Image context for object detection, object context for part detection

Abel Gonzalez-Garcia

Doctor of Philosophy
Institute of Perception, Action and Behaviour
School of Informatics
University of Edinburgh
2017

Abstract

Objects and parts are crucial elements for achieving automatic image understanding. The goal of the object detection task is to recognize and localize all the objects in an image. Similarly, semantic part detection attempts to recognize and localize the object parts. This thesis proposes four contributions. The first two make object detection more efficient by using active search strategies guided by image context. The last two involve parts. One of them explores the emergence of parts in neural networks trained for object detection, whereas the other improves on part detection by adding object context.

First, we present an active search strategy for efficient object class detection. Modern object detectors evaluate a large set of windows using a window classifier. Instead, our search sequentially chooses what window to evaluate next based on all the information gathered before. This results in a significant reduction on the number of necessary window evaluations to detect the objects in the image. We guide our search strategy using image context and the score of the classifier.

In our second contribution, we extend this active search to jointly detect pairs of object classes that appear close in the image, exploiting the valuable information that one class can provide about the location of the other. This leads to an even further reduction on the number of necessary evaluations for the smaller, more challenging classes.

In the third contribution of this thesis, we study whether semantic parts emerge in Convolutional Neural Networks trained for different visual recognition tasks, especially object detection. We perform two quantitative analyses that provide a deeper understanding of their internal representation by investigating the responses of the network filters. Moreover, we explore several connections between discriminative power and semantics, which provides further insights on the role of semantic parts in the network.

Finally, the last contribution is a part detection approach that exploits object context. We complement part appearance with the object appearance, its class, and the expected relative location of the parts inside it. We significantly outperform approaches that use part appearance alone in this challenging task.

Acknowledgements

First and foremost, I would like to express my heartfelt gratitude to my supervisor, Vittorio Ferrari. He has been an enlightening force that has shaped my research skills in a truly particular manner. His unmatched passion is incredibly inspiring and his dedicated support is absolutely priceless. I would also like to thank Alexander Vezhnevets, whose vast knowledge and kind assistance were genuinely appreciated during the first years of my PhD and truly missed during the latter.

Before my PhD, I was very fortunate to have exceptional supervision. I would like to thank my Master thesis supervisor, Iain Murray, for his admirable support. My sincere appreciation must also go to Robert Benavente and Maria Vanrell. My interest in Computer Vision and my first contact with academic research both started with them at the CVC and for this I am truly grateful.

The four years of my PhD would not have been the same without the awesomeness of the CALVIN research group. For this, I am indebted to Dimitris Papadopoulos, Vicky Kalogeiton, Holger Caesar, Davide Modolo, Paul Henderson, Buyu Liu, Miaojing Shi, Anestis Papazoglou, Jasper Uijlings, Luca del Pero, Michele Volpi, and Alexander Vezhnevets. The engaging coffee breaks, the enjoyable after-work beers, and the fun night outs in Edinburgh provided excellent intermissions of my PhD life.

I would like to thank my committee of examiners, Bob Fisher and Thomas Mensink, for taking the time to examine my work and provide valuable corrections. Also, thanks to the School of Informatics at the University of Edinburgh for their facilities and to the ERC Starting Grant VisCul for funding my research.

I am profoundly grateful to my family, who has been my longest standing support. Special thanks to my parents, who not only encouraged the entirety of my education, but also enabled it. Making them proud has been a motivating force to keep going. I hope that in the future my research in Computer Vision will contribute to improving their lives.

Finally, I owe my deepest gratitude to my wonderful wife. During these years, she has been my indisputable number one fan and my main source of motivation. She has provided unconditional and kindhearted support, from detailed attention during stressing deadline times to skilled proof-reading of my drafts. This thesis is as much hers as it is mine.

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 of the material presented in this thesis is described in the following papers:

- Gonzalez-Garcia Abel, Vezhnevets Alexander, Ferrari Vittorio. An active search strategy for efficient object class detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (**CVPR**), 2015.

