house expert annotators (Brostow et al., 2009; Xiao et al., 2010; Everingham et al., 2015), that is not feasible for large-scale datasets with millions of images. Crowd-sourced annotations bring their own problems: Annotators need to be instructed and trained for the task. This is more difficult than for in-house annotators, possibly due to the heterogeneous educational, cultural and language background of crowd-sourced workers. Furthermore, crowd-sourced workers are typically paid per image and therefore have no incentive to spend a lot of time on the instructions. They also need to be closely supervised to deliver acceptable quality. This may be due to a lack of identification with the goals of the scientific project or an incentive to maximize profit by delivering poor quality. The most important tool to filter annotators is to occasionally compare their answers to known answers to "gold-standard" questions (Su et al., 2012). Furthermore a complex task should be divided into atomic and simple subtasks. As an example, to annotate COCO (Lin et al., 2014), the authors use the following tasks: 1) label the categories present in an image 2) mark all objects 3) segment each object. Eventually the resulting annotations are noisier than with expert annotators. Therefore multiple annotators are required to verify or redo each others work until they achieve consensus (Lin et al., 2014; Caesar et al., 2018). When the label space is large, asking a user about the presence of each object class in an image is extremely time consuming. Deng et al. (2014b) show how to significantly reduce the time consumption, when the labels are correlated, sparse and naturally form a hierarchy. Their algorithm proceeds through the label hierarchy and dynamically selects the next query.

Several works use *human-in-the-loop* or interactive annotation methods to speedup annotation. Rother et al. (2004) present Grabcut, which requires an annotator to draw only a loose rectangle around an object. It automatically infers objects mask by solving a graph cut optimization that takes into account texture and edge information. TouchCut (Wang et al., 2014) requires a single touch or click from an annotator. It uses a level set method to fuse edge, texture and geometry sampled around the touch point. This technique can be extended from image to video segmentation. Castrejon et al. (2017) present Polygon-RNN, which is a recurrent neural network that predicts the next point of a polygon at each step of the annotation. The annotator can then correct a predicted point if necessary. Papadopoulos et al. (2016) propose a scheme for training object detectors which only requires annotators to verify bounding boxes produced automatically by the learning algorithm. This is substantially faster while delivering

detectors that are almost as good as in the fully supervised case.

2.2.2 Image-level annotations

An alternative to fully supervised learning is *weakly supervised learning*, where a method is trained with a lesser level of supervision than it outputs. Unless specified, this thesis refers to weakly supervised learning when training from image-level labels instead of boxes or masks. This level of annotation is much less time-consuming as objects do not have to be localized in the image. In fact, the web provides an almost unlimited and free resource for image-level tags. However special consideration needs to be taken for particularly noisy labels (Raykar et al., 2009; Cinbis et al., 2014; Zhang et al., 2015b). In practice the performance of weakly supervised approaches is typically about 60% to 80% of the fully supervised performance (Pathak et al., 2015a; Hoffman et al., 2014).

Weakly supervised semantic segmentation is a hard task, as priors need to be encoded in the method that could otherwise be learned from data in the fully supervised case. Therefore region-based approaches (Sec. 2.1.3.2) are important in the weakly supervised setup (Vezhnevets et al., 2011, 2012; Zhang et al., 2014; Xu et al., 2014; Caesar et al., 2015; Zhang et al., 2015b; Krapac and Segvic, 2016), as they typically assign a single label to each superpixel or region. Depending on the type of regions used this may enforce priors on spatial smoothness as well as coherent texture, edges and color. This does not mean that fully convolutional approaches (Sec. 2.1.3.1) do not have such priors, but they are more implicit (e.g. in the large receptive fields or in post-processing with a CRF). Given a suitable initialization, most weakly supervised region-based works alternate between two steps: relabeling of the regions and retraining of the classifier (Vezhnevets et al., 2011, 2012; Xu et al., 2014; Caesar et al., 2015; Zhang et al., 2015b). Vezhnevets et al. (2012) define a CRF over superpixels, using a maximum expected agreement model and an algorithm based on Gaussian processes to solve it. Using CNNs, perhaps the simplest weakly supervised approach is the Multiple Instance Learning (MIL) (Dietterich et al., 1997) technique of Pathak et al. (2015b). They use an FCN and identify the highest scoring pixel in the heat map of the classes present in the image. They then compute a standard cross-entropy log-loss on those highest scoring pixels. This means that they only check whether all image-level labels are present in the prediction, but not whether other classes are absent. Furthermore this method is very sensitive to the initialization of the network. Without initialization by

pretraining for whole image classification, the network converges to a solution that outputs only the most common class (typically *background*). It should be noted that such pretraining does not invalidate the weakly supervised protocol, as pretraining requires only image-level labels. It also shows again that pretraining and finetuning are a useful tool to transfer knowledge of low-level filters of the CNN across domains and tasks. Bearman et al. (2016) improve upon this loss formulation by including an additional image-level label absence term that penalizes high scores for predicted classes that are not in the ground-truth annotations of that image. Kolesnikov and Lampert (2016) introduce the Seed Expand Constrain (SEC) model (Fig. 2.12) that follows three guiding principles: to *seed* a model with weak localization cues, to *expand* the object based on information about which classes can occur in an image and to *constrain* the segmentations to coincide with object boundaries. Figuring out the spatial extent of an object is perhaps the biggest challenge in weakly supervised semantic segmentation. Stuff classes have "unspecified spatial extent" (Xiao et al., 2010), which may exacerbate the problem. While edges in the image may be sufficient to delineate things, stuff is more dependent on texture and we require a larger area to compute reliable statistics. On the other hand, Ion et al. (2011) argue that "inference is harder to constrain for [things], which [require] longer-range interactions among groups of measurements". Bearman et al. (2016) include an objectness prior (Alexe et al., 2010) in their training loss to accurately infer the spatial extent of things. Similar in spirit to objectness, but without the need to train an external objectness measure, Saleh et al. (2016) use a built-in foreground-background prior. They assume that a network pretrained for an object recognition task learns to focus on things and their parts, rather than stuff. Oh et al. (2017) use saliency to estimate prior knowledge on the object extent and image statistics. Tokmakov et al. (2016) incorporate unsupervised motion cues and train from weakly labeled videos. Zhang et al. (2015b) use label co-occurrence statistics. Bilen et al. (2014b) use vertical symmetry and mutual exclusion terms to encode domain specific knowledge in weakly supervised object detection. Papandreou et al. (2015); Pathak et al. (2015a) use class-specific constraints, such as a foreground-background bias, encouraging a minimum or maximum number of pixels with a particular label and size constraints.

Several authors have presented frameworks that use varying levels of supervision (Hong et al., 2015; Papandreou et al., 2015; Xu et al., 2015a; Javanmardi et al., 2016). These works fully annotate a small subset of the images and use lesser amounts of supervision or none at all for the remaining images. Dehghani et al. (2017) train a

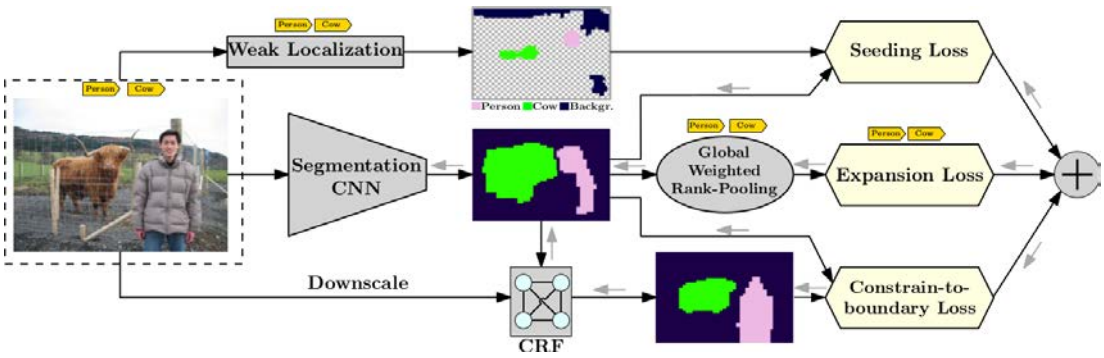


Figure 2.12: The SEC model. Image taken from Kolesnikov and Lampert (2016).

target network in a weakly supervised fashion, but use an additional confidence network that is trained on a small set of fully supervised data. The confidence network is a meta learner used to calibrate the learning rate of the target network for each batch. This way noisy annotations receive a lower learning rate.

It should be noted here that objectness or saliency measures and foreground-background biases are particularly successful on thing-only datasets (Everingham et al., 2015). In this work we attempt to do away with this recent narrow focus on things in weakly supervised learning (see Chapter 3 and 5) and restore the balance between stuff and things.

2.2.3 Other forms of annotation

Weakly supervised learning is not restricted to image-level annotations. A number of different forms of supervision have been proposed. For training object detectors, Papadopoulos et al. (2017b) ask annotators to click on the center of an imaginary bounding box of the object. They propose a method to derive a bounding box from these clicks. This is much faster and close to the performance of manually drawn boxes. Papadopoulos et al. (2017a) avoid the cognitive load of the bounding box drawing task by clicking on four extreme points of an object. This is faster than the traditional way of drawing bounding boxes, while delivering the same quality of boxes. Papadopoulos et al. (2014) record human eye tracks of annotators instructed to find an object and derive approximate bounding boxes from these fixations.

In semantic segmentation, several authors use box supervision instead of masks (Dai et al., 2015a; Khoreva et al., 2017). Bearman et al. (2016) use point annotations for stuff and things in semantic segmentation. In the Seed Expand Constrain framework (see Sec. 2.2.2) this would be equivalent to providing ground-truth for the seed

step. Others use scribbles/squiggles that are one of the most user-friendly ways of interacting and favored for annotating stuff that has no well-defined shape (Xu et al., 2015a; Bearman et al., 2016; Lin et al., 2016a). We hypothesize that compared to points, squiggles come at a similar annotation cost, but provide additional information about the spatial extent of an object.

Finally, a plethora of works use image captions to learn about the objects in the image (Ozcan et al., 2011; Guillaumin et al., 2008; Vinyals et al., 2015). Similar to image tags, these can be efficiently mined from the web, while taking into account possible noise in the labels. Contrary to image tags, captions come with additional information regarding the relations between and attributes of stuff and things in the image.

Furthermore, some methods are trained in an unsupervised way, i.e. without labels. Sankaranarayanan et al. (2017); Yi et al. (2017); Zhu et al. (2017) perform unsupervised domain transfer using Generative Adversarial Networks (GAN). Saito et al. (2017) show that they can learn a model of the road class in car-based imagery simply from location priors (the road is assumed to be at the bottom center of the image). Self-supervised learning is a variant of unsupervised learning where labels are "manufactured" through unlabeled data (Larsson et al., 2017). Papazoglou and Ferrari (2013) use motion boundaries to separate foreground objects from the background in a video. Doersch et al. (2015) extract random pairs of images patches and learn to predict the position of one image patch relative to another. Fernando et al. (2017) use odd-one out learning to identify one unrelated item in a set, e.g. multiple videos in correct and one video in incorrect temporal order. Larsson et al. (2017) convert color images to grayscale and then use colorization to restore the original color image. Networks that are trained in a self-supervised way are often useful initialization for other tasks and may one day remove the need for pretraining with image-level labels.

2.3 Stuff and Things

In this section we look at the role of stuff and things in existing datasets and give a brief overview of works that use stuff. For a definition of stuff and things please refer to Sec. 1.1.

2.3.1 Datasets

Here we present existing dataset that include stuff labels, things labels or both. In Sec. 6.3.3 we compare these datasets to our new dataset COCO-Stuff.

Stuff-only datasets. Early stuff datasets (Brodatz, 1966; Dana et al., 1999; Lazebnik et al., 2005; Caputo et al., 2010) focused on texture classification and had simple images completely covered with a single textured patch. The more recent Describable Textures Dataset (Cimpoi et al., 2014) instead collects textured patches in the wild, described by human-centric attributes. A related task is material recognition (Sharan et al., 2014; Bell et al., 2013, 2015). Although the recent Materials in Context dataset (Bell et al., 2015) features realistic and difficult images, they are mostly restricted to indoor scenes with man-made materials. For the task of semantic segmentation, the Stanford Background dataset (Gould et al., 2009) offers pixel-level annotations for seven common stuff categories and a single *foreground* category (confounding all thing classes). All stuff-only datasets above have no distinct thing classes, which make them inadequate to study the relations between stuff and thing classes.

Thing-only datasets. These datasets have bounding box or outline-level annotations of things, e.g. PASCAL VOC (Everingham et al., 2015), ILSVRC (Russakovsky et al., 2015) and COCO (Lin et al., 2014). They have pushed the state-of-the-art in Computer Vision, but the lack of stuff annotations limits the ability to understand the whole scene.

Stuff and thing datasets. Some datasets have pixel-wise stuff and thing annotations (Table 2.2). Early datasets like MSRC 21 (Shotton et al., 2006), NYUD (Silberman et al., 2012), CamVid (Brostow et al., 2009), KITTI (Geiger et al., 2012) and SIFT Flow (Liu et al., 2011) annotate less than 50 classes on less than 5,000 images. More recent large-scale datasets like Barcelona (Tighe and Lazebnik, 2010), LM+SUN (Tighe and Lazebnik, 2013b), PASCAL Context (Mottaghi et al., 2014), Cityscapes (Cordts et al., 2016) and ADE20K (Zhou et al., 2017b) annotate tens of thousands of images with hundreds of classes. The Barcelona (Tighe and Lazebnik, 2010) dataset comes with an additional binary stuff/thing label that is "somewhat arbitrary" according to (Tighe and Lazebnik, 2011) (see also Sec. 1.1). PASCAL Context (Mottaghi et al., 2014) complements the PASCAL VOC (Everingham et al., 2015) dataset with stuff (and thing) labels, similar to COCO-Stuff and COCO. We compare our COCO-Stuff to these datasets in Sec. 6.3.3.

Dataset	Images	Classes	Stuff classes	Thing classes	Year
MSRC 21	591	21	6	15	2006
KITTI	203	14	9	4	2012
CamVid	700	32	13	15	2008
Cityscapes	25,000	30	13	14	2016
SIFT Flow	2,688	33	15	18	2009
Barcelona	15,150	170	31	139	2010
LM+SUN	45,676	232	52	180	2010
PASCAL Context	10,103	540	152	388	2014
NYUD	1,449	894	190	695	2012
ADE20K	25,210	2,693	1,242	1,451	2017
COCO-Stuff	163,957	172	91	80	2018

Table 2.2: An overview of datasets with pixel-level stuff and thing annotations. COCO-Stuff, which we present in Chapter 6, is the largest existing dataset with dense stuff and thing annotations. The number of stuff and thing classes are estimated given the definitions in Sec. 6.2. Sec. 6.3.3 shows that COCO-Stuff also has more *usable* classes than any other dataset.

2.3.2 Usage of stuff and things

Hundreds of works in the literature have been using the distinction between stuff and things. The majority of them use it to analyze datasets or results on specific classes. Several works present statistics that show that stuff covers the majority of the pixels in an image (Tighe and Lazebnik, 2013a; Mottaghi et al., 2013, 2014; Zhang et al., 2015a). Some argue that stuff is easier to segment than things (Tighe and Lazebnik, 2010; Ion et al., 2011; Liu et al., 2011; Tighe and Lazebnik, 2013a; Tighe et al., 2014; Zhang et al., 2015a; Xu et al., 2015a; Zhou et al., 2017b), to which we present a counter-example in Sec. 6.4.3.

Some works explicitly learn stuff and things in their method. Tighe and Lazebnik (2013a) use a superpixel-based retrieval method for stuff and an object detector for thing instances. Interestingly, they show that training a thing detector only for thing classes achieves a lower performance than when training it for both, stuff and things. Ladicky et al. (2010b); Kim et al. (2012); Sun et al. (2013) use higher order potentials

in a CRF to enforce label consistency between hypotheses of thing detections and stuff segments. Sun et al. (2013) additionally model the geometric relationships (e.g. above, in front) between things and assume that stuff regions form the background. Dai et al. (2015b) present a convolutional feature masking approach for joint stuff and thing segmentation. The only difference between stuff and things in this approach is the different type of region proposals: While things use the raw segments, stuff proposals are generated using a *segment pursuit* approach to use the largest possible proposal that is still pure in terms of stuff labels. Brahmbhatt et al. (2017) present a two-stream network called StuffNet. One stream of the network (Deeplab) is trained to segment stuff, the other (Faster R-CNN) is trained to detect things. They show that the joint stuff-thing features improve thing detection. They suggest to train stuff segmentation only once and transfer to other datasets, which is an idea also promoted in this thesis. In a similar fashion, Zhou et al. (2017b) use a two-stream network called Cascade Segmentation Module. The stuff stream outputs stuff classes and a generic objectness term. The thing stream segments things and only those pixels with a high objectness are taken into account. Both streams are merged to produce the final output. It should be noted that these methods make the distinction between stuff and things explicit. Tighe et al. (2014) point out that in future work they would prefer to explore ways that do not to rely on a hard things-vs-stuff split.

Others have shown that stuff is a useful contextual cue to detect things and vice versa (Rabinovich et al., 2007; Heitz and Koller, 2008; Kim et al., 2012; Mottaghi et al., 2014; Shi et al., 2017a). Heitz and Koller (2008) distinguish between stuff-stuff, thing-thing and stuff-thing context and focus on the latter. In Sec. 5 we use all of these, as well as second order (stuff-thing-thing) transfer. Finally, Xie et al. (2016) annotate urban street scenes using different protocols for stuff and things. They annotate things (*car*, *caravan*, *box*) as cuboids and ellipsoids in 3D and stuff (*road*, *sidewalk*, *grass*) as 2D polygons from a bird's eye view that are then extruded into 3D.

Chapter 3

Joint calibration for semantic segmentation

3.1 Introduction

Semantic segmentation is the task of assigning a class label to each pixel in an image (see Fig. 3.1 and Sec. 2.1). In the fully supervised setting, we have ground-truth labels for all pixels in the training images. In the weakly supervised setting, class labels are only given at the image-level. We tackle both settings in a single framework which builds on region-based classification.

Our framework addresses three important problems common to region-based semantic segmentation. First of all, objects naturally occur at different scales within an image (Carreira and Sminchisescu, 2010; Uijlings et al., 2013). Performing recognition at a single scale inevitably leads to regions covering only parts of an object which may have ambiguous appearance, such as *wheels* or *fur*, and to regions straddling over multiple objects, whose classification is harder due to their mixed appearance. Therefore many recent methods operate on pools of regions computed at multiple scales, which have a much better chance of containing some regions covering complete objects (Plath et al., 2009; Carreira and Sminchisescu, 2010; Carreira et al., 2012; Li et al., 2013; Hariharan et al., 2014; Girshick et al., 2014; Zhang et al., 2015b). However, this leads to overlapping regions which may lead to conflicting class predictions at the pixel-level. These conflicts need to be properly resolved.

Secondly, classes are often imbalanced (Farabet et al., 2013; Tighe and Lazebnik, 2013a; Vezhnevets et al., 2011; Kekeç et al., 2014; Xu et al., 2014; Yang et al., 2014; Long et al., 2015; Sharma et al., 2014, 2015; Xu et al., 2015a; Mostajabi et al., 2015;

Figure 3.1: Semantic segmentation is the task of assigning class labels to all pixels in the image. During training, with full supervision we have ground-truth labels of all pixels. With weak supervision we only have labels at the image-level. For more information refer to Sec. 2.1.

Byeon et al., 2015; Shuai et al., 2015): *cars* and *grass* are frequently found in images while *tricycles* and *gravel* are much rarer. Due to the nature of most classifiers, without careful consideration these rare classes are largely ignored: even if the class occurs in an image the system will rarely predict it. Since class frequencies typically follow a power law distribution, this problem becomes increasingly important with the modern trend towards larger datasets with more and more classes.

Finally, classes compete: a pixel can only be assigned to a single class (e.g. it cannot belong to both *sky* and *airplane*). To properly resolve such competition, a semantic segmentation framework should take into account predictions for multiple classes jointly.

In this chapter we address these three problems with a joint calibration method over an ensemble of SVMs, where the calibration parameters are optimized over all classes, and for the final evaluation criterion, i.e. the accuracy of pixel-level labeling, as opposed to simply region classification (Plath et al., 2009; Girshick et al., 2014; Hariharan et al., 2014). While each SVM is trained for a single class, their joint calibration deals with the competition between classes. Furthermore, the criterion we optimize for explicitly accounts for class imbalance. Finally, competition between overlapping regions is resolved through maximization: each pixel is assigned the highest scoring class over all regions covering it. We jointly calibrate the SVMs for optimal pixel labeling *after* this maximization, which effectively takes into account conflict resolution between overlapping regions. Experiments on the popular SIFT Flow (Liu et al., 2011) dataset show a considerable improvement over the state-of-the-art at the time of first publication of this work (Caesar et al., 2015) in both the fully and weakly supervised setting (+6% and +10%, respectively).

This chapter is an updated version of our paper published at the British Machine Vision Conference 2015 (Caesar et al., 2015). Later we added results using Fast R-CNN instead of R-CNN and VGG-16 instead of AlexNet (see Sec. 2.1.2 and 2.1.3.2). In Sec. 3.5 we contextualize this work in the literature and in Sec. 3.6 we present future work.

3.2 Related work

Early works on semantic segmentation used pixel or patch based features over which they define a Condition Random Field (CRF) (Shotton et al., 2009; Verbeek and Triggs, 2007). Many modern successful works use region-level representations, both in the fully supervised (Boix et al., 2012; Carreira et al., 2012; Hariharan et al., 2014; Girshick et al., 2014; Li et al., 2013; Plath et al., 2009; Tighe and Lazebnik, 2010, 2011; Tighe et al., 2014; Yang et al., 2014; Sharma et al., 2014, 2015; George, 2015; Mostajabi et al., 2015) and weakly supervised (Vezhnevets et al., 2011, 2012; Xu et al., 2014; Zhang et al., 2014; Xu et al., 2015a; Zhang et al., 2015b) settings. A few recent works use CNNs to learn a direct mapping from image to pixel labels (Farabet et al., 2013; Sharma et al., 2014, 2015; Pinheiro and Collobert, 2014, 2015; Shuai et al., 2015; Zheng et al., 2015; Papandreou et al., 2015; Schwing and Urtasun, 2015; Lin et al., 2016b), although some of them (Farabet et al., 2013; Sharma et al., 2014, 2015) use region-based post-processing to impose label smoothing and to better respect object boundaries. Other recent works use CRFs to refine the CNN pixel-level predictions (Zheng et al., 2015; Schwing and Urtasun, 2015; Lin et al., 2016b; Chen et al., 2015a; Papandreou et al., 2015). For more information we refer the reader to Chapter 2. In this work we focus on region-based semantic segmentation, which we discuss in light of the three problems raised in the introduction.

Overlapping regions. Traditionally, semantic segmentation systems use superpixels (Boix et al., 2012; Tighe and Lazebnik, 2010, 2011; Tighe et al., 2014; Yang et al., 2014; George, 2015; Mostajabi et al., 2015; Sharma et al., 2014, 2015), which are non-overlapping regions resulting from a single-scale oversegmentation. However, appearance-based recognition of superpixels is difficult as they typically capture only parts of objects, rather than complete objects. Therefore, many recent methods use overlapping multi-scale regions (Carreira and Sminchisescu, 2010; Carreira et al., 2012; Hariharan et al., 2014; Girshick et al., 2014; Li et al., 2013; Plath et al., 2009;

Zhang et al., 2015b). However, these may lead to conflicting class predictions at the pixel-level. Carreira et al. (2012) address this simply by taking the maximum score over all regions containing a pixel. Both Hariharan et al. (2014) and Girshick et al. (2014) use non-maximum suppression, which may give problems for nearby or interacting objects (Li et al., 2013). Li et al. (2013) predict class overlap scores for each region at each scale. Then they create superpixels by intersecting all regions. Finally, they assign overlap scores to these superpixels using maximum composite likelihood (i.e. taking all multi-scale predictions into account). Plath et al. (2009) use classification predictions over a segmentation hierarchy to induce label consistency between parent and child regions within a tree-based CRF framework. After solving their CRF formulation, only the smallest regions (i.e. leaf-nodes) are used for class prediction. In the weakly supervised setting, most works use superpixels (Vezhnevets et al., 2011, 2012; Xu et al., 2014, 2015a) and so do not encounter problems of conflicting predictions. Zhang et al. (2014) use overlapping regions to enforce a form of class-label smoothing, but they all have the same scale. A different Zhang et al. (2015b) use overlapping region proposals at multiple scales in a CRF.

Class imbalance. As the PASCAL VOC dataset (Everingham et al., 2010) is relatively balanced, most works that experiment on it did not explicitly address this issue (Boix et al., 2012; Carreira et al., 2012; Hariharan et al., 2014; Girshick et al., 2014; Li et al., 2013; Plath et al., 2009; Long et al., 2015; Pinheiro and Collobert, 2015; Zheng et al., 2015; Schwing and Urtasun, 2015; Lin et al., 2016b; Chen et al., 2015a). On highly imbalanced datasets such as SIFT Flow (Liu et al., 2011), Barcelona (Tighe and Lazebnik, 2010) and LM+SUN (Tighe and Lazebnik, 2013a), rare classes pose a challenge. This is observed and addressed by Tighe and Lazebnik (2013a) and Yang et al. (2014): for a test image, only a few training images with similar context are used to provide class predictions, but for rare classes this constraint is relaxed and more training images are used. Vezhnevets et al. (2011) balance rare classes by normalizing scores for each class to range $[0, 1]$. A few works (Mostajabi et al., 2015; Xu et al., 2014, 2015a) balance classes by using an inverse class frequency weighted loss function.

Competing classes. Several works train one-vs-all classifiers separately and resolve labeling through maximization (Carreira et al., 2012; Hariharan et al., 2014; Girshick et al., 2014; Li et al., 2013; Plath et al., 2009; Tighe and Lazebnik, 2013a; Mostajabi et al., 2015; Pinheiro and Collobert, 2015). This is suboptimal since the scores of

different classes may not be properly calibrated. Instead, Tighe and Lazebnik (2010, 2013a) and Yang et al. (2014) use Nearest Neighbor classification which is inherently multi-class. In the weakly supervised setting appearance models are typically trained in isolation and remain uncalibrated (Vezhnevets et al., 2011, 2012; Zhang et al., 2014; Xu et al., 2014, 2015a). To the best of our knowledge, Boix et al. (2012) is the only work in semantic segmentation to perform joint calibration of SVMs. While this enables to handle competing classes, in their work they use non-overlapping regions. In contrast, in our work we use overlapping regions where conflicting predictions are resolved through maximization. In this setting, joint calibration is particularly important, as we will show in Sec. 3.4. As another difference, Boix et al. (2012) address only full supervision whereas we address both full and weak supervision in a unified framework.

3.3 Method

3.3.1 Model

We represent an image by a set of overlapping regions (Uijlings et al., 2013) described by CNN features (Girshick et al., 2014) (Sec. 3.3.4). Our semantic segmentation model infers the label o_p of each pixel p in an image:

$$o_p = \underset{c,\, r \ni p}{\operatorname{argmax}} \; \sigma(w_c \cdot x_r,\, a_c, b_c) \tag{3.1}$$

As appearance models, we have a separate linear SVM w_c per class c. These SVMs score the features x_r of each region r. The scores are calibrated by a sigmoid function σ, with different parameters a_c, b_c for each class c. The argmax returns the class c with the highest score over all regions that contain pixel p. This involves maximizing over classes for a region, and over the regions that contain p.

During training we find the SVM parameters w_c (Sec. 3.3.2) and calibration parameters a_c and b_c (Sec. 3.3.3). The training of the calibration parameters takes into account the effects of the two maximization operations, as they are optimized for the output pixel-level labeling performance (as opposed to simply accuracy in terms of region classification).

3.3.2 SVM training

Fully supervised. In this setting we are given ground-truth pixel-level labels for all images in the training set (Fig. 3.1). This leads to a natural subdivision into ground-truth regions, i.e. non-overlapping regions perfectly covering a single class. We use these as positive training samples. However, such idealized samples are rarely encountered at test time since there we have only imperfect region proposals (Uijlings et al., 2013). Therefore we use as additional positive samples for a class all region proposals which overlap heavily with a ground-truth region of that class (i.e. Intersection-over-Union greater than 50% (Everingham et al., 2010)). As negative samples, we use all regions from all images that do not contain that class. In the SVM loss function we apply inverse frequency weighting in terms of the number of positive and negative samples.

Weakly supervised. In this setting we are only given image-level labels on the training images (Fig. 3.1). Hence, we treat region-level labels as latent variables which are updated using an alternated optimization process (as in (Vezhnevets et al., 2011, 2012; Xu et al., 2014; Zhang et al., 2015b; Xu et al., 2015a)). To initialize the process, we use as positive samples for a class all regions in all images containing it. At each iteration we alternate between training SVMs based on the current region labeling and updating the labeling based on the current SVMs (by assigning to each region the label of the highest scoring class). In this process we keep our negative samples constant, i.e. all regions from all images that do not contain the target class. In the SVM loss function we apply inverse frequency weighting in terms of the number of positive and negative samples.

3.3.3 Joint calibration

We now introduce our joint calibration procedure, which addresses three common problems in semantic segmentation: (1) conflicting predictions of overlapping regions, (2) class imbalance, and (3) competition between classes.

To better understand the problem caused by overlapping regions, consider the example of Fig. 3.2. It shows three overlapping regions, each with different class predictions. The final goal of semantic segmentation is to output a pixel-level labeling, which is evaluated in terms of pixel-level accuracy. In our framework we employ a winner-takes all principle: each pixel takes the class of the highest scored region which contains it. Now, using uncalibrated SVMs is problematic (second row in Fig. 3.2). SVMs

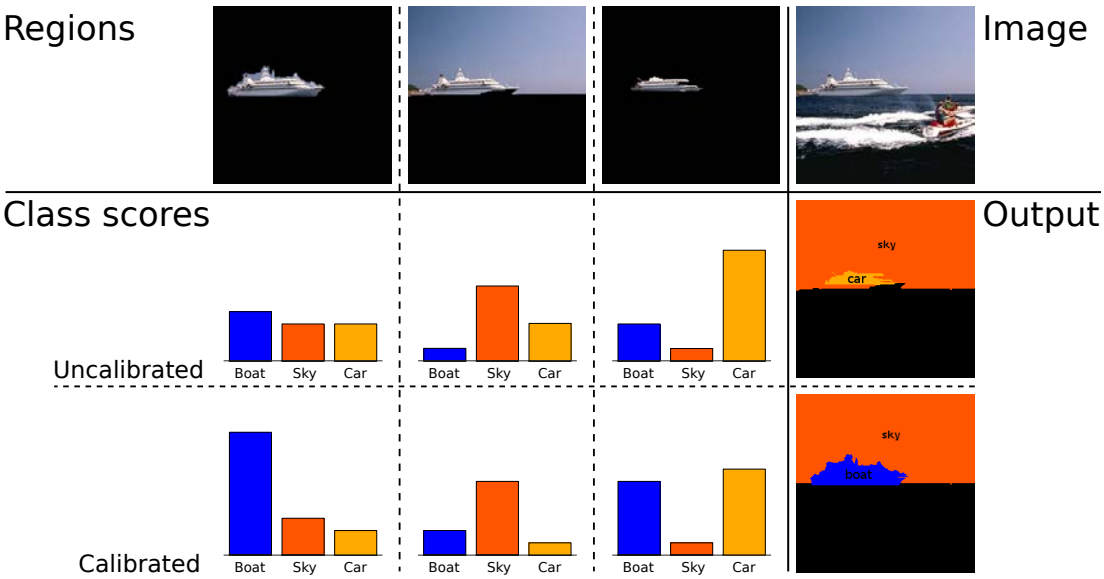


Figure 3.2: The first row shows multiple region proposals (left) extracted from an image (right). The following rows show the per-class SVM scores of each region (left) and the pixel-level labeling (right). Row 2 shows the results before and row 3 after joint calibration.

are trained to predict class labels at the region-level, not the pixel-level. However, different regions have different area, and, most importantly, not all regions contribute all of their area to the final pixel-level labeling: Predictions of small regions may be completely suppressed by bigger regions (e.g. in Fig. 3.2, row 3, the inner *boat* region is suppressed by the prediction of the complete *boat*). In other cases, bigger regions may be *partially* overwritten by smaller regions (e.g. in Fig. 3.2 the *boat* region partially overwrites the prediction of the larger *boat+sky* region). Furthermore, the SVMs are trained in a one-vs-all manner and are unaware of other classes. Hence they are unlikely to properly resolve competition between classes even within a single region. The problems above show that without calibration, the SVMs are optimized for the wrong criterion. We propose to jointly calibrate SVMs for the correct criterion, which corresponds better to the evaluation measure typically used for semantic segmentation (i.e. class accuracy). We do this by applying sigmoid functions σ to all SVM outputs:

$$\sigma(w_c \cdot x_r,\ a_c, b_c) = (1 + \exp(a_c \cdot w_c \cdot x_r + b_c))^{-1} \tag{3.2}$$

where a_c, b_c are the calibration parameters for class c. We calibrate the parameters of all classes jointly by minimizing a loss function $\mathcal{L}(o,l)$, where o is the pixel labeling output of our method on the full training set ($o = \{o_p; p = 1 \ldots P\}$) and l the ground-truth labeling.

We emphasize that the pixel labeling output o is the result *after* the maximization over classes and regions in Eq. 3.1. Since we optimize for the accuracy of this final output labeling, and we do so jointly over classes, our calibration procedure takes into account both problems of conflicting class predictions between overlapping regions and competition between classes. Moreover, we also address the problem of class imbalance, as we compensate for it in our loss functions below.

Fully supervised loss. In this setting our loss directly evaluates the desired performance measure, which is typically class accuracy (Sec. 2.1.1) (Tighe and Lazebnik, 2010; Sharma et al., 2014; Yang et al., 2014; Farabet et al., 2013; Long et al., 2015)

$$\mathcal{L}(o,l) = 1 - \frac{1}{C}\sum_{c=1}^{C}\frac{1}{P_c}\sum_{p;l_p=c}[l_p = o_p] \tag{3.3}$$

where l_p is the ground-truth label of pixel p, o_p is the output pixel label, P_c is the number of pixels with ground-truth label c, and C is the number of classes. $[\cdot]$ is 1 if the condition is true and 0 otherwise. The inverse frequency weighting factor $1/P_c$ deals with class imbalance.

Weakly supervised loss. Also in this setting the performance measure is typically class accuracy (Vezhnevets et al., 2011, 2012; Xu et al., 2015a; Zhang et al., 2015b). Since we do not have ground-truth pixel labels, we cannot directly evaluate it. We do however have a set of ground-truth image labels l_i which we can compare against. We first aggregate the output pixel labels o_p over each image m_i into output image labels $o_i = \cup_{p\in m_i} o_p$. Then we define as loss the difference between the ground-truth label set l_i and the output label set o_i, measured by the Hamming distance between their binary vector representations

$$\mathcal{L}(o,l) = \sum_{i=1}^{I}\sum_{c=1}^{C}\frac{1}{I_c}|l_{i,c} - o_{i,c}| \tag{3.4}$$

where $l_{i,c} = 1$ if label c is in l_i, and 0 otherwise (analog for $o_{i,c}$). I is the total number of training images. I_c is the number of images having ground-truth label c, so the loss is weighted by the inverse frequency of class labels, measured at the image-level. Note how also in this setting the loss looks at performance after the maximization over classes and regions (Eq. 3.1).

Optimization. We want to minimize our loss functions over the calibration parameters a_c, b_c of all classes. This is hard, because the output pixel labels o_p depend on these parameters in a complex manner due to the max over classes and regions in

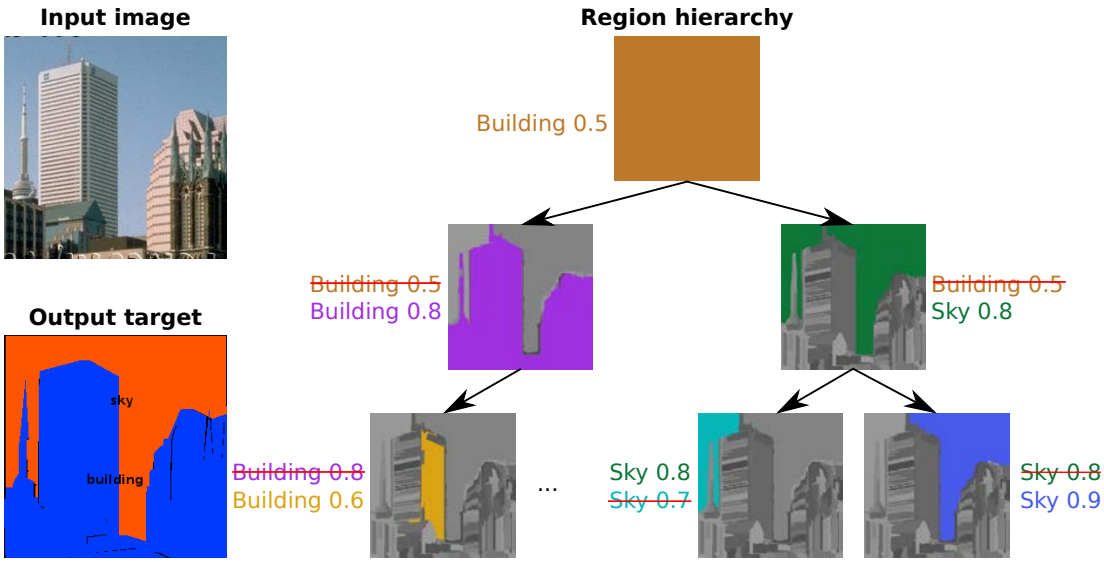


Figure 3.3: Our efficient evaluation algorithm uses the bottom-up structure of Selective Search region proposals to simplify the spatial maximization. We start from the root node and propagate the maximum score with its corresponding label down the tree. We label the image based on the labels of its superpixels (leaf nodes).

Eq. 3.1, and because of the set-union aggregation in the case of the weakly supervised loss. Therefore, we apply an approximate minimization algorithm based on coordinate descent. Coordinate descent is different from gradient descent in that it can be used on arbitrary loss functions that are not differentiable, as it only requires their evaluation for a given setting of parameters.

Coordinate descent iteratively applies line search to optimize the loss over a single parameter at a time, keeping all others fixed. This process cycles through all parameters until convergence. As initialization we use constant values ($a_c = -7$, $b_c = 0$). During line search we consider 10 equally spaced values (a_c in $[-12, -2]$, b_c in $[-10, 10]$).

This procedure is guaranteed to converge to a local minimum on the search grid. While this might not be the global optimum, in repeated trials we found the results to be rather insensitive to initialization. Furthermore, in our experiments the number of iterations was roughly proportional to the number of parameters.

Efficient evaluation. On a typical training set with $C = 30$ classes, our joint calibration procedure evaluates the loss thousands of times. Hence, it is important to evaluate pixel-level accuracy quickly. As the model involves a maximum over classes and a maximum over regions at every pixel, a naive per-pixel implementation would be prohibitively expensive. Instead, we propose an efficient technique that exploits the nature

of the Selective Search region proposals (Uijlings et al., 2013), which form a bottom-up hierarchy starting from superpixels. As shown in Fig. 3.3, we start from the region proposal that contains the entire image (root node). Then we propagate the maximum score over all classes down the region hierarchy. Eventually we assign to each superpixel (leaf nodes) the label with the highest score over all regions that contain it. This label is assigned to all pixels in the superpixel. To compute class accuracy, we normally need to compare each pixel label to the ground-truth label. However since we assign the same label to all pixels in a superpixel, we can precompute the ground-truth label distribution for each superpixel and use it as a lookup table. This reduces the runtime complexity for an image from $O(P_i \cdot R_i \cdot C)$ to $O(R_i \cdot C)$, where P_i and R_i are the number of pixels and regions in an image respectively, and C is the number of classes.

Why no Platt scaling. At this point the reader may wonder why we do not simply use Platt scaling (Platt, 1999) as is commonly done in many applications. Platt scaling is used to convert SVM scores to range $[0,1]$ using sigmoid functions, as in Eq. 3.2. However, in Platt scaling the parameters a_c, b_c are optimized for each class in isolation, ignoring class competition. The loss function $\mathcal{L}_c$ in Platt scaling is the cross-entropy function

$$\mathcal{L}_c(\sigma_c, l) = -\sum_r t_{r,c} \log(\sigma_c(x_r)) + (1 - t_{r,c}) \log(1 - \sigma_c(x_r)) \tag{3.5}$$

where N_+ is the number of positive samples, N_- the number of negative samples, and $t_{r,c} = \frac{N_+ + 1}{N_+ + 2}$ if $l_r = c$ or $t_{r,c} = \frac{1}{N_- + 2}$ otherwise; l_r is the region-level label. This loss function is inappropriate for semantic segmentation because it is defined in terms of accuracy of training samples, which are regions, rather than in terms of the final pixel-level accuracy. Hence it ignores the problem of overlapping regions. There is also no inverse frequency term to deal with class imbalance. We experimentally compare our method with Platt scaling in Sec. 3.4.

3.3.4 Implementation details

Region proposals. We use Selective Search (Uijlings et al., 2013) region proposals using a subset of the "Fast" mode: we keep the similarity measures, but we restrict the scale parameter k to 100 and the color-space to RGB. This leads to two bottom-up hierarchies of one initial oversegmentation (Felzenszwalb and Huttenlocher, 2004).

Features. We show experiments with features generated by two CNNs (AlexNet (Krizhevsky et al., 2012), VGG-16 (Simonyan and Zisserman, 2015)) using the Caffe

implementations (Jia, 2013). We use the R-CNN (Girshick et al., 2014) framework for AlexNet, and Fast R-CNN (Girshick, 2015) for VGG-16, in order to maintain high computational efficiency. Regions are described using all pixels in a tight bounding box. Since regions are free-form, Girshick et al. (2014) additionally propose to set pixels not belonging to the region to zero (i.e. not affecting the convolution). However, in our experiments this did not improve results so we do not use it. For the weakly supervised setting we use the CNNs pre-trained for image classification on ILSVRC 2012 (Russakovsky et al., 2015). For the fully supervised setting we finetune them on the training set of SIFT Flow (Liu et al., 2011) (i.e. the semantic segmentation dataset we experiment on). For both settings, following Girshick et al. (2014) we use the output of the FC6 layer of the CNN as features.

SVM training. Like Girshick et al. (2014) we set the regularization parameter C to a fixed value in all our experiments. The SVMs minimize the L2 loss for region classification. We use hard-negative mining to reduce memory consumption.

3.4 Experiments

Datasets. We evaluate our method on the challenging SIFT Flow dataset (Liu et al., 2011). It consists of 2488 training and 200 test images, pixel-wise annotated with 33 class labels. The class distribution is highly imbalanced in terms of overall region count as well as pixel count. As evaluation measure we use the popular class accuracy measure (Tighe and Lazebnik, 2010; Tighe et al., 2014; Yang et al., 2014; Farabet et al., 2013; Long et al., 2015; Sharma et al., 2015; Pinheiro and Collobert, 2014; Vezhnevets et al., 2012; Xu et al., 2015a; Zhang et al., 2015b). For both supervision settings we report results on the test set.

Fully supervised setting. Table 3.1 evaluates various versions of our model in the fully supervised setting, and compares to other works on SIFT Flow. Using AlexNet features and uncalibrated SVMs, our model achieves a class accuracy of 28.7%. If we calibrate the SVM scores with traditional Platt scaling results do not improve (27.7%). Using our proposed joint calibration to maximize class accuracy improves results substantially to 55.6%. This shows the importance of joint calibration to resolve conflicts between overlapping regions at multiple scales, to take into account competition between classes, and generally to optimize a loss mirroring the evaluation measure.

Fig. 3.4 (column "SVM") shows that larger background regions (i.e. *sky*, *building*)

Method	Class acc.
Byeon et al. (2015)	22.6%
Tighe and Lazebnik (2010)	29.1%
Pinheiro and Collobert (2014)	30.0%
Shuai et al. (2015)	39.7%
Tighe and Lazebnik (2013a)	41.1%
Kekeç et al. (2014)	45.8%
Sharma et al. (2014)	48.0%
Yang et al. (2014)	48.7%
George (2015)	50.1%
Farabet et al. (2013)	50.8%
Long et al. (2015)	51.7%
Sharma et al. (2015)	52.8%
Ours SVM (AlexNet)	28.7%
Ours SVM+PS (AlexNet)	27.7%
Ours SVM+JC (AlexNet)	55.6%
Ours SVM+JC (VGG-16)	**59.2%**
Souly et al. (2017)	46.7%
Lin et al. (2017)	53.4%
Cheng et al. (2017)	53.6%
Hu et al. (2017)	58.6%
Ngan Le et al. (2017)	61.0%
Fan and Ling (2018)	61.1%
Caesar et al. (2016b)	**64.0%**
Huang et al. (2017a)	**64.0%**

Table 3.1: Class accuracy in the fully supervised setting. We show results for our model on the test set of SIFT Flow using uncalibrated SVM scores (SVM), traditional Platt scaling (PS) and joint calibration (JC). We also compare to other works published before (top) and after (bottom) the initial release of our paper (Caesar et al., 2015).

Method	Class acc.
Vezhnevets et al. (2011)	14.0%
Vezhnevets et al. (2012)	21.0%
Zhang et al. (2014)	27.7%
Xu et al. (2014)	27.9%
Zhang et al. (2015b)	32.3%
Xu et al. (2015a)	35.0%
Xu et al. (2015a) (transductive)	41.4%
Ours SVM (AlexNet)	21.2%
Ours SVM+PS (AlexNet)	16.8%
Ours SVM+JC (AlexNet)	37.4%
Ours SVM+JC (VGG-16)	**44.8%**
Shi et al. (2017b)	23.8%
Shi et al. (2017b) (transductive)	31.2%

Table 3.2: Class accuracy in the weakly supervised setting setting. We show results for our model on the test set of SIFT Flow using uncalibrated SVM scores (SVM), traditional Platt scaling (PS) and joint calibration (JC). We also compare to other works published before (top) and after (bottom) the initial release of our paper (Caesar et al., 2015).

swallow smaller foreground regions (i.e. *boat*, *awning*). Many of these small objects become visible after calibration (column "SVM+JC"). This issue is particularly evident when working with overlapping regions. Consider a large region on a *building* which contains an *awning*. As the surface of the *awning* is small, the features of the large region will be dominated by the *building*, leading to strong classification score for the *building* class. When these are higher than the classification score for *awning* on the small *awning* region, the latter gets overwritten. Instead, this problem does not appear when working with superpixels (Boix et al., 2012). A superpixel is either part of the *building* or part of the *awning*, so a high scoring *awning* superpixel cannot be overwritten by neighboring *building* superpixels. Hence, joint calibration is particularly important when working with overlapping regions.

Using the deeper VGG-16 CNN the results improve further, leading to our final performance 59.2%. At the time of publication of our paper (Caesar et al., 2015), this result outperformed the state-of-the-art (Sharma et al., 2015) by 6.4%. Since then, the state-of-the-art has improved to 64.0% (Caesar et al., 2016b; Huang et al., 2017a)

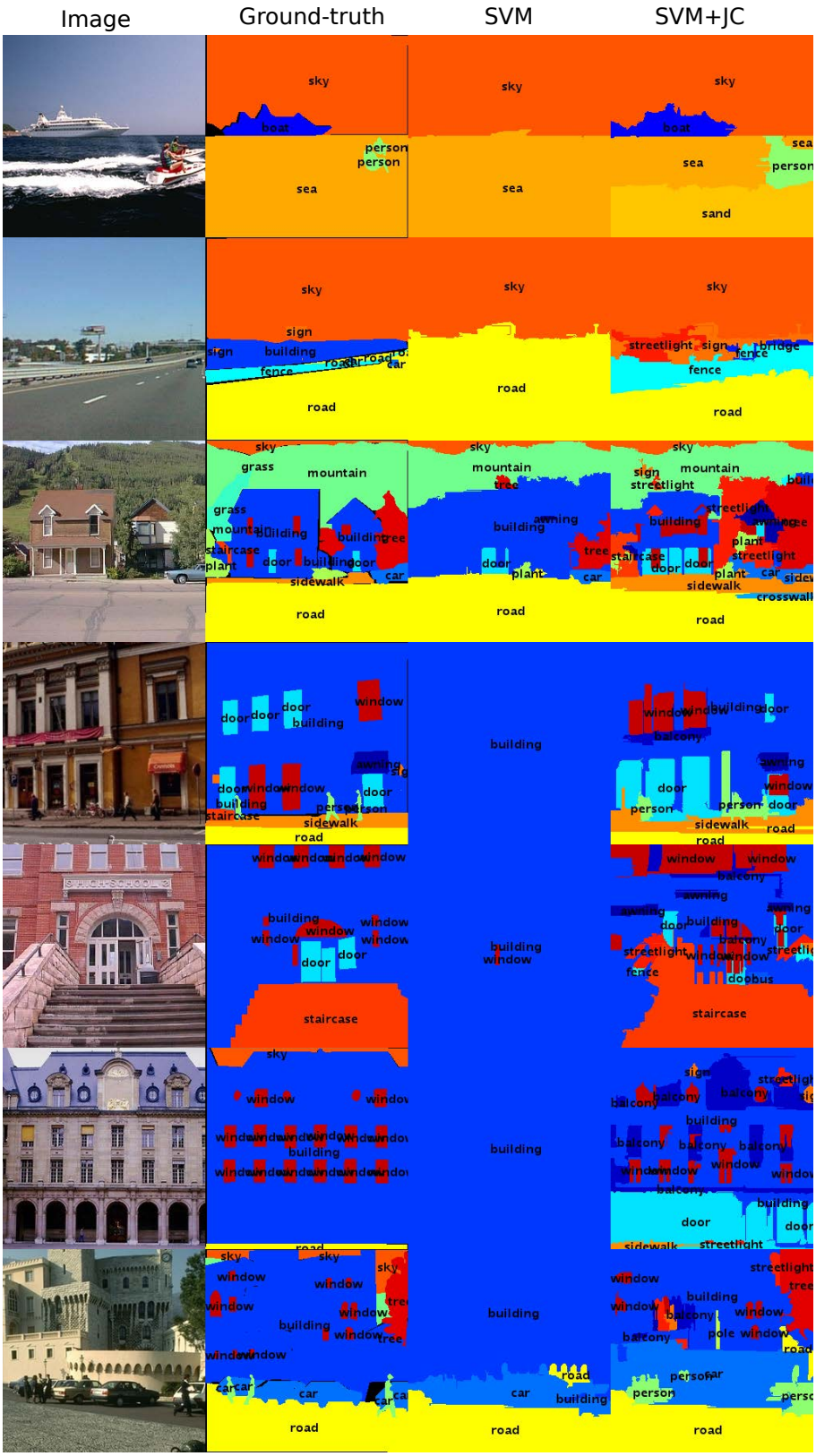


Figure 3.4: Fully supervised semantic segmentation on SIFT Flow. We present uncalibrated SVM results (SVM) and jointly calibrated results (SVM+JC), both with VGG-16.

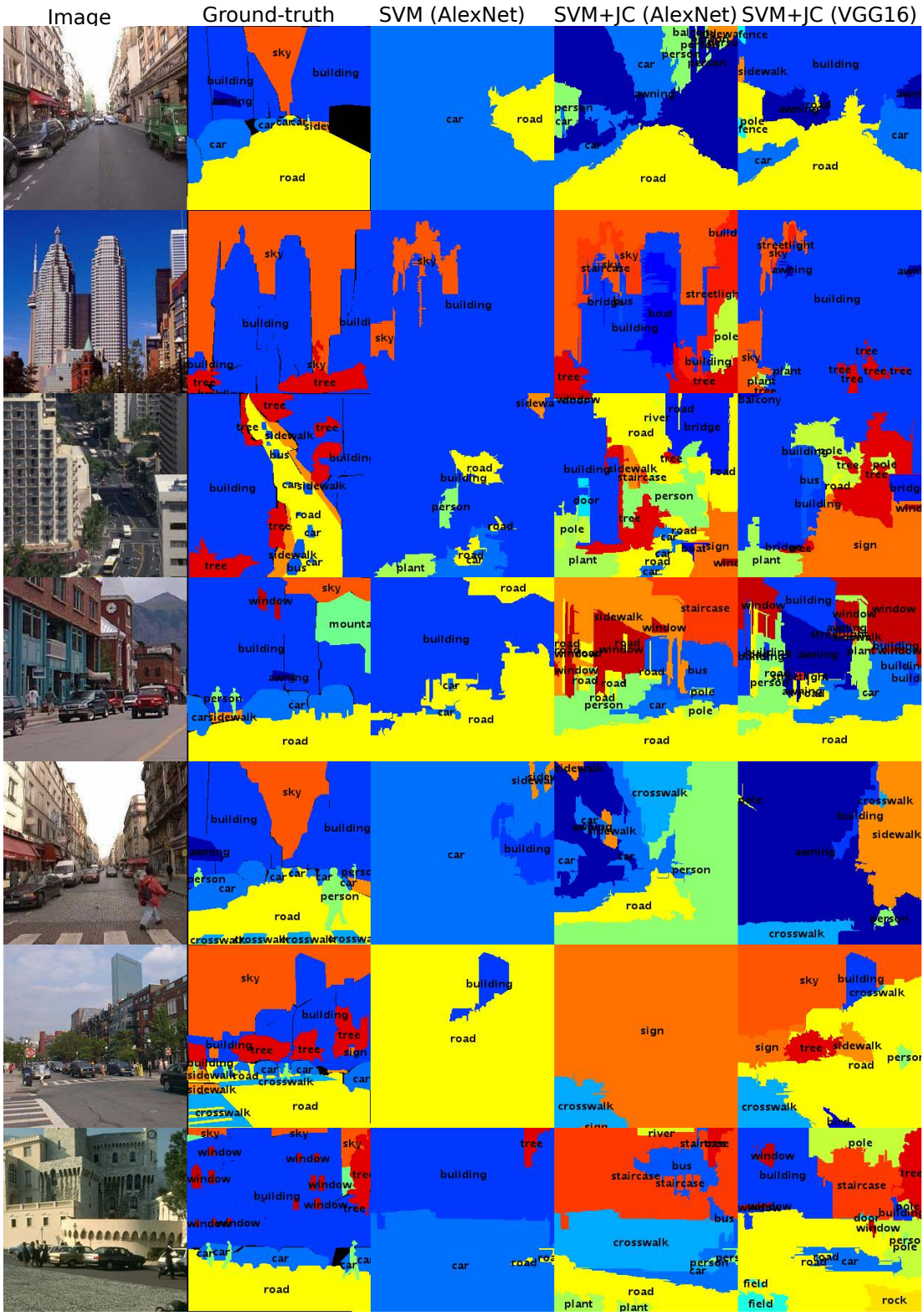


Figure 3.5: Weakly supervised semantic segmentation on SIFT Flow. We present uncalibrated SVM results (SVM) with AlexNet, jointly calibrated results (SVM+JC) with AlexNet, and with VGG-16.

Regions	Class acc.
Felzenszwalb and Huttenlocher (2004)	43.4%
Uijlings et al. (2013)	55.6%

Table 3.3: Comparison of single-scale (Felzenszwalb and Huttenlocher, 2004) and multi-scale (Uijlings et al., 2013) regions using SVM+JC (AlexNet).

Weakly supervised setting. Table 3.2 shows results in the weakly supervised setting. The model with AlexNet and uncalibrated SVMs achieves an accuracy of 21.2%. Using traditional Platt scaling the result is 16.8%, again showing it is not appropriate for semantic segmentation. Instead, our joint calibration almost doubles accuracy (37.4%). Using the deeper VGG-16 CNN results improve further to 44.8%.

Fig. 3.5 illustrates the power of our weakly supervised method. Again rare classes appear only after joint calibration. We compare to the state-of-the-art at the time of publication our our paper (Caesar et al., 2015): We outperform Xu et al. (2015a) (35.0%) by 9.8% in this setting. Xu et al. (2015a) additionally report results on the transductive setting (41.4%), where all (unlabeled) test images are given to the algorithm during training. Since the initial publication of our paper, Shi et al. (2017b) released new results, both in the regular (23.8%) and transductive setting (31.2%). Neither matches our best result.

Region proposals. To demonstrate the importance of multi-scale regions, we also analyze oversegmentations that do not cover multiple scales. To this end, we keep our framework the same, but instead of Selective Search (Uijlings et al., 2013) region proposals we used a single oversegmentation using the method of Felzenszwalb and Huttenlocher (2004), for which we optimize the scale parameter. As Table 3.3 shows, Selective Search regions outperform the oversegmentation by a good margin of 12.2% in the fully supervised setting. This confirms that overlapping multi-scale regions are superior to non-overlapping oversegmentations.

CNN finetuning. As described in 3.3.4 we finetune our network for detection in the fully supervised case. Table 3.4 shows that this improves results by 6.2% compared to using a CNN trained only for image classification on ILSVRC 2012.

Finetuned	Class acc.
no	49.4%
yes	55.6%

Table 3.4: Effect of CNN finetuning in the fully supervised setting using SVM+JC (AlexNet).

3.5 Discussion

In this section we present preliminary studies that we conducted to find suitable region proposals. We discuss why region proposal are suitable to find stuff and things. Then we give an overview of later works that have been inspired by our original paper (Caesar et al., 2015).

Region proposal evaluation. In preliminary studies we evaluated the quality of leading region proposal methods (Carreira and Sminchisescu, 2010; Uijlings et al., 2013; Arbeláez et al., 2014; Krähenbühl and Koltun, 2014; Rantalankila et al., 2014) on the SIFT Flow dataset (Liu et al., 2011). We compared the region proposals to regions (connected components) extracted from the segmentation ground-truth. The two metrics used were Mean Average Best Overlap (MABO) and recall at an IOU of 0.5. We found that Selective Search (Uijlings et al., 2013) outperforms all other methods on both metrics at ≥ 300 region proposals per image. We hypothesize that we need at least this number of region proposals for our method to cover all objects at multiple scales. Due to these observations we believe that Selective Search proposals are well suited for semantic segmentation.

Region proposals and stuff. Here we discuss why region proposal based methods are not just suitable for things, but also for stuff. Several authors point out that their proposal methods are suitable for stuff and things (Carreira and Sminchisescu, 2010; Uijlings et al., 2013), while others are specifically designed for thing classes (Endres and Hoiem, 2010; Zitnick and Dollár, 2014). We observe that suitable methods tend to 1) cover most of the image 2) take into account regions of homogeneous texture and 3) use diversification techniques to cover a variety of different proposals. Revisiting the region proposal evaluation experiments presented earlier, we found that the performance on both metrics is significantly higher for stuff than for thing classes. While the type of images and the granularity of the stuff and thing classes in SIFT Flow play a role here (see Sec. 6.4), this empirically shows that region proposals are indeed suitable

for stuff. Furthermore we note that the SIFT Flow dataset was specifically chosen for our experiment, as it covers stuff and thing classes and has a long tail label distribution which is typical for such datasets.

Later works inspired by ours. Two works appearing after our BMVC 2015 paper (Caesar et al., 2015) have been inspired by it. Zhu et al. (2016) present a system for weakly supervised facial analysis that uses a calibration technique for binary SVMs. They use a softmax function to enable the system to deal with multi-class and mixed-class problems. Bulo et al. (2017a) use a loss max-pooling layer to adaptively re-weight the contributions of each pixel based on the observed loss. This targets both inter- and intra-class imbalance.

3.6 Future work

In this section we present future work related to this chapter.

Multi-class classification. In Sec. 3.3 we use a separate one-vs-all SVM classifier for each class. Instead of training an SVM for each class, we could also train a single multiclass SVM. For a target label y and model parameters w_y, Crammer and Singer (2001) propose the following linear multi-class hinge loss $l(y)$:

$$l(y) = \max\left(0, 1 + \max_{t \neq y} w_t x - w_y x\right) \tag{3.6}$$

We assume that this significantly speedups training, while achieving a comparable level of performance. However, the use of a multi-class classifier does not address the problems described in Sec. 3.3.3 and joint calibration remains essential.

Speeding up joint calibration. A limitation of the work presented in this chapter is its poor scalability to a huge number of classes, which is very important for our overall project. One reason for this is the use of one-vs-all Support Vector Machines (SVM) which have to be trained separately for each class. In a first step to overcome this limitation, we replace SVMs trained on CNN features with the final layer activations of the CNN (Girshick, 2015). By moving from R-CNN (Girshick et al., 2014) to Fast R-CNN (Girshick, 2015), we are able to further speed up network training and testing by avoiding redundant computation on convolutional layers. We also speed up the joint calibration component that uses a sigmoid to calibrate per-class scores. In this work we suggested a coordinate descent scheme to optimize one sigmoid parameter at a

time. To parallelize this sequential algorithm, we let a number of threads optimize one sigmoid parameter each. The resulting solution is the one with the highest validation set performance of all partial solutions. This algorithmic modification allows for a more efficient implementation and results in a 400x speedup for large-scale datasets with about 200 classes. Future work could focus on further speeding up the region proposal extraction and network training.

Apply joint calibration to other methods. As future work one could apply the joint calibration technique as a post-processing step to current state-of-the-art methods, including fully convolutional approaches (see Sec. 2.1.3.1). For datasets with class imbalance this should yield significant improvements in metrics that give the same weight to all classes (see Sec. 2.1.1). Even in the case of a balanced dataset, optimizing the slope of the sigmoid function is crucial to take into account class competition and (if applicable) overlapping regions. As an analogy to end-to-end training, our method allows for approximate "to-the-end" training of any metric. This is particularly interesting for the mean IOU metric, where no closed form solution exists to compute the gradients (see Sec. 2.1.4.2).

Per-class evaluation metrics In Sec. 2.1.1 we discussed commonly used evaluation criteria for semantic segmentation. In Sec. 2.1.4.1 we introduce the challenges related to class competition and class imbalance. In this chapter we presented a method that is able to calibrate any semantic segmentation method for the final evaluation criterion. But what is the performance of the underlying method *independent of its* calibration? Can we compute performance metrics that do not take into account class imbalance and class competition? It turns out that we can use existing metrics from other tasks – such as mean Average Precision (mAP) – on the pixel-level. To compute the AP for a particular class, we sort all pixels by their score for that class and compute the precision at each score threshold. Unfortunately this is computationally expensive. For the test set of our COCO-Stuff dataset (see Sec. 6), this requires sorting and storing about 10^{10} scores per class. Online sorting algorithms and random subsampling may be able to speedup this process and find suitable approximations. Furthermore, precision can be computed per image and then aggregated over the entire dataset, to avoid having to store the scores for every pixel. Future work could evaluate whether this is a useful metric for semantic segmentation and how to efficiently compute it for large datasets. This approach will make the submissions of semantic segmentation challenges more comparable, as the metric cannot be altered by class reweighting via cross-validation.

It also allows the creator of a new semantic segmentation method to focus on one class at a time, just as in object detection. Similarly, it may allow us to design and evaluate branches of a network that are specific to stuff *or* things (Zhou et al., 2017b; Brahmbhatt et al., 2017), without the performance of one set of classes being altered by the other.

3.7 Conclusion

We addressed three common problems in semantic segmentation based on region proposals: (1) overlapping regions yield conflicting class predictions at the pixel-level; (2) class imbalance leads to classifiers unable to detect rare classes; (3) one-vs-all classifiers do not take into account competition between multiple classes. We proposed a joint calibration strategy which optimizes a loss defined over the final pixel-level output labeling of the model, after maximization over classes and regions. This tackles all three problems: *joint* calibration deals with multi-class predictions, while our loss explicitly deals with class imbalance and is defined in terms of pixel-wise labeling rather than region classification accuracy. As a result we take into account conflict resolution between overlapping regions. In the fully supervised setting, our method outperformed the state-of-the-art on the SIFT Flow dataset (Liu et al., 2011) at the time of publication (Caesar et al., 2015), but it has been surpassed since then. In the weakly supervised setting, our work still outperforms the state-of-the-art on SIFT Flow today.

Chapter 4

Region-based semantic segmentation with end-to-end training

4.1 Introduction

We address the task of semantic segmentation, labeling each pixel in an image with a semantic class. Currently, there are two main paradigms: classical region-based approaches (Boix et al., 2012; Caesar et al., 2015; Carreira et al., 2012; Dai et al., 2015b; George, 2015; Girshick et al., 2014; Hariharan et al., 2014; Li et al., 2013; Mostajabi et al., 2015; Plath et al., 2009; Sharma et al., 2014, 2015; Tighe and Lazebnik, 2010, 2013a; Tighe et al., 2014; Yang et al., 2014; Mottaghi et al., 2014) and, inspired by the Convolutional Neural Network (CNN) revolution, fully convolutional approaches (Chen et al., 2015a; Dai et al., 2015a; Eigen and Fergus, 2015; Farabet et al., 2013; Hariharan et al., 2015; Long et al., 2015; Noh et al., 2015; Pinheiro and Collobert, 2014; Zheng et al., 2015).

In the fully convolutional approach the idea is to directly learn a mapping from image pixels to class labels using a CNN (see Sec. 2.1.3.1). This results in a single model, directly optimized end-to-end for the task at hand, including the intermediate image representations (i.e. the hidden layers in the network). However, the spatial support on which predictions are based are fixed-size square patches of the input image. Intuitively, this is suboptimal since: (I) Objects are free-form rather than square, so ideally the intermediate representations should take this into account. (II) Objects do not have a fixed size, but occur at various scales. Hence many patches either cover pieces of multiple objects and mix their representations, or cover a piece of an object, which is sometimes difficult to recognize in isolation (e.g. a patch on the belly of a cow).

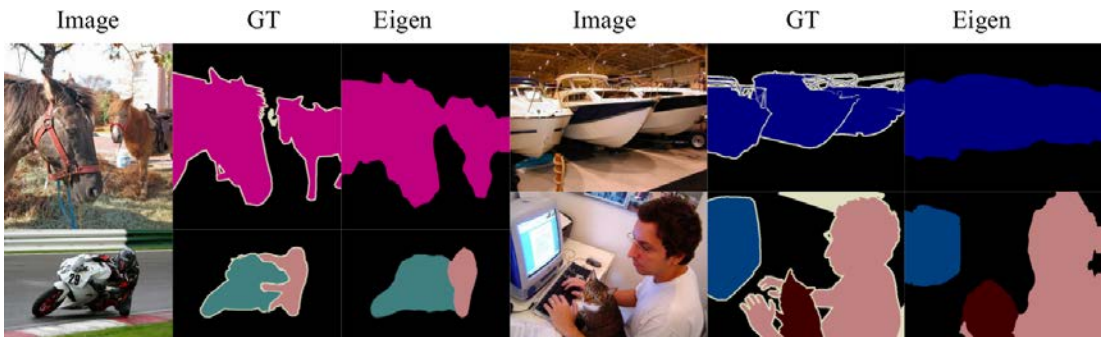


Figure 4.1: Fully convolutional methods typically produce fuzzy object boundaries, as illustrated here by examples from Eigen and Fergus (2015).

An additional problem is that fully convolutional methods typically make predictions at a coarse resolution, which often results in inaccurate object boundaries (Chen et al., 2015a; Eigen and Fergus, 2015; Farabet et al., 2013; Hariharan et al., 2015; Noh et al., 2015; Zheng et al., 2015). Fig. 4.1 illustrates this on example outputs of Eigen and Fergus (2015).

In the region-based approach, the image is first segmented into coherent regions, which are described by image features (Boix et al., 2012; Caesar et al., 2015; Carreira et al., 2012; Dai et al., 2015b; George, 2015; Girshick et al., 2014; Hariharan et al., 2014; Li et al., 2013; Mostajabi et al., 2015; Plath et al., 2009; Sharma et al., 2014, 2015; Tighe and Lazebnik, 2010, 2013a; Tighe et al., 2014; Yang et al., 2014) (see Sec. 2.1.3.2). Typically many regions are extracted at multiple scales (Caesar et al., 2015; Carreira et al., 2012; Dai et al., 2015b; Girshick et al., 2014; Hariharan et al., 2014; Li et al., 2013; Plath et al., 2009; Sharma et al., 2014, 2015), capturing complete objects and canonical object parts (e.g. faces) which in turn facilitates recognition. Furthermore, the segmentation process delivers regions which follow object boundaries quite well. However, these methods generally first extract region features and then train a classifier optimized for classifying regions rather than for the final semantic segmentation criterion (i.e. pixel-level labeling) (Caesar et al., 2015; Carreira et al., 2012; Dai et al., 2015b; Girshick et al., 2014; Hariharan et al., 2014; Li et al., 2013; Plath et al., 2009). Hence, while these methods benefit from the power of multi-scale, overlapping regions, they cannot be trained end-to-end for semantic segmentation.

In this work we want the best of both worlds. We propose a region-based semantic segmentation model with an accompanying end-to-end training scheme based on a CNN architecture (Fig. 4.2c). To enable this we introduce a novel, differentiable region-to-pixel layer which maps from regions to image pixels. We insert this layer

before the final classification layer, enabling the use of a pixel-level loss which allows us to directly optimize for semantic segmentation. Conceptually, our region-to-pixel layer ignores regions which have low activations for all classes and which therefore do not impact the final labeling. This is in contrast to all multi-scale region-based methods where such regions incorrectly affect training (Caesar et al., 2015; Carreira et al., 2012; Dai et al., 2015b; Girshick et al., 2014; Hariharan et al., 2014; Li et al., 2013; Plath et al., 2009). Additionally, we introduce a differentiable Region-of-Interest pooling layer which operates on the final convolutional layer in the spirit of Fast R-CNN (Girshick, 2015), but which is adapted for free-form regions like (Dai et al., 2015b; Sharma et al., 2014, 2015). Note how we use region proposals from a separate pre-processing stage. By end-to-end we mean training all parameters for the final pixel-level loss, rather than for region classification (see Sec. 2.1.4.2).

To summarize, our contributions are: (1) We introduce a region-to-pixel layer which enables full end-to-end training of semantic segmentation models based on multi-scale overlapping regions. (2) We introduce a Region-of-Interest pooling layer specialized for free-form regions. (3) At the time of publication of our ECCV 2016 paper (Caesar et al., 2016b), we obtained state-of-the-art results on the SIFT Flow and the PASCAL Context datasets, in terms of class accuracy. Our approach delivers crisp object boundaries, as demonstrated in Fig. 4.5 and Sec. 4.4.3. This chapter is a modified version of our paper published at the European Conference on Computer Vision 2016 (Caesar et al., 2016b). In Sec. 4.5 we contextualize this work in the literature and in Sec. 4.6 we present future work. We release the source code of our method at: https://github.com/nightrome/matconvnet-calvin

Relation to Chapter 3. In Chapter 3 we presented a technique to calibrate a region-based method "to-the-end" for the pixel-level evaluation criterion. In this chapter we improve upon this approach by mapping from pixels to regions and allowing for end-to-end training of the *entire* framework. If trained for a suitable loss, both methods takes into account three problems of region-based semantic segmentation: overlapping regions, class imbalance and class competition (see Sec. 3.1). This improvement allows us to drastically simplify our method and remove Support Vector Machines and the joint calibration step. It also leads to a significantly higher performance (see Sec. 4.4.2).

4.2 Related Work

4.2.1 Region-based semantic segmentation

Region-based semantic segmentation methods first extract free-form regions (Carreira and Sminchisescu, 2010; Uijlings et al., 2013; Endres and Hoiem, 2010; Arbeláez et al., 2014) from an image and describe them with features. Afterwards a region classifier is trained. At test time, region-based predictions are mapped to pixels, usually by labeling a pixel according to the highest scoring region that contains it. Region-based methods generally yield crisp object boundaries (Boix et al., 2012; Caesar et al., 2015; Carreira et al., 2012; Dai et al., 2015b; George, 2015; Girshick et al., 2014; Hariharan et al., 2014; Li et al., 2013; Mostajabi et al., 2015; Plath et al., 2009; Sharma et al., 2014, 2015; Tighe and Lazebnik, 2010, 2013a; Tighe et al., 2014; Yang et al., 2014; Mottaghi et al., 2014). Fig. 4.2b shows a prototypical architecture for such an approach (which we modernized by basing it on Fast R-CNN (Girshick, 2015)). We discuss several aspects below.

Multi-scale vs single-scale regions. Several region-based methods use an oversegmentation to create small, non-overlapping regions (Boix et al., 2012; George, 2015; Mostajabi et al., 2015; Tighe and Lazebnik, 2010, 2013a; Tighe et al., 2014; Yang et al., 2014). Intuitively however, objects are more easily recognized as a whole than by looking at small object parts individually. The inherent multi-scale aspect of recognition is adequately captured in many recent works using multi-scale, overlapping regions (Caesar et al., 2015; Carreira et al., 2012; Dai et al., 2015b; Girshick et al., 2014; Hariharan et al., 2014; Li et al., 2013; Plath et al., 2009; Sharma et al., 2014, 2015).

Training criterion. The final criterion is pixel-level prediction of class labels. However, we use overlapping regions whose predictions are in competition with each other on the pixel level. Typically, many methods initially ignore this by simply training a classifier to predict region labels (Caesar et al., 2015; Carreira et al., 2012; Dai et al., 2015b; Girshick et al., 2014; Hariharan et al., 2014; Li et al., 2013; Plath et al., 2009), which is *different* from semantic segmentation (Sec. 4.3.1). At test time one labels a pixel by simply taking the maximum over all regions containing it (Caesar et al., 2015; Carreira et al., 2012; Dai et al., 2015b; Girshick et al., 2014; Hariharan et al., 2014). A few works partially addressed the mismatch between training and test time through a post-processing stage using graphical models (Li et al., 2013; Plath et al., 2009) or

by joint calibration (Caesar et al., 2015). However, none of them does full end-to-end training.

Region representations. Most older works use hand-crafted region-based features (Boix et al., 2012; Carreira et al., 2012; George, 2015; Li et al., 2013; Plath et al., 2009; Tighe and Lazebnik, 2010, 2013a; Tighe et al., 2014; Yang et al., 2014) often based on (Carreira et al., 2012; Tighe and Lazebnik, 2010). More recent works instead use the top convolutional layers of a pre-trained CNN (e.g. Krizhevsky et al. (2012); Simonyan and Zisserman (2015)) as feature representations (Caesar et al., 2015; Dai et al., 2015b; Girshick et al., 2014; Hariharan et al., 2014; Mostajabi et al., 2015; Sharma et al., 2014, 2015). These representations can be free-form respecting the shape of the region (Dai et al., 2015b; Girshick et al., 2014; Hariharan et al., 2014; Mostajabi et al., 2015; Sharma et al., 2014, 2015) or simply represent the bounding box around the region (Caesar et al., 2015; Girshick et al., 2014). Furthermore regions can be cropped out from the image before being fed to the network (Girshick et al., 2014; Hariharan et al., 2014; Mostajabi et al., 2015) or one can create region representations from a convolutional layer (Dai et al., 2015b; Sharma et al., 2014, 2015), termed Region-of-Interest (ROI) pooling (Girshick, 2015) or Convolutional Feature Masking (Dai et al., 2015b). CNN representations become more powerful when further trained for the task. In Caesar et al. (2015); Girshick et al. (2014); Hariharan et al. (2014) they train CNNs, but for the task of region classification, not for semantic segmentation.

4.2.2 Fully convolutional semantic segmentation

Fully convolutional methods learn a direct mapping from pixels to pixels, which was pioneered by Shotton et al. (2009) in the pre-CNN era. Early CNN-based approaches train relatively shallow end-to-end networks (Farabet et al., 2013; Pinheiro and Collobert, 2014), whereas more recent works use much deeper networks whose weights are initialized by pre-training on the ILSVRC (Russakovsky et al., 2015) image classification task (Chen et al., 2015a; Dai et al., 2015a; Eigen and Fergus, 2015; Hariharan et al., 2015; Long et al., 2015; Noh et al., 2015; Zheng et al., 2015). The main insight to adapt these networks for semantic segmentation was to re-interpret the classification layer as 1x1 convolutions (Sermanet et al., 2014; Long et al., 2015). A prototypical model is illustrated in Fig. 4.2a.

Square receptive fields. All fully convolutional methods have receptive fields of fixed shape (square) (Chen et al., 2015a; Dai et al., 2015a; Eigen and Fergus, 2015; Farabet et al., 2013; Hariharan et al., 2015; Long et al., 2015; Noh et al., 2015; Pinheiro and Collobert, 2014; Zheng et al., 2015). However, since objects are free-form this may be suboptimal.

Multi-scale. Recognition is a multi-scale problem, which is addressed by using two strategies: *(I) Multi-scale representations.* Using skip-layer connections (Bishop, 1995; Ripley, 1996), representations from different convolutional layers can be combined (Eigen and Fergus, 2015; Hariharan et al., 2015; Long et al., 2015; Pinheiro and Collobert, 2014). This leads to multi-scale representations of a predetermined size. *(II) Multi-scale application.* In Hariharan et al. (2015); Noh et al. (2015) they train and apply their method on multi-scale, rectangular image crops. However, this results in a mismatch between training time, where each crop is considered separately, and test time, where predictions of multiple crops are combined before evaluation.