- Gonzalez-Garcia Abel, Modolo Davide, Ferrari Vittorio. Do semantic parts emerge in Convolutional Neural Networks? In *International Journal of Computer Vision* (**IJCV**), 2017

- Gonzalez-Garcia Abel, Modolo Davide, Ferrari Vittorio. Objects as context for part detection. In *arXiv*:1703.09529 preprint, 2017. To be submitted to **CVPR**, 2018. The code for this paper has been made publicly available at `http://github.com/agonzgarc/parts-object-context`.

Authorship of the second contribution is shared with fellow PhD student Davide Modolo. Davide contributed substantially to the development of this work (40%). We designed and developed most of the analysis presented in this chapter together. Nonetheless, for the purpose of providing an explicit division of our contributions, I take credit for the following: sec. 4.3.1, sec. 4.3.2 (except for "differences between part sizes"), sec. 4.4, and sec. 4.5.

(Abel Gonzalez-Garcia)

To my wife.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

We live in a highly visual world. As humans, we heavily rely on visual input for a multitude of our everyday tasks. This dependence has been on the rise in the last few years, as images are more ubiquitous every day. The availability of cheap digital cameras, especially those in smartphones, have greatly contributed to the monumental increase in available imagery. Social media platforms (e.g. Facebook, Instagram) and messaging applications with image support (e.g. Whatsapp, SnapChat) have paved the way for the image as a central means of communication. Humans are inherently good at understanding images and extracting vast amounts of information from them. With just a brief glance at an image depicting a fairly complex scene, we can obtain a great deal of information. For example, we can easily recognize where the scene is, what actions are taking place, and what are the main actors and objects. We can also have a notion of the scene's 3D layout and the arrangement of its components. We can even make predictions regarding future events in the scene. Nonetheless, the development of computer algorithms that automatically achieve such detailed level of understanding is a very challenging research task.

Computer vision is a science field that aims at automatically analyzing and understanding visual inputs. Commonly, the visual input is a still image, although other types of inputs may be used, such as videos (Marin-Jimenez et al., 2014; Karpathy et al., 2014) or depth images (Shotton et al., 2011; Ye and Yang, 2014). Given an input image, different computer vision tasks act at multiple levels of detail. The coarsest tasks label the image as a whole. Some examples are *scene recognition* (Xiao et al., 2010; Juneja et al., 2013) and *image classification* (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015), which choose from a finite set of classes to categorize the whole scene or the most prominent object in the image, respectively. Similarly, *action recog-*

nition (Desai and Ramanan, 2012; Gkioxari et al., 2015) determines the main activity occurring in the image, whereas *image captioning* (Vinyals et al., 2015; Karpathy and Fei-Fei, 2015) describes the image using natural language. Some tasks belong to a finer level and label specific areas of the image. For example, *object detection* (Felzenszwalb et al., 2010b; Girshick, 2015) recognizes the objects present in the image and localizes where they appear. With even further detail, *part detection* (Chen et al., 2014; Wang et al., 2015b) recognizes and localizes the semantic parts that compose the objects, and *pose estimation* (Liu et al., 2014; Pishchulin et al., 2016) predicts the inner spatial arrangement of the objects.

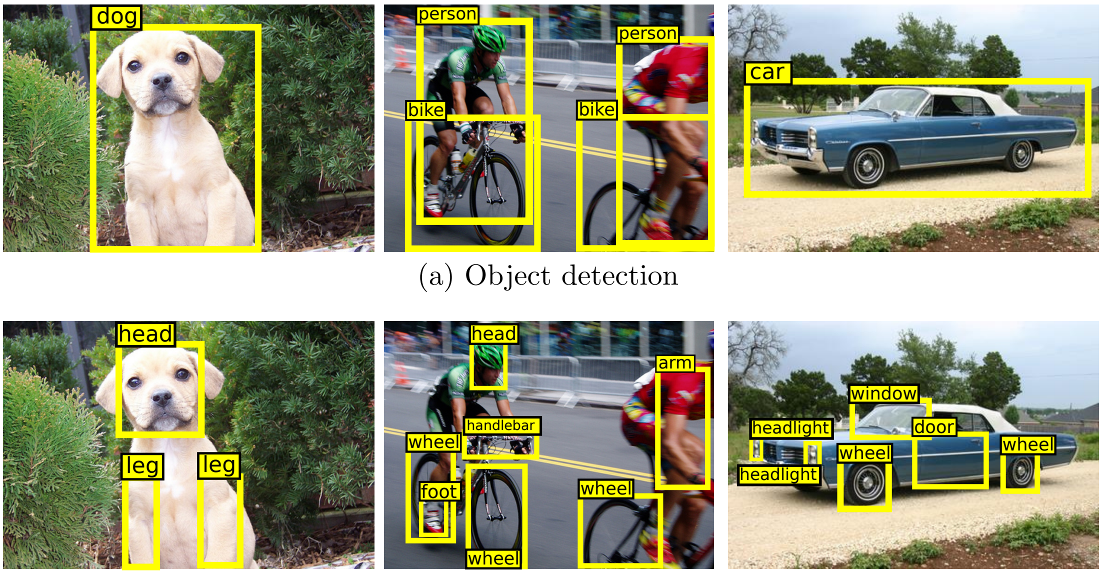
This thesis addresses two computer vision tasks: object and part detection. We define these tasks and list some applications in sec. 1.1. Sec. 1.2 describes the main contributions of this thesis. Finally, sec. 1.3 presents how the thesis is organized.

1.1 Object and part detection

Objects are arguably one of the most relevant image components. For this reason, object detection can be considered among the central tasks of computer vision. Object detection has a two-fold goal. First, it attempts to recognize all the objects present in an image by categorizing them into a fixed, finite set of object classes. Second, it tries to localize each object instance by identifying the region of the image where the object appears. In this thesis, localization is performed by placing an axis-aligned rectangular region, called *bounding-box*, around each object instance (fig. 1.1).

Object detection in real-world images is a highly challenging task. An object class may span a wide range of possibly disparate object appearances. For example, the object class ‘car’ ranges from small compact cars to large pickup trucks, whose appearances are notably distinct. This is called *intra-class variation*, and it results in object detection approaches having to learn an ample variety of object appearances, with the intention of maximizing the number of detected object instances. Similarly, objects are depicted in a multitude of poses and viewpoints, thus greatly affecting their appearance. Moreover, external conditions such as illumination or low image resolution add further difficulty to the task, by limiting information usable for recognition. Finally, objects may appear truncated or partially occluded, especially in cluttered images.

In part detection, each class is an object part, a piece of a bigger object composed of multiple parts. Therefore, the aim is recognizing and localizing each part of the



(a) Object detection

(b) Part detection

Figure 1.1: *Examples of outputs for the (a) object detection and (b) part detection tasks. Note the greater difficulty of part detection due to the smaller instance size and the unclear separation between some classes. Each detection has an associated score, not shown here for simplicity.*

object to obtain a finer characterization of the object. Some works (Fergus et al., 2003; Arbeláez et al., 2012; Song et al., 2014; Endres et al., 2013; Tsai et al., 2015; Zhang et al., 2014b; Juneja et al., 2013; Simon and Rodner, 2015) understand as object part any image patch that is discriminative for an object class. In this thesis, we follow another line of work (Wah et al., 2011a; Sun and Savarese, 2011; Ukita, 2012; Azizpour and Laptev, 2012; Liu et al., 2012; Zhang et al., 2013; Chen et al., 2014; Vedaldi et al., 2014; Liu et al., 2014; Zhang et al., 2014b; Chen et al., 2014; Gkioxari et al., 2015; Wang et al., 2015b; Wang and Yuille, 2015; Yang et al., 2016) and refer to parts as *semantic parts*, i.e. object regions that are interpretable and nameable by humans, such as ‘arm’ or ‘head’[1].

The part detection task is generally more challenging than its object counterpart for several reasons. First, parts are smaller than whole objects. This fact decreases their resolution in images and increases the difficulty of localization. Moreover, the boundary between two different parts is often not very clear. For example, the legs of a dog are smoothly attached to its torso without an obvious separation (fig. 1.1). On the other hand, parts exhibit lower intra-class variation, which offsets to some extent

[1] Unless otherwise stated, ‘part detection’ refers to semantic part detection throughout this thesis.

the extra difficulty. As in the object detection case, in this thesis we localize parts up to a bounding-box.