Fuzzy object boundaries. It is widely acknowledged that fully convolutional approaches yield rather fuzzy object boundaries (Chen et al., 2015a; Eigen and Fergus, 2015; Farabet et al., 2013; Hariharan et al., 2015; Noh et al., 2015; Zheng et al., 2015). A variety of strategies address this. *(I) Multi-scale.* The multi-scale methods discussed above (Eigen and Fergus, 2015; Hariharan et al., 2015; Long et al., 2015; Noh et al., 2015; Pinheiro and Collobert, 2014) include a fine scale resulting in improved object boundaries. *(II) Conditional Random Fields (CRFs).* CRFs are a classical tool to refine pixel-wise labelings and are used as post-processing step by (Chen et al., 2015a; Farabet et al., 2013; Noh et al., 2015; Zheng et al., 2015). Notably, Zheng et al. (2015) reformulate the CRF as a recurrent neural network (CRF-RNN) enabling them to train the whole network including convolutional layers in an end-to-end fashion. *(III) Post-processing by region proposals.* Finally, Farabet et al. (2013) averages pixel-wise network outputs over regions from an oversegmentation.

4.2.3 Our method.

We propose a model based on free-form, multi-scale, overlapping regions. We design a partially differentiable region-to-pixel layer enabling end-to-end training for semantic segmentation. Additionally we introduce a ROI pooling layer, which is free-form (Dai et al., 2015b; Sharma et al., 2014, 2015) yet also differentiable (Girshick, 2015).

4.3 Method

Sec. 4.3.1 presents a baseline model that is representative for modern region-based semantic segmentation (Caesar et al., 2015; Dai et al., 2015b; Girshick et al., 2014; Hariharan et al., 2014) (Fig. 4.2b), and explains its shortcomings. Sec. 4.3.2-4.3.5 present our framework, which addresses these issues (Fig. 4.2c).

4.3.1 Region-based semantic segmentation

Model. Fig. 4.2b presents a typical region-based semantic segmentation architecture. It modernizes Caesar et al. (2015); Dai et al. (2015b); Girshick et al. (2014); Hariharan et al. (2014) by using the Region-of-Interest pooling layer of Girshick (2015). We use this model as a baseline in our experiments (Sec. 4.4).

The input to the network are images and free-form regions (Uijlings et al., 2013). The image is fed through several convolutional layers. A Region-of-Interest pooling layer (Girshick, 2015) creates a feature representation of the tight bounding boxes around each region. These region features are then fed through several fully connected layers and a classification layer, followed by a softmax, resulting in region-level predictions. At test time, these predictions are mapped from regions to pixels: each pixel p is assigned the label o_p with the highest probability over all classes and all regions containing p:

$$o_p = \operatorname*{argmax}_c \max_{r \ni p} \operatorname*{softmax}_c S_{r,c} \tag{4.1}$$

Here $S_{r,c}$ denotes the classifier scores for region r and class c (i.e. activations of the classification layer).

Training. The training procedure searches for the network parameters that minimize a cross-entropy log-loss $\mathcal{L}$ over regions:

$$\mathcal{L} = -\sum_c \frac{1}{R} \sum_{r=1}^{R} y_{r,c} \log \operatorname*{softmax}_c S_{r,c} \tag{4.2}$$

Here R indicates the number of regions in the training set and $y_{r,c} \in \{0,1\}$ is a ground-truth label indicating whether region r has label c. The network is trained with Stochastic Gradient Descent (SGD) with momentum. To update the network weights, one needs to compute the partial derivatives of the loss with respect to the weights. These derivatives depend on the partial derivatives of the loss with respect to the outputs of the respective layer.

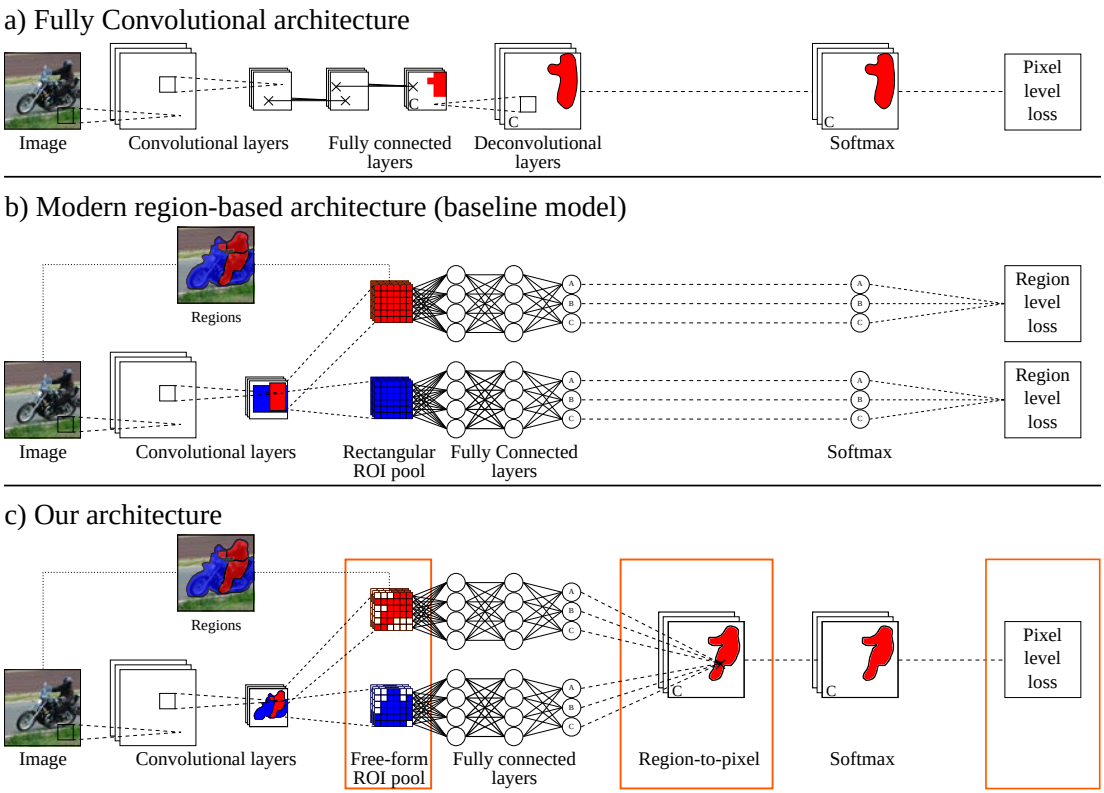


Figure 4.2: Overview of three semantic segmentation architectures. We show only layers with trainable parameters, softmax and loss layers. We omit all pre- and post-processing steps. a) shows the class of fully convolutional architectures that are end-to-end trainable, but do not have regions. b) shows the baseline model, representative for modern region-based architectures. It is not end-to-end trainable for the desired pixel labeling criterion. c) shows our suggested architecture, which pools activations of each region in a free-form manner, maps the region-level predictions to pixels and computes a loss at the pixel level. Hence our method combines regions and end-to-end training. Our main contributions are highlighted by orange boxes.

Problems. A first problem arises because the softmax is applied before pixel assignment in Eq. 4.1: (I) regions with low but highly varying activation scores are unsure about the class, but can still yield high probabilities due to the softmax. Intuitively, this means that such non-discriminative regions can wrongly affect the final prediction.

More importantly, since $\max_{r \ni p}$ occurs at test time (Eq. 4.1), but not at training time (Eq. 4.2), the pixel-wise evaluation criterion at test time is *different* from the region-level optimization criterion at training time. This has several consequences: (II) While during training *all* regions affect the network, at test time most regions are ignored. (III) It is unclear what are good region training examples for achieving good performance at test time: Are positive examples only ground-truth regions? Or should we use also region proposals which partially overlap with the ground-truth? And with what threshold? What overlap are negative proposals allowed to have to count as negative examples? Hence one has to select overlap thresholds for positive and negative examples empirically using test time evaluations. (IV) Regions with different size have the same weight. (V) The network is not trained end-to-end for semantic segmentation, but for the intermediate task of region classification instead. Hence both the classification layer and the representation layers will be suboptimal for the actual semantic segmentation task.

4.3.2 End-to-end training for region-based semantic segmentation

Model. To combine the paradigms of region-based semantic segmentation and end-to-end training, we map from regions to pixels as in Eq. 4.1, but *before* the softmax and loss computation on a pixel-level:

$$o_p = \operatorname*{argmax}_c \operatorname*{softmax}_c \max_{r \ni p} S_{r,c} \tag{4.3}$$

This region-to-pixel layer is shown in Fig. 4.2c. It brings two benefits. At training time, having the region-to-pixel layer before the loss enables optimizing a pixel-level loss. Furthermore, having the region-to-pixel layer before the softmax ensures that the class score for each pixel is taken from the region with the highest activation score, hence each class can be recognized at its appropriate scale.

Training. In Eq. 4.2 the baseline model computes a cross-entropy log-loss on the region-level. Here instead we compute a log-loss on the pixel-level:

$$\mathcal{L} = -\sum_c \frac{1}{P} \sum_{p=1}^{P} y_{p,c} \log \operatorname*{softmax}_c S_{p,c} \tag{4.4}$$

Here P indicates the number of pixels in the training set, $y_{p,c} \in \{0,1\}$ indicates whether pixel p has ground-truth label c, and $S_{p,c} = \max_{r \ni p} S_{r,c}$ is the pixel-level score for class c. As in Sec. 4.3.1 we train the network using SGD. To determine the partial derivatives of our region-to-pixel layer, we observe that it does not have any weights and we only need to compute the subgradients of the loss with respect to the region-level scores $S_{r,c}$:

$$\frac{\partial \mathcal{L}}{\partial S_{r,c}} = \sum_{p \in r \mid r = \operatorname{argmax}_{r' \ni p'} S_{r',c}} \frac{\partial \mathcal{L}}{\partial S_{p,c}} \tag{4.5}$$

This means that for each class we map each pixel-level gradient to the region with the highest score among all regions that include the pixel. If multiple pixels per class map to the same region, their gradient contributions are summed.

Advantages. Our model addresses all problems raised in Sec. 4.3.1: (I) Pixels are always labeled according to the relevant region with the highest activation score for that class. (II) Regions which do not affect the pixel-level prediction are ignored during training. (III) Since we evaluate pixels there is no need to assign class labels to regions for training. (IV) The pixel-level loss is agnostic to different sizes of region proposals. (V) We train our method end-to-end for the actual semantic segmentation criterion, resulting in properly optimized classifiers and region representations.

4.3.3 Pooling on free-form regions

Model. While the baseline model classifies free-form regions, their feature representations are computed on the bounding box. This is suboptimal as the regions can take highly irregular shapes. We propose here a free-form Region-of-Interest (ROI) pooling layer which computes representations taking into account only pixels actually in the region (Fig. 4.2c):

$$S^R_{i,d,r} = \max_{j \mid \phi(j) = i,\, \delta_{j,r} = 1} S^C_{j,d} \tag{4.6}$$

Here $S^R_{i,d,r}$ is the ROI pooling activation for ROI coordinate i, channel d and region r. For each ROI coordinate and channel we maximize over the corresponding coordinate j in the convolutional map $S^C_{j,d}$, considering only points inside the region, i.e. $\delta_{j,r} = 1$. The mapping ϕ from convolutional map coordinates to ROI ones is done as in Girshick (2015); He et al. (2014), but operates on a free-form region rather than a bounding box.

Training. During the forward pass the highest scoring convolutional map coordinate $\pi(i,d,r)$ for each ROI coordinate and channel is computed as:

$$\pi(i,d,r) = \underset{j \mid \phi(j)=i,\, \delta_{j,r}=1}{\operatorname{argmax}} S^C_{j,d} \tag{4.7}$$

We use the technique of Girshick (2015) to backpropagate through the pooling layer, computing the subgradients of the loss with respect to each coordinate in the last convolutional feature map. For each coordinate and channel in the ROI pooling output of a region, the gradients are passed to the convolutional feature map coordinate with the highest activations during the forward pass:

$$\frac{\partial \mathcal{L}}{\partial S^C_{j,d}} = \sum_r \sum_{i \mid \pi(i,d,r)=j} \frac{\partial \mathcal{L}}{\partial S^R_{i,d,r}} \tag{4.8}$$

Advantages. Our free-form region representations focus better on the region of interest, leading to purer representations. Additionally, they solve a common problem with bounding boxes: when objects of two classes occur in a contained-in relationship (i.e. a bird in the sky), their free-form region proposals degenerate to the same bounding box. Hence higher network layers will receive two identical feature vectors for two different regions covering different classes. This leads to confusion between the two classes, both at training and test time.

Incorporating region context. Several works have shown that including local region context improves semantic segmentation (Dai et al., 2015b; Girshick et al., 2014; Hariharan et al., 2015), as many object classes appear in a characteristic context (e.g. a lion is more likely to occur in the savanna than indoors). We take into account region context by performing ROI pooling also on their bounding boxes using Girshick (2015). Hence we combine the advantages of using context with the advantages of free-form region representations.

As shown in Fig. 4.3, we combine region and bounding box representations using one of two strategies: *(I) Tied weights.* We use the same fully connected layers with the same weights for both region and bounding box representations and add the corresponding activations scores after the classification layers. Hence the number of network parameters stays the same and the region and its context are handled identically. *(II) Separate weights.* We concatenate the representations of region and bounding box before applying the consecutive fully connected layers. This strategy roughly doubles the total number of weights of our overall network architecture, but can develop separate classifiers for each representation.

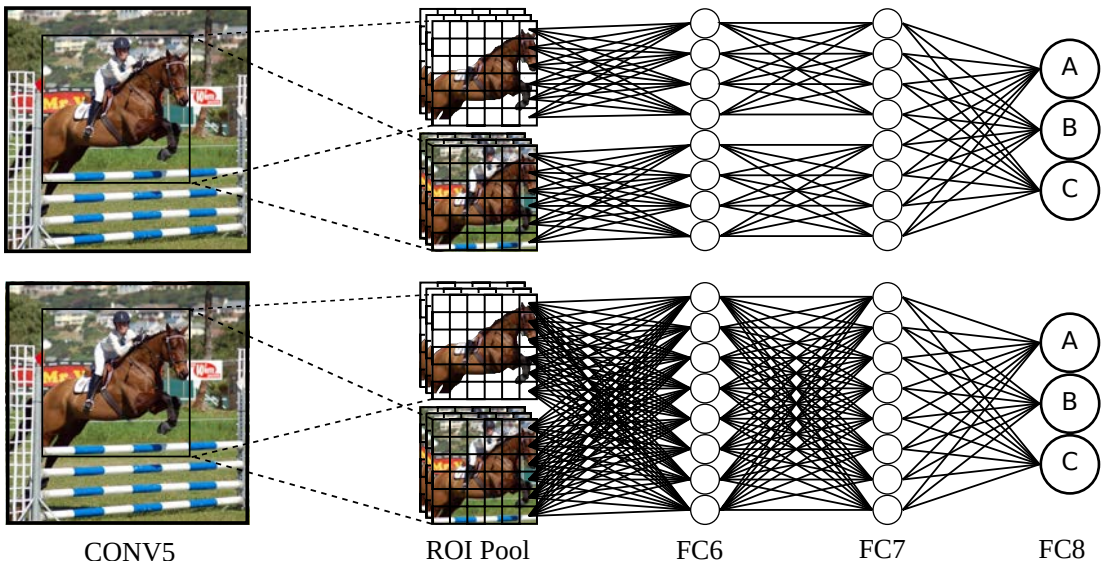


Figure 4.3: We combine free-form region representations, which focus on the appearance of the region itself, with bounding box based representations, which also capture context. We combine them using tied weights (above) and separate weights (below).

Since ROI pooling on bounding boxes and free-form regions are both differentiable, the combined representations are also differentiable and allow for end-to-end training. We compare all representations experimentally (Table 4.4).

Relation to Girshick (2015); Girshick et al. (2014); Dai et al. (2015b). Girshick (2015b) use a differentiable ROI pooling layer in Fast R-CNN for bounding boxes only. Girshick et al. (2014) use free-form regions in R-CNN for semantic segmentation. For each region proposal they set the color values of the background pixels to zero. In our scheme we do not alter the image pixels of the input but pool exclusively over pixels inside the region. Dai et al. (2015b) perform Convolutional Feature Masking on the last convolutional feature map, followed by a Spatial Pyramid Pooling layer (He et al., 2014), but did not backpropagate through this layer. Both Dai et al. (2015b); Girshick et al. (2014) combined free-form and bounding box representations. Only Dai et al. (2015b) took representations after the convolutional layers, but their model was not able to perform backpropagation. Both Dai et al. (2015b); Girshick et al. (2014) optimized for region classification instead of semantic segmentation.

4.3.4 Attention to rare classes

Pixel-level class frequencies are often imbalanced (Caesar et al., 2015; Mostajabi et al., 2015; Sharma et al., 2014, 2015; Tighe and Lazebnik, 2013a; Yang et al., 2014; Eigen and Fergus, 2015; Farabet et al., 2013; Kekeç et al., 2014; Byeon et al., 2015; Shuai et al., 2015). This is typically addressed by using an inverse class frequency weighting $\frac{1}{P_c}$ (Mostajabi et al., 2015; Sharma et al., 2014; Eigen and Fergus, 2015; Farabet et al., 2013). Since we have a pixel-level loss, we can simply plug this into Eq. 4.4. However, we found that rare classes lead to large weight updates resulting in exploding gradients and numerical problems. To avoid these issues, we re-normalize the inverse frequency weights by a factor Z so that the total sum of weights for each training image is 1:

$$\frac{1}{Z}\sum_{c}\frac{1}{P_c}\sum_{p=1}^{P} y_{p,c} = 1. \tag{4.9}$$

4.3.5 Efficient evaluation of the pixel-level loss

Evaluating the loss for each pixel separately is computationally expensive and redundant, because different pixels belonging to the same highest scoring region for a class are assigned the same score $S_{r,c}$. Hence we partition the set of region proposals for a training image into a set of non-overlapping, single-class regions using the ground-truth. We then reformulate Eq. 4.4 into an equivalent loss in terms of these regions. This reduces the cost of loss evaluation by a factor 1000.

4.4 Experiments

4.4.1 Setup

Datasets. We evaluate our method on two challenging datasets: SIFT Flow (Liu et al., 2011) and PASCAL Context (Mottaghi et al., 2014). SIFT Flow contains 33 classes in 2688 images. The dataset is known for its extreme class imbalance (Liu et al., 2011; Farabet et al., 2013; Eigen and Fergus, 2015). We use the provided fixed split into 2488 training images and 200 test images.

PASCAL Context provides complete pixel-level annotations for both things and stuff classes in the popular PASCAL VOC 2010 (Everingham et al., 2015) dataset. It contains 4998 training and 5105 validation images. As there is no dedicated test set available, we use the validation images exclusively for testing. We use the 59 classes

plus background commonly used in the literature (Dai et al., 2015b,a; Long et al., 2015; Zheng et al., 2015).

Evaluation measures. For a detailed overview of the evaluation measures, we refer the reader to Sec. 2.1.1.

Network. We use the popular classification network VGG-16 (Simonyan and Zisserman, 2015) pre-trained for image classification on ILSVRC 2012 (Russakovsky et al., 2015). We use the layers up to CONV5, discarding all higher layers, as the basis of our network. We then append a free-form ROI pooling layer (Sec. 4.3.3), a region-to-pixel layer, a softmax layer and pixel-level loss (Sec. 4.3.2, Fig. 4.2c). To include local context, we combine region and entire bounding box using separate weights (Sec. 4.3.3).

Regions. We use Selective Search (Uijlings et al., 2013), which delivers three sets of region proposals, one per color space (RGB, HSV, LAB). During training we change the set of region proposals in each mini-batch to have a more diverse set of proposals without the additional overhead of having three times as many regions. We use region proposals with a minimum size of 100 pixels for SIFT Flow, and 400 pixels for PASCAL Context. This results in an average of 370 proposals for SIFT Flow and 150 proposals for PASCAL Context, for each of the three color spaces. Additionally we use all ground-truth regions at training time. This is especially important for very small objects that are not tightly covered by region proposals.

Training. The network is trained using Stochastic Gradient Descent (SGD) with momentum. For 20 epochs we use a learning rate of 1e-3, followed by 10 epochs using learning rate 1e-4. All other SGD hyperparameters are taken from Fast R-CNN (Girshick, 2015). We use either an inverse-class frequency weighted loss (referred to as *balanced* below) or a natural frequency weighted loss (*imbalanced*).

4.4.2 Main results

SIFT Flow. We compare our method to other works on SIFT Flow test in Table 4.1. We first compare in the balanced setting, which takes rare classes into account. Hence we train our model for the loss described in Sec. 4.3.4 and compare to methods using class accuracy. We achieve 64.0%, which substantially outperformed the state-of-the-art at the time of publication of our paper (Caesar et al., 2016b), including the fully convolutional method (Eigen and Fergus, 2015) by +8.3% and the region-based

method (Caesar et al., 2015) by +8.4%. Since the publication of our paper, only Huang et al. (2017a) have matched our results in terms of class accuracy.

We also compare in the imbalanced setting using pixel accuracy, which mainly measures performance on common classes. Hence we train our model for the loss in Eq. 4.4. This yields a competitive 84.3% pixel accuracy, outperforming most previous methods, and coming close to the state-of-the-art at the time of publication of our paper (Eigen and Fergus, 2015) (86.8%). Since then, the state-of-the-art in pixel accuracy has further increased to 89.3% (Fan and Ling, 2018).

PASCAL Context. We also evaluate our method on the recent PASCAL Context dataset (Mottaghi et al., 2014). In Table 4.2 we show the results using either a balanced or an imbalanced loss. Our balanced model achieves 49.9% class accuracy, outperforming the only work that reported results for that measure at the time of publication of our paper, (Long et al., 2015), by +3.4%. At the same time, our imbalanced model achieved competitive results on pixel accuracy (62.4%) and reasonable results on mean IOU (32.5%). Since the publication of our paper (Caesar et al., 2016b), the state-of-the-art has improved significantly on all three metrics.

Qualitative analysis. Fig. 4.4 and 4.5 show example labelings generated by our method on SIFT Flow test and PASCAL Context validation. Notice how our method accurately adheres to object boundaries, such as *buildings* (Fig. 4.4e, 4.4h), *birds* (Fig. 4.5a, 4.5c) and *boat* (Fig. 4.5i). This is one of the advantages of using a region-based approach. Furthermore, our method correctly identifies small objects like *pole* (Fig. 4.4a) and the *streetlight* (Fig. 4.4d). This is facilitated by our method's ability to adaptively select the scale on which to do recognition. Finally, notice that our method sometimes even correctly labels parts of the image missing in the ground-truth, such as *fence* (Fig. 4.4d) and *cat whiskers* (Fig. 4.5d).

4.4.3 Extra analysis

Accuracy at object boundaries. Following Kohli et al. (2009); Krähenbühl and Koltun (2011), we evaluate the performance on image pixels that are within 4 pixels of a ground-truth object boundary. We compare our method to the MatConvNet (Vedaldi and Lenc, 2015) reimplementation of Fully Convolutional Networks (FCN) (Long et al., 2015) in Table 4.3. On SIFT Flow test, FCN-16s obtains 37.9% class accuracy on boundaries, while our method gets to 57.3%. When evaluated on all pixels

Method	Class Acc.	Pixel Acc.
Byeon et al. (2015)	22.6	68.7
Gould et al. (2014)	25.7	78.4
Tighe and Lazebnik (2010)	29.4	76.9
Pinheiro and Collobert (2014)	30.0	76.5
Gatta et al. (2014)	32.1	78.7
Singh and Kosecka (2013)	33.8	79.2
Shuai et al. (2015)	39.7	80.1
Tighe and Lazebnik (2013a)	41.1	78.6
Kekeç et al. (2014)	45.8	70.4
Sharma et al. (2014)	48.0	79.6
Yang et al. (2014)	48.7	79.8
George (2015)	50.1	81.7
Farabet et al. (2013)	50.8	78.5
Long et al. (2015)	51.7	**85.2**
Sharma et al. (2015)	52.8	80.9
Caesar et al. (2015)	55.6	-
Eigen and Fergus (2015)	55.7	**86.8**
Ours	**64.0**	**84.3**
Souly et al. (2017)	46.7	83.4
Lin et al. (2017)	53.4	88.1
Cheng et al. (2017)	53.6	86.4
Hu et al. (2017)	58.6	88.0
Ngan Le et al. (2017)	61.0	84.7
Fan and Ling (2018)	61.1	**89.3**
Huang et al. (2017a)	**64.0**	87.8

Table 4.1: Evaluation on SIFT Flow test. We show results for our model trained for either a balanced or an imbalanced loss. We also compare to other works published before (top) and after (bottom) the publication of our paper (Caesar et al., 2016b). If multiple results are given, we report the maximum result for each metric.

Method	Class Acc.	Pixel Acc.	Mean IOU
Carreira et al. (2012)	-	-	18.1
Dai et al. (2015b)	-	-	34.4
Long et al. (2015)	46.5	**65.9**	35.1
Dai et al. (2015a)	-	-	35.7
Zheng et al. (2015)	-	-	**39.3**
Dai et al. (2015a) (add. boxes)	-	-	40.5
Ours	**49.9**	62.4	32.5
Park et al. (2017)	52.0	-	39.7
Bansal et al. (2017)	51.5	-	41.4
Abdulnabi et al. (2017)	53.5	73.0	42.1
Lin et al. (2017)	53.9	71.5	43.3
Shen et al. (2017)	-	-	44.4
Fan and Ling (2018)	**57.7**	75.1	45.3
Chen et al. (2017)	-	-	45.7
Hu et al. (2017)	56.6	73.5	45.8
Hung et al. (2017)	-	73.8	46.5
Huang et al. (2017b)	57.2	**75.3**	**48.1**

Table 4.2: Evaluation on PASCAL Context validation. We show results using a balanced and an imbalanced version of our method. We also compare to other works published before (top) and after (bottom) the publication of our paper (Caesar et al., 2016b). If multiple results are given, we report the maximum result for each metric. Results for Carreira et al. (2012) are taken from the errata of Mottaghi et al. (2014). Dai et al. (2015a) train using additional bounding box annotations.

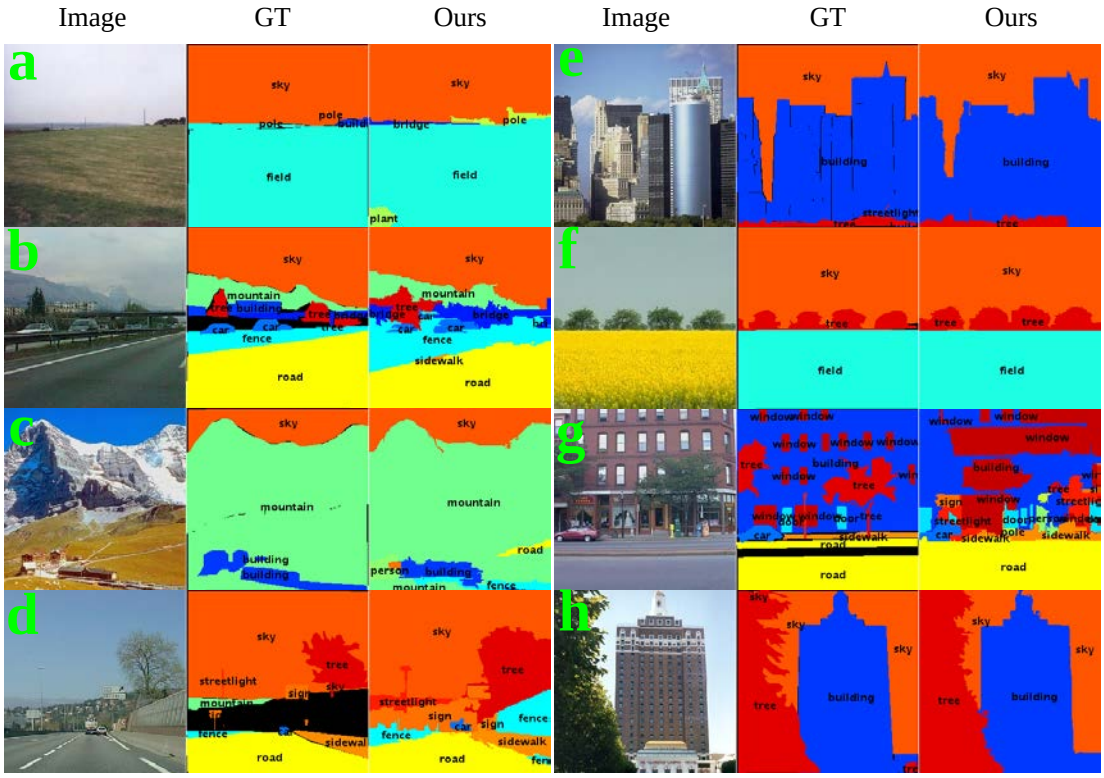


Figure 4.4: Example labelings on SIFT Flow test. We show an image, the ground-truth labeling and the output of our balanced model.

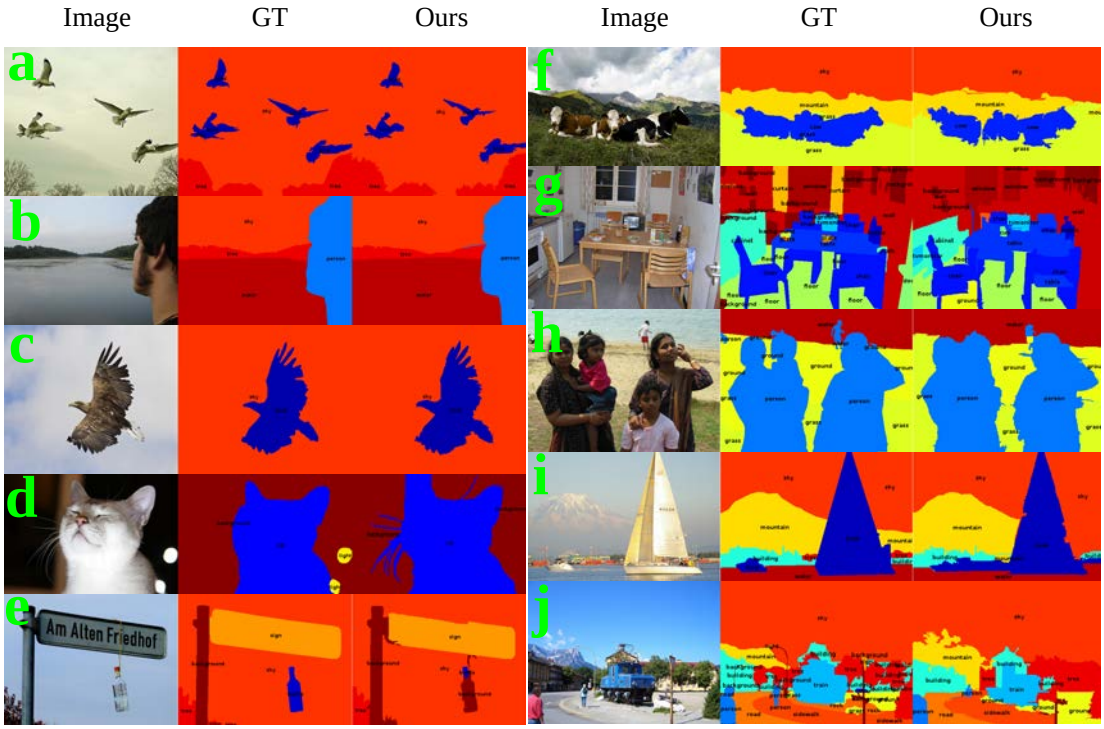


Figure 4.5: Example labelings on PASCAL Context validation. We show an image, the ground-truth labeling and the output of our imbalanced model.

	Boundaries	**Full image**
FCN-16s	37.9	49.3
Ours	57.3	64.0
difference	*+19.4*	*+14.7*
FCN-16s	34.0	48.1
Ours	38.9	49.9
difference	*+4.9*	*+1.8*

Table 4.3: Class accuracy at object boundaries on SIFT Flow test (top) and PASCAL Context validation (bottom). Improvements on boundaries are consistently larger than on full images.

ROI pooling		**Class Acc.**
bounding box		62.3
region		62.8
region + box	tied weights	63.4
region + box	separate weights	64.0
bounding box	purely rect.	59.3

Table 4.4: Results on SIFT Flow test using free-form pooling, bounding box pooling or both. We also report results when regions are rectangular even in the region-to-pixel layer (purely rectangular).

in the image, FCN-16s brings 49.3%, vs 64.0% by our method. Hence, our method is +19.4% better on boundaries and +14.7% on complete images. Analogously, on PASCAL Context we get +4.9% on boundaries and +1.8% on complete images. Since our improvements are consistently larger on object boundaries, we conclude that our method is especially good at capturing them, compared to the basic FCN architecture (Fig. 4.2a).

End-to-end training. Our region-to-pixel layer enables end-to-end training of region-based semantic segmentation models. We analyze how this end-to-end training influences performance, by comparing the baseline model (Fig. 4.2b) to our model (Fig. 4.2c). To isolate the effect of end-to-end training, in both models we perform ROI pooling on the bounding box only. Hence all components of the two models are iden-

tical, apart from the region-to-pixel layer and the loss they are trained for. On SIFT Flow test the baseline model achieves a pixel accuracy of 60.9%, compared to our 83.7%. We conclude that end-to-end training yields considerable accuracy gains over the baseline architecture in Fig. 4.2b.

Softmax before max. Our application of the max before the softmax (Eq. 4.3) enables us to recognize each object at its appropriate scale (Sec. 4.3.2). However, using the softmax before the max (Eq. 4.1) yields an alternative model. Interestingly, on SIFT Flow test our proposed order outperforms the alternative by +8.7% class accuracy.

Importance of multi-scale regions. We argue that overlapping, multi-scale regions are important to unleash the full potential of region-based methods. To show this, we train and test our model with non-overlapping regions (Felzenszwalb and Huttenlocher, 2004). This yields 60.0% class accuracy on SIFT Flow test, which is below the results when using multi-scale overlapping regions (64.0% class accuracy).

Free-form versus bounding box representations. We analyze the influence of the different representations resulting from different ROI pooling methods (Sec. 4.3.3). Keeping all else constant, we compare (I) free-form ROI pooling, (II) bounding box ROI pooling, (III) their combination with tied weights and (IV) their combination with separate weights. Results are shown in Table 4.4.

Free-form representations perform +0.5% better than bounding box representations, demonstrating that focusing accurately on the object is better. Their combination does even better, yielding another +0.6% gain with tied weights (same number of model parameters) and +0.6% with separate weights. Hence both representations are complementary and best treated separately.

In all above experiments the region-to-pixel layer operates on free-form regions. To verify the importance of the free-form regions themselves, we perform an extra experiment using purely rectangular regions (both in the region-to-pixel layer and during ROI pooling). This lowers class accuracy by -4.7%, demonstrating the value of free-form regions.

4.5 Discussion

In this section we discuss later works that have been inspired by our ECCV 2016 (Caesar et al., 2016b) paper and compare our results to other methods released since the

publication of our paper.

Later works inspired by ours. Two have built on our paper. Liu et al. (2017) use a more specialized version of our free-form ROI pooling to detect ships using rotated bounding boxes. Lee et al. (2017) extend the concept of free-form ROI pooling to semantic lines in an image (e.g. symmetry, horizontal, diagonal and vertical lines).

Comparison to other methods. He et al. (2017b) run our published code on other datasets and compare it to 13 leading methods including their own, FCN (Long et al., 2015), Deeplab (Chen et al., 2015a), CRF-RNN (Zheng et al., 2015). They use a multi-view approach to combine several images of the same scene, which is an advantage over the single-view setup used by our method. Despite that, the results rank our method first on the SUN3D (Xiao et al., 2013) dataset and second on the NYUDv2 (Silberman et al., 2012) dataset in terms of class accuracy. This shows that our method is also suitable for highly cluttered indoor scenes, which it was not designed for. Fig. 4.6 shows a qualitative comparison of the results. Contrary to other methods, our method produces crisp object boundaries, particularly on the straight edges of the doors in images 3 and 4. Furthermore our method tends to favor small and rare classes, such as the lamp on the left in image 2, which is partially occluded and therefore missed by most other methods.

As pointed out in Sec. 4.4.2, our class accuracy results on SIFT Flow (Liu et al., 2011) have only been matched by one method (Huang et al., 2017a) after the publication of our ECCV 2016 (Caesar et al., 2016b) paper. This shows the potential of our approach that is optimized end-to-end for the final pixel-level evaluation criterion. It may also be an indicator that performance on this dataset is converging and that larger datasets are required, particularly when dealing with class imbalance. In Sec. 6 we present a new large scale dataset that is more than 60x bigger than SIFT Flow.

4.6 Future work

In this section we present future work related to this chapter.