Object and part detection are narrowly related and both tasks may benefit from exploiting the interplay between them. In one direction, parts contain valuable information regarding the objects to which they belong. The presence of some parts might give excellent clues about the object class, e.g. detecting a beak indicates that there must be a bird. Furthermore, the appearance of individual parts might also be very discriminative for the recognition of the object. For instance, we can distinguish different types of vehicles by the appearance of their wheels. This direction of the interplay between objects and parts is already successfully exploited by part-based object detection models in the literature (Felzenszwalb et al., 2010b; Azizpour and Laptev, 2012; Chen et al., 2014; Tsai et al., 2015). In the other direction, objects provide an excellent contextual support for the parts they contain: if we detect a person, we can expect to find parts such as the head, arms, or legs at specific relative locations. We exploit this direction in chapter 5, where we perform part detection using the object as context.

Applications. Object detection has abundant real-world applications. From assisting visually impaired people by describing scenes in movies or helping them navigate the world (Hub et al., 2004; Chen and Yuille, 2005), to the use of robot agents in critical situations such as natural disasters or nuclear accidents (Collet et al., 2009; Lee et al., 2017). Another relevant example is self-driving cars (Geiger et al., 2012; Wu et al., 2017), for which accurately detecting pedestrians, street signs, or other cars, is a crucial feature when maximizing the safety of such systems. Moreover, part detection delivers a more comprehensive image understanding, enabling reasoning about object-part interactions in semantic terms. Parts are necessary to obtain rich object characterizations and image descriptions at a fine level, e.g. “a person is grabbing a dog by its tail” or “the car has its headlights turned on”. Furthermore, only by correctly decomposing objects into parts will robot agents be able to fully interact with the world, such as grabbing a briefcase by its handle, or opening the door of a car. Given the multiple benefits of semantic parts, part-based models have gained attention for tasks such as fine-grained recognition (Zhang et al., 2014a; Lin et al., 2015; Zhang et al., 2016; Parkhi et al., 2012), object class detection and segmentation (Chen et al., 2014; Wang et al., 2015b), articulated pose estimation (Liu et al., 2014; Sun and Savarese, 2011; Ukita, 2012; Yang et al., 2016), and attribute prediction (Zhang et al., 2013; Vedaldi et al., 2014; Gkioxari et al., 2015).

1.2 Contributions

This thesis has four main contributions, two about objects and two about parts. The first two tackle the object detection task. They develop visual search strategies guided by image context to efficiently detect objects in images, for one or multiple classes, respectively. The third contribution is an analysis regarding the emergence of semantic parts in neural networks trained for object detection. Finally, the last contribution improves on part detection by incorporating object context. More concretely, the contributions of this thesis are:

- An active search strategy for efficient object class detection (chapter 3). Modern object detectors evaluate a large set of windows using a window classifier. Our active search sequentially chooses what window to evaluate next based on all the information gathered before. We guide our search strategy with two different cues, the classifier score and the image context, exploiting the statistical relation between the appearance of a window and its location relative to the object, as observed in the training set. This results in a significant reduction of the number necessary window evaluations to detect the objects in the image. This contribution was published at CVPR 2015 (Gonzalez-Garcia et al., 2015).

- A multi-class joint active search (chapter 3). We extend our active search to jointly detect pairs of object classes that appear close in the image. We first find both classes as a unit, since this is generally an easier task than independently finding the individual classes. Then, we branch off into individual searches for each class, exploiting the information that one class can provide about the location of the other. This leads to an even further reduction of the number of necessary evaluations for the challenging classes.

- We study whether semantic parts emerge in Convolutional Neural Networks trained for object detection, among other visual recognition tasks (chapter 4). We perform two quantitative analyses that shed some light on this by investigating the responses of the network filters. The first uses annotated ground-truth bounding-boxes of parts, whereas the second draws its conclusions by directly asking human annotators. Moreover, we explore several connections between discriminative power and semantics. We find out which are the most discriminative filters and semantic parts for object recognition. This enables to gain an even deeper understanding of the role of semantic parts in the network. This contribution has been accepted for publication (subject to minor revision) at IJCV (Gonzalez-Garcia et al., 2017a).

- A part detection approach that takes object context into account (chapter 5). We leverage object information, such as the object appearance and its class, to enhance the part representation used for part recognition. We also model the expected relative location of parts inside the objects based on their appearance. Our model effectively combines all the available cues, significantly outperforming approaches that use part appearance alone in the challenging task of part detection. This contribution will be submitted to CVPR 2018 and is available as preprint (Gonzalez-Garcia et al., 2017b).

1.3 Organization

This thesis is organized as follows. This introductory chapter briefly presents object and part detection, the two computer vision tasks approached in this thesis. Chapter 2 provides a background exposition of both tasks, and presents the background of the use of context for recognition, which is a common element in several contributions of the thesis. Chapter 3 covers the first two contributions, whereas chapters 4 and 5 detail the third and fourth contributions, respectively. Each chapter includes related work relative to that specific contribution, a detailed description of the employed methods, a series of conducted experiments, a discussion of the obtained results, and an outlook on possible future directions. Finally, chapter 6 presents the general conclusions of this thesis and additional future work. The acronyms used throughout the thesis are reported in full length in appendix A. Appendix B lists all object and part classes used in chapters 4 and 5.

Chapter 2

Background

This chapter explores the background of the two computer vision tasks addressed in this thesis, namely object and part detection. We present a brief history of both tasks along with their most relevant recent techniques. Additionally, we give an overview of several approaches that incorporate context for the object detection task, as this is a cornerstone of this thesis.

2.1 Object detection

Object detection is arguably one of the main tasks of modern computer vision and, as such, has been extensively researched. Therefore, an overwhelmingly large number of methods approach object detection, and the summary provided here is far from being exhaustive. We present an overview of the most relevant landmark works that have kept the field moving forward, with special focus on those that are explicitly used in this thesis. First, we describe how object detection is evaluated.

Evaluation. Object detection approaches take a test image as input and output object *detections*, composed of bounding-boxes around the objects of each class and their associated scores (fig. 1.1). To evaluate the method's performance in this task, we need to determine whether each object has been successfully localized and recognized. Everingham et al. (2006) introduced a widely used measure to evaluate localization, *Intersection-over-Union* (IoU), defined as

$$IoU = \frac{area(B_d \cap B_{gt})}{area(B_d \cup B_{gt})}, \tag{2.1}$$

where B_d is a detection bounding-box and B_{gt} is a ground-truth bounding-box. A detection for a particular class is usually considered correct if its IoU with any ground-truth bounding-box of the class is greater than 0.5.

Datasets are commonly divided into a *training* set and a *test* set. Sometimes, a *validation* set is also provided for learning the model's hyperparameters. Methods are trained on the training set and evaluated on the test set, by running object detectors for all classes on every test image. We can compute the method's *precision* as the fraction of correct detections among all detections, where a detection is considered correct as aforementioned. The method's *recall* is computed as the fraction of all instances in the test set that have been correctly detected. The standard evaluation protocol ranks all detections by score and computes a Precision-Recall curve. This curve expresses how precise the method is at different levels of recall. The *Average Precision* (AP) measure is then used to summarize the shape of the curve. It averages the precision values at all recall points of the curve. In practice, Everingham et al. (2006) defines 11 equally spaced recall thresholds, interpolates the precision at each of them, and then computes the average of the resulting precision values. When considering multiple classes, the measure of performance becomes *mean Average Precision* (mAP), which is the mean of the individual AP values for each class.

Despite the ubiquitousness of this standard evaluation methodology, it might be inadequate in providing a complete disclosure of the method's performance. For example, the selection of the training, validation, and test data sets is arbitrary and might induce significant biases. Since these sets are inmutable in the standard protocol, authors often overfit their methods to the chosen test set, thus reducing the method's generalization ability. An alternative evaluation methodology based on a cross-validation technique would lead to a more reliable performance assesment.