Atrous convolutions. Max poolings after convolutional layers are responsible for the reduction in resolution in CNNs. Atrous convolutions have had a big impact on semantic segmentation (see Sec. 2.1.3.1). They allow for dense feature extraction on high resolution feature maps, instead of sparse feature extraction on low resolution

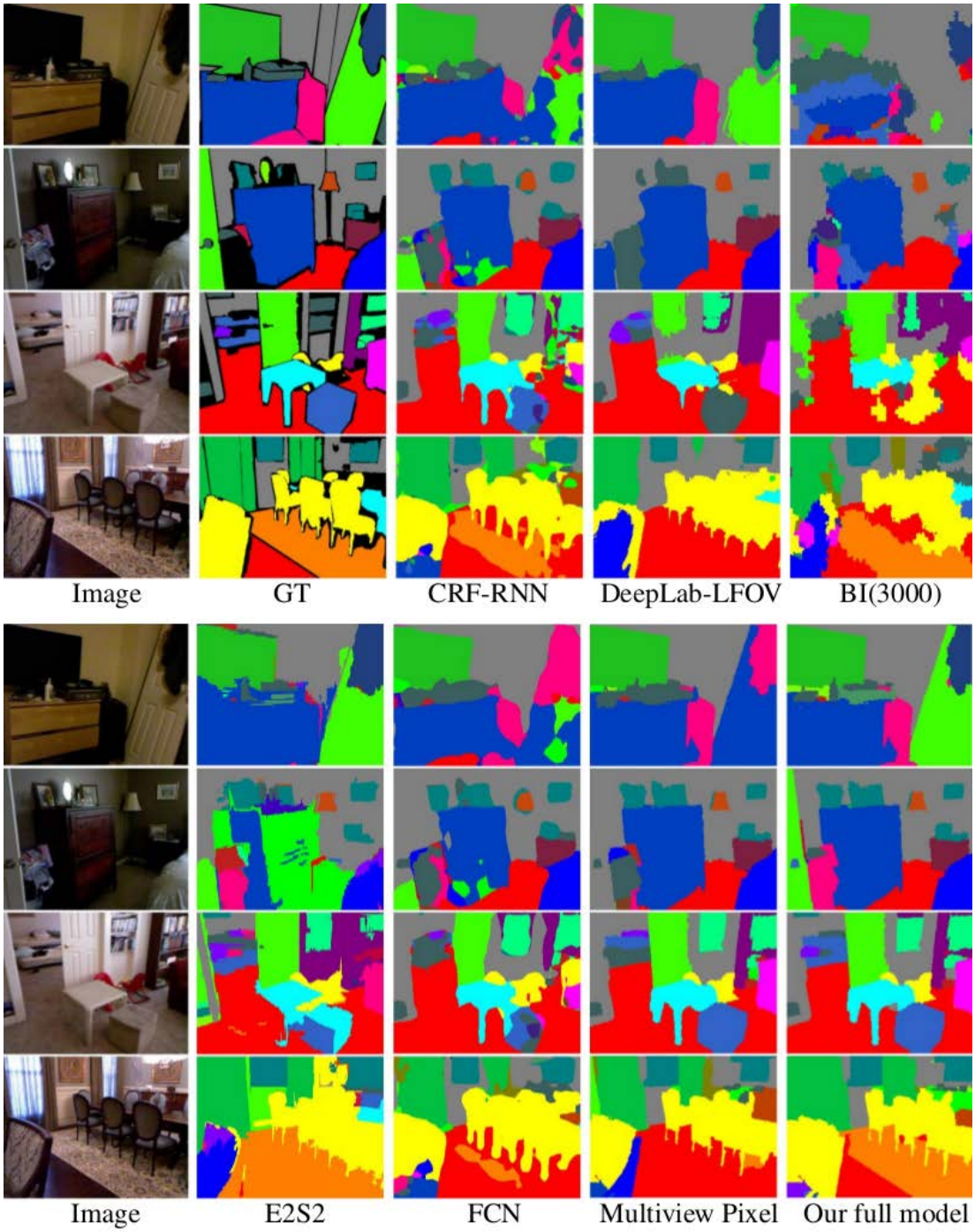


Figure 4.6: Comparison of the results of leading semantic segmentation methods on the NYUDv2 (Silberman et al., 2012) dataset. We show four example images, the desired ground-truth (GT) labeling and the outputs of seven methods. E2S2 is the method presented in this chapter. The last two methods correspond to the methods of He et al. (2017b). Some methods were remove to increase readability. Image modified from He et al. (2017b).

feature maps. As part of future work we propose to integrate atrous convolutions into region-based methods. We expect a further increase in performance. However it should be noted that the authors of Deeplab (Chen et al., 2015a) use a combination of atrous convolutions and bilinear interpolation to achieve a good efficiency/accuracy trade-off.

Region Proposal Networks. Although region proposals have proven to be beneficial in many applications, they are subject to frequent criticism. Note that our method is not limited to any particular region proposal algorithm, but in Sec. 4.3.5 we achieve significant speedups for bottom-up proposal methods. Several works mention region proposal computation as a bottleneck at test time (Ren et al., 2015; Redmon et al., 2016). Therefore Faster R-CNN (Ren et al., 2015) introduces a Region Proposal Network that shares convolutional features with the detection network and enables nearly cost-free region proposals (see Sec. 2.1.3.2). Future work could focus on the integration of Region Proposal Networks into our method.

Instance segmentation. In recent years, the instance segmentation task (see Sec. 2.1) has received a lot of attention. Mask R-CNN (He et al., 2017a) has shown that region-based methods can achieve state-of-the-art performance on this task. Our ECCV 2016 (Caesar et al., 2016b) work builds on Fast R-CNN (Girshick, 2015), which is a predecessor of Mask R-CNN (He et al., 2017a). Hence future work could modify our method accordingly and apply it to instance segmentation. In Sec. 4.4.3 we have shown that end-to-end training yields considerable accuracy gains over a baseline architecture. It would be interesting to compare whether the gains are equally big when using ROIAlign (He et al., 2017a) instead of ROI pooling.

4.7 Conclusion

We propose a region-based semantic segmentation model with an accompanying end-to-end training scheme based on a CNN architecture. This architecture combines the advantages of crisp object boundaries and adaptive, multi-scale representations found in region-based methods with end-to-end training directly optimized for semantic segmentation found in fully convolutional methods. We achieve this by introducing a differentiable region-to-pixel layer and a differentiable free-form ROI pooling layer. In terms of class pixel accuracy, our method outperformed the state-of-the-art at the time of publication on two datasets, achieving 49.9% on PASCAL Context and 64.0% on SIFT Flow.

Chapter 5

Weakly supervised object localization using things and stuff transfer

5.1 Introduction

The goal of object class detection is to place a tight bounding box on every instance of an object class. Given an input image, many object detectors (Girshick et al., 2015; Girshick, 2015; Cinbis et al., 2014, 2016) first extract object proposals (Alexe et al., 2010; Uijlings et al., 2013; Zitnick and Dollár, 2014) and then score them with a classifier to determine their probabilities of containing an instance of the class. Manually annotated bounding boxes are typically required for training (full supervision).

Annotating bounding boxes is tedious and time-consuming. In order to reduce the annotation cost, many previous works learn the detector in a weakly supervised setting (Bilen et al., 2014b, 2015; Cinbis et al., 2016; Deselaers et al., 2010; Russakovsky et al., 2012; Siva and Xiang, 2011; Song et al., 2014a,b; Shi and Ferrari, 2016), i.e. given a set of images known to contain instances of a certain object class, but without their locations. This weakly supervised object localization (WSOL) bypasses the need for bounding box annotation and substantially reduces annotation time.

Despite the low annotation cost, the performance of WSOL is considerably lower than that of full supervision. To improve WSOL, various advanced cues can be added (see Sec. 2.2), e.g. objectness (Deselaers et al., 2010; Alexe et al., 2012a; Cinbis et al., 2016; Siva and Xiang, 2011; Tang et al., 2014; Shi and Ferrari, 2016), which gives an estimation of how likely a proposal contains an object; co-occurrence among multiple classes in the same training images (Shi et al., 2012); object size estimates based on an auxiliary dataset with size annotations (Shi and Ferrari, 2016); and appearance

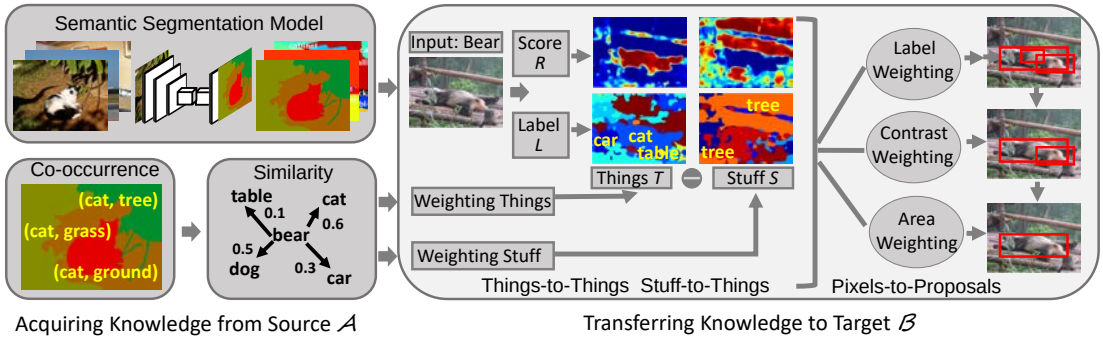


Figure 5.1: An overview of our things and stuff transfer (TST) method. We acquire the 1) segmentation model 2) co-occurrence relation and 3) similarity relation from the source $\mathcal{A}$ and transfer them to the target $\mathcal{B}$. We use the segmentation model to generate two maps: thing (T) and stuff (S) maps; each of them contains one score (R) map and one label (L) map. The knowledge of class similarity and co-occurrence is specifically transferred as weighting functions to the thing and stuff label maps. Based on the transferred knowledge, we propose three scoring schemes (label weighting, contrast weighting, and area weighting) to propagate the information from pixels to proposals. The rightmost image column illustrates some highly ranked proposals in the image by gradually adopting the three schemes.

models transferred from object classes with bounding box annotations to new object classes (Guillaumin and Ferrari, 2012; Hoffman et al., 2014; Rochan and Wang, 2015).

There are two types of classes that can be transferred from a source set with manually annotated locations: things and stuff (see Sec. 1.1). Things have a specific spatial extent and shape (e.g. *helicopter*, *cow*, *car*), while stuff does not (e.g. *sky*, *grass*, *road*). Current transfer works mostly focus on transferring appearance models among similar thing classes (Guillaumin and Ferrari, 2012; Rochan and Wang, 2015; Hoffman et al., 2014) (*things-to-things*). In contrast, using stuff to find things (Heitz and Koller, 2008; Lee and Grauman, 2011) is largely unexplored, particularly in the WSOL setting (*stuff-to-things*).

In this chapter, we propose to help WSOL for classes where location annotations are not available, by transferring things and stuff knowledge from a source set with available annotations. The source and target classes might share similar appearance (e.g. *bear fur* is similar to *cat fur*) or appear against similar background (e.g. *horse* and *sheep* appear against *grass*). To exploit this, we acquire three types of knowledge from the source set: a segmentation model trained on both thing and stuff classes; similarity relations between target and source classes; and co-occurrence relations between thing

and stuff classes in the source. The segmentation model is used to generate thing and stuff segmentation maps on a target image, while the class similarity and co-occurrence knowledge help refining them. We then incorporate these maps as new cues into a multiple instance learning framework (MIL) (Dietterich et al., 1997), propagating the transferred knowledge from the pixel level to the object proposal level.

In extensive experiments, we show that our method: (1) improves over a standard MIL baseline on three datasets: ILSVRC (Russakovsky et al., 2015), COCO (Lin et al., 2014), PASCAL VOC 2007 (Everingham et al., 2010); (2) outperforms the things-to-things transfer method (Rochan and Wang, 2015) and recent WSOL methods (Bilen and Vedaldi, 2016; Cinbis et al., 2016; Wang et al., 2015) on VOC 2007; (3) outperforms another things-to-things transfer method (LSDA by Hoffman et al. (2014)) on ILSVRC.

This chapter is a modified version of our paper published at the International Conference on Computer Vision 2017 (Shi et al., 2017a). The first author is Miaojing Shi. However, I was responsible for acquiring the source knowledge (see Sec. 5.4): the segmentation model, the similarity and co-occurrence relations. Furthermore I ran some of the experiments and was involved in all steps of the paper creation process, except its initial conception.

5.2 Related work

Weakly supervised object localization. In WSOL the training images are known to contain instances of a certain object class but their locations are unknown. The task is both to localize the objects in the training images and to learn a detector for the class.

Due to the use of strong CNN features (Girshick et al., 2014; Krizhevsky et al., 2012), recent works on WSOL (Bilen et al., 2015; Cinbis et al., 2016; Song et al., 2014a; Wang et al., 2015; Bilen and Vedaldi, 2016; Shi and Ferrari, 2016) have shown remarkable progress. Moreover, researchers also tried to incorporate various advanced cues into the WSOL process, e.g. objectness (Cinbis et al., 2016; Deselaers et al., 2010; Siva and Xiang, 2011; Tang et al., 2014), object size (Shi and Ferrari, 2016), co-occurrence (Shi et al., 2012) among classes, and transferring appearance models of the source thing classes to help localize similar target thing classes (Guillaumin and Ferrari, 2012; Rochan and Wang, 2015; Hoffman et al., 2014). This chapter introduces a new cue called things and stuff transfer (TST), which learns a semantic segmentation model from the source on both things and stuff annotations and transfers its knowledge

to help localize the target thing class.

Transfer learning. The goal of transfer learning is to improve the learning of a target task by leveraging knowledge from a source task (Pan and Yang, 2010). It is intensively studied in image classification, segmentation and object detection (Aytar and Zisserman, 2011, 2012; Tommasi et al., 2010; Lampert et al., 2009; Rohrbach et al., 2010; Ott and Everingham, 2011; Stark et al., 2009). Many methods use the parameters of the source classifiers as priors for the target model (Aytar and Zisserman, 2011, 2012; Tommasi et al., 2010). Other works (Lampert et al., 2009; Rohrbach et al., 2010) transfer knowledge through an intermediate attribute layer, which captures visual qualities shared by many object classes (e.g. *striped*, *yellow*). A third family of works transfer object parts between classes (Aytar and Zisserman, 2012; Ott and Everingham, 2011; Stark et al., 2009), e.g. *wheels* between *cars* and *bicycles*.

In this work we are interested in the task where we have the location annotations in the source and transfer them to help learn the classes in the target (Shi et al., 2012; Kuettel et al., 2012; Rochan and Wang, 2015; Guillaumin and Ferrari, 2012; Lee and Grauman, 2011; Heitz and Koller, 2008). We categorize the transfer into two types: 1) *Things-to-things.* Guillaumin and Ferrari (2012) transferred spatial location, appearance, and context information from the source thing classes to localize the things in the target; Shi et al. (2012) and Rochan and Wang (2015) follow a similar spirit to Guillaumin and Ferrari (2012); while Kuettel et al. (2012) instead transferred segmentation masks. 2) *Stuff-to-things.* Heitz and Koller (2008) proposed a context model to utilize stuff regions to find things, in a fully supervised setting for the target objects; Lee and Grauman (2011) also made use of stuff annotations in the source to discover things in the target, in an unsupervised setting.

Our work offers several new elements over these: (1) we encode the transfer as a combination of both *things-to-things* and *stuff-to-things*; (2) we propose a model to propagate the transferred knowledge from the pixel level to the proposal level; (3) we introduce a second order transfer, i.e. *stuff-to-things-to-things*.

5.3 Overview of our method

In this section we define the notations and introduce our method on a high level, providing some details for each part.

Notations. We have a source set $\mathcal{A}$ and a target set $\mathcal{B}$. We have every image pixelwise annotated for both stuff and things in $\mathcal{A}$; whereas we have only image-level labels for images in $\mathcal{B}$. We denote by $\mathcal{A}^T$ the set of thing classes in $\mathcal{A}$, and a^t an individual thing class; analogue we have $\mathcal{A}^S$ and a^s for stuff classes in $\mathcal{A}$ and $\mathcal{B}^T$ and b^t for thing classes in $\mathcal{B}$. Note that there are no stuff classes in $\mathcal{B}$, as datasets labeled only by thing classes are more common in practice (e.g. PASCAL VOC (Everingham et al., 2015), ImageNet (Russakovsky et al., 2015), COCO (Lin et al., 2014)).

Method overview. Our goal is to conduct WSOL on $\mathcal{B}$, where the training images are known to contain instances of a certain object class but their locations are unknown. A standard WSOL approach, MIL, treats images as bags of object proposals (Dietterich et al., 1997; Alexe et al., 2010; Uijlings et al., 2013; Zitnick and Dollár, 2014) (instances). The task is both to localize the objects (select the best proposal) in the training images and to learn a detector for the target class. To improve MIL, we transfer knowledge from $\mathcal{A}$ to $\mathcal{B}$, incorporating new cues into it.

Fig. 5.1 illustrates our transfer. We first acquire three types of knowledge in the source $\mathcal{A}$ (Sec. 5.4): 1) a semantic segmentation model (Sec. 5.4.1), 2) the thing class similarities between $\mathcal{A}$ and $\mathcal{B}$ (Sec. 5.4.2) and 3) the co-occurrence frequencies between thing and stuff classes in $\mathcal{A}$ (Sec. 5.4.3). Afterwards, we transfer the knowledge to $\mathcal{B}$ (Sec. 5.5). Given an image in $\mathcal{B}$, we first use the segmentation model to generate the thing (T) and stuff (S) maps of it (Sec. 5.5.1). T contains one score map (R) and one label (L) map, so does S. The segmentation model transfers knowledge generically to every image in $\mathcal{B}$. Building upon its result, we propose three proposal scoring schemes: label weighting (LW, Sec. 5.5.2), contrast weighting (CW, Sec. 5.5.3), and area weighting (AW, Sec. 5.5.4). These link the pixel level segmentation to the proposal level score. In each scheme, two scoring functions are proposed separately on thing and stuff maps. We combine the three schemes to provide an even better proposal score to help MIL (Sec. 5.5.5).

Scoring schemes. LW transfers the similarity and co-occurrence relations as weighting functions to the thing and stuff label maps, respectively. Since we do not have stuff annotations on $\mathcal{B}$, we conduct the co-occurrence knowledge transfer as a second-order transfer by finding the target class' most similar thing class in $\mathcal{A}$. We believe that the target class should appear against a similar background with its most similar class. For example, in Fig. 5.1 target class *bear*'s most similar class in $\mathcal{A}$ is *cat*, LW up-weights the *cat* score on T and its frequently co-occurring *tree* score on S.

LW favors small proposals with high weighted scores. To counter this effect, we introduce the CW score. It measures the dissimilarity of a proposal to its surroundings, measured on the thing/stuff score maps (Fig. 5.3). CW up-weights proposals that are more likely to contain an entire object in T or an entire stuff region in S.

Finally, the AW score encourages proposals to incorporate as much as possible of the connected components of pixels on a target's K most similar classes in $\mathcal{A}$ (e.g. Fig. 5.1: the *cat* area in the T map). While CW favors objects in general, AW focuses on objects of the target class in particular.

5.4 Acquiring knowledge from the source $\mathcal{A}$

5.4.1 Segmentation model

We employ the popular fully convolutional network (FCN-16s) (Long et al., 2015) to train an end-to-end semantic segmentation model on both thing and stuff classes of $\mathcal{A}$. Given a new image, the FCN model is able to predict a likelihood distribution over all classes at each pixel. Notice that the FCN model is first pretrained for image classification on ILSVRC 2012 (Russakovsky et al., 2015), then fine-tuned for semantic segmentation on $\mathcal{A}$. While it is possible that some of the target classes are seen during pretraining, only image-level labels are used. Therefore the weakly supervised setting still holds for the target classes.

5.4.2 Similarity relations

We compute the thing class similarities $V(a^t, b^t)$ between any thing class pair (a^t, b^t). We propose two similarity measures to compute V as follows:

Appearance similarity. Every image in $\mathcal{A}$ or $\mathcal{B}$ is represented by a 4096-dimensional CNN feature vector covering the whole image, using the output of the FC7 layer of the AlexNet CNN architecture (Krizhevsky et al., 2012). The similarity of two images is the inner product of their feature vectors. The similarity $V_{\text{APP}}(a^t, b^t)$ is therefore the average similarity between images in a^t and images in b^t.

Semantic similarity. We compute the commonly used Lin similarity (Lin, 1998) $V_{\text{SEM}}(a^t, b^t)$ between two nouns b^t and a^t in the WordNet hierarchy (Fellbaum, 1998).

5.4.3 Co-occurrence relation

We denote by $U(a^s, a^t)$ the co-occurrence frequency of any stuff and thing class pair (a^s, a^t) in $\mathcal{A}$. This frequency is computed and normalized over all the images in $\mathcal{A}$.

5.5 Transferring knowledge to the target $\mathcal{B}$

This section transfers the source knowledge to the target set $\mathcal{B}$. In this set, we have access only to image-level labels, but no location annotations. We call the classes that are listed on the image-level label list *target classes*. Given a new image of class b^t, we first use the FCN model trained on $\mathcal{A}$ to generate the thing (T) and stuff (S) segmentations separately (Sec. 5.5.1). Then we introduce three proposal scoring schemes to propagate the information from pixel level to proposal level (Sec. 5.5.2 – 5.5.4). Finally we combine the three scoring schemes into a single window score (Sec. 5.5.5). The scoring scheme parameters are learned in Sec. 5.5.6.

5.5.1 Generating thing and stuff segmentations

We apply the trained FCN model (Sec. 5.4.1) to a target image in $\mathcal{B}$. Usually, the output semantic segmentation is obtained by maximizing over all the class scores at each pixel (Long et al., 2015; Chen et al., 2015a; Eigen and Fergus, 2015; Farabet et al., 2013; Noh et al., 2015; Pinheiro and Collobert, 2014). In this chapter, we instead generate two output segmentations, one for things T and one for stuff S. We denote i as the i-th pixel in the image. We use $R^T = \{r_i^T\}$ and $L^T = \{l_i^T\}$ to denote the score (R) and label (L) maps for T. They are generated by keeping the maximum score and the corresponding label over all the thing classes $\mathcal{A}^T$ at each pixel i. Similar to R^T and L^T, $R^S = \{r_i^S\}$ and $L^S = \{l_i^S\}$ are generated by keeping the maximum score over all the stuff classes $\mathcal{A}^S$ at each pixel.

Fig. 5.1 shows an example of a *bear* image (target). The thing and stuff maps are produced by the semantic segmentation model. The R heatmaps indicate the probability of assigning a certain thing or stuff label to each pixel. Building upon these heatmaps, we propose three proposal scoring schemes to link the pixel level result to the proposal level score (Sec. 5.5.2 – 5.5.4). These try to give high scores to proposals containing the target class.

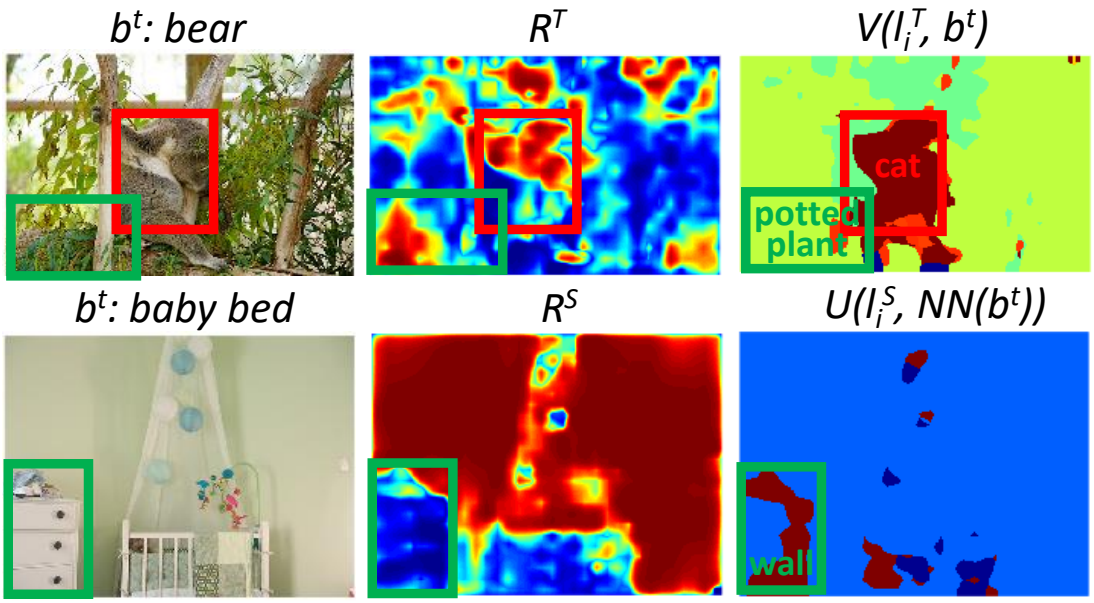


Figure 5.2: Label weighting example. Top: thing label weighting (class *bear*); bottom: stuff label weighting (class *baby bed*). R^T and R^S denote the thing and stuff score heatmaps, respectively; while $V(l_i^T, b^t)$ and $U(l_i^S, \mathrm{NN}(b^t))$ denote the thing and stuff label weighting heatmaps. We illustrate some proposals in each image. We print the dominantly predicted labels in the proposals to show how label weighting favors b^t's NN class in thing maps and its frequently co-occurring stuff class in stuff maps.

5.5.2 Label weighting (LW)

Because *bear* is more similar to *cat* than to *table*, we want to up-weight the proposal area in the thing map if it is predicted as *cat*. Meanwhile, because *bear* frequently appears against *tree*, we also want to up-weight the proposal area in the stuff map if it is predicted as *tree*. To do this, we transfer the knowledge of similarity and co-occurrence relations acquired in the source to the target class (*bear*), and use both relations to modulate the segmentation scores in T and S. Both relations and segmentation scores play a role in the label weighting proposal scoring scheme.

Thing label weighting. We can generate a thing label weighting map depending on how close the predicted class l_i^T at pixel i in L^T is to the target class b^t. The thing label (l_i^T) weight is given by the class similarity score $V(l_i^T, b^t)$ (Sec. 5.4.2). In Fig. 5.1 the target class *bear* is more similar to *cat* than to *table*. If a pixel is predicted as *cat*, then we assign a high label weight, otherwise we assign a low one.

Stuff label weighting. We do not have stuff annotations in $\mathcal{B}$. To conduct the stuff label weighting, we first find b^t's most similar thing class in $\mathcal{A}^T$ according to a similarity relation V (we denote it by $\mathrm{NN}(b^t)$). We believe that b^t should appear against a similar background (stuff) as its most similar thing class $\mathrm{NN}(b^t)$. We employ the co-occurrence frequency $U(l_i^S, \mathrm{NN}(b^t))$ of $\mathrm{NN}(b^t)$ as the corresponding stuff label weight for l_i^S at pixel i as stuff label weighting L^S.

In Fig. 5.1, *cat* frequently co-occurs with *trees*, and so does *bear*. So, if a certain pixel is predicted as *tree*, it gets assigned a high stuff label weight.

Proposal scoring. To score the proposals in an image, we multiply the label weights $V(l_i^T, b^t)$ and $U(l_i^S, \mathrm{NN}(b^t))$ with the segmentation scores r_i^T and r_i^S at each pixel. The weighting scheme is conducted separately on T and S. Given a window proposal w, we average the weighted scores inside w:

$$\begin{aligned} \mathrm{LW}^t(w, \alpha^t) &= f\left(\tfrac{1}{|w|} \textstyle\sum_{i \in w} r_i^T V(l_i^T, b^t), \alpha^t\right) \\ \mathrm{LW}^s(w, \alpha^s) &= f\left(\tfrac{1}{|w|} \textstyle\sum_{i \in w} r_i^S U(l_i^S, \mathrm{NN}(b^t)), \alpha^s\right) \end{aligned} \tag{5.1}$$

where $|w|$ denotes the size of w (area in pixels). We apply an exponential function $f(x) = \exp(\alpha \cdot x)$ to both thing and stuff LWs, α^t and α^s are the parameters.

Fig. 5.2 offers two examples (*bear* and *baby bed*) for our thing and stuff label weighting schemes. The red proposal in the top row is mostly classified as a *cat* and the green proposal as a *potted plant*. Both proposals have high scores in the thing score map R^T, but the red proposal has a higher thing label weight $V(l_i^T, b^t)$, because *cat* is more similar to *bear* than to *potted plant*. In contrast, the green proposal in the bottom row has low scores in R^S but a high label weight $U(l_i^T, \mathrm{NN}(b^t))$, as *baby bed* co-occurs more frequently with *wall*.

Notice that the thing label weighting can be viewed as a first-order transfer where the information goes directly from the source thing classes to the target thing classes. Instead, the stuff label weighting can be viewed as second-order transfer where the information first goes from the source stuff classes to the source thing classes, and then to the target thing classes. To the best of our knowledge, such second-order transfer has not been proposed before.

5.5.3 Contrast weighting (CW)

The LW scheme favors small proposals with high label weights, which typically cover only part of an object (top right image in Fig. 5.1). To counter this effect, contrast

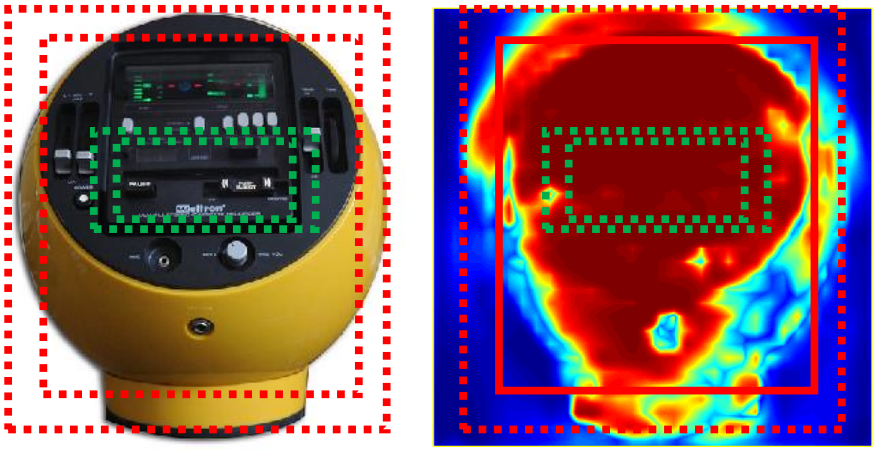


Figure 5.3: Contrast weighting example. An image of a *tape player* and some of its window proposals (left). CW^t is computed on the thing score map R^t (right). The red proposal has a higher contrast CW^t (with its surrounding dashed ring) than the green one.

weighting (CW) measures the dissimilarity of a proposal to its immediate surrounding area on the thing/stuff score maps. It up-weights proposals that are more likely to contain an entire object or an entire stuff region.

The surrounding $Surr(w,\theta)$ of a proposal w is a rectangular ring obtained by enlarging it by a factor θ in all directions (Alexe et al., 2010) (Fig. 5.3, the yellow ring). The CW between a window and its surrounding ring is computed as the Chi-square distance between their score map (R) histograms $h(\cdot)$

$$\mathrm{CW}(w,\theta) = \chi^2(h(w), h(Surr(w,\theta))) \tag{5.2}$$

We apply the CW scheme on both R^T and R^S and obtain $\mathrm{CW}^t(w,\theta^t)$ and $\mathrm{CW}^s(w,\theta^s)$. In Fig. 5.3 the red proposal has a higher CW^t score compared to the green one.

5.5.4 Area weighting (AW)

Thing area weighting. Fig. 5.4 gives an example of an *electric fan* and its semantic segmentation map. Its 3-NN classes in terms of appearance similarity (Sec. 5.4.2) are *table*, *chair* and *people*. Between the white and yellow proposals, the CW scheme gives a bigger score to the white one, because its contrast is high. Instead, the yellow proposal incorporates most of the *electric fan* area, but is unfortunately predicted as *table* and *chair*. The thing area weighting scheme helps here boosting the yellow proposal's score. We find the K-NN classes of b^t in $\mathcal{A}^T$ by using one of the similarity

measures in Sec. 5.4.2. Given a window w, we denote by $Area(w, b^t)$ the segment areas of any K-NN(b^t) inside w; while $Area(O(w), b^t)$ is the area that expands the current segments to their connected components $O(w)$ inside and outside w. We measure the area ratio between the segments and their corresponding connected components:

$$Ratio^t(w) = \frac{Area(w, b^t)}{Area(O(w), b^t)} \tag{5.3}$$

If none of the K-NN classes occurs in w, we simply set $Ratio^t$ to zero. Throughout this chapter, K is set to 3.

Stuff area weighting. In Fig. 5.4 among the three proposals, the green one is the best detection of the fan. However, its score is not the highest according to LW^t, CW^t and AW^t, as it contains some stuff area (*wall*) surrounding the *electric fan*. A bounding box usually has to incorporate some stuff area to fit an object tightly, as objects are rarely perfectly rectangle-shaped. We propose to up-weight a window w if stuff occupies a small but non-zero fraction of the window. We denote with $Ratio^s(w)$ the percentage of stuff pixels in window w.

For thing and stuff area weighting we apply a cumulative distribution function (CDF) of the normal distribution

$$\begin{aligned} \text{AW}^t(w, \mu^t, \sigma^t) &= \text{CDF}(Ratio^t(w) | \mu^t, \sigma^t) \\ \text{AW}^s(w, \mu^s, \sigma^s) &= \text{CDF}(Ratio^s(w) | \mu^s, \sigma^s) \end{aligned} \tag{5.4}$$

where μ^t and σ^t are the mean and standard deviation. We choose $\mu^t = \mu^s = 0$ and σ^t, σ^s are free parameters (Sec. 5.5.6).

5.5.5 Combining the scoring schemes

For each proposal in an image, the above scoring schemes can be independently computed, each on the thing and stuff map. The scoring schemes tackle different problems, and are complementary to each other. This sections combines them to give our final TST (things and stuff transfer) window score W.

All the scoring functions on the thing map are multiplied together as a thing score $W^t = \text{LW}^t * \text{CW}^t * \text{AW}^t$. This gives a higher score if a proposal mostly contains a target thing labeled as present in that image. Similarly, we have the stuff score $W^s = \text{LW}^s * \text{CW}^s * \text{AW}^s$, which gives a higher score if a proposal mostly contains stuff. To combine the thing and stuff scores, we simply subtract W^s from W^t

$$W = W^t - W^s \tag{5.5}$$

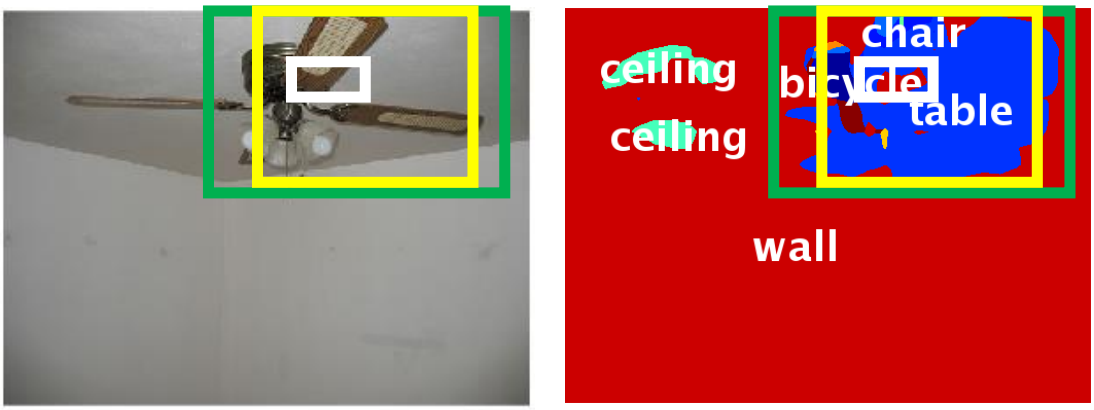


Figure 5.4: Area weighting example. An image of an *electric fan* (left) and its semantic segmentation (right). Thing area weighting favors the yellow proposal compared to the white one, as it incorporates most of the connected component area of *table* and *chair*. Stuff area weighting further favors the green proposal as it allows certain stuff area in a proposal as the surrounding area of *electric fan*.

5.5.6 Parameter learning

In the WSOL setting, we do not have the ground-truth bounding box annotations in the target set $\mathcal{B}$. Thus we learn the score parameters α^t, α^s, θ^t, θ^s, σ^t and σ^s on the source set $\mathcal{A}$, where we have ground-truth, by optimizing for the proxy measures described in Sec. 5.10. We train the semantic segmentation model on the *train* set of $\mathcal{A}$, and then apply it to the *val* set of $\mathcal{A}$. For each image in the *val* set, we rank all its proposals using (5.5). We jointly learn the score parameters by maximizing the performance over the entire validation set of $\mathcal{A}$.

5.6 Overall system

In WSOL, given the target training set in $\mathcal{B}$ with image-level labels, the goal is to localize the object instances in it and to train good object detectors for the target test set. We explain here how we build a complete WSOL system by building on a MIL framework and incorporating our transfer cues into it.

Basic MIL. We build a Basic MIL pipeline as follows. We represent each image in the target set $\mathcal{B}$ as a bag of object proposals extracted using Edge Boxes (Zitnick and Dollár, 2014). They return about 5,000 proposals per image, likely to cover all objects. Following Girshick et al. (2014); Bilen et al. (2014b); Song et al. (2014a,b); Wang et al.

(2015), we describe the proposals by the output of the FC7 layer of the AlexNet CNN architecture (Krizhevsky et al., 2012). The CNN model is pre-trained for whole-image classification on ILSVRC (Russakovsky et al., 2015), using the Caffe implementation (Jia, 2013). This produces a 4,096-dimensional feature vector for each proposal. Based on this feature representation for each target class, we iteratively build an SVM appearance model (object detector) in two alternating steps: (1) Re-localization: in each positive image, we select the highest scoring proposal by the SVM. This produces the positive set which contains the current selection of one instance from each positive image. (2) Re-training: we train the SVM using the current selection of positive samples, and all proposals from the negative images as negative samples. As in Deselaers et al. (2010); Siva and Xiang (2011); Cinbis et al. (2016); Guillaumin and Ferrari (2012); Tang et al. (2014), we also linearly combine the SVM score with a general measure of objectness (Alexe et al., 2010; Zitnick and Dollár, 2014). This leads to a higher MIL baseline.

Incorporating things and stuff transfer (TST). We incorporate our things and stuff transfer (TST) into Basic MIL by linearly combining the SVM score with our proposal scoring function (5.5). Note how the behavior of (5.5) depends on the class similarity measure used within it (either appearance or semantic similarity, Sec. 5.4.2).

Deep MIL. Basic MIL uses an SVM on top of fixed deep features as the appearance model. Now we change the model to fine-tune all layers of the deep network during the re-training step of MIL. We take the output of Basic MIL as an initialization for two additional MIL iterations. During these iterations, we use Fast R-CNN (Girshick et al., 2015).

5.7 Experiments

5.7.1 Datasets and evaluation protocol

We use one source set $\mathcal{A}$ (PASCAL Context) and several different target sets $\mathcal{B}$ in turn (ILSVRC-20, COCO-07 and PASCAL VOC 2007). Each target set contains a training set and a test set. We perform WSOL on the target training set to localize objects within it. Then we train a Fast R-CNN (Girshick, 2015) detector from it and apply it on the target test set.

Method	**APP**	**SEM**
Basic MIL	39.7	
DT ≈ Rochan and Wang (2015) (transfer only)	15.0	-
DT + MIL ≈ Rochan and Wang (2015) (full)	39.5	-
TST	46.7	46.0
Basic MIL + Objectness (Zitnick and Dollár, 2014)	47.6	
DT ≈ Rochan and Wang (2015) (transfer only)	34.5	-
DT + MIL ≈ Rochan and Wang (2015) (full)	49.1	-
TST	52.7	52.5
Deep MIL + Objectness (Zitnick and Dollár, 2014)	48.4	
TST	54.0	53.8
TST + ILSVRC-dets	-	**55.1**

Table 5.1: CorLoc on ILSVRC-20; DT: direct transfer; DT+MIL: direct transfer plus MIL. TST is our method; ILSVRC-dets: Sec. 5.7.2, last paragraph. The transfers are guided by either appearance (APP) or semantic (SEM) class similarity.