2.1.1 Early approaches

Early work in the area focused on the simpler problem of recognizing specific objects (e.g. my car) as opposed to object classes (e.g. any car). The first specific object recognition methods (Lowe, 1987; Huttenlocher and Ullman, 1987; Rothwell et al., 1992) tried to geometrically match the shape of a model object to edge segments discovered in the test image. Such geometric methods are bound to fail when the object's shape is heavily altered, for example when objects appear in varying poses or under different illumination conditions. To overcome this limitation, appearance-based methods (Murase and Nayar, 1995; Lowe, 1999; Schmid and Mohr, 1996; Rothganger et al., 2003; Mahamud and Hebert, 2003; Tuytelaars and Van Gool, 2004; Moreels et al., 2004; Ferrari et al., 2004) started modeling the local appearance of object patches instead of the object's overall shape. This results in an increased robustness to pose and illumination changes.

The core steps of most appearance-based methods are finding a set of interest points in the test image, describing the appearance of local patches around them, and matching them with similarly extracted features from training images. A widely used interest point detector is the *Harris-Laplacian* operator of Mikolajczyk and Schmid (2004), which is invariant to affine transformations such as rotation, scale, or translation. The appearance of local patches is commonly described with local invariant descriptors such as *Scale-Invariant Feature Transform* (SIFT) (Lowe, 1999). The SIFT descriptor is a summary of the gradient information around a local neighborhood of each interest point. It places a 4×4 rectangular grid centered at the point and computes a spatial histogram of the gradient directions, which are quantized into 8 bins and weighted by gradient magnitude. Finally, the descriptor is then contrast normalized to make it more robust to illumination changes. For the last step of the appearance-based methods, features are matched between training and test images by using some distance function between descriptors.

2.1.2 Sliding-window

The next generation of object detectors tackled the more general problem of recognizing object classes rather than specific objects. Instead of relying on interest points, many successful object class detectors (Viola and Jones, 2001; Schneiderman and Kanade, 2004; Dalal and Triggs, 2005; Felzenszwalb et al., 2010b) adopted the *sliding-window* paradigm. Sliding-window first partitions the image into a large, regular grid of windows at multiple locations and scales. Then, it applies a window classifier to every window in the grid to determine whether they contain an object of the target class or not. Finally, the local maxima among all the windows become the output detections.

Viola and Jones (2001) employed this paradigm to develop a powerful and efficient face detector. Their success can be largely attributed to the use of simple and fast to compute features called Haar-like wavelets, and to its efficient cascaded detector based on Adaboost (Freund and Schapire, 1997). This detector consists in a cascade of weak classifiers, each of which performs decisions based on features previously selected as discriminative. The cascade is arranged such that those windows that are unlikely to contain faces are quickly disregarded. Therefore, only the most promising windows are processed by strong classifiers. Since the strong classifiers are also the slowest, this detector is extremely efficient.

Another successful sliding-window detector is the person detector of Dalal and Triggs (2005). They introduced *Histogram of Gradients* (HOG), a feature descriptor closely related to SIFT (Lowe, 1999). In fact, its three basic steps are the same: gradient computation, weighted vote with quantization into cells for different locations and orientations, and contrast normalization. The main difference resides in the contrast normalization step, which in the HOG descriptor is performed on overlapping spatial cells. As window classifier, Dalal and Triggs (2005) use a *Support Vector Machine* (SVM) on HOG features. The detector's last step is Non-Maxima Suppression (NMS), which fuses multiple detections of the same object. An important aspect of this detector is the incorporation of *hard-negative mining*. This bootstrapping technique first runs the full detection pipeline on training images and collects hard-negatives, i.e. misclassified negative windows. Then, it retrains the window classifiers using these hard-negatives as examples, which substantially reduces the false-positive rate.

The HOG detector of Dalal and Triggs (2005) assumes that people are always upright and fully visible. Despite its unprecedented performance, this assumption severely limits the detector's applicability, as people appear in a wide range of articulated poses in real-world images. Moreover, occlusions are commonplace, especially in cluttered scenes. The *Deformable Part Model* (DPM) of Felzenszwalb et al. (2010b) extends the HOG detector to overcome this limitation and to detect more object classes. DPM takes inspiration from the old pictorial structure approach of Fischler and Elschlager (1973), which models objects as a collection of parts and the relative locations between them. The idea of modeling objects as constellations of rigid parts had already attracted some attention in the field (Weber et al., 2000; Fergus et al., 2005; Felzenszwalb and Huttenlocher, 2005), but none of these approaches were close to DPM's outstanding performance.

DPM consists of a set of HOG templates that act as filters at two different levels (fig. 2.1a). The *root filter* models the appearance of the whole object. The *part filters* model the appearance of each of the parts, and work at twice the resolution of the root filter. DPM follows a star structure centered at the root node (Felzenszwalb and Huttenlocher, 2005), and each part is expected to lie at a particular relative location with respect to the center. In practice, parts do not always appear exactly at the expected locations, but also in their close surroundings. DPM models this by assigning deformation costs to each of the parts through a quadratic function on the part displacements. Therefore, parts further away from their expected locations are more severely penalized. Moreover, objects may appear under different part arrangements. For example, the legs of person are located below the root if the person is upright, but on the sides if the person is lying down. For this reason, the DPM detector is actually a mixture model with several components, each accounting for a different aspect variation of the object (e.g. viewpoints).

At test time, DPM is applied in a sliding-window manner, where each component outputs a score for every window. The score for each component is computed as a weighted sum of the root filter score, the score of each part filter, and their deformation costs. The final score is the maximum score over all components. Since parts annotations are not given, the window classifier is a *latent* SVM (Yu and Joachims, 2009) for which the part locations are latent variables.

2.1.3 Bag-of-Words and object proposals

Alternatively, *Bag-of-Words* (BoW) models (Csurka et al., 2004; Sivic et al., 2005; Harzallah et al., 2009; Prest et al., 2012; Uijlings et al., 2013) disregard all spatial relations within the object. Inspired by the text analysis literature, these models represent objects as an orderless collection of features. More specifically, BoW models extract and cluster visual features (e.g. SIFT (Lowe, 1999)) from a set of training images. Each cluster, or *visual word*, represents one of the recurring visual patterns that compose the images, analogously to how words compose texts. The set of all visual words form the *visual codebook*, i.e. the word dictionary in the text analogy. An image region can now be represented by a histogram of visual words (fig. 2.1b). Some approaches also add first and second order statistics to enhance the representation (e.g. Fisher vectors (Perronnin and Dance, 2007; Perronnin et al., 2010; Cinbis et al., 2013) and VLAD (Jégou et al., 2010)). Originally, BoW models were introduced for whole-image classification (Csurka et al., 2004; Lazebnik et al., 2006), and worked by inputting the BoW representation into an image classifier, e.g. SVM, that predicted the presence of the object in the image. Afterwards, such BoW models were used as window classifiers for object detection (Harzallah et al., 2009; Prest et al., 2012; Uijlings et al., 2013).

Fig. 2.1c shows different ways in which the features used in the representation can be extracted from the image region. Common approaches use interest points (Csurka