Class	monk	pizz	rabb	stra	tpla	turt	wiro	whal					Avg. (8)
LSDA	22.9	27.6	40.2	6.8	19.1	31.9	8.6	**20.3**					22.2
Deep MIL+Obj.	18.0	29.7	32.8	19.6	27.0	27.0	5.9	2.9					20.4
+TST (APP)	23.8	32.5	40.7	**24.8**	28.6	25.1	**9.9**	5.1					23.8
+TST (SEM)	22.9	31.2	**45.2**	18.7	**30.3**	28.2	8.1	6.2					23.9
+ ILSVRC-dets	**24.5**	**33.9**	44.5	18.4	28.4	**32.1**	9.9	5.7					**24.7**
Class	ant	bbed	bask	bear	burr	butt	cell	cmak	efan	elep	gfis	gcar	Avg. (20)
LSDA	-	-	-	-	-	-	-	-	-	-	-	-	-
Deep MIL+Obj.	39.2	24.2	0.2	13.0	16.5	28.9	29.7	8.9	**39.1**	34.4	9.1	40.3	22.3
+TST (APP)	**39.9**	**31.0**	0.6	16.8	11.3	32.2	32.0	6.0	34.9	38.4	**13.6**	**65.1**	25.6
+TST (SEM)	34.1	26.8	0.6	19.7	16.8	31.7	**32.6**	8.6	31.2	37.2	11.5	57.8	25.0
+ ILSVRC-dets	34.1	24.7	**3.3**	**21.5**	**18.6**	**35.1**	**32.6**	**9.1**	32.9	**38.8**	11.1	58.5	25.9

Table 5.2: mAP performance on the test set of ILSVRC-20. All our methods start from DeepMIL with objectness. For comparison we show the performance on the 8 classes common to our target set and that of LSDA (Hoffman et al., 2014) (top) and the 12 remaining classes not evaluated by LSDA (bottom).

Evaluation protocol. We quantify localization performance in the target training set with the CorLoc measure (Bilen et al., 2015; Cinbis et al., 2016; Deselaers et al., 2010; Shi et al., 2015; Wang et al., 2015; Bilen and Vedaldi, 2016). We quantify object detection performance on the target test set using mean Average Precision (mAP). As in most previous WSOL methods (Bilen et al., 2014b, 2015; Cinbis et al., 2014, 2016; Deselaers et al., 2010; Russakovsky et al., 2012; Siva and Xiang, 2011; Song et al., 2014a,b; Wang et al., 2015), our scheme returns exactly one bounding box per class per training image. At test time the object detector is capable of localizing multiple objects of the same class in the same image (and this is captured in the mAP measure).

Source set: PASCAL Context. PASCAL Context (Mottaghi et al., 2014) augments PASCAL VOC 2010 (Everingham et al., 2010) with class labels at every pixel. As in Mottaghi et al. (2014), we select the 59 most frequent classes. We categorize them into *things* and *stuff*. There are 40 thing classes, including the original 20 PASCAL classes and new classes such as *book*, *cup* and *window*. There are 19 stuff classes, such as *sky*, *water* and *grass*. We train the semantic segmentation model (Sec. 5.4.1) on the *train* set of $\mathcal{A}$ and set the score parameters (Sec. 5.5.6) on the *val* set, using the 20 PASCAL classes from $\mathcal{A}$ as targets.

Target set: ILSVRC-20. The ILSVRC (Russakovsky et al., 2015) dataset originates from the ImageNet dataset (Deng et al., 2009), but is much harder (Russakovsky et al., 2015). As the target training set we use the *train60k* subset (Girshick et al., 2014) of ILSVRC 2014. As the target test set we use the 20k images of the validation set. To conduct WSOL on *train60k*, we carefully select 20 target classes: *ant*, *baby-bed*, *basketball*, *bear*, *burrito*, *butterfly*, *cello*, *coffee-maker*, *electric fan*, *elephant*, *goldfish*, *golfcart*, *monkey*, *pizza*, *rabbit*, *strainer*, *tape-player*, *turtle*, *waffle-iron* and *whale*. ILSVRC-20 contains 3,843 target training set images and 877 target test set images. This selection is good because: (1) they are visually considerably different from any source class; (2) they appear against similar background classes as the source classes, so we can show the benefits of stuff transfer; (3) they are diverse, covering a broad range of object types.

Target set: COCO-07. The COCO 2014 (Lin et al., 2014) dataset has fewer object classes (80) than ILSVRC (200), but more instances. COCO is generally more difficult than ILSVRC for detection, as objects are smaller (Lin et al., 2014). There are also more instances per image: 7.7 in COCO compared to 3.0 in ILSVRC (Lin et al., 2014).

We select 7 target classes to carry out WSOL: *apple*, *giraffe*, *kite*, *microwave*, *snowboard*, *tennis racket* and *toilet*. COCO-07 contains 11,489 target training set images and 5,443 target test set images.

Target set: PASCAL VOC 2007. The PASCAL VOC 2007 (Everingham et al., 2015) dataset is one of the most important object detection datasets. It includes 5,011 training (*trainval*) images and 4,952 test images, which we directly use as our target training set and target test set, respectively. For our experiments we use all 20 thing classes in VOC 2007. Since the thing classes in our source set (PASCAL Context) overlap with those of VOC 2007, when doing our TST transfer to a target class we remove it from the sources. For example, when we transfer to *dog* in VOC 2007, we remove *dog* from the FCN model trained on PASCAL Context.

5.7.2 ILSVRC-20

Table 5.1 presents results for our method (TST) and several alternative methods on ILSVRC-20.

Our transfer (TST). Our results (TST) vary depending on the underlying class similarity measure used, either appearance (APP) or semantic (SEM) (Sec. 5.4.2). TST (APP) leads to slightly better results than TST (SEM). We achieve a +7% improvement in CorLoc (46.7) compared to Basic MIL without objectness, and +5% improvement (52.7) over Basic MIL with objectness. Hence, our transfer method is effective, and is complementary to objectness. Fig. 5.5 shows example localizations by Basic MIL with objectness and TST (APP). We present a detailed ablation study of the LW, AW and CW scores in the supplementary material in Sec. 5.10.

Comparison to direct transfer (DT). We compare here to a simpler way to transfer knowledge. We train a fully supervised object detector for each source thing class. Then, for every target class we find the most similar source class from the 40 PASCAL Context thing classes, and use it to directly detect the target objects. For the appearance similarity measure (APP) all NN classes of ILSVRC-20 are part of PASCAL VOC and PASCAL Context. Therefore we have bounding box annotations for these classes. However, for the semantic similarity measure (SEM) not all NN classes of ILSVRC-20 are part of PASCAL VOC. Therefore we do not have bounding box annotations for these classes and cannot apply DT. DT is similar to the 'transfer only' method in Rochan and Wang (2015) (see Sec. 4.2 and Table 2 in Rochan and Wang (2015)).

As Table 5.1 shows, the results are quite poor as the source and target classes are visually quite different, e.g. the most similar class to *ant* according to APP is *bird*; while for *waffle-iron*, it is *table*; for *golfcart*, it is *person*. This shows that the transfer task we address (from PASCAL Context to ILSVRC-20) is challenging and cannot be solved by simply using object detectors pre-trained on the source classes.

Comparison to direct transfer with MIL (DT+MIL). We improve the direct transfer method by using the DT detector to score all proposals in a target image, and then combining this score with the standard SVM score for the target class during the MIL re-localization step. This is very similar to the full method of Rochan and Wang (2015) and is also close to Guillaumin and Ferrari (2012). The main difference from Rochan and Wang (2015) is that we train the target class' SVM model in an MIL framework (Sec. 5.6), whereas Rochan and Wang (2015) simply trains it by using proposals with high objectness as positive samples.

As Table 5.1 shows, DT+MIL performs substantially better than DT alone, but it only slightly exceeds MIL without transfer, again due to the source and target classes being visually different (+1.5% over Basic MIL with objectness). Importantly, our method (TST) achieves higher results, demonstrating that it is a better way to transfer knowledge (+5% over Basic MIL with objectness).

Deep MIL. As Table 5.1 shows, Deep MIL improves slightly over Basic MIL (from 47.6 to 48.4, both with objectness). When built on Deep MIL, our TST transfer raises CorLoc to 54.0 (APP) and 53.8 (SEM), a +5% improvement over Deep MIL (confirming what we observed when building on Basic MIL). Table 5.2 shows the mAP of Deep MIL and our method (TST) on the test set. The observed improvements in CorLoc on the training set nicely translate to better mAP on the test set (+3.3% over Deep MIL).

Comparison to LSDA (Hoffman et al., 2014). We compare to LSDA (Hoffman et al., 2014), which trains fully supervised detectors for 100 classes of the ILSVRC 2013 dataset (sources) and transfers to the other 100 classes (targets). We report in Table 5.2 the mAP on the 8 classes common to both their target set and ours. On these 8 classes, we improve on Hoffman et al. (2014) by +1.7% mAP while using a substantially smaller source set (5K images in PASCAL Context, compared to 105K images in their 100 source classes from ILSVRC 2013).

Furthermore, we can also incorporate detectors for their 100 source classes in our method, in a similar manner as for the DT+MIL method. For each target class we use

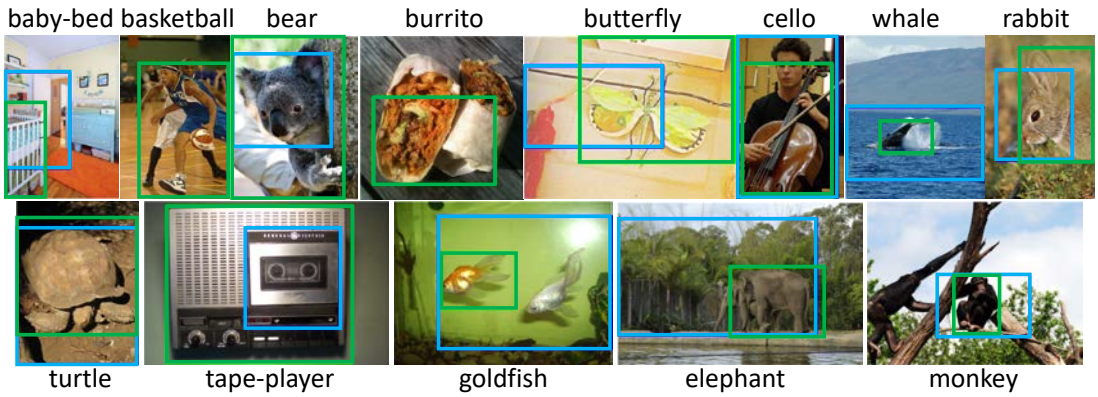


Figure 5.5: We show localizations on ILSVRC-20 of Basic MIL with objectness (blue) and our TST (APP) method (green).

Method	training (CorLoc)	test (mAP)
Deep MIL + Obj.	15.8	9.1
+TST (SEM)	18.0	11.0
+TST (APP)	**18.8**	**11.3**

Table 5.3: CorLoc and mAP on COCO-07. Objectness (Zitnick and Dollár, 2014) is added on top of the baseline. TST (SEM) and TST (APP) are separately added to the baseline with objectness.

the detector of the 3 most similar source classes as a proposal scoring function during MIL's re-localization step. We choose the SEM measure to guide the transfer as it is fast to compute. This new scoring function is referred to as ILSVRC-dets in Table 5.1 and 5.2. When using the ILSVRC-dets score, our mAP improves further, to a final value $+2.5\%$ better than LSDA (Hoffman et al., 2014).

5.7.3 COCO-07

Table 5.3 presents results on COCO-07, which is a harder dataset. Compared to Deep MIL with objectness, our transfer method improves CorLoc by $+3.0\%$ and mAP by $+2.2\%$ (APP). We present a detailed ablation study of the LW, AW and CW scores in the supplementary material in Sec. 5.10.

5.7.4 PASCAL VOC 2007

Table 5.4 presents results on PASCAL VOC 2007. As our baseline system, we use both objectness and multifolding (Cinbis et al., 2016) in Deep MIL. This performs at 50.7 CorLoc and 28.1 mAP. Our transfer method TST strongly improves CorLoc to 59.9 (+9.2%) and mAP to 33.8 (+5.7%).

Comparison to Rochan and Wang (2015). They present results on this dataset in a transfer setting, by using detectors trained in a fully supervised setting for all 200 classes of ILSVRC (excluding the target class). Adopting their protocol, we also use those detectors in our method (analog to the LSDA comparison above). This leads to our highest CorLoc of 60.8, which outperforms Rochan and Wang (2015), as well as recent WSOL works (Wang et al., 2015; Bilen and Vedaldi, 2016; Cinbis et al., 2016) (which do not use such transfer) at the time of publication of our ICCV 2017 (Shi et al., 2017a) paper. For completeness, we also report the corresponding mAPs. Our mAP 34.5 matches the result of Bilen and Vedaldi (2016) based on their 'S' neural network, which corresponds to the AlexNet we use. They propose an advanced WSOL technique that integrates both recognition and detection tasks to jointly train a weakly supervised deep network, whilst we build on a weaker MIL system. We believe our contributions are complementary: we could incorporate our TST transfer cues into their WSOL technique and get even better results.

Finally, we note that our experimental protocol guarantees no overlap in either images nor classes between source and target sets (Sec. 5.7.1). However, in general VOC 2007 and PASCAL Context (VOC 2010) share similar attributes, which makes this transfer task easier in our setting.

5.8 Future work

In this section we present future work related to this chapter.

Finding stuff. We used stuff and things in first- and second-order transfer to find things. We have not used this model to find stuff, primarily because the object localization task is not suitable for stuff classes. Future work could investigate this option in semantic segmentation. Most components of this work can be adapted to semantic segmentation of stuff and things. The LW, CW and AW weighting schemes can be modified to use masks instead of boxes (see Sec. 5.5.2, 5.5.3 and 5.5.4). To compute

Method	ILSVRC-dets	CorLoc	mAP
Wang et al. (2015)		48.5	31.6
Bilen and Vedaldi (2016) (S)		54.2	**34.5**
Cinbis et al. (2016)		54.2	28.6
Rochan and Wang (2015)	✓	58.8	-
Deep MIL + Obj. + MF		50.7	28.1
+TST (SEM)		59.9	33.8
+TST (SEM)	✓	**60.8**	**34.5**

Table 5.4: Performance on PASCAL VOC 2007. We start from Deep MIL with objectness (Zitnick and Dollár, 2014) and multifolding (Cinbis et al., 2016) as a baseline. Then we add our method TST (SEM) to it. Rochan and Wang (2015) do not report mAP. (S) denotes the S model (roughly AlexNet) in Bilen and Vedaldi (2016), which corresponds to the network architecture we use in all experiments. ILSVRC-dets indicates using detectors trained from ILSVRC during transfer.

the final segmentation, we can follow the standard protocol of computing the class with the highest score for each pixel.

End-to-end training. A recurring theme in this thesis is end-to-end training (see Sec. 2.1.4.2 and Chapter 3 and 4). End-to-end training typically leads to higher performance and requires less (re-)engineering efforts. Future work could modify the part of our model that transfers knowledge to the target set (see the box on the right in Fig. 5.1) to enable end-to-end training. This requires converting our three weighting schemes (see Sec. 5.5.2, 5.5.3 and 5.5.4) into differentiable layers in a neural network. Additionally, one could modify our method to learn our *entire* model end-to-end, including the source knowledge (see Sec. 5.4). While this may yield higher performance on the target dataset it is trained for, it presumably decreases the performance on other target datasets. Currently we determine the model weighting parameters via cross-validation on a held-out set (see Sec. 5.5.6). Future work could learn those parameters to reduce the manual tuning effort required when adapting our method to new datasets.

Annotation time and performance. In Sec. 5.7.2 we have shown that our method outperforms LSDA (Hoffman et al., 2014) in that particular setting. We use 21x less source images for training than LSDA, albeit with a stronger form of supervision (segmentations versus boxes). Future work could compare both methods in

terms of the trade-off between annotation time and performance. Another factor to take into account is that both methods use different source datasets. While PASCAL Context (Mottaghi et al., 2014) has stuff and thing annotations, ILSVRC 2013 (Russakovsky et al., 2015) only contains things. Particularly interesting is the question how the performance scales to much larger source datasets (such as COCO-Stuff in Chapter 6). We hypothesize that our method scales better, as we also use non appearance-based cues (semantic similarity in Sec. 5.4.2 and co-occurrence in Sec. 5.4.3) and a more accurately delineated source set.

5.9 Conclusion

We present weakly supervised object localization using things and stuff transfer. We transfer knowledge by training a semantic segmentation model on the source set and using it to generate thing and stuff maps on a target image. Class similarity and co-occurrence relations are also transferred and used as weighting functions. We devise three proposal scoring schemes on both thing and stuff maps and combine them to produce our final TST score. We plug the score into an MIL pipeline and show significant improvements on the ILSVRC-20, VOC 2007 and COCO-07 datasets. We compare favorably to two previous transfer works (Rochan and Wang, 2015; Hoffman et al., 2014).

5.10 Supplementary material

This supplemental material complements the chapter in two aspects: 1) We provide the proxy measures to detail the parameter learning process of our method. 2) We conduct an ablation study to demonstrate the contribution of each component in the proposed system.

5.10.1 Proxy measures

We propose two proxy measures to jointly learn the score parameters by maximizing the performance over the entire validation set in $\mathcal{A}$ (Sec. 5.5.6):

1. Rank: the highest rank of any proposal whose intersection-over-union (IOU) with ground-truth bounding box is > 0.5.

2. CorLoc@1: the percentage of images in which the highest scoring proposal localizes an object of the target class correctly (IOU > 0.5).

These two measures characterize well whether a proposal scoring function gives a higher score to the target objects than to other proposals. Hence they are good proxy measures for their usefulness within MIL. The behavior of the proposal scoring functions (Eqn. 5.5) depends on the class similarity measure used within them. Referring to Sec. 5.4.2, the guided similarities can be either appearance (APP) or semantic similarity (SEM).

Results. We notice that roughly the same parameters are obtained from both criteria. Now we test how well they work on two of our target sets: ILSVRC-20 and COCO-07. We gradually add each proposal scoring scheme from Sec. 5.5.2 – 5.5.4 and denote them by +LW, +CW, and +AW in Fig. 5.6. Both Rank and CorLoc@1 are gradually improved: using APP we achieve the highest CorLoc@1: 22.9 on ILSVRC-20 and 6.2 on COCO-07; and the highest Rank: 0.94 on ILSVRC-20 and 0.89 on COCO-07. SEM is lower than APP: 19.2 and 4.3 in terms of CorLoc@1, and 0.94 and 0.83 in terms of Rank, on ILSVRC-20 and COCO-07, respectively. Comparing our proposal scoring schemes with a modern version of objectness (Zitnick and Dollár, 2014), we see that both perform similarly well. In Sec. 5.7.2 and 5.7.3 we integrate our scheme with objectness and achieve a big improvement (+5%), which shows that both are complementary.

5.10.2 Ablation study

We report here an ablation study to offer the justification of each component in our proposed system. We incorporate the LW, AW and CW scores (Sec. 5.5.2 – 5.5.4) separately into the Basic MIL framework (Sec. 5.7.1). We report experiments on ILSVRC-20 in Table 5.5 in the same protocol as in Table 5.1 (guided by appearance similarity APP column). The three scores bring +0.9%, +3.3%, and +3.2% CorLoc on top of Basic MIL's 39.7%. This demonstrates that each individual score brings an improvement. Moreover, we also tried combining multiple scores: LW+CW reaches 44.0%, AW+CW reaches 45.8%, and using all three scores AW+LW+CW gives us the highest CorLoc 47.6%. Here we can see that LW brings an additional improvement of +1.8% when added to AW+CW. This shows that by carefully designing and integrating each component into our system, we are able to boost the overall performance over each individual component or any two of them.

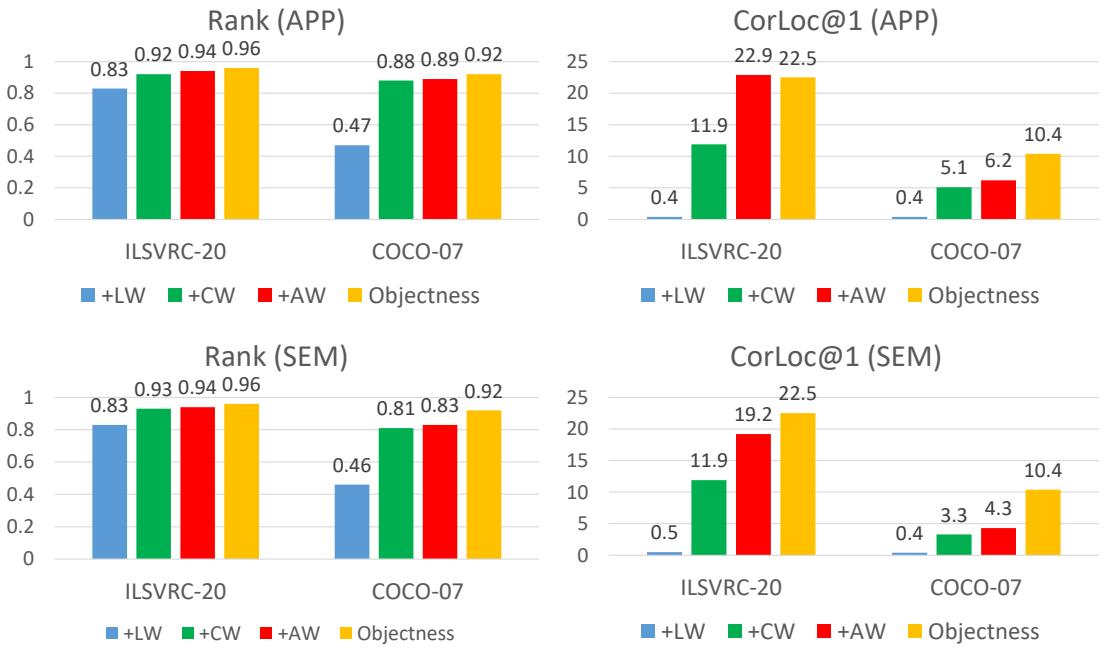


Figure 5.6: Proxy measures on ILSVRC-20 and COCO-07. Our transfer is guided by the appearance similarity (top: APP) as well as the semantic similarity (bottom: SEM). Performance is measured by Rank (left) and CorLoc@1 (right).

Method	LW	CW	AW	CorLoc
Basic MIL				39.7
	+			40.6
		+		43.0
			+	42.9
	+	+		44.0
		+	+	45.8
	+	+	+	47.6

Table 5.5: Ablation study on ILSVRC-20. LW: label weighting; CW: contrast weighting; AW: area weighting. We start from Basic MIL and incorporate LW, CW, AW, or any of their combination into it. We report the CorLoc.

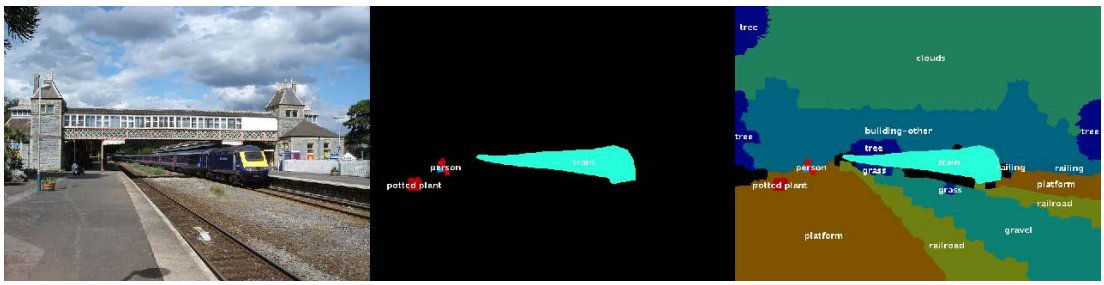
Chapter 6

COCO-Stuff: Thing and stuff classes in context

6.1 Introduction

Most of the recent object detection efforts have focused on recognizing and localizing thing classes, such as *cat* and *car*. Such classes have a specific size (Forsyth et al., 1996; Heitz and Koller, 2008) and shape (Forsyth et al., 1996; Tighe and Lazebnik, 2013a; Uijlings et al., 2013; Mottaghi et al., 2013; Endres and Hoiem, 2014; Dai et al., 2015b), and identifiable parts (e.g. a *car* has *wheels*). Indeed, the main recognition challenges (Everingham et al., 2015; Russakovsky et al., 2015; Lin et al., 2014) are all about things. In contrast, much less attention has been given to stuff classes, such as *grass* and *sky*, which are amorphous and have no distinct parts (e.g. a piece of *grass* is still *grass*). In this chapter we ask: Is this strong focus on things justified?

To appreciate the importance of stuff, consider that it makes up the majority of our visual surroundings. For example, *sky*, *walls* and most *ground* types are *stuff*. Furthermore, stuff often determines the type of a scene, so it can be very descriptive for an image (e.g. in a beach scene the *beach* and *water* are the essential elements, more so than *people* and *volleyball*). Stuff is also crucial for reasoning about things: Stuff captures the 3D layout of the scene and therefore heavily constrains the possible locations of things. The contact points between stuff and things are critical for determining depth ordering and relative positions of things, which supports understanding the relations between them. Finally, stuff provides context helping to recognize small or uncommon things, e.g. a metal thing in the sky is likely an *aeroplane*, while a metal thing in the water is likely a *boat*. For all these reasons, stuff plays an important role



A large long **train** on a steel **track**.
A blue and yellow transit **train** leaving the **station**.
A **train** crossing beneath a city **bridge** with brick **towers**.
A **train** passing by an over **bridge** with a railway **track** (..).
A **train** is getting ready to leave the train **station**.

Figure 6.1: (left) An example image, (middle) its thing annotations in COCO (Lin et al., 2014) and (right) enriched stuff and thing annotations in COCO-Stuff. Just having the *train*, *person*, *bench* and *potted plant* does not tell us much about the context of the scene, but with stuff and thing labels we can infer the position and orientation of the *train*, its stuff-thing interactions (*train* leaving the *station*) and thing-thing interactions (*person* waiting for a different *train*). This is also visible in the captions written by humans. Whereas the captions only mention one thing (*train*), they describe a multitude of different stuff classes (*track*, *station*, *bridge*, *tower*, *railway*), stuff-thing interactions (*train* leaving the *station*, *train* crossing beneath a city *bridge*) and spatial arrangements (on, beneath).

in scene understanding and we feel it deserves more attention.

In this chapter we introduce the COCO-Stuff dataset, which augments the popular COCO (Lin et al., 2014) with pixel-wise annotations for a rich and diverse set of 91 stuff classes. The original COCO dataset already provides outline-level annotation for 80 thing classes. The additional stuff annotations enable the study of stuff-thing interactions in the complex COCO images. To illustrate the added value of our stuff annotations, Fig. 6.1 shows an example image, its annotations in COCO and COCO-Stuff. The original COCO dataset offers location annotations only for the *train*, *potted plant*, *bench* and *person*, which are not sufficient to understand what the scene is about. Indeed, the image captions written by humans (also provided by COCO) mention the *train*, its interaction with stuff (i.e. *track*), and the spatial arrangements of the *train* and its surrounding stuff. All these elements are necessary for scene understanding and show how COCO-Stuff offers much more comprehensive annotations.

This chapter makes the following contributions: (1) We introduce COCO-Stuff,

which augments the original COCO dataset with stuff annotations. (2) We introduce an annotation protocol for COCO-Stuff which leverages the existing thing annotations and superpixels. We demonstrate both the quality and efficiency of this protocol (Sec. 6.3). (3) Using COCO-Stuff, we analyze the role of stuff from multiple angles (Sec. 6.4): (a) the importance of stuff and thing classes in terms of their surface cover and how frequently they are mentioned in image captions; (b) the spatial relations between stuff and things, highlighting the rich contextual relations that make COCO-Stuff unique; (c) we compare the performance of a modern semantic segmentation method on thing and stuff classes.

This chapter is a modified version of our paper published at the Computer Vision and Pattern Recognition (CVPR) 2018 (Caesar et al., 2018) conference. Hoping to further promote research on stuff and stuff-thing contextual relations, we release COCO-Stuff and the trained segmentation models online[1].

6.2 Related Work

Defining things and stuff. For a definition of stuff and things we refer the reader to Sec. 1.1. For examples of the role of stuff and things in the literature, as well as existing datasets, we refer the reader to Sec. 2.3.

Annotating datasets. For an overview of efficient technique for dataset annotation and learning, we refer the reader to Sec. 2.2. In this work we introduce a new annotation protocol to obtain high quality pixel-wise stuff annotations at low human costs by using superpixels and by exploiting the existing detailed thing annotations of COCO (Lin et al., 2014) (Sec. 6.3.2).

6.3 The COCO-Stuff dataset

The Common Objects in COntext (COCO) (Lin et al., 2014) dataset is a large-scale dataset of images of high complexity. COCO has been designed to enable the study of thing-thing interactions, and features images of complex scenes with many small objects, annotated with very detailed outlines. However, COCO is missing stuff annotations. In this chapter we augment COCO by adding dense pixel-wise stuff annotations. Since COCO is about complex, yet natural scenes containing substantial

[1] http://calvin.inf.ed.ac.uk/datasets/coco-stuff

Figure 6.2: Annotated images from the COCO-Stuff dataset with dense pixel-level annotations for stuff and things. To emphasize the depth ordering of stuff and thing classes we use bright colors for thing classes and darker colors for stuff classes.

areas of stuff, COCO-Stuff enables the exploration of rich relations between things and stuff. Therefore COCO-Stuff offers a valuable stepping stone towards complete scene understanding.

Fig. 6.2 presents several annotated images from the COCO-Stuff dataset, showcasing the complexity of the images, the large number and diversity of stuff classes, the high level of accuracy of the annotations, and the completeness in terms of surface coverage of the annotations. We have annotated all 164K images in COCO 2017: training (118K), val (5K), test-dev (20K) and test-challenge (20K).

6.3.1 Defining stuff labels.

COCO-Stuff contains 172 classes: 80 thing, 91 stuff, and 1 class *unlabeled*. The 80 thing classes are the same as in COCO (Lin et al., 2014). The 91 stuff classes are curated by an expert annotator. The class *unlabeled* is used in two situations: if a label does not belong to any of the 171 predefined classes, or if the annotator cannot infer the label of a pixel.

Before annotation, we choose to predefine our label set. This contrasts with a common choice in semantic segmentation to have annotators use free-form text labels (Tighe and Lazebnik, 2010, 2013b; Mottaghi et al., 2014). However, using free-form labels leads to several problems. First of all, it leads to an extremely large number of classes, many having only a handful of examples. This makes most classes unusable for recognition purposes, as observed in (Mottaghi et al., 2014; Zhou et al., 2017b). Furthermore, different annotators typically use several synonyms to indicate the same

class. These need to be merged a posteriori (Tighe and Lazebnik, 2010; Xiao et al., 2014). Even after merging, classes might not be consistently annotated. For example, PASCAL Context (Mottaghi et al., 2014) includes the classes *bridge* and *footbridge*, which are in a parent-child relationship. If one image has *bridge* annotations and another image has *footbridge* annotations, both can describe the same concept (i.e. *footbridge*), or the *bridge* can be another type of *bridge* and therefore describe a different concept. Similarly, in SIFT Flow (Liu et al., 2011) some images have *field* annotations, whereas others have *grass* annotations. These concepts are semantically overlapping, but are neither synonymous nor in a parent-child relationship. A region with a *grass field* could be annotated as *grass* or as *field* depending on the annotator.

To prevent such inconsistencies, we decided to predefine a set of mutually exclusive stuff classes, similarly to how the COCO thing classes were defined. Additionally, we organized our classes into a label hierarchy, e.g. classes like *cloth* and *curtain* have *textile* as parent, while classes like *moss* and *tree* have *vegetation* as parent (Fig. 6.3). The super-categories *textile* and *vegetation* have *indoor* and *outdoor* as parents, respectively. The top-level nodes in our hierarchy are generic classes *stuff* and *thing*.

To choose our set of stuff labels, the expert annotator used the following criteria: stuff classes should (1) be mutually exclusive; (2) in their ensemble, cover the vast majority of the stuff surface appearing in the dataset; (3) be frequent enough; (4) have a good level of granularity, around the base level for a human. However, these criteria conflict with each other: if we label all vegetations as *vegetation*, the labels are too general. On the other extreme, if we create a separate class for every single type of *vegetation*, the labels are too specific and infrequent. Therefore, as shown in Fig. 6.3, for every super-category like *vegetation*, we explicitly list its most frequent subclasses as choices for the annotator to pick (e.g. *straw*, *moss*, *bush* and *grass*). And there is one additional subclass *vegetation-other* to be picked to label any other case of *vegetation*. This achieves the coverage goal, while avoiding to scatter the data over many small classes. For some super-categories (*floor*, *wall* and *ceiling*) we are particularly interested in the material they are made of. Therefore we include the material type in the class definition (e.g. *wall-brick*, *wall-concrete* and *wall-wood*). This enables further analysis of the materials present in a scene.

Our label set fulfills all design criteria (1-4): (1) the mutual exclusivity of labels is by design and enforced through having annotators only use the leaves of our hierarchy as labels (Fig. 6.3). For the other criteria we need to look at pixel-level frequencies after dataset collection: (2) only 6% of the pixels are *unlabeled*, which is satisfac-

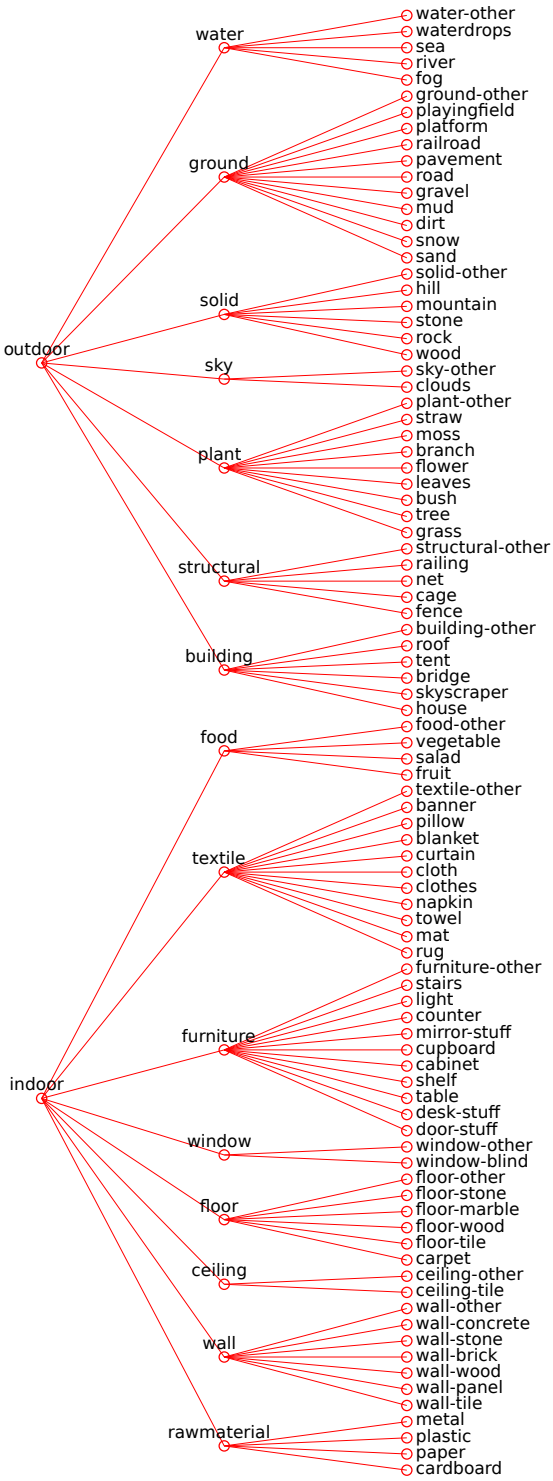


Figure 6.3: The stuff label hierarchy of the COCO-Stuff dataset. Stuff classes are divided into *outdoor* and *indoor*, each further divided into super-categories (e.g. *floor*, *plant*), and finally into leaf-level classes (e.g. *marble floor, grass*). The labels used by the annotators form the leaf nodes of the tree. Furniture classes can be interpreted as either things or stuff, depending on the imaging conditions. A full list of descriptions is available online[1].

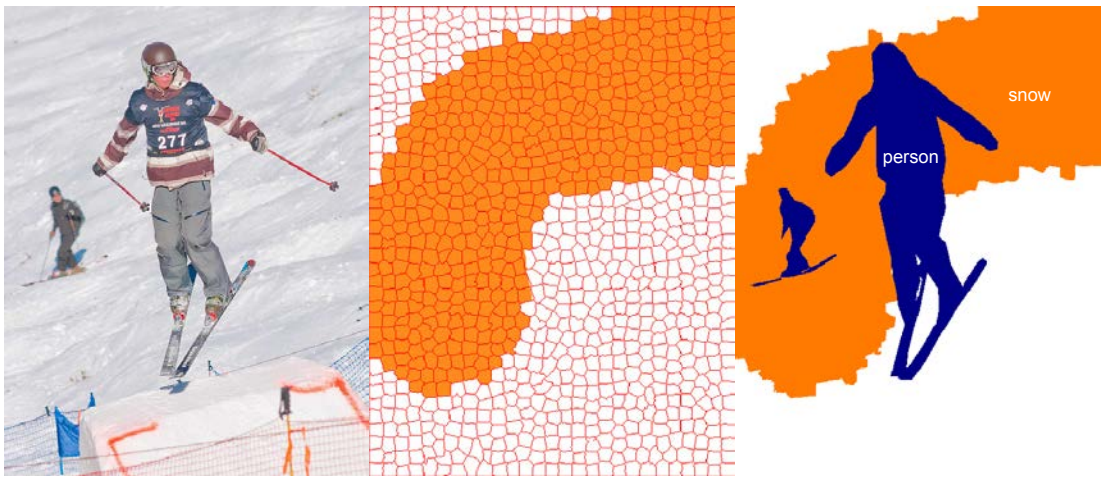


Figure 6.4: Example of a) an image, b) the superpixel-based stuff annotation and c) the final labeling. The annotator can quickly annotate large stuff regions (*snow*) with a single mouse stroke using a paintbrush tool. Thing (*person*) annotations are copied from the COCO dataset. The transparency of each layer can be regulated to get a better overview. This approach dramatically reduces annotation time and yields a very accurate labeling, especially at stuff-thing boundaries.

tory; (3, 4) interestingly, all our stuff classes have pixel frequencies in the same range of the COCO thing classes (Fig. 6.5) and they also follow a similar distribution and granularity (Fig. 6.3). Intuitively, having both thing and stuff classes follow similar distributions makes the dataset well suited to analyze stuff-thing relations.

6.3.2 Annotation protocol and analysis

Protocol. We developed a very efficient protocol, specialized for labeling stuff classes at the pixel-level. We first partition each image into 1,000 superpixels using SLICO (Achanta et al., 2012), which adheres very well to boundaries and gives superpixels of homogeneous size (Fig. 6.4). Superpixels remove the need for manually delineating the exact boundaries between two regions of different classes. As superpixels respect boundaries, it is enough to mark which superpixels belong to which class, which is a lot faster to do. Moreover, the evenly spaced and sized SLICO superpixels result in a labeling task natural for humans (as opposed to superpixel algorithms which yield regions that greatly vary in size (Felzenszwalb and Huttenlocher, 2004)). We accelerate the annotation process by providing annotators a size-adjustable paintbrush tool, which enables labeling large regions of stuff very efficiently (Fig. 6.4b).