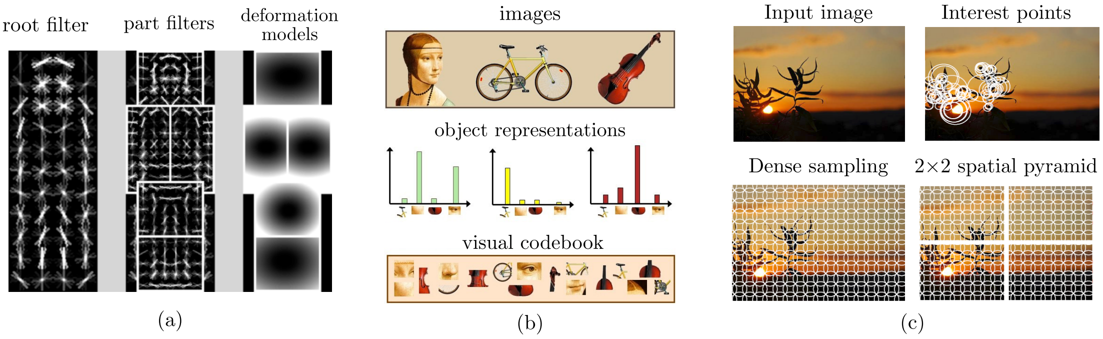


Figure 2.1: *(a) HOG templates and deformation models used in DPM (Felzenszwalb et al., 2010b) for class person. (b) Example of BoW object representations, adapted from Fei-Fei et al. (2005). (c) Different sampling strategies used in BoW models, adapted from van de Sande and Gevers (2010)*

et al., 2004; Sivic et al., 2005) or a regular, possibly dense grid (Vogel and Schiele, 2002; Fei-Fei and Perona, 2005; Uijlings et al., 2013). Some weak spatial information may also be added by using a *spatial pyramid* (Lazebnik et al., 2006). This method divides the image region in different subregions and independently pools visual words from each of them. The final representation is the concatenation of the histograms for each region, thus multiplying the dimensionality of the BoW representation by the number of subregions.

An example of a successful BoW approach is the *UvA* detector of Uijlings et al. (2013), winner of the PASCAL VOC2012 detection challenge (Everingham et al., 2012). They describe each window using a densely sampled 3x3 spatial pyramid BoW representation. The codebook is created using a Random Forest on RGB-SIFT (Van De Sande et al., 2010) descriptors previously reduced with Principal Component Analysis (PCA), and it has 4096 words. Overall, their window descriptor has 36864 dimensions. To classify each window, they use a SVM with a histogram intersection kernel on these features. Given the high dimensionality of the features and the complexity of the kernel, this window classifier is too expensive to be applied in a sliding-window fashion.

For this reason, recent and highly accurate detectors like UvA evaluate their window classifiers only on smaller window sets produced by *object proposals* generators (Alexe et al., 2010; Manen et al., 2013; Van de Sande et al., 2011; Dollar and Zitnick, 2014). Object proposals are sets of class-independent candidate windows likely to cover all the objects in the image. These sets are generally rather small, in the order of a few thousand windows, as opposed to the hundreds of thousands in the sliding-window paradigm. The seminal work of Alexe et al. (2010), *Objectness*, was the first to introduce such object proposals for object detection, but many followed shortly after. Among the most notorious we find Selective Search (Van de Sande et al., 2011), Randomized Prim's (Manen et al., 2013), CPMC (Carreira and Sminchisescu, 2012), BING (Cheng et al., 2014), and Edge Boxes (Zitnick and Dollár, 2014). Hosang et al. (2016) presents a thorough analysis that explores the properties and performance of an extensive list of object proposal methods. In general, object proposals enabled the use of computationally expensive classifiers. They have been later applied to other classifiers beyond BoW, resulting especially useful for Convolutional Neural Network classifiers (Girshick et al., 2014; Girshick, 2015).

2.1.4 Convolutional Neural Networks

In the last few years, the state-of-the-art in object detection has been clearly dominated by approaches based on *Convolutional Neural Networks* (CNNs). Back in 1998, LeCun et al. (1998) showed how a CNN was an excellent model to perform handwritten digit recognition. However, several limitations prevented scaling up this model for object recognition in high-resolution, real-world images. First, CNNs are very slow to train, even for small input resolutions. Therefore, increasing the model's input resolution rendered CNN architectures effectively untrainable. Second, the enormous amount of model parameters requires abundant amounts of data for effective training. Finally, models with such number of parameters tend to overfit to the training data.

Image classification. The real CNN breakthrough in mainstream computer vision occurred in 2012, when Krizhevsky et al. (2012) overcame these limitations through an efficient GPU implementation, the use of great amounts of training data (Russakovsky et al., 2015), and a new regularization technique called *dropout*. Their model became the winner of the ImageNet ILSVRC2012 image classification challenge (Russakovsky