We improve annotation efficiency even further by leveraging the highly accurate

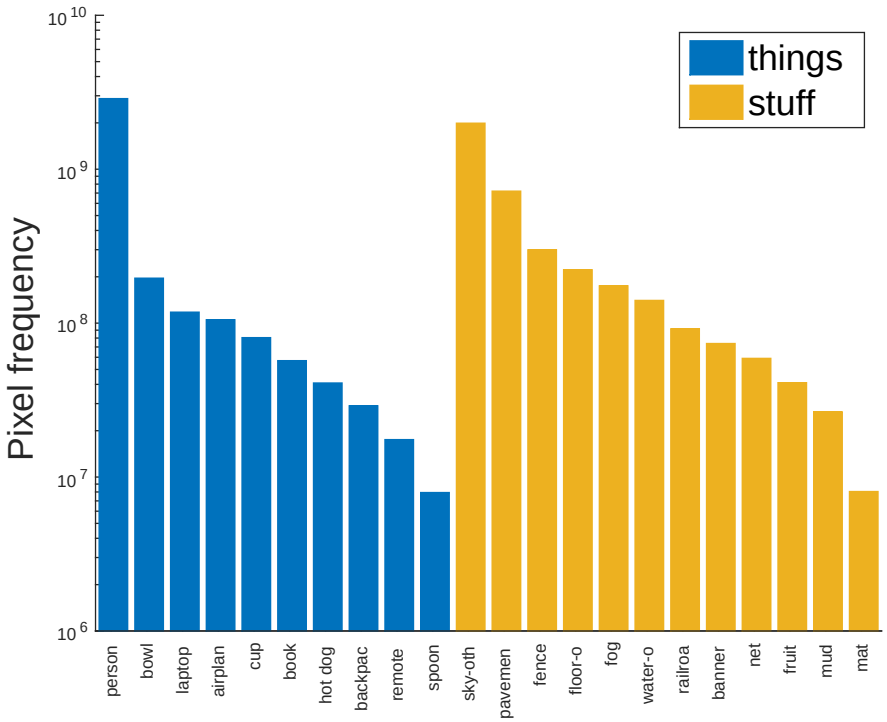


Figure 6.5: Pixel-level frequencies for some of the classes in the trainval set of COCO-Stuff. For clarity, we show about $1/8$ of all classes. We can see that stuff and thing classes follow a similar pixel frequency distribution.

thing outlines available from COCO (Lin et al., 2014) (Fig. 6.4c). We show annotators images with *thing overlays*, and pixels belonging to things are clamped and unaffected by the annotator's brush. This results in a lightweight experience, where the annotator merely needs to select a stuff class (like *snow*) and brush over the foreground object. In fact, because of the high annotation accuracy of COCO things, our technique results in extremely precise stuff outlines at stuff-thing boundaries, often beyond the accuracy of superpixel boundaries.

As a final element in our protocol, we present our stuff labels to the annotators using the full hierarchy. In initial trials we found that, compared to presenting them in a list, this reduces the look-up time of labels significantly. This annotation protocol yields an annotation time of only three minutes to annotate stuff in one of the COCO images, which are very complex (Fig. 6.2). We release the superpixels and the annotation tool online to allow for further analysis.

We annotated 10K images with our protocol using in-house annotators. Afterwards, we collaborated with the startup Mighty AI to adapt our protocol for crowdsourcing and annotate all remaining images of COCO-Stuff.

Analysis of superpixels. We study here the quality-speed trade-off of using superpixels. We ask a single annotator to annnotate 10 COCO images three times, once for each of three different modalities: (1) superpixel annotation, as we do for COCO-Stuff; (2) polygon annotation, the de facto standard (Cordts et al., 2016; Mottaghi et al., 2014; Zhou et al., 2017b) and (3) freedraw annotation, which consists of directly annotating pixels with a very accurate size-adjustable paintbrush tool, but *without* aid from superpixels. The freedraw annotations attempt to get as close to pixel-level accuracy as possible, and we use them as ground-truth reference in this analysis.

Table 6.1 shows the results for superpixel, polygon and freedraw annotation. Compared to the freedraw reference, polygons and superpixels are much faster (1.5x and 2.8x). Computing pixel-level labeling agreement w.r.t. freedraw reveals that both polygons and superpixels lead to very accurate annotations (96%-97%). We also asked the annotator to re-annotate the images with the same modality, enabling to measure 'self agreement'. Interestingly the self agreement of freedraw is in the same range as the agreement of superpixels and polygons w.r.t. freedraw. This shows that the differences across annotation modalities are of similar magnitude to the natural variations within a single modality, even by a single annotator. Hence, all three modalities are about as accurate.

Furthermore we simulate our stuff annotation protocol on two other datasets which were originally annotated with polygons: SIFT Flow (Liu et al., 2011) and PASCAL Context (Mottaghi et al., 2014). For each image we label each superpixel with the majority stuff label in the ground-truth annotations. We then overlay the existing thing annotations. This protocol achieves 98.3% agreement with the ground-truth on SIFT Flow and 98.4% on PASCAL Context. These findings show that superpixel annotation is faster than conventional polygon annotation, while providing almost the same annotations.

We found that a dominant factor for the differences in annotation time across images is their boundary complexity. Boundary complexity is defined as the ratio of pixels that have any neighboring pixel with a different semantic label (as in the boundary evaluation in Caesar et al. (2016b); Kohli et al. (2009); Krähenbühl and Koltun (2011)). Fig. 6.6 analyzes the relationship between boundary complexity and annotation time of an image using different annotation modalities. The linear trendlines show that there is a clear correlation between annotation time and boundary complexity. We can see that the slopes of the freedraw and polygon annotation trendlines are 3.4x and 2.0x steeper than for superpixels. This is one of the main reasons why superpixels

Modality	Speedup	Reference agreement	Self agreement
Superpixels	2.8	96.1%	98.7%
Polygons	1.5	97.3%	97.0%
Freedraw	1.0	-	96.6%

Table 6.1: A quantitative comparison of different stuff annotation modalities. We use freedraw annotation as a reference in the 'Speedup' and 'Reference agreement' columns. The self-agreement between repeated runs of the same annotation modality decreases with weaker constraints on the possible labelings.

yield such big improvements in annotation time on average.

Analysis of thing overlays. We analyze thing overlays in terms of the annotation speedup they bring and the quality they lead to. For this we perform superpixel and freedraw annotation with and without thing overlays. We achieve significant speedups when using thing overlays with freedraw annotation (1.8x) and also with superpixel annotation (1.2x). Furthermore, the agreement of superpixel annotation w.r.t. the freedraw reference is identical with and without thing overlays (96.1% in both cases). This shows that thing overlays achieve a significant speedup without any loss in quality.

Moreover, 46.8% of the boundary pixels in COCO-Stuff have a neighboring pixel that belongs to a thing class. Therefore using thing overlays significantly decreases the boundary complexity and leads to a larger speedup for freedraw annotation than for superpixel annotation.

Across-annotator agreement. Following Zhou et al. (2017b); Cordts et al. (2016) we annotate 30 images by 3 annotators each. For each image we compute the label agreement between each pair of annotators and average over all pairs. The mean label agreement in COCO-Stuff is 73.6%, compared to 66.8% for ADE20K (Zhou et al., 2017b).

6.3.3 Comparison to other datasets.

COCO-Stuff has the largest number of images of any semantic segmentation dataset (164K). In particular, MSRC 21 (Shotton et al., 2006), KITTI (Geiger et al., 2012), CamVid (Brostow et al., 2009), SIFT Flow (Liu et al., 2011) and NYUD (Silberman

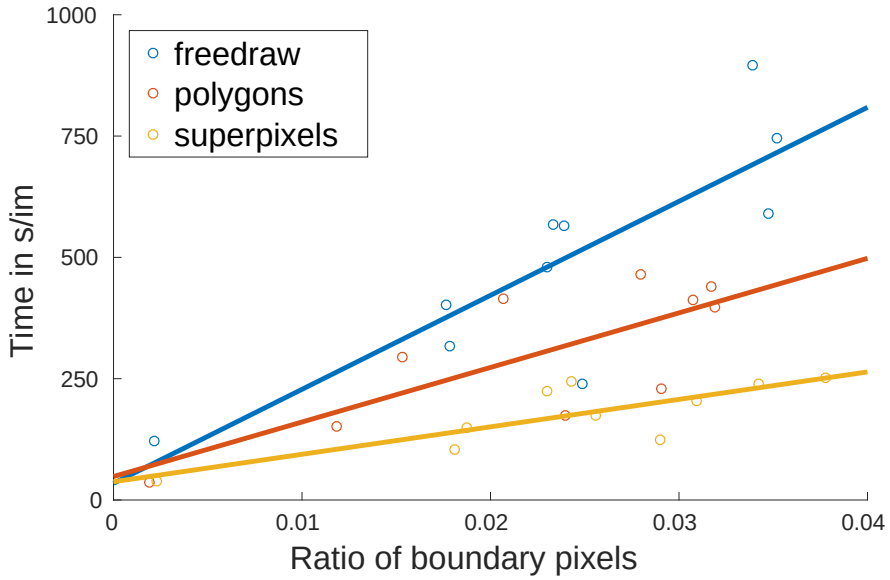


Figure 6.6: Annotation time versus image boundary complexity. Each circle indicates an image annotated using one of three modalities. The trendlines show that annotation time for some modalities increases faster with boundary complexity than for others.

et al., 2012) all have less than 5,000 images (Table 2.2). COCO-Stuff is also much richer in both the number of stuff and thing classes than MSRC 21 (Shotton et al., 2006), KITTI (Geiger et al., 2012), CamVid (Brostow et al., 2009), Cityscapes (Cordts et al., 2016) and SIFT Flow (Liu et al., 2011). Compared to the Barcelona (Tighe and Lazebnik, 2010) and LM+SUN (Tighe and Lazebnik, 2013b) datasets, it has 3× and 2× more stuff classes, respectively.

PASCAL Context (Mottaghi et al., 2014) and ADE20K (Zhou et al., 2017b) are the most similar datasets to COCO-Stuff. On the surface they appear to have a very large numbers of classes (540 and 2,693), but in practice most classes are rare. The authors of those datasets define a set of classes deemed *usable* for experiments (i.e. the most frequent 60 classes in PASCAL Context and 150 classes in ADE20K). In Fig. 6.7 we show the number of stuff classes that occur in at least x images, for varying thresholds x, on the trainval sets of three datasets. COCO-Stuff has more usable stuff classes than PASCAL Context and ADE20K for *any* threshold, e.g. for $x = 1,000$, there are 5 stuff classes in PASCAL Context, 20 in ADE20K and 84 in COCO-Stuff. This means that 92% of the stuff classes in COCO-Stuff occur in at least $1,000$ images. Furthermore, both PASCAL Context and ADE20K use free-form label names, which lead to annotations at different granularities and hence ambiguities, as discussed in Sec. 6.3.1. In contrast, in COCO-Stuff all labels are mutually exclusive and at a comparable level

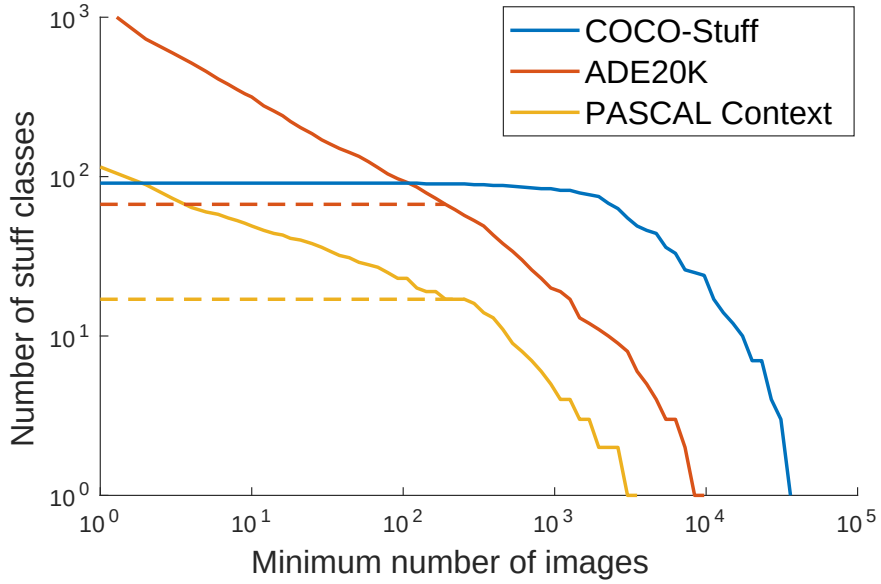


Figure 6.7: The number of stuff classes occurring in at least x images for varying thresholds of x. Solid lines indicate the full datasets, dashed lines the versions with only *usable* classes. Statistics are computed on the trainval sets of three datasets.

of granularity. Finally, PASCAL Context and ADE20K are annotated with overlapping polygons. Hence some pixels have multiple conflicting labels at the boundaries between things and stuff. In COCO-Stuff instead, each pixel has exactly one label.

To conclude, COCO-Stuff is a very large dataset of highly complex images. It has the largest number of *usable* stuff and thing classes with pixel-level annotations. Moreover, by building on COCO it also has natural language captions, further supporting rich scene understanding.

6.4 Analysis of stuff and things

In this section we leverage COCO-Stuff to analyze various relations between stuff and things: we analyze the relative importance of stuff and thing classes (Sec. 6.4.1); study spatial contextual relations between stuff and things (Sec. 6.4.2); and analyze the behavior of semantic segmentation methods on stuff and things (Sec. 6.4.3). To preserve the integrity of the test set annotations, all experiments in this section are run on the trainval set of COCO-Stuff.

Level	Stuff	Things
Pixels	**69.1%**	30.9%
Regions	**69.4%**	30.6%
Caption nouns	38.2%	**61.8%**

Table 6.2: Relative frequency of stuff and thing classes in pixel-level annotations and caption nouns in COCO-Stuff.

6.4.1 Importance of stuff and things

We quantify the relative importance of stuff and things using two criteria: surface cover and human descriptions.

Surface cover. We measure the frequencies of stuff and thing pixels in the COCO-Stuff annotations. Table 6.2 shows that the majority of pixels are stuff (69.1%). We also compute statistics for the labeled *regions* in COCO-Stuff, i.e. connected components in the pixel annotation map. We use such regions as a proxy for class instances, as stuff classes do not have instances. We see that 69.4% of the regions are stuff and 30.6% things.

Human descriptions. Although stuff classes cover the majority of the image surface, one might argue they are just irrelevant background pixels. The COCO dataset is annotated with five captions per image (Lin et al., 2014), which have been written explicitly to describe its content, and therefore capture the most relevant aspects of the image for a human. To emphasize the importance of stuff for scene understanding, we also analyze these captions, counting how many nouns point to things and stuff respectively. We use a Part-Of-Speech (POS) tagger (Toutanova et al., 2003) to automatically detect nouns. Then we manually categorize the 600 most frequent nouns as stuff (e.g. *street*, *field*, *water*, *building*, *beach*) or things (e.g. *man*, *dog*, *train*), ignoring nouns that do not represent physical entities (e.g. *game*, *view*, *day*).

Table 6.2 shows the relative frequency of these nouns. Stuff covers more than a third of the nouns (38.2%). This clearly shows the importance of stuff according to the COCO image captions.

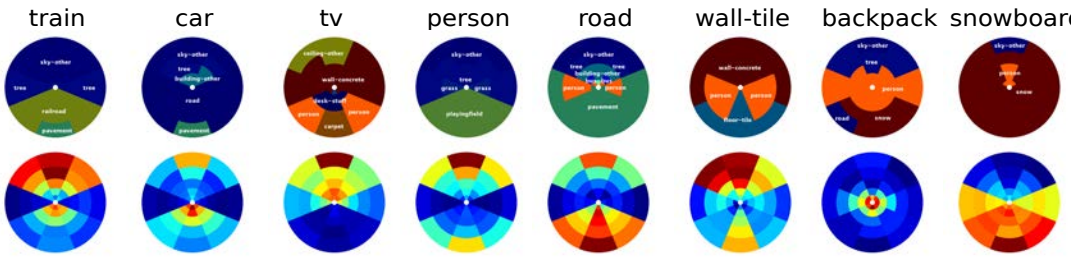


Figure 6.8: Spatial context visualizations. (Top) Each disc is for a different reference class and shows the most likely other class at each direction and distance bin. (Bottom) The conditional probabilities of the most common class in each bin, as a measure of confidence. The values are normalized for each reference class and range from low (blue) to high (red). We also show examples for classes with high (person) and low (snowboard) mean entropy.

6.4.2 Spatial context between stuff and things

Methodology. We analyze spatial context by considering the relative image position of one class with respect to another. For simplicity, here we explain how to compute the spatial context for one particular reference class, i.e. *car* (Fig. 6.8, second column). The explanation is analogous for all other classes. For every image containing a *car*, we extract a set of *car* regions, i.e. connected components of *car* pixels in the annotation map. Next we compute a histogram of image pixels surrounding the *car* regions, with two spatial dimensions (distance, angle) and one dimension for the class label. To determine in which spatial bin a certain pixel lands, we (1) compute the distance between the pixel and the nearest point in the *car* region (normalized by image size); (2) compute the relative angle with respect to the center of mass of the *car* region.

Results. Fig. 6.8 shows the spatial context of eight reference classes. This visualization reveals several interesting contextual relations. *Trains* are typically found above *railroads* (thing-stuff). *TVs* are typically found in front of *persons* (thing-thing). *Tiled walls* occur above *tiled floors* (stuff-stuff), and *roads* are flanked by *persons* on both sides (stuff-thing). Note that these contextual relations are not necessarily symmetric: most *cars* appear above a *road*, but many *roads* have other things above them.

For each reference class and spatial bin we also show the conditional probability of the most likely other class as a measure of confidence (Fig. 6.8, bottom). In most cases the highest confidence is in regions above (*sky*, *wall*, *ceiling*) or below (*road*, *pavement*, *snow*) the reference region, but rarely to the left or right. Since vertical

relations are mostly support relations (e.g. 'on top of'), this suggests that support is the most informative type of context. For some classes the highest confidence region is also very close to the reference region, often indicating that another object is attached to the reference one (*person* close to *backpack*).

As the figure shows, some classes have a rich and diverse context, composed of many other classes (e.g. *tv*, *road*), while some classes have a simpler context (e.g. *snowboards* always appear in the middle of *snow*). We quantify the complexity of a reference class as the entropy of the conditional probability distribution, averaged over all other classes and spatial bins. The classes with highest mean entropy are *wood*, *metal* and *person*, and those with the lowest are *snowboard*, *airplane* and *playingfield*. On average, we find that stuff classes have a significantly higher mean entropy than things (3.40 vs. 3.02), showing they appear in more varied contexts. We also find that the mean entropy is rather constant over distances (small: 3.21, big: 3.23) and directions (left: 3.19, right: 3.18, down: 3.20, up: 3.15).

Comparing the mean entropy of different datasets, taking into account all classes, we find that COCO-Stuff has the highest (3.22), followed by the 60 *usable* classes of PASCAL Context (2.42), the 150 *usable* classes of ADE20K (2.18) and SIFT Flow (1.20). This shows the contextual richness of COCO-Stuff.

6.4.3 Semantic segmentation of stuff and things

We now analyze how a modern semantic segmentation method (Chen et al., 2015a) performs on COCO-Stuff. We compare the performance on stuff and thing classes and hope to establish a baseline for future experiments on this dataset.

Protocol. We use the popular DeepLab V2 (Chen et al., 2015a) based on the VGG-16 network (Simonyan and Zisserman, 2015) pre-trained on the ILSVRC classification dataset (Russakovsky et al., 2015). We use the following experimental protocol: train on the 118K training images and test on the 5K val images. To evaluate performance we use four criteria commonly used in the literature (Long et al., 2015; Eigen and Fergus, 2015; Caesar et al., 2016b): (1) *pixel accuracy* is the percentage of correctly labeled pixels in the dataset, (2) *class accuracy* computes the average of the per-class accuracies, (3) *mean Intersection-over-Union (IOU)* divides the number of pixels of the intersection of the predicted and ground-truth class by their union, averaged over classes (Everingham et al., 2015), (4) *frequency weighted (FW) IOU* is per-class IOU weighted by the pixel-level frequency of each class.

Results for all images and classes. Table 6.3 shows the results using all images (row "118K (train)"). DeepLab achieves an mIOU of 33.2% over all classes. A detailed comparison of leading methods can be found online[1].

Benefits of a large dataset. One reason for the recent success of deep learning methods is the advent of large-scale datasets (Russakovsky et al., 2015; Ionescu et al., 2015; Zhou et al., 2017b). Inspired by Sun et al. (2017), we want to test whether the performance of semantic segmentation models plateaus at current dataset sizes or whether it benefits from larger datasets. Following the above protocol, we train multiple DeepLab models with different amounts of training data, keeping all training parameters fixed. Table 6.3 shows the resulting performance on the validation set (rows from 1K to 118K). We can see that for all metrics, performance significantly increases as the training set grows. We hypothesize that even deeper network architectures (He et al., 2016) could benefit even more from large training sets.

Is stuff easier than things? Several works found that stuff is easier to segment than things (Tighe and Lazebnik, 2010; Ion et al., 2011; Liu et al., 2011; Tighe and Lazebnik, 2013a; Tighe et al., 2014; Zhang et al., 2015a; Xu et al., 2015a; Zhou et al., 2017b). We argue that this is due to their choice of dataset, rather than a general observation. Most datasets only include a small number of very frequent and coarse-grained stuff classes, such as *sky* and *grass* (Table 2.2). In contrast, COCO-Stuff features a larger number of relevant stuff labels at a similar level of granularity as the existing thing labels. It has a similar number of stuff and thing classes, and a similar pixel frequency distribution for both (see Fig. 6.5).

As Table 6.3 (bottom) shows, on COCO-Stuff DeepLap performs substantially better on thing classes than on stuff. This shows that stuff is harder to segment than things in COCO-Stuff, a dataset where both stuff and things are similarly distributed. Therefore we argue that stuff is not generally easier than things.

6.5 Impact of COCO-Stuff

In this section we discuss the impact of our COCO-Stuff paper and dataset. We present works that are inspired by COCO-Stuff and briefly mention two segmentation challenges.

Training images	Class accuracy	Pixel accuracy	Mean IOU	FW IOU
1K	24.1%	46.1%	15.9%	31.0%
5K	33.8%	52.7%	23.1%	37.5%
10K	36.9%	54.6%	25.5%	39.6%
20K	40.2%	57.5%	28.6%	42.6%
40K	43.0%	61.1%	31.4%	45.7%
80K	44.9%	63.4%	32.9%	47.4%
118K (train)	**45.1%**	**63.6%**	**33.2%**	**47.6%**
stuff	33.5%	58.2%	24.0%	45.6%
things	**58.3%**	**75.7%**	**43.6%**	**58.4%**

Table 6.3: Rows 1K to 118K: Performance of Deeplab V2 with VGG-16 with varying amounts of training data. We can see that for all metrics, performance significantly increases for larger datasets. Last two rows: Performance of the same model on stuff and thing classes using all 118K training images in COCO.

Later works inspired by ours. We released an earlier version of the COCO-Stuff dataset with 10K images (Caesar et al., 2016a) in December 2016. Since then, COCO-Stuff has quickly become a standard benchmark for semantic segmentation (Shuai et al., 2017; Wang et al., 2017; Bulo et al., 2017b; Hu et al., 2017). Yoo et al. (2017) study the tasks of objectness estimation and semantic segmentation using a compound eye camera inspired by insects. They then transform the images in COCO-Stuff to simulate a compound eye camera view. Du and Davis (2017) use COCO-Stuff for portrait segmentation and editing. Smith et al. (2017) explore the linguistic distinction between *mass* and *count* nouns in the visual modality. Although related, they find that these do not strictly correspond to our stuff and thing categories, as some borderline stuff classes are countable (e.g. mountain, tree). Using their own dataset, they find that mass classes exhibit a lower intra-image and inter-image variance in CNN activations than count classes. We assume that this insight also applies to stuff and thing classes and future work could try to find semantic segmentation methods that take this into account. Furthermore, COCO-Stuff may become one of the default datasets for the panoptic segmentation task (Kirillov et al., 2018) (see Sec. 2.1). This task requires knowing the depth ordering between thing instances, which are currently being annotated by Kirillov et al. (2018).

COCO challenges. To help promote the COCO-Stuff dataset and the mission of this thesis to a broader community, we conduct segmentation challenges open to the public. At ICCV 2017 we launched a new track for COCO Stuff Segmentation, as part of the Joint Workshop of the COCO and Places Challenges. At ECCV 2018 we will launch the COCO Panoptic Segmentation challenge as part of the Joint COCO and Mapillary Recognition Challenge Workshop.

6.6 Future work

In this section we present future work related to this chapter.

Linguistic analysis of stuff and things. In Sec. 6.4.1 we studied the importance of stuff and things in image captions, counting how many nouns point to things and stuff respectively. As future work we envision a more detailed study of the linguistic role of stuff and things. The captions in Fig. 6.1 seem to indicate that things often form the subject of the sentence, thereby taking an active role. Stuff on the contrary seems to be the object of the sentence, taking a passive role. Future work could study these relations and try to find exceptions to this rule. It would be interesting to look at the subject in captions for images that do not contain any things. Furthermore, one could study the prepositions linking particular stuff-thing, thing-thing and stuff-stuff pairs (e.g. cow *on* grass). Finally, stuff and things are very related to the *mass and count* (Cheng, 1973; Fieder et al., 2014) distinction in linguistics. It would be interesting to see whether the human annotators that wrote the captions only use count quantifiers (a, two, few) for things and mass quantifiers (much, little, an amount of) for stuff. Closely related, one could look at the use of singular and plural nouns and nouns that are only have a singular form (grass).

Human-in-the-loop annotation for stuff. Many human-in-the-loop annotation schemes result in a reduced annotation time, while achieving a comparable level of quality to traditional annotation schemes (see Sec. 2.2.1). Future work could investigate how suitable methods like Polygon RNN (Castrejon et al., 2017) are for stuff annotation. Polygon-RNN is a recurrent neural network that repeatedly predicts the next point of a polygon along the boundary of a thing instance. If necessary, the annotator can correct a predicted point, which in turn updates the prediction of other points. It is not clear whether such a method can be adapted to stuff annotation, as stuff has neither clearly defined boundaries nor instances (see Sec. 1.1). Precisely because of this problem we

use our superpixel-based stuff annotation protocol in Sec. 6.3.2. Future work could create a human-in-the-loop version of our protocol, where a classifier predicts the labels for each superpixel and the user can correct them, thereby triggering an update of the labels of other superpixels. Although the SLICO (Achanta et al., 2012) superpixels that we use are essentially parameter-free, the desired number of superpixels per image is one parameter, which we fixed to 1000. A human-in-the-loop scheme could let the user define this parameter per image or even for each part of the image.

Hierarchy-aware semantic segmentation. In Sec. 6.3 we present the label hierarchy of COCO-Stuff, which augments the existing hierarchy in COCO (Lin et al., 2014). For many applications of semantic segmentation, mislabelings between semantically related classes (e.g. *wall-stone* and *wall-concrete*) are less critical than mislabelings between semantically unrelated classes (e.g. *sky* and *water*). Semantic segmentation on COCO-Stuff may benefit from hierarchy-aware approaches such as Deng et al. (2014a) which model mutual exclusion, overlap and subsumption relations between classes in a label relation graph. Furthermore hierarchical evaluation may help to scale up classification to thousands of classes, as only a subset of the classes needs to be evaluated at each level of the hierarchy.

6.7 Conclusion

We introduced the large-scale COCO-Stuff dataset. COCO-Stuff enriches the COCO dataset with dense pixel-level stuff annotations. We used a specialized stuff annotation protocol to efficiently label each pixel. Our dataset features a diverse set of stuff classes. In combination with the existing thing annotations in COCO it allows us to perform a detailed analysis of stuff and the rich contextual relations that make our dataset unique. We have shown that (1) stuff is important: Stuff classes cover the majority of the image surface and more than a third of the nouns in human descriptions of an image; (2) many classes show frequent patterns of spatial context, and stuff classes appear in more varied contexts than things; (3) stuff is not generally easier to segment than things; (4) the larger training set that COCO-Stuff offers improves the semantic segmentation performance.

Chapter 7

Additional works

In this chapter we present additional topics that we worked on as part of this thesis. These are unpublished works and we point out the potential for future work.

7.1 ImageNet-Stuff: Augmenting ILSVRC with stuff

Chapters 3 and 5 use image-level tags to learn under weak supervision. Image-level tags are cheaper to annotate and therefore allow us to scale our learning more efficiently (see Sec. 2.2.1). Several works crawl the web to learn from noisy image-level tags (Fergus et al., 2005; Ferrari and Zisserman, 2007; Vijayanarasimhan and Grauman, 2008; Li and Fei-Fei, 2010; Modolo and Ferrari, 2017) or add additional annotations to create fully labeled datasets (Deng et al., 2009; Lin et al., 2014; Everingham et al., 2015; Zhou et al., 2017a). Datasets crawled from the web typically have only a few labels per image, compared to the exhaustive image-level annotations required by most weakly supervised semantic segmentation methods. To demonstrate the feasibility of weakly supervised semantic segmentation, most works use datasets labeled at the pixel-level and then extract image-level tags from them (Vezhnevets et al., 2011; Xu et al., 2015a; Kolesnikov and Lampert, 2016). This means that these methods do not actually save any annotation time, which is the purpose of weakly supervised learning. Furthermore, we found that directly annotating actual image-level labels results in additional problems. To study these problems, we annotated a medium-scale dataset called the *ImageNet-Stuff* dataset. The dataset creation, experiments, results and future work are presented below.

Dataset creation. ImageNet-Stuff is a precursor to COCO-Stuff. When we created the ImageNet-Stuff dataset in 2014, ILSVRC (Russakovsky et al., 2015), which is

based on ImageNet, was arguably the most important challenge in object detection. We build on the challenge dataset and augment the existing bounding boxes for 200 thing classes with image-level stuff annotations. We sample 20K images from the 60K images added to ILSVRC in 2014. These images have about 4.5 stuff classes per image and are therefore well suited to study stuff and things in context.

Image-level annotations. To create the image-level stuff annotations we design a special protocol. We show an image to the annotator, but mask the bounding boxes of all known things during the annotation process using a semi-transparent black color. This lets an annotator focus on the stuff regions. The annotation process is crowd-sourced using the CrowdFlower platform. For this reason we need to drastically simplify the annotation process. The instructions explicitly tell the annotators to annotate "background regions", rather than giving definitions of stuff and things (see Sec. 1.1). Due to the language barrier, we decide to replace lengthy instructions by example images with correct and incorrect answers. Preliminary experiments show that this improves the resulting annotations. On average 3.8 annotators annotate the same image. By measuring the label agreement between different annotators, we can estimate the correctness of a label. The relative frequencies can also tell us how relevant the label is for that particular image.

The annotation of a single image with on average 4.2 labels takes 48s (median) and we pay an hourly wage of 3.5 USD (median). Note that the mean annotation time per image amounts to 74s, because we do not control when crowd-sourced workers take a break. During internal test runs we also saw that annotators frequently like to go back to the instruction examples to compare their work to the guidelines. Papadopoulos et al. (2014) report hourly payments of about 16 USD for expert annotators, which tend to work more concentrated. Xu et al. (2015b) report hourly payments of 6 USD for letting crowd-sourced users play a game. Su et al. (2012) report a median of 26s for the drawing of a bounding box. This means that for us annotating an image with a single stuff tag takes about 44% of the time that it takes annotators in Su et al. (2012) to draw a bounding box, which is arguably more time consuming. We did not choose to restrict the possible answers for the user to a predefined set. We let the annotator specify the label names as free-form text. This allows the annotator to add missing labels. But as described in Sec. 6.3, it has the disadvantage of producing ambiguous and semantically overlapping classes. Therefore we need to carefully review, filter and merge these free-form labels in the next step.

Figure 7.1: Image-level annotations overlaid on example images of the ImageNet-Stuff dataset. The images shown are selected to include instances of the *car* class. All known thing instances are masked with a black box. Together with the ground-truth image-level tags, we show the absolute frequency that indicates how many annotators used the same label or synonyms thereof.

Label filtering. In the next step we filter the image-level labels. Starting from all labels (308,744), we remove duplicates per image (286,748) and list all unique labels (18,751). We perform automatic spell-checking on all labels and review all misspellings manually (4,251). In the case of an object being specified by an attribute (e.g. *muddy floor*), we also use the object and the attribute as separate labels. The resulting 4,059 unique labels that occur at least 5 times are manually categorized as stuff, things or other. The thing classes are removed. The stuff classes are manually categorized into natural (261), man-made (313) and hybrid categories (33). The other classes are categorized into materials (83) and attributes (250). A total of 940 classes remain. The 200 most common stuff classes all occur at least 80 times. Fig. 7.1 shows example images from the ImageNet-Stuff dataset that are known to have the *car* class inside them. As described above, all known thing instances are masked with a black box. Looking at the image-level stuff labels, we can see that most images contain the stuff class *road*. This indicates that there is a strong co-occurrence between *car* and *road*.

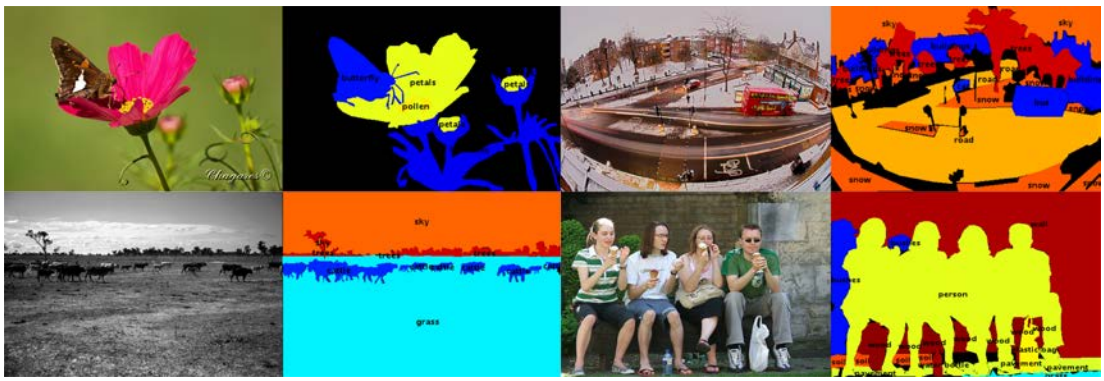


Figure 7.2: Example images and their pixel-level annotations on the ImageNet-Stuff dataset. We can see the complexity and diversity of the images that cover stuff and things. The annotations include even fine details.

Pixel-level annotations. As discussed in Sec. 2.2.1, we can train semantic segmentation using large amounts of data with image-level labels. To perform a meaningful evaluation of our method we require at least a small test set annotated at the pixel-level. The collection of image-level annotations takes little time, is easy and can be distributed to many annotators. Collection of pixel-level annotations is very costly, difficult to explain and therefore typically done by in-house expert annotators (Everingham et al., 2015; Mottaghi et al., 2014; Cordts et al., 2016). We let 4 students annotate 500 validation and 500 test images in ImageNet-Stuff with polygons for 200 stuff and 200 thing classes. Using a modified version of the LabelMe (Russell et al., 2008) annotation tool, the annotators draw polygons for each class in the image. Given the large number of classes, not all classes can be shown to or memorized by the annotators. Instead we showed only the classes known to be present in the image from the image-level annotations gathered above. Fig. 7.2 shows example images from ImageNet-Stuff and their pixel-level annotations. The quality of the annotations is very high and even fine details are annotated. For the image-level annotations we find that it takes on average 1.2 minutes to annotate an image at a cost of 0.03 GBP. For the pixel-level annotations the average is 10.1 minutes for an image at a cost of 2.0 GBP. The differences in price, time and quality are mainly due the aspect of crowd-sourcing versus in-house annotation.

Experiments. To evaluate the usefulness of and identify possible issues with learning from weakly annotated datasets, we run preliminary experiments on the stuff classes of the ImageNet-Stuff dataset. We ignore the thing classes to focus on learning from

weak annotations. For fully supervised experiments we use FCN-32s by Long et al. (2015). For weakly supervised experiments we train the same FCNs using the image-level loss of Bearman et al. (2016). The three experiments are as follows: 1) Fully supervised: Train on 500 pixel-level annotated images. 2) Weakly supervised: Train on 19,000 image-level annotated images. 3) Semi supervised: Fully supervised initialization, then weakly supervised fine-tuning. Note that the despite its name, the semi-supervised experiment actually uses more annotated images than the fully supervised experiment (19,500). None of the networks is pretrained on ImageNet (Krizhevsky et al., 2012). All experiments are evaluated in terms of class accuracy on the same 500 test images. We find that the fully supervised setup achieves a class accuracy of 13.2%, weakly supervised at most 5.3% and semi supervised 16.1%. The results in the fully supervised case are straightforward, but not the best, as only 500 images are used to train the 200 stuff classes, which clearly is not sufficient. The weakly supervised case shows that training with noisy image-level labels gives bad results, although still 10x better than random chance. The results fluctuate a lot depending on the initialization and the randomly selected mini-batches. The semi supervised results are the most interesting. Despite the weakly supervised annotations being 2x cheapter than the fully supervised ones, they significantly increase the performance of the fully supervised scenario by $+2.9\%$.

The results show that there is still a long way to go. Labels are often noisy and incomplete. Future work could try to find a way to deal with noise (Zhang et al., 2015b) and incorporate the confidence scores retrieved from the number of annotators that agree on a single label. We observe that many of the ground-truth image-level tags are not present in the image in the same way as when extracted from pixel-level annotations. Examples include occlusion, semi-transparency (e.g. *windows*, *raindrops*, *fences*) and even non-material classes (e.g. *sunlight*, *humidity*), although the latter were successfully filtered in the previous step. We will need to handle such cases, e.g. by explicitly modeling occlusions. Inspecting the resulting segmentations, we find that they often consist of many small patches. We assume that additional constraints on the spatial extent of the object will alleviate this problem. Furthermore we stress that the fully labeled annotations in the semi supervised case are only used as an initialization. Using recent state-of-the-art methods with ImageNet pretraining and strong priors (see Sec. 2.2.1), we hope to achieve the same performance in the weakly supervised case.

Annotating and learning from ImageNet-Stuff influenced many of the design decisions for COCO-Stuff. In particular, the 91 stuff labels in COCO-Stuff are derived

from the most common labels in ImageNet-Stuff. We realized that the polygon annotation protocol (used for the test set) is not efficient on stuff classes, where boundaries are hard to delineate. From our experiments we also concluded that current methods are not ready to use *real* weakly annotated datasets. Future work could try to focus more closely on the differences between weakly and fully annotated datasets.

7.2 Automatic photo popups from stuff and things

In Chapter 6 we presented the largest existing dataset for stuff and things. We studied the spatial context relations of stuff and things in 2D. However, analyzing the context of a 3D scene in a 2D image is ambiguous. If a *car* is directly above a *road* in 2D, it may be above or next to the *road* in 3D. It may touch the *road* or be far away. Therefore we propose to lift the 2D images of COCO-Stuff into 3D space, to be able to study stuff and things without ambiguity. This is often referred to as a 3D popup model.

Early works on automatic photo popup creation have shown impressive results (Criminisi et al., 2000; Hoiem et al., 2005b; Delage et al., 2006; Saxena et al., 2008), but typically use a lot of simplifying assumptions, such as a flat ground and vertical walls and are restricted to indoor images. Hoiem et al. (2005b) use geometric labels (*horizontal*, *vertical*, *sky*) and superpixels as a prior on uniformly labeled regions. Saxena et al. (2008) use more sophisticated occlusion boundaries and monocular depth information.

Our approach. Similar to most related methods, we fit a plane to each semantic region. We improve upon these methods by looking at the semantic labels of the image, especially in relation to the differences between stuff and things. We hypothesize that some classes are likely to be vertical (*building*, *person*, *car*), horizontal (*road*, *grass*) or porous (*rock*, *bush*). We estimate the support relations (Silberman et al., 2012; Guo and Hoiem, 2013) between regions (e.g. *person on bench*) and "stitch" together the ground and vertical planes, thereby enforcing depth continuity and refining depth and normal estimates. Preliminary experiments show that with enough images these relations can be learned. The camera model is defined by camera height and viewing angle, which also define the ground plane.

We harness recent deep learning methods for depth and normal estimation from monocular images. Gathering ground-truth depth data (e.g. with a laser scanner) is expensive and most datasets only include indoor scenes (Silberman et al., 2012). In our

case it is even impossible, as the images in COCO and hence COCO-Stuff are gathered from the web. Zoran et al. (2015); Chen et al. (2016b) train depth estimation methods from ordinal relationships between two points. These can be annotated efficiently by humans and Chen et al. (2016b) publish a large-scale dataset with ordinal relationships, which is not restricted to indoor images. Combined with the dense depth annotations on the NUYD (Silberman et al., 2012) dataset, they achieve state-of-the-art performance on both indoor and outdoor depth estimation. We use the pretrained models of Chen et al. (2016b) on COCO-Stuff. We define an energy function that includes all of the local and global constraints mentioned above. We first optimize the orientation and then the translation of each region plane using grid search. Then we update the local camera parameters and loop until convergence. We annotate 100 images with ordinal relationships to find suitable model parameters via cross-validation.

Preliminary results. Fig. 7.3 shows preliminary results of our photo popup method. We can see that our method nicely deals with the ground, vehicle and background. Future work will try to focus on improving the model, e.g. by allowing for arbitrary rotation and translation and dealing with more complex images with multiple ground planes. We observe that sometimes an object can cut through the ground-plane, which should not be possible. We also plan to speed up the inference procedure. Similar to Hoiem et al. (2008), using the known real-world height of commonly occurring thing classes (e.g. *person*) in COCO, we plan to estimate the scale of stuff and things in the scene. Furthermore, the knowledge of vanishing points (Li et al., 2010) might improve our model.

The resulting 3D models will allow us to study stuff and things in context in 3D. They will also allow us to query a large-scale dataset for stuff and thing classes by viewpoint, scale, depth, orientation and 3D spatial relations between classes. This will improve the understanding of the characteristics of stuff classes and allow us to train semantic segmentation with additional levels of annotation.

7.3 Cycle consistency in semantic segmentation

Beyond traditional evaluation metrics. In Sec. 2.1.1 we discussed existing evaluation criteria for semantic segmentation. We presented their advantages and disadvantages and pointed out that no single criterion is able to satisfy all possible requirements. In fact, we hypothesize that end-to-end learning leads to methods "overfitting" to the

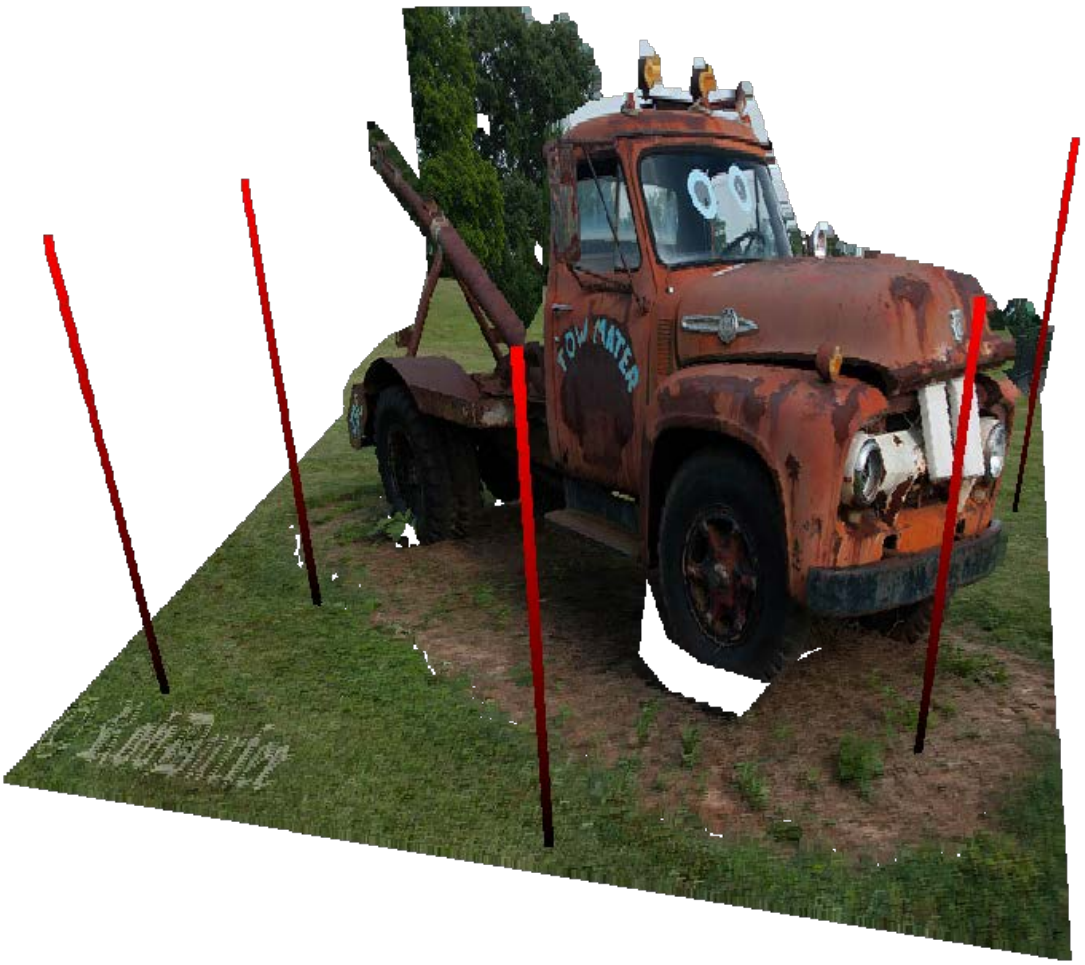


Figure 7.3: Preliminary results of our photo popup method. Red lines show the normals of the ground plane sampled at multiple locations to indicate the camera model.

criterion they are optimized for. Similarly on the conceptual level, methods that overfit to the criterion used in the most important semantic segmentation challenges are more likely to be selected. This may have adverse affects on the real-world applicability of leading methods. A variation of this idea is known in other fields as Goodhart's law: "When a measure becomes a target, it ceases to be a good measure." (Strathern, 1997)

Here we present a new idea that removes the need for the somewhat arbitrary evaluation criterion on the output of the semantic segmentation method. We argue that a good segmentation is one that reconstructs the original image well. The segmentation can be seen as a semantic compression of the image. This is visualized in Fig. 7.4. A semantic segmentation method assigns a label to each pixel in an image. The image can be "reconstructed" from the pixel labels, which is called image synthesis. Note that a huge number of images map to the same segmentation, as each object can vary

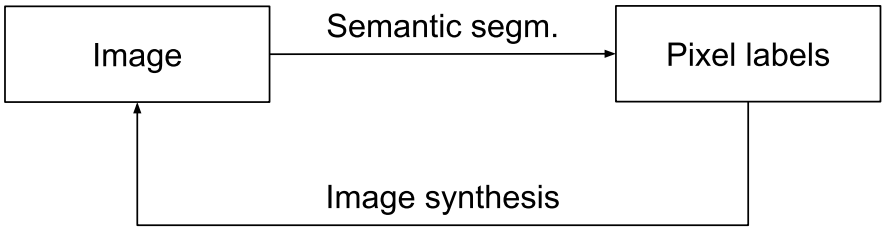


Figure 7.4: The concept of cycle consistency (Zhu et al., 2017) in semantic segmentation. Semantic segmentation assigns a label to each pixel in an image. Image synthesis is used to generate plausible images conditioned on the pixel labels. The original and the synthesized image should be as similar as possible.

in appearance. Our approach replaces the evaluation criterion on the segmentation with an evaluation criterion on the image, which is arguably easier to find. Zhu et al. (2017) use the L1 distance between the synthesized image and the original image. Future work will try to find more suitable criteria that allow pixel values to differ while preserving the semantics. For example, if a blue car is synthesized instead of a red car this should not affect the criterion. This idea is closely related to the concepts of a cycle consistency loss (Zhu et al., 2017) and dual learning (Yi et al., 2017). Zhu et al. (2017) use the cycle consistency loss to translate an image from a source to a target domain in absence of paired examples. To the best of our knowledge related concepts have not been used in semantic segmentation.

Application to weakly supervised learning. Cycle consistency is particularly useful in weakly supervised semantic segmentation. In Sec. 7.1 we discuss that even a weakly annotated dataset requires a test set with pixel labels to evaluate performance. Using a cycle consistency evaluation criterion, performance can be evaluated on the synthesized images, removing the need for pixel-level labels. Regardless of the evaluation criterion, optimizing cycle consistency can also be seen as a useful prior on the generally underconstrained problem of weakly supervised semantic segmentation (see Sec. 2.2.2 on related priors).

Our approach. Now we present early work that attempts to optimize a weakly supervised semantic segmentation network for cycle consistency. For the semantic segmentation step in Fig. 7.4 we use a state-of-the-art method, such as a Deeplab (Chen et al., 2017). Image synthesis is usually done with Generative Adversarial Networks (GAN) (Goodfellow et al., 2014) that consist of a generator that synthesizes images and a discriminator that attempts to distinguish real and fake images. We use the Conditional

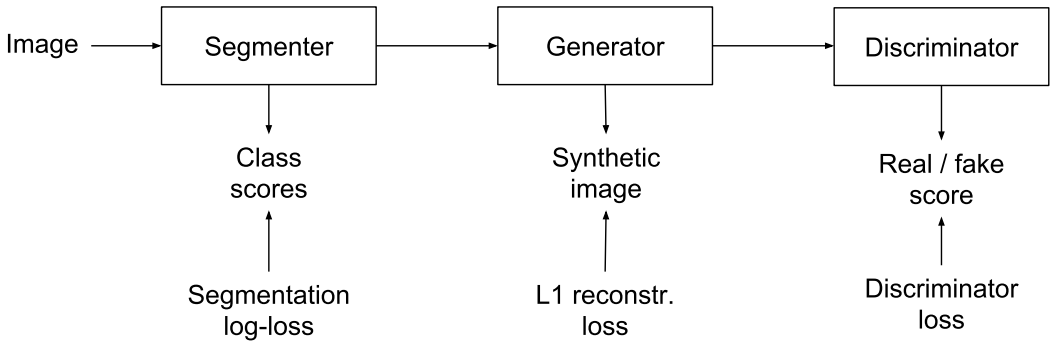


Figure 7.5: Overview of our proposed cycle consistency framework with 3 components and their respective outputs and losses: segmenter, generator and discriminator.

GAN (Mirza and Osindero, 2014) of Isola et al. (2016) to condition the image synthesis on the pixel labels. We chain the two networks together by passing the segmentation outputs as inputs to the GAN. As shown in Fig. 7.5, we now have three components: a segmenter that predicts semantic classes, a generator that synthesizes images and a discriminator that decides whether an image is real or fake. Each component has its own loss and the total loss can be any combination of the three. Since we train the whole network end-to-end, we expect semantic segmentation and image synthesis to benefit from each other.

A modification of our approach may be able to perform unsupervised semantic segmentation. The segmenter can be trained exclusively for image reconstruction, instead of the segmentation log-loss. By enforcing additional smoothness priors on the output labeling, the segmenter might be forced to learn semantically coherent regions. However, even if that is the case, the resulting segmentations are only accurate up to a permutation of labels.

Preliminary results. Here we present preliminary results of our approach. At the current stage our method is facing several issues, which future work will try to address:

Collusion between segmenter and generator. Fig. 7.6a shows an example, where we train the entire framework in an unsupervised way only for image reconstruction and not for semantic segmentation. We are able to perform a near-perfect reconstruction, although only two classes appear in the segmentation. This is because the segmenter tends to encode the image in the semantic class scores, such that the generator can perform a perfect pixel-level reconstruction. This is not surprising, as the scores have a higher dimensionality than the RGB image (C 64-bit channels for C classes, compared to 3 8-bit channels).

We propose to address this issue by introducing a bottleneck in the network. We present class labels to the generator, instead of the class scores. However since the argmax function is non-differentiable and we cannot backpropagate gradients through it. We circumvent this by propagating the labels directly to the class with the highest score. Fig. 7.6b shows results for a fully supervised version of our framework. The image reconstruction method now only sees the labels as shown in the third column and the segmenter can no longer pass on information through the class scores. This is visible in the black background labels at the bottom of the image. Whereas the initial version was able to reconstruct cars in that region, the more "semantic" second version only produces gray-colored noise.

Insufficient quality of image reconstruction methods. GANs have achieved breakthrough successes in image synthesis. However, for the time being they only work well on specific datasets. These datasets have a fixed spatial setup and low visual variability, e.g. front-facing building facade views (Tyleček and Šára, 2013) or car-mounted camera images (Cordts et al., 2016) are commonly used in the literature. They also tend to have the same semantic classes present in every image, which is detrimental to weakly supervised learning: if the same two classes occur in most of the images, how will the learner be able to tell them apart, given only image-level labels? In fact, our experiments show that GANs do not work well on datasets that work well for weakly supervised methods and vice versa. We expect future work to resolve most of the problems of GAN training (e.g. mode collapse (Goodfellow et al., 2014; Arjovsky et al., 2017)) and weakly supervised semantic segmentation (sensitivity to bad initialization (Pathak et al., 2015b,a)).

Lack of stuff annotations. As pointed out in Sec. 2.2.1 most recent weakly supervised works operate on the PASCAL VOC (Everingham et al., 2015) datasets. In these datasets only things are annotated and the background is assigned a single label. As shown in Fig 7.6c this leads to poor image reconstruction, as only about 27% of the pixels have a label. In fact, the network learns to always surround animals with a green background, although that is not generally the case. Fig 7.6d presents an example trained and tested on the MSRC 21 (Shotton et al., 2006) dataset. It includes dense stuff and thing labels and is therefore more suitable for image reconstruction. Future work will experiment on densely labeled datasets like PASCAL Context (Mottaghi et al., 2014) and COCO-Stuff (Caesar et al., 2018).

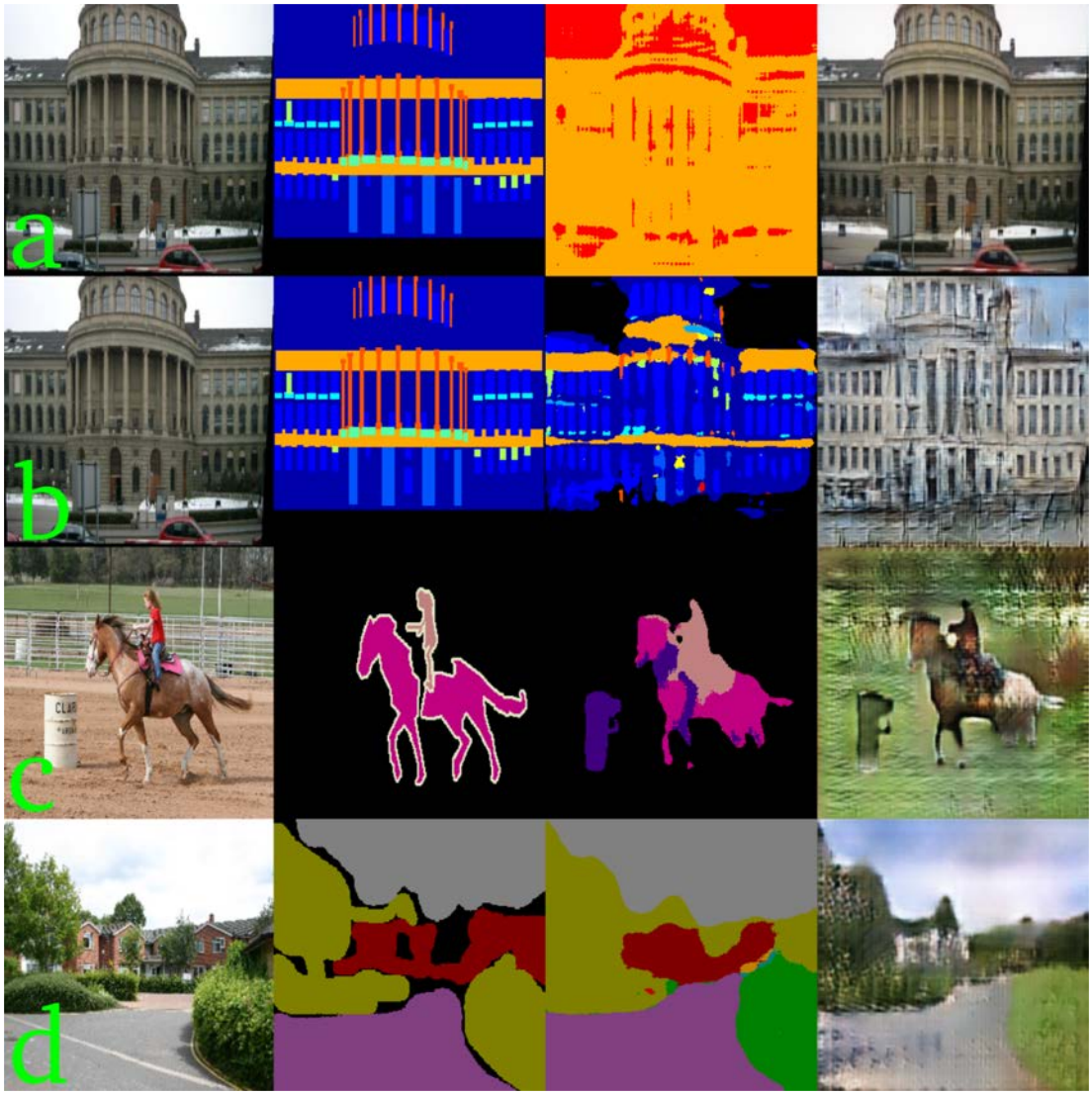


Figure 7.6: Cycle consistency examples. Each example shows an image, its ground-truth semantic labels, the segmenter outputs and the reconstructed images. The first row shows an unsupervised example where due to the collusion between segmenter and generator the image is reconstructed perfectly, despite the labels not being "semantic". The following rows show working examples of the training for cycle consistency priors. For simplicity these are trained under full supervision.

Chapter 8

Future work

In this chapter we present potential future work. Note that future work specific to each paper can be found in the respective chapters (see Sec. 3.6, 4.6, 5.8 and 6.6).

8.1 Applications for stuff and things

In Chapter 1 we discussed the importance of stuff and things for many different applications. Building on a large collection of related works (Forsyth et al., 1996; Adelson, 2001; Xiao et al., 2010; Endres and Hoiem, 2010; Ion et al., 2011; Tighe and Lazebnik, 2013a; Uijlings et al., 2013; Mottaghi et al., 2013; Feldman, 2003; Dai et al., 2015b), we studied stuff and things and devised new methods and datasets related to stuff and things. As future work we propose to apply our insights, models and datasets to new applications beyond semantic segmentation and object detection. While the role of context has been studied in detail for most applications, we believe that many fields could benefit from making the distinction between stuff and things explicit.

As an example application, crowd counting (Wu and Nevatia, 2005; Chan et al., 2008; Wang and Wang, 2011; Zhang et al., 2019; Liu et al., 2018) is the task of estimating the number of people in a crowd from an image. This task is related to object detection, but we are not interested in the actual outlines of a person. Furthermore, since persons are often heavily occluded, special techniques are required, such as density estimation (Chan et al., 2008; Zhang et al., 2019). One challenge of this task are false positive person "detections" in areas with high frequency patterns (such as windows in buildings). Using stuff, future work may be able to discard these false positives. But not all stuff classes should be treated equally. While some stuff classes specifically exclude the presence (or limit the number) of persons, others are

frequently found below or next to a person (see our spatial context study in Sec. 6.4.2). Our COCO-Stuff dataset (see Chapter 6) features both, stuff annotations and complex COCO (Lin et al., 2014) images with dozens of persons annotated at the instance-level (see Fig. 6.2), and is therefore suitable to study these relations.

Another application where stuff may be beneficial is object or person tracking (Gavrila and Davis, 1996; Ioffe and Forsyth, 2001; Forsyth et al., 2005; Andriluka et al., 2008; Breitenstein et al., 2009). Specific stuff classes may define where a person can go (*grass* or *pavement*) and where it cannot (*fence* or *wall*). Furthermore, the type of stuff may even limit the speed at which a person is able to move (*water*). These insights may also be useful to the task of autonomous driving. Autonomous vehicles need to create a 3D model of their surroundings, both visible and beyond. Stuff and thing classes are essential to create this model and model the intentions of all other actors in the scene.

8.2 Performance differences between stuff and things

In Sec. 6.4.3 we compare the performance of a standard semantic segmentation method on stuff and thing classes. We find that the performance is higher on thing than on stuff classes. Other authors make the opposite observation on other datasets (Tighe and Lazebnik, 2010, 2013a; Tighe et al., 2014; Liu et al., 2011; Ion et al., 2011). To understand why that is the case, future work could control for the different criteria of stuff and things (see Sec. 1.1). This idea is inspired by Shelhamer et al. (2016), where they mask parts of the images used in semantic segmentation. These masks are applied on foreground or background pixels at training or test time. Foreground roughly corresponds to thing classes and background to stuff classes. They show that masking the background only leads to minor decrease in performance. Interestingly they achieve decent performance when only using black and white images, that show only the shape of the object.

We envision a detailed ablation test of how the performance of a semantic segmentation method is affected by the individual criteria that define stuff and things. For that purpose we suggest to modify certain aspects of the images at training and test time and measure the performance on the test set. The modifications need not be limited to masking, but can include any form of editing and composition of new images (e.g. using Generative Adversarial Networks conditioned on segmentation maps (Mirza and Osindero, 2014)). We suggest the following modifications, closely inspired by the stuff

and thing criteria in Sec. 1.1:

1. Shape: Controlling for shape is not trivial. We can alter things to limit the effects of shape, e.g. by taking crops of thing instances or blurring the boundaries between different objects. Alternatively we can add shapes to stuff, e.g. by adding artificial but characteristic boundaries to each stuff class.
2. Size: To control for region size, we can subsample images with particular region sizes, mask out subregions, scale existing regions or compose entirely new images.
3. Parts: To control for the effect of parts, we can disassemble the part composition of thing instances and reassemble them in a random manner (cf. *bag of words* and *bag of feature* representations (Belongie et al., 2001; Lowe, 2004)). This would not remove parts, but at least destroy their common spatial configuration. Furthermore we can compose images consisting only of a single part, to alleviate the effect of co-occurrence relations between parts. However, we need to make sure that the parts used are not characteristic of multiple thing classes (wheels can be found in cars and bikes).
4. Instances: It is not clear how to modify things to lose the instances and countability property, other than the modifications mentioned above.
5. Texture: We can remove the characteristic texture of stuff classes. Similar to Shelhamer et al. (2016), we can set the stuff classes to a single color. Alternatively we can add specific textures to each thing class, e.g. by copying from a stuff class.

The main difficulty with the modifications suggested above is that the image needs to "look realistic" to the network, i.e. follow the same distribution. Otherwise the network can easily spot and learn the modifications. Furthermore it may be helpful to look at the per-class performance to figure out which criteria are important for particular classes.

8.3 Towards lifelong learning

Current approaches in scene understanding are catered directly for the problem they are trying to solve. New datasets are annotated from scratch, instead of including existing datasets. New methods are trained from scratch, instead of exploiting existing trained

models (with the notable exception of ImageNet pretraining (Krizhevsky et al., 2012)). If artificial intelligence is to become more human-like, we need to be able to benefit from previous knowledge.

Several works have shown that even larger datasets are beneficial to train existing methods (see Sun et al. (2017) and Chapter 6). Computational resources available for a certain budget are often assumed to increase exponentially over time (cf. Moore's law (Moore, 1965)). Therefore we expect future models to grow superlinearly in terms of the number of trainable parameters and required data. On the contrary, with traditional approaches there is no reason why human annotated datasets should grow superlinearly. Hence we propose to combine the data annotation efforts of various groups to create a single larger dataset. This requires standardization of annotation formats, the classes to be used and their license. Instead of releasing a single version of the dataset, datasets can grow continuously in size (see the SUN datasets (Xiao et al., 2010)) and new types of annotations can be added to existing datasets (such as our stuff annotations and the keypoints and captions added to COCO (Lin et al., 2014)). Furthermore, by democratizing access to the annotation tools, a larger public may become annotators (see the web interface of LabelMe (Russell et al., 2008)). To motivate such a larger public, proper incentives are essential, e.g. Von Ahn and Dabbish (2004) let annotators play a game against each other to annotate an image. Eventually the proliferation of cheap sensors in electronic devices will give us abundant data, but likely with noisy labels and lesser forms of supervision. The user should be particularly incentivized to annotate regions of an object where annotations are scarce.

Current CNNs require a large amount of manual engineering of the architecture for the task at hand. Approaches such as Convolutional Neural Fabrics (Saxena and Verbeek, 2016) automatically select the optimal architecture for a given task. For such purposes a publicly available catalog of Neural Network layers and other machine learning tricks may be beneficial. Similarly a catalog of generic knowledge sources and priors may be useful to be able to benefit from prior experience. Generic knowledge and priors both help to make learning more data efficient. Curriculum learning (Bengio et al., 2009) has been shown to improve generalization, by presenting easy examples first during training. We propose to use curriculum learning to start from generic stuff classes and then learn more fine-grained subclasses. Similarly, we could go from a generic objectness (Alexe et al., 2010) measure to thousands of rare objects with only few training samples available. Alternatively, like Lee and Grauman (2011) we could use objectness as an indicator of difficulty for thing instances and present samples in

decreasing order of objectness. Such schedules could also successively decrease the level of supervision and increase the expected level of noise in the presented samples. We envision an online learning approach that is closer to how humans learn. Instead of training on all samples at once, new data could be added repeatedly. One problem we face when fine-tuning neural networks to a new domain is *catastrophic forgetting* (McCloskey and Cohen, 1989). Machine learning will need to devise new methods to allow machines to reuse prior knowledge. Furthermore embodiment of artificial intelligence and the ability to interact with the environment seem essential to learn strong priors (such as the knowledge of gravity and other basic laws of physics) and develop a more human-like intelligence. All of these will bring us closer to the long-term goals of curriculum learning (Bengio et al., 2009) and lifelong learning (Thrun, 1996).

Chapter 9

Conclusion

Here we give an overview of the core insights of this thesis:

- In the past, stuff has received less attention than things. We give a definition of stuff and things and their actual and perceived differences. We discuss why stuff and things are essential and why it is important to restore the balance between stuff and things in scene understanding (Sec. 1). Furthermore we present important background knowledge for this thesis and how stuff and things have been used in the literature (Sec. 2).

- Region-based approaches for semantic segmentation have a great potential. However three essential problems need to be taken into account, particularly when looking at stuff and thing classes together: class imbalance, class competition and overlapping regions. We presented a technique that optimizes for an arbitrary evaluation criterion and thereby takes into account all of these problems (Sec. 3).

- Fully convolutional methods are conceptually simple and efficient and allow for end-to-end training. Region-based methods produce crisp object boundaries. We combine the advantages of both by enabling to train a region-based method end-to-end. Region-based approaches with multi-scale regions are able to capture thing instances at their canonical size and stuff regions at their most discriminative scale. Using a novel region-to-pixel layer enables us to classify stuff and things directly at the pixel-level. It thus presents a unified approach to learn stuff and things end-to-end (Sec. 4).

- Standard fully supervised learning approaches do not scale well to thousands of

classes. Therefore we require techniques like weakly supervised learning and domain transfer to learn efficiently and reuse existing knowledge. In this context prior knowledge becomes essential to solve a heavily underconstrained problem. We show that the knowledge of stuff and things forms an excellent source of generic knowledge. This knowledge can be transferred across domains, using similarity relations and co-occurrence to work as contextual and appearance forces (Sec. 5).

- We require large-scale datasets to be able to study stuff and things in context. Therefore we gathered the largest existing dataset with a dedicated protocol for stuff and things. This enables new insights regarding the annotation protocol, relative importance, spatial context and role of stuff and things in semantic segmentation (Sec. 6).

We hope that this thesis will raise awareness for the importance of stuff classes in human perception and computer vision. By sharing our insights, methods and datasets with the public, we want to spur further research on stuff and things and eventually restore the balance between stuff and things in scene understanding.

Appendix A

Survey of semantic segmentation papers

For this thesis we survey the semantic segmentation literature. We create records of every paper we found that presents semantic segmentation experiments. Furthermore we note which datasets they use in their experiments, excluding special purpose datasets, that are limited to faces, birds, horses, flowers, medical images or 3D semantic segmentation. If there are multiple versions of the same paper, such as pre-releases on arXiv.org and conference papers, we use only the latest version. We acknowledge that this collection is not unbiased, but stress that this is the largest existing collection on the web[1]. Last updated on March 7, 2018, the collection includes 356 papers and 43 datasets. Without replicating the entire collection here, we present some insightful statistics below.

Task names. In Sec. 2.1 we presented the various synonyms for the task of semantic segmentation. Given our collection of papers that are known to include semantic segmentation experiments, we gather statistics from their titles to see how they name this task. The results are as follows: 133 papers use *semantic segmentation*, 20 papers use *scene parsing*, 17 use *semantic image segmentation*, 12 *scene labeling* and 11 the more general *scene understanding*. Furthermore, the mean publication year of *semantic segmentation* is 4 to 10 months higher than of any of the other synonyms. This shows what we already suspected, that semantic segmentation was a very heterogeneous field in its infant years, but that it has achieved significant standardization recently. Particularly the work of Long et al. (2015) contributed to this standardization, in terms of the task name, default method and the evaluation metrics used.

[1] See https://github.com/nightrome/really-awesome-semantic-segmentation

Venue	WACV	ICRA	NIPS	BMVC	PAMI	ECCV	ICCV	CVPR	arXiv
Papers	7	7	7	13	14	19	21	55	156

Table A.1: Publication venues and the number of papers published there.

Publication venues. Now we analyze the venue (conference, journal or arXiv) that each paper was published in. Table A.1 shows the venues with more than 5 papers. Compared to other fields, there are many more conference than journal (only PAMI) publications. We hypothesize that this is due to the field moving forward very quickly, which is more inline with the short turnaround time of conferences. The bulk of the papers in our collection are so far exclusively published on arXiv. Some of these papers may be published at conferences or journals soon. Publishing of pre-releases on arXiv is a recent trend that was not the case before 2014, according to this data. We hypothesize that this is another sign of the fast pace of this field, where new ideas are often published even before extensive experiments are available and long before acceptance in a conference or journal.

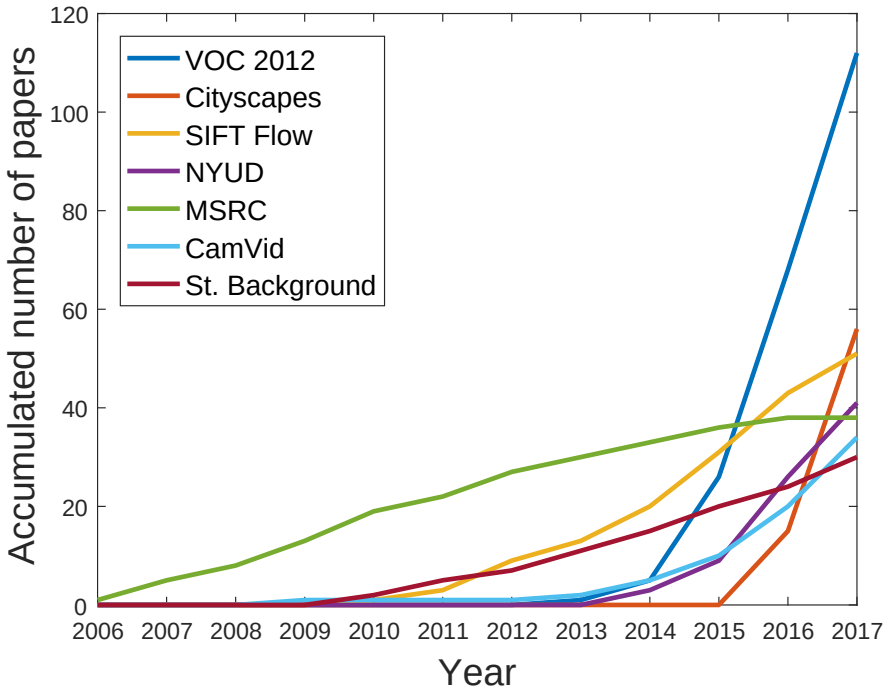


Figure A.1: The number of papers that experiment on a dataset accumulated over time.

Dataset importance. We present statistics of the number of papers in our collection that include experiments on a particular dataset. Fig. A.1 gives an overview of the 7 most commonly used datasets. We exclude the year 2018, as it is not over yet. We ob-

serve an exponential rise in the number of papers from 4 in 2007 to 139 in 2017 alone. However we do not predict this trend to continue, as there are now diminishing returns in terms of performance on most datasets and other tasks like instance segmentation and panoptic segmentation (see Sec. 2.1) are gaining more interest.

We can see that MSRC (Shotton et al., 2006) is the only very early dataset still in use, but that the community is increasingly interested in large-scale datasets. PASCAL VOC 2012 (Everingham et al., 2015) has established itself as the standard benchmark for semantic segmentation, despite not having stuff labels. This reinforces the main message of this thesis, that there is an imbalance between stuff and things in semantic segmentation. In terms of general purpose stuff and thing datasets, SIFT Flow (Liu et al., 2011) is the most popular dataset. Recently, self-driving car datasets are rapidly attracting interest (Brostow et al., 2009; Cordts et al., 2016). These cover at least some very common stuff classes, such as *road* and *building*.

Appendix B

List of acronyms

ASPP - Atrous Spatial Pyramid Pooling
AW - Area Weighting
CNN - Convolutional Neural Network
CPMC - Constrained Parametric Min-Cuts
CDF - Cumulative Distribution Function
CRF - Conditional Random Field
CUDA - Compute Unified Device Architecture
CW - Contrast Weighting
FCN - Fully Convolutional Network
GPU - Graphics Processing Unit
IOU - Intersection Over Union
LBP - Local Binary Pattern
LSDA - Large Scale Detection Through Adaptation
LW - Label Weighting
mAP - mean Average Precision
MIL - Multiple Instance Learning
MRF - Markov Random Field
R-CNN - Regions with CNN features
ReLU - Rectified Linear Unit
ROI - Region Of Interest
RNN - Recurrent Neural Network
RPN - Region Proposal Network
SEC - Seed Expand Constrain
SIFT - Scale Invariant Feature Transform

SVM - Support Vector Machine

TPU - Tensor Processing Unit

VGG - Visual Geometry Group

WSOL - Weakly Supervised Object Localization

Bibliography

Abdulnabi, A. H., Winkler, S., and Wang, G. (2017). Beyond forward shortcuts: Fully convolutional master-slave networks (MSNets) with backward skip connections for semantic segmentation. *arXiv*, arXiv preprint arXiv:1707.05537.

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Susstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. on PAMI*, 34(11):2274–2282.

Adelson, E. (2001). On seeing stuff : The perception of materials by humans and machines. In *SPIE proceedings series*, pages 1–12. Society of Photo-Optical Instrumentation Engineers.

Alexe, B., Deselaers, T., and Ferrari, V. (2010). What is an object? In *CVPR*.

Alexe, B., Deselaers, T., and Ferrari, V. (2012a). Measuring the objectness of image windows. *IEEE Trans. on PAMI*.

Alexe, B., Heess, N., Teh, Y., and Ferrari, V. (2012b). Searching for objects driven by context. In *NIPS*.

Andriluka, M., Roth, S., and Schiele, B. (2008). People-tracking-by-detection and people-detection-by-tracking. In *CVPR*.

Arbeláez, P., Maire, M., Fowlkes, C., and Malik, J. (2009). From contours to regions: An empirical evaluation. In *CVPR*.

Arbeláez, P., Pont-Tuset, J., Barron, J. T., Marques, F., and Malik, J. (2014). Multiscale combinatorial grouping. In *CVPR*.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. *arXiv*, arXiv preprint arXiv:1701.07875.

Arnab, A., Jayasumana, S., Zheng, S., and Torr, P. (2016). Higher order conditional random fields in deep neural networks. In *ECCV*.

Arnab, A. and Torr, P. H. S. (2016). Bottom-up instance segmentation using deep higher-order crfs. In *BMVC*.

Aytar, Y. and Zisserman, A. (2011). Tabula rasa: Model transfer for object category detection. In *ICCV*.

Aytar, Y. and Zisserman, A. (2012). Enhancing exemplar SVMs using part level transfer regularization. In *BMVC*.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. on PAMI*.

Bansal, A., Chen, X., Russell, B., Gupta, A., and Ramanan, D. (2017). Pixelnet: Representation of the pixels, by the pixels, and for the pixels. *arXiv*, arXiv preprint arXiv:1702.06506.

Bearman, A., Russakovsky, O., Ferrari, V., and Fei-Fei, L. (2016). What's the point: Semantic segmentation with point supervision. In *ECCV*.

Bell, S., Upchurch, P., Snavely, N., and Bala, K. (2013). OpenSurfaces: A richly annotated catalog of surface appearance. In *SIGGRAPH*.

Bell, S., Upchurch, P., Snavely, N., and Bala, K. (2015). Material recognition in the wild with the materials in context database. In *CVPR*.

Bell, S., Zitnick, C. L., Bala, K., and Girshick, R. (2016). Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*.

Belongie, S., Malik, J., and Puzicha, J. (2001). Matching shapes. In *ICCV*.

Bengio, J., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *ICML*.

Bilen, H., Pedersoli, M., Namboodiri, V. P., and Van Gool, L. (2014a). Object classification with adaptable regions. In *CVPR*.

Bilen, H., Pedersoli, M., and Tuytelaars, T. (2014b). Weakly supervised object detection with posterior regularization. In *BMVC*.

Bilen, H., Pedersoli, M., and Tuytelaars, T. (2015). Weakly supervised object detection with convex clustering. In *CVPR*.

Bilen, H. and Vedaldi, A. (2016). Weakly supervised deep detection networks. In *CVPR*.

Bishop, C. (1995). Neural networks for pattern recognition. *Oxford University Press*.

Boix, X., Gonfaus, J., van de Weijer, J., Bagdanov, A. D., Serrat, J., and Gonzàlez, J. (2012). Harmony potentials: Fusing global and local scale for semantic image segmentation. *IJCV*.

Brahmbhatt, S., Christensen, H., and Hays, J. (2017). Stuffnet: Using 'stuff' to improve object detection. In *Proc. WACV*.

Breitenstein, M., Reichlin, F., and Gool, L. V. (2009). Robust tracking-by-detection using a detector confidence particle filter. In *ICCV*.

Brodatz, P. (1966). *Textures: A Photographic Album for Artists and Designers*. Dover Publications.

Brostow, G. J., Fauqueur, J., and Cipolla, R. (2009). Semantic object classes in video: A high-definition ground truth database. *Patt. Rec. Letters*, 30(2):88–97.

Bulo, S. R., Neuhold, G., and Kontschieder, P. (2017a). Loss max-pooling for semantic image segmentation. In *CVPR*.

Bulo, S. R., Porzi, L., and Kontschieder, P. (2017b). In-place activated batchnorm for memory-optimized training of dnns. *arXiv*, arXiv preprint arXiv:1712.02616.

Burel, G. and Carel, D. (1994). Detection and localization of faces on digital images. *Patt. Rec. Letters*.

Byeon, W., Breuel, T. M., Raue, F., and Liwicki, M. (2015). Scene labeling with LSTM recurrent neural networks. In *CVPR*.

Caesar, H., Uijlings, J., and Ferrari, V. (2015). Joint calibration for semantic segmentation. In *BMVC*.

Caesar, H., Uijlings, J., and Ferrari, V. (2016a). COCO-Stuff: Thing and stuff classes in context. *arXiv*, arXiv preprint arXiv:1612.03716.

Caesar, H., Uijlings, J., and Ferrari, V. (2016b). Region-based semantic segmentation with end-to-end training. In *ECCV*.

Caesar, H., Uijlings, J., and Ferrari, V. (2018). COCO-Stuff: Thing and stuff classes in context. In *CVPR*.

Caputo, B., Hayman, E., Fritz, M., and Eklundh, J.-O. (2010). Classifying materials in the real world. *Image and Vision Computing*, 28(1):150–163.

Carreira, J., Caseiro, R., Batista, J., and Sminchisescu, C. (2012). Semantic segmentation with second-order pooling. In *ECCV*.

Carreira, J. and Sminchisescu, C. (2010). Constrained parametric min-cuts for automatic object segmentation. In *CVPR*.

Castrejon, L., Kundu, K., Urtasun, R., and Fidler, S. (2017). Annotating object instances with a Polygon-RNN. In *CVPR*.

Chan, A. B., Liang, Z.-S. J., and Vasconcelos, N. (2008). Privacy preserving crowd monitoring: Counting people without people models or tracking. In *CVPR*.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2015a). Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. on PAMI*.

Chen, L.-C., Schwing, A., Yuille, A., and Urtasun, R. (2015b). Learning deep structured models. In *ICML*.

Chen, L.-C., Yang, Y., Wang, J., Xu, W., and Yuille, A. L. (2016a). Attention to scale: Scale-aware semantic image segmentation. In *CVPR*.

Chen, W., Fu, Z., Yang, D., and Deng, J. (2016b). Single-image depth perception in the wild. In *NIPS*.

Cheng, C., Koschan, A., Chen, C.-H., Page, D. L., and Abidi, M. A. (2012). Outdoor scene image segmentation based on background recognition and perceptual organization. *IEEE Transactions on Image Processing*.

Cheng, C. Y. (1973). Response to moravcsik. *Approaches to Natural Language*, pages 286–288.

Cheng, F., He, X., and Zhang, H. (2017). Stacked learning to search for scene labeling. *IEEE Transactions on Image Processing*.

Cheng, M.-M., Zhang, Z., Lin, W.-Y., and Torr, P. (2014). Bing: Binarized normed gradients for objectness estimation at 300fps. In *CVPR*.

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. (2014). Describing textures in the wild. In *CVPR*.

Cinbis, R., Verbeek, J., and Schmid, C. (2014). Multi-fold mil training for weakly supervised object localization. In *CVPR*.

Cinbis, R., Verbeek, J., and Schmid, C. (2016). Weakly supervised object localization with multi-fold multiple instance learning. *IEEE Trans. on PAMI*.

Cogswell, M., Lin, X., Purushwalkam, S., and Batra, D. (2014). Combining the best of graphical models and convnets for semantic segmentation. *arXiv*, 1412.4313v2.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *CVPR*.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*.

Crammer, K. and Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292.

Criminisi, A., Reid, I., and Zisserman, A. (2000). Single view metrology. *IJCV*.

Csurka, G., Larlus, D., and F., P. (2013). What is a good evaluation measure for semantic segmentation? In *BMVC*.

Dai, J., He, K., and Sun, J. (2015a). Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *ICCV*.

Dai, J., He, K., and Sun, J. (2015b). Convolutional feature masking for joint object and stuff segmentation. In *CVPR*.

Dalal, N. and Triggs, B. (2005). Histogram of Oriented Gradients for human detection. In *CVPR*.

Dana, K., Van Ginneken, B., Nayar, S., and Koenderink, J. (1999). Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics (TOG)*, 18(1):1–34.

Dehghani, M., Severyn, A., Rothe, S., and Kamps, J. (2017). Learning to learn from weak supervision by full supervision. In *Workshop at NIPS*.

Delage, E., Lee, H., and Ng, A. Y. (2006). A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image. In *CVPR*.

Deng, J., Ding, N., Jia, Y., Frome, A., Murphy, K., Bengio, S., Li, Y., Neven, H., and Adam, H. (2014a). Large-scale object classification using label relation graphs. In *ECCV*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *CVPR*.

Deng, J., Russakovsky, O., Krause, J., Bernstein, M. S., Berg, A., and Fei-Fei, L. (2014b). Scalable multi-label annotation. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 3099–3102. ACM.

Deselaers, T., Alexe, B., and Ferrari, V. (2010). Localizing objects while learning their appearance. In *ECCV*.

Dietterich, T. G., Lathrop, R. H., and Lozano-Perez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71.

Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *ICCV*.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). DeCAF: A deep convolutional activation feature for generic visual recognition. In *ICML*.

Du, X. and Davis, L. (2017). Boundary-sensitive network for portrait segmentation. *arXiv*, arXiv preprint arXiv:1712.08675.

Duda, R. O. and Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*.

Eigen, D. and Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*.

Endres, I. and Hoiem, D. (2010). Category independent object proposals. In *ECCV*.

Endres, I. and Hoiem, D. (2014). Category-independent object proposals with diverse ranking. *IEEE Trans. on PAMI*, 36(2):222–234.

Everingham, M., Eslami, S., van Gool, L., Williams, C., Winn, J., and Zisserman, A. (2015). The PASCAL visual object classes challenge: A retrospective. *IJCV*.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The PASCAL Visual Object Classes (VOC) Challenge. *IJCV*.

Fan, H. and Ling, H. (2018). Dense recurrent neural networks for scene labeling. *arXiv*, arXiv preprint arXiv:1801.06831.

Fan, H., Mei, X., Prokhorov, D., and Ling, H. (2010). Multi-level contextual RNNs with attention model for scene labeling. *IEEE Transactions on Intelligent Transportation Systems*.

Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013). Learning hierarchical features for scene labeling. *IEEE Trans. on PAMI*, 35(8):1915–1929.

Feldman, J. (2003). What is a visual object? *Trends in Cognitive Sciences*, 7.6:252–256.

Fellbaum, C. (1998). Wordnet: An on-line lexical database.

Felzenszwalb, P., Girshick, R., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part based models. *IEEE Trans. on PAMI*, 32(9).

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *IJCV*.

Fergus, R., Fei-Fei, L., Perona, P., and Zisserman, A. (2005). Learning object categories from google's image search. In *ICCV*.

Fernando, B., Bilen, H., Gavves, E., and Gould, S. (2017). Self-supervised video representation learning with odd-one-out networks. In *CVPR*.

Ferrari, V. and Zisserman, A. (2007). Learning visual attributes. In *NIPS*, pages 433–440.

Fieder, N., Nickels, L., and Biedermann, B. (2014). Representation and processing of mass and count nouns: a review. *Frontiers in psychology 5*.

Forsyth, D., Malik, J., Fleck, M., Greenspan, H., Leung, T., Belongie, S., and Bregler, C. (1996). Finding pictures of objects in large collections of images. In *International Workshop on Object Representation in Computer Vision*.

Forsyth, D. A., Arikan, O., Ikemoto, L., O'Brien, J., and Ramanan, D. (2005). Computational studies of human motion: part 1, tracking and motion synthesis. *Found. Trends. Comput. Graph. Vis.*, 1(2-3):77–254.

Gadde, R., Jampani, V., Kiefel, M., and Gehler, P. V. (2016). Superpixel convolutional networks using bilateral inceptions. In *ECCV*.

Galasso, F., Cipolla, R., and Schiele, B. (2012). Video segmentation with superpixels. In *ACCV*.

Gatta, C., Romero, A., and van de Veijer, J. (2014). Unrolling loopy top-down semantic feedback in convolutional deep networks. In *Workshop at CVPR*.

Gavrila, D. M. and Davis, L. S. (1996). 3d model-based tracking of humans in action: a multi-view approach. In *CVPR*.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*.

George, M. (2015). Image parsing with a wide range of classes and scene-level context. In *CVPR*.

Girshick, R. (2015). Fast R-CNN. In *ICCV*.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*.

Girshick, R., Iandola, F., Darrell, T., and Malik, J. (2015). Deformable part models are convolutional neural networks. In *CVPR*, pages 437–446.

Gonzalez-Garcia, A. (2017). *Image context for object detection, object context for part detection*. PhD thesis, University of Edinburgh.

Gonzalez-Garcia, A., Vezhnevets, A., and Ferrari, V. (2015). An active search strategy for efficient object class detection. In *CVPR*.

Goodfellow, I., Pouget-Abadle, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *NIPS*.

Gould, S., Fulton, R., and Koller, D. (2009). Decomposing a scene into geometric and semantically consistent regions. In *ICCV*.

Gould, S., Zhao, J., He, X., and Zhang, Y. (2014). Superpixel graph label transfer with learned distance metric. In *ECCV*.

Guillaumin, M. and Ferrari, V. (2012). Large-scale knowledge transfer for object localization in imagenet. In *CVPR*.

Guillaumin, M., Mensink, T., Verbeek, J., and Schmid, C. (2008). Automatic face naming with caption-based supervision. In *CVPR*.

Guo, R. and Hoiem, D. (2013). Support surface prediction in indoor scenes. In *ICCV*.

Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. (2014). Simultaneous detection and segmentation. In *ECCV*.

Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. (2015). Hypercolumns for object segmentation and fine-grained localization. In *CVPR*.

He, K., Gkioxari, G., Dollar, P., and Girshick, R. (2017a). Mask R-CNN. *arXiv*, arXiv preprint arXiv:1703.6870.

He, K., Zhang, X., Ren, S., and Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*.

He, X., Zemel, R., and Carreira-Perpinan, M. (2004). Multiscale conditional random fields for image labelling. In *CVPR*.

He, Y., Chiu, W.-C., Keuper, M., and Fritz, M. (2017b). STD2P: RGBD semantic segmentation using spatio-temporal data-driven pooling. In *CVPR*.

Heitz, G. and Koller, D. (2008). Learning spatial context: Using stuff to find things. In *ECCV*.

Henderson, P. and Ferrari, V. (2016). Automatically selecting inference algorithms for discrete energy minimisation. In *ECCV*, pages 235–252.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv*, arXiv preprint arXiv:1207.0580.

Hoffman, J., Guadarrama, S., Tzeng, E., Hu, R., and Donahue, J. (2014). LSDA: Large scale detection through adaptation. In *NIPS*.

Hoiem, D., Efros, A. A., and Hebert, M. (2005a). Automatic photo pop-up. *ACM Transactions on Graphics*.

Hoiem, D., Efros, A. A., and Hebert, M. (2005b). Automatic photo pop-up. In *SIGGRAPH*.

Hoiem, D., Efros, A. A., and Hebert, M. (2008). Putting objects in perspective. *IJCV*, 80(1):3–15.

Holschneider, M., Kronland-Martinet, R., Morlet, J., and Tchamitchian, P. (1990). A real-time algorithm for signal analysis with the help of the wavelet transform. *Wavelets*.

Hong, S., Noh, H., and Han, B. (2015). Decoupled deep neural network for semi-supervised semantic segmentation. In *NIPS*.

Hong, S., Oh, J., Han, B., and Lee, H. (2016). Learning transferrable knowledge for semantic segmentation with deep convolutional neural network. In *CVPR*.

Hosang, J., Benenson, R., and Schiele, B. (2017). Learning non-maximum suppression. In *CVPR*.

Hu, H., Deng, Z., Zhou, G.-T., Sha, F., and Mori, G. (2017). Labelbank: Revisiting global perspectives for semantic segmentation. *arXiv*, arXiv preprint arXiv:1703.09891.

Huang, Q., Wang, W., Zhou, K., You, S., and Neumann, U. (2017a). Scene labeling using gated recurrent units with explicit long range conditioning. *arXiv*, arXiv preprint arXiv:1611.07485.

Huang, Q., Xia, C., Hao, W., Li, S., Wang, Y., Song, Y., and Jay Kuo, C.-C. (2017b). Semantic segmentation with reverse attention. In *BMVC*.

Hung, W.-C., Tsai, Y.-H., and Shen, X. (2017). Scene parsing with global context embedding. In *ICCV*.

Ioffe, S. and Forsyth, D. (2001). Human tracking with mixtures of trees. In *ICCV*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

Ion, A., Carreira, J., and Sminchisescu, C. (2011). Probabilistic joint image segmentation and labeling. In *NIPS*, pages 1827–1835.

Ionescu, C., Vantzos, O., and Sminchisescu, C. (2015). Training deep networks with structured layers by matrix backpropagation. In *ICCV*.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2016). Image-to-image translation with conditional adversarial networks. In *CVPR*.

Javanmardi, M., Sajjadi, M., Liu, T., and Tasdizen, T. (2016). Unsupervised total variation loss for semi-supervised deep learning of semantic segmentation. *arXiv*, arXiv preprint arXiv:1605.01368.

Jia, Y. (2013). Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/.

Kanizsa, G. (1979). Organization in vision: Essays on gestalt perception. *Praeger Publishers*.

Kekeç, T., Emonet, R., Fromont, E., Trémeau, A., and Wolf, C. (2014). Contextually constrained deep networks for scene labeling. In *BMVC*.

Khoreva, A., Benenson, R., Hosang, J., Hein, M., and Schiele, B. (2017). Simple does it: Weakly supervised instance and semantic segmentation. In *CVPR*.

Kim, B., Sun, M., Kohli, P., and Savarese, S. (2012). Relating things and stuff by high-order potential modeling. In *ECCV*.

Kirillov, A., He, K., Girshick, R., Rother, C., and Dollar, P. (2018). Panoptic segmentation. *arXiv*, arXiv preprint arXiv:1801.00868.

Kohli, P., Ladicky, L., and Torr, P. (2009). Robust higher order potentials for enforcing label consistency. *IJCV*.

Kolesnikov, A. and Lampert, C. (2016). Seed, expand and constrain: Three principles for weakly-supervised image segmentation. In *ECCV*.

Komodakis, N. and Paragios, N. (2009). Beyond pairwise energies: Efficient optimization for higher-order mrfs. In *CVPR*.

Krähenbühl, P. and Koltun, V. (2011). Efficient inference in fully connected CRFs with gaussian edge potentials. In *NIPS*, pages 109–117.

Krähenbühl, P. and Koltun, V. (2011). Efficient inference in fully connected CRFs with gaussian edge potentials. In *NIPS*.

Krähenbühl, P. and Koltun, V. (2014). Geodesic object proposals. In *ECCV*.

Krapac, J. and Segvic, S. (2016). Weakly-supervised semantic segmentation by redistributing region scores back to the pixels. In *GCPR*.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*.

Kuettel, D., Guillaumin, M., and Ferrari, V. (2012). Segmentation Propagation in ImageNet. In *ECCV*.

Ladicky, L., Russel, C., Kohli, P., and Torr, P. (2010a). Graph cut based inference with co-occurrence statistics. In *ECCV*.

Ladicky, L., Russell, C., and Kohli, P. (2009). Associative hierarchical CRFs for object class image segmentation. In *ICCV*.

Ladicky, L., Sturgess, P., Alahari, K., Russell, C., and Torr, P. H. S. (2010b). What, where and how many? combining object detectors and crfs. In *ECCV*.

Lampert, C., Nickisch, H., and Harmeling, S. (2009). Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*.

Larsson, G., Maire, M., and Shakhnarovich, G. (2017). Colorization as a proxy task for visual understanding. In *CVPR*.

Lazebnik, S., Schmid, C., and Ponce, J. (2005). A sparse texture representation using local affine regions. *IEEE Trans. on PAMI*, 27(8):1265–1278.

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

Lee, J.-T., Kim, H.-U., Lee, C., and Kim, C.-S. (2017). Semantic line detection and its applications. In *ICCV*.

Lee, Y. J. and Grauman, K. (2011). Learning the easy things first: Self-paced visual category discovery. In *CVPR*.

Li, B., Peng, K., Ying, X., and Zha, H. (2010). Simultaneous vanishing point detection and camera calibration from single images. In *International Symposium on Intelligent Data Analysis*.

Li, F., Carreira, J., Lebanon, G., and Sminchisescu, C. (2013). Composite statistical inference for semantic segmentation. In *CVPR*.

Li, L.-J. and Fei-Fei, L. (2010). Optimol: automatic online picture collection via incremental model learning. *IJCV*, 88(2):147–168.

Lim, J., Arbeláez, P., and Gu, C. (2009). Context by region ancestry. In *ICCV*.

Lin, D. (1998). An information-theoretic definition of similarity. In *ICML*.

Lin, D., Dai, J., Jia, J., He, K., and Sun, J. (2016a). Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. In *CVPR*.

Lin, G., Shen, C., Reid, I., and van dan Hengel, A. (2016b). Efficient piecewise training of deep structured models for semantic segmentation. In *CVPR*.

Lin, G., Shen, C., van den Hengel, A., and Reid, I. (2017). Exploring context with deep structured models for semantic segmentation. *IEEE Trans. on PAMI*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. (2014). Microsoft COCO: Common objects in context. In *ECCV*.

Liu, C., Yuen, J., and Torralba, A. (2011). Nonparametric scene parsing via label transfer. *IEEE Trans. on PAMI*, 33(12):2368–2382.

Liu, X., van de Weijer, J., and Bagdanov, A. D. (2018). Leveraging unlabeled data for crowd counting by learning to rank. In *CVPR*.

Liu, Z., Hu, J., Weng, L., and Yang, Y. (2017). Rotated region based CNN for ship detection. In *ICIP*.

Liu, Z., Li, X., and Luo, P. a. (2016). Semantic image segmentation via deep parsing network. In *CVPR*.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *CVPR*.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110.

Malisiewicz, T., Gupta, A., and Efros, A. (2011). Ensemble of exemplar-svms for object detection and beyond. In *ICCV*.

Manen, S., Guillaumin, M., and Van Gool, L. (2013). Prime object proposals with randomized prim's algorithm. In *ICCV*.

McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv*, arXiv preprint arXiv:1411.1784.

Modolo, D. and Ferrari, V. (2017). Learning semantic part-based models from google images. *IEEE Trans. on PAMI*.

Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics Magazine*, page 4.

Mostajabi, M., Yadollahpour, P., and Shakhnarovich, G. (2015). Feedforward semantic segmentation with zoom-out features. In *CVPR*.

Mottaghi, R., Chen, X., Liu, X., Cho, N.-G., Lee, S.-W., Fidler, S., Urtasun, R., and Yuille, A. (2014). The role of context for object detection and semantic segmentation in the wild. In *CVPR*.

Mottaghi, R., Fidler, S., Yao, J., Urtasun, R., and Parikh, D. (2013). Analyzing semantic segmentation using hybrid human-machine CRFs. In *CVPR*, pages 3143–3150.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *ICML*.

Ngan Le, T. H., Nhan Duong, C., Han, L., Luu, K., Savvides, M., and Pal, D. (2017). Deep contextual recurrent residual networks for scene labeling. *arXiv*, arXiv preprint arXiv:1704.03594.

Nickolls, J., Buck, I., Garland, M., and Skadron, K. (2008). Scalable parallel programming with CUDA. *Queue*, 6.2.

Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *ICCV*.

Nowozin, S. (2014). Optimal decisions from probabilistic models: The intersection-over-union case. In *CVPR*.

Oh, S., Benenson, R., Khoreva, A., Akata, Z., Fritz, M., and Schiele, B. (2017). Exploiting saliency for object segmentation from image level labels. In *CVPR*.

Ojala, T., Pietikaeinen, M., and Maeenpaeae, T. (2002). Multiresolution gray scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. on PAMI*.

Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42(3):145–175.

Ott, P. and Everingham, M. (2011). Shared parts for deformable part-based models. In *CVPR*.

Ozcan, M., Jie, L., Ferrari, V., and Caputo, B. (2011). A large-scale database of images and captions for automatic face naming. In *BMVC*.

Pan, S. K. and Yang, Q. (2010). A survey on transfer learning. *IEEE Trans. on KDE*.

Papadopoulos, D. P., Clarke, A. D. F., Keller, F., and Ferrari, V. (2014). Training object class detectors from eye tracking data. In *ECCV*.

Papadopoulos, D. P., Uijlings, J. R., Keller, F., and Ferrari, V. (2017a). Extreme clicking for efficient object annotation. In *ICCV*.

Papadopoulos, D. P., Uijlings, J. R., Keller, F., and Ferrari, V. (2017b). Training object class detectors with click supervision. In *CVPR*.

Papadopoulos, D. P., Uijlings, J. R. R., Keller, F., and Ferrari, V. (2016). We don't need no bounding-boxes: Training object class detectors using only human verification. In *CVPR*.

Papandreou, G., Chen, L.-C., Murphy, K., and Yuille, A. L. (2015). Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *ICCV*.

Papazoglou, A. and Ferrari, V. (2013). Fast object segmentation in unconstrained video. In *ICCV*.

Park, H., Jeong, J., Yoo, Y., and Kwak, N. (2017). Superpixel-based semantic segmentation trained by statistical process control. In *BMVC*.

Paszke, A., Chaurasia, A., Kim, S., and Culurciello, E. (2017). ENet: A deep neural network architecture for real-time semantic segmentation. In *ICLR*.

Pathak, D., Krähenbühl, P., and Darrell, T. (2015a). Constrained convolutional neural networks for weakly supervised segmentation. In *ICCV*.

Pathak, D., Shelhamer, E., Long, J., and Trevor, D. (2015b). Fully convolutional multi-class multipe instance learning. In *ICLR Workshop*.

Pinheiro, P. and Collobert, R. (2014). Recurrent convolutional neural networks for scene parsing. In *ICML*.

Pinheiro, P. and Collobert, R. (2015). From image-level to pixel-level labeling with convolutional networks. In *CVPR*.

Plath, N., Toussaint, M., and Nakajima, S. (2009). Multi-class image segmentation using conditional random fields and global classification. In *ICML*.

Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*.

Pont-Tuset, J., Guiu, M. A. F., and Smolic, A. (2015). Semi-automatic video object segmentation by advanced manipulation of segmentation hierarchies. In *International Workshop on Content-Based Multimedia Indexing*.

Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., and Belongie, S. (2007). Objects in context. In *ICCV*.

Rahman, M. A. and Wang, Y. (2016). Optimizing intersection-over-union in deep neural networks for image segmentation. In *International Symposium on Visual Computing*. Springer.

Rahtu, E., Kannala, J., and Blaschko, M. (2011). Learning a category independent object detection cascade. In *ICCV*.

Ranjbar, M. (2013). *Optimizing non-decomposable loss functions in structured prediction*. PhD thesis, Simon Fraser University.

Rantalankila, P., Kannala, J., and Rahtu, E. (2014). Generating object segmentation proposals using global and local search. In *CVPR*.

Raykar, V. C., Shipeng, Y., Zhao, L. H., Jerebko, A., Florin, C., Valadez, G. H., Bogoni, L., and Moy, L. (2009). Supervised learning from multiple experts: whom to trust when everyone lies a bit. In *ICML*.

Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). CNN features off-the-shelf: An astounding baseline for recognition. In *DeepVision workshop at CVPR*.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *CVPR*.

Ren, M. and Zemel, R. (2017). End-to-end instance segmentation with recurrent attention. In *CVPR*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*.

Ripley, B. (1996). Pattern recognition and neural networks. *Cambridge University Press*.

Rochan, M. and Wang, Y. (2015). Weakly supervised localization of novel objects using appearance transfer. In *CVPR*.

Rohrbach, M., Stark, M., Szarvas, G., Gurevych, I., and Schiele, B. (2010). What helps where - and why? semantic relatedness for knowledge transfer. In *CVPR*.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*.

Rother, C., Kolmogorov, V., and Blake, A. (2004). Grabcut: Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., and Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *IJCV*.

Russakovsky, O., Lin, Y., Yu, K., and Fei-Fei, L. (2012). Object-centric spatial pooling for image classification. In *ECCV*.

Russell, B. C., Murphy, K. P., and Freeman, W. T. (2008). LabelMe: a database and web-based tool for image annotation. *IJCV*.

Saito, S., Kerola, T., and Tsutsui, S. (2017). Superpixel clustering with deep features for unsupervised road segmentation. *arXiv*, arXiv preprint arXiv:1711.05998.

Saleh, F., Akbarian, M. S. A., Salzmann, M., Petersson, L., Gould, S., and Alvarez, J. M. (2016). Built-in foreground/background prior for weakly-supervised semantic segmentation. In *ECCV*.

Sankaranarayanan, S., Balaji, Y., Jain, A., Lim, S.-N., and Chellappa, R. (2017). Unsupervised domain adaptation for semantic segmentation with GANs. *arXiv*, arXiv preprint arXiv:1711.06969.

Saxena, A., Sun, M., and Ng, A. Y. (2008). Make3d: Learning 3d scene structure from a single still image. *IEEE Trans. on PAMI*.

Saxena, S. and Verbeek, J. (2016). Convolutional neural fabrics. In *NIPS*.

Schwing, A. and Urtasun, R. (2015). Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*.

Sharan, L., Rosenholtz, R., and Adelson, E. (2014). Material perception: What can you see in a brief glance? *Journal of Vision*, 14(9).

Sharma, A., Tuzel, O., and Jacobs, D. W. (2015). Deep hierarchical parsing for semantic segmentation. In *CVPR*.

Sharma, A., Tuzel, O., and Liu, M.-Y. (2014). Recursive context propagation network for semantic scene labeling. In *NIPS*.

Shelhamer, E., Long, J., and Darrell, T. (2016). Fully convolutional networks for semantic segmentation. *IEEE Trans. on PAMI*.

Shen, T., Lin, G., Shen, C., and Reid, I. (2017). Learning multi-level region consistency with dense multi-label networks for semantic segmentation. In *Proc. Intl. Joint Conf. on Artificial Intelligence*.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Trans. on PAMI*.

Shi, M., Caesar, H., and Ferrari, V. (2017a). Weakly supervised object localization using things and stuff transfer. In *ICCV*.

Shi, M. and Ferrari, V. (2016). Weakly supervised object localization using size estimates. In *ECCV*.

Shi, Z., Hospedales, T., and Xiang, T. (2015). Bayesian joint modelling for object localisation in weakly labelled images. *IEEE Trans. on PAMI*.

Shi, Z., Siva, P., and Xiang, T. (2012). Transfer learning by ranking for weakly supervised object annotation. In *BMVC*.

Shi, Z., Yang, Y., Hospedales, T. M., and Xiang, T. (2017b). Weakly supervised image annotation and segmentation with objects and attributes. *IEEE Trans. on PAMI*.

Shotton, J., Winn, J., Rother, C., and Criminisi, A. (2006). TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*.

Shotton, J., Winn, J., Rother, C., and Criminisi, A. (2009). TextonBoost for image understanding: Multi-class object recognition and segmentation by jointly modeling appearance, shape and context. *IJCV*, 81(1):2–23.

Shuai, B., Wang, G., Zuo, Z., Wang, B., and Zhao, L. (2015). Integrating parametric and non-parametric models for scene labeling. In *CVPR*.

Shuai, B., Zuo, Z., Wang, B., and Wang, G. (2017). Scene segmentation with dag-recurrent neural networks. *IEEE Trans. on PAMI*.

Shuai, B., Zuo, Z., Wang, G., and Wang, B. (2016). DAG-recurrent neural networks for scene labeling. In *CVPR*.

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *ECCV*.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Singh, G. and Kosecka, J. (2013). Nonparametric scene parsing with adaptive feature relevance and semantic context. In *CVPR*.

Siva, P. and Xiang, T. (2011). Weakly supervised object detector learning with model drift detection. In *ICCV*.

Smith, D., Pezzelle, S., Franzon, F., Zanini, C., and Bernardi, R. (2017). Can you see the (linguistic) difference? Exploring mass/count distinction in vision. In *12th International Conference on Computational Semantics*.

Song, H., Girshick, R., Jegelka, S., Mairal, J., Harchaoui, Z., and Darell, T. (2014a). On learning to localize objects with minimal supervision. In *ICML*.

Song, H., Lee, Y., Jegelka, S., and Darell, T. (2014b). Weakly-supervised discovery of visual pattern configurations. In *NIPS*.

Souly, N., Spampinato, C., and Shah, M. (2017). Semi supervised semantic segmentation using generative adversarial network. In *ICCV*.

Stark, M., Goesele, M., and Schiele, B. (2009). A shape-based object class model for knowledge transfer. In *ICCV*.

Stone, J. E., Gohara, D., and Shi, G. (2010). OpenCL: A parallel programming standard for heterogeneous computing systems. *Computing in science & engineering*, 12(3):66–73.

Strathern, M. (1997). improving ratings: audit in the british university system. *European review*, 5(3):305–321.

Su, H., Deng, J., and Fei-Fei, L. (2012). Crowdsourcing annotations for visual object detection. In *AAAI Human Computation Workshop*.

Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*.

Sun, M., Kim, B., Kohli, P., and Savarese, S. (2013). Relating things and stuff via object property interactions. *IEEE Trans. on PAMI*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *CVPR*.

Tang, K., Joulin, A., Li, L.-J., and Fei-Fei, L. (2014). Co-localization in real-world images. In *CVPR*.

Thrun, S. (1996). *Explanation-based neural network learning: A lifelong learning approach*. Kluwer Academic Publishers.

Tighe, J. and Lazebnik, S. (2010). Superparsing: Scalable nonparametric image parsing with superpixels. In *ECCV*.

Tighe, J. and Lazebnik, S. (2011). Understanding scenes on many levels. In *ICCV*.

Tighe, J. and Lazebnik, S. (2013a). Finding things: Image parsing with regions and per-exemplar detectors. In *CVPR*.

Tighe, J. and Lazebnik, S. (2013b). Superparsing - scalable nonparametric image parsing with superpixels. *IJCV*, 101(2):329–349.

Tighe, J., Niethammer, M., and Lazebnik, S. (2014). Scene parsing with object instances and occlusion ordering. In *CVPR*.

Tishby, N., Pereira, F. C., and Bialek, W. (2000). The information bottleneck method. *arXiv*, arXiv preprint physics/0004057.

Tokmakov, P., Alahari, K., and Schmid, C. (2016). Weakly-supervised semantic segmentation using motion cues. In *ECCV*.

Tommasi, T., Orabona, F., and Caputo, B. (2010). Safety in numbers: Learning categories from few examples with multi model knowledge transfer. In *CVPR*. IEEE.

Toutanova, K., Klein, D., Manning, C., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*.

Tyleček, R. and Šára, R. (2013). Spatial pattern templates for recognition of objects with regular structure. In *GCPR*.

Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. (2013). Selective search for object recognition. *IJCV*.

Van de Sande, K., Uijlings, J., Gevers, T., and Smeulders, A. (2011). Segmentation as selective search for object recognition. In *ICCV*.

Vanderbrug, G. J. and Rosenfeld, A. (1977). Two-stage template matching. In *IEEE Transactions on Computers*, volume C-26, pages 384–393.

Vedaldi, A. and Lenc, K. (2015). Matconvnet – convolutional neural networks for MATLAB. In *ACM Multimedia*.

Verbeek, J. and Triggs, B. (2007). Region classification with markov field aspect models. In *CVPR*.

Vezhnevets, A., Ferrari, V., and Buhmann, J. M. (2011). Weakly supervised semantic segmentation with multi image model. In *ICCV*.

Vezhnevets, A., Ferrari, V., and Buhmann, J. M. (2012). Weakly supervised structured output learning for semantic segmentation. In *CVPR*.

Vijayanarasimhan, S. and Grauman, K. (2008). Keywords to visual categories: Multiple-instance learning for weakly supervised object categorization. In *CVPR*.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *CVPR*.

Viola, P. and Jones, M. (2001). Robust real-time object detection. *IJCV*.

Von Ahn, L. and Dabbish, L. (2004). Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, pages 319–326. ACM.

von Helmholtz, H. (1867). Handbuch der physiologischen optik. *Voss*.

Wang, C., Ren, W., Zhang, J., Huang, K., and Maybank, S. (2015). Large-scale weakly supervised object localization via latent category learning. *IEEE Transactions on Image Processing*, 24(4):1371–1385.

Wang, J. and Yuille, A. (2015). Semantic part segmentation using compositional model combining shape and appearance. In *CVPR*.

Wang, M. and Wang, X. (2011). Automatic adaptation of a generic pedestrian detector to a specific traffic scene. In *CVPR*.

Wang, T., Han, B., and Collomosse, J. (2014). Touchcut: Fast image and video segmentation using single-touch interaction. *CVIU*.

Wang, Z., Gu, F., Lischinski, D., Cohen-Or, D., Tu, C., and Chen, B. (2017). Neuron-level selective context aggregation for scene segmentation. *arXiv*, arXiv preprint arXiv:1711.08278.

Wertheimer, N. (1923). Untersuchungen zur Lehre von der Gestalt II. *Psycologische Forschung, 4*, pages 301–350.

Wu, B. and Nevatia, R. (2005). Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *ICCV*.

Xiao, J., Ehinger, K. A., Hays, J., Torralba, A., and Oliva, A. (2014). SUN database: Exploring a large collection of scene categories. *IJCV*, pages 1–20.

Xiao, J., Hays, J., Ehinger, K., Oliva, A., and Torralba, A. (2010). SUN database: Large-scale scene recognition from Abbey to Zoo. In *CVPR*.

Xiao, J., Owens, A., and Torralba, A. (2013). SUN3D: A database of big spaces reconstructed using SfM and object labels. In *ICCV*.

Xie, J., Kiefel, M., Sun, M.-T., and Geiger, A. (2016). Semantic instance annotation of urban scenes by 3d to 2d label transfer. In *CVPR*.

Xu, J., Schwing, A., and Urtasun, R. (2014). Tell me what you see and i will show you where it is. In *CVPR*.

Xu, J., Schwing, A. G., and Urtasun, R. (2015a). Learning to segment under various forms of weak supervision. In *CVPR*.

Xu, P., Ehinger, K. A., Zhang, Y., Finkelstein, A., Kulkarni, S. R., and Xiao, J. (2015b). Turkergaze: Crowdsourcing saliency with webcam based eye tracking. *arXiv preprint arXiv:1504.06755*.

Yamaguchi, K., Kiapour, M. H., Ortiz, L. E., and Berg, T. L. (2012). Parsing clothing in fashion photographs. In *CVPR*.

Yan, Z., Zhang, H., Jia, Y., Breuel, T., and Yu, Y. (2016). Combining the best of convolutional layers and recurrent layers: A hybrid network for semantic segmentation. *arXiv*, arXiv preprint arXiv:1603.04871.

Yang, J., Price, B., Cohen, S., and Yang, M.-H. (2014). Context driven scene parsing with attention to rare classes. In *CVPR*.

Yi, Z., Zhang, H., Tan, P., and Gong, M. (2017). Dualgan: Unsupervised dual learning for image-to-image translation. In *ICCV*.

Yoo, H., Cha, G., and Oh, S. (2017). Understanding visual information with compound eye camera. In *International Conference on Control, Automation and Systems*.

Yu, F. and Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. In *ICLR*.

Zhang, L., Gao, Y., Xia, Y., Lu, K., Shen, J., and Ji, R. (2014). Representative discovery of structure cues for weakly-supervised image segmentation. In *IEEE Transactions on Multimedia*, volume 16, pages 470–479.

Zhang, L., Shi, M., and Chen, Q. (2019). Crowd counting via scale-adaptive convolutional neural network. In *Proc. WACV*.

Zhang, R., Candra, S. A., Vetter, K., and Zakhor, A. (2015a). Sensor fusion for semantic segmentation of urban scenes. In *ICRA*.

Zhang, W., Zeng, S., Wang, D., and Xue, X. (2015b). Weakly supervised semantic segmentation for social images. In *CVPR*.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. (2015). Conditional random fields as recurrent neural networks. In *ICCV*.

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., and Torralba, A. (2017a). Places: A 10 million image database for scene recognition. *IEEE Trans. on PAMI*.

Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A. (2017b). Scene parsing through ADE20K dataset. In *CVPR*.

Zhu, C., Zheng, Y., Luu, K., Hoang Ngan Le, T., Bhagavatula, C., and Savvides, M. (2016). Weakly supervised facial analysis with dense hyper-column features. In *CVPR*.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*.

Zitnick, C. L. and Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In *ECCV*.

Zoran, D., Isola, P., D., K., and Freeman, W. T. (2015). Learning ordinal relationships for mid-level vision. In *ICCV*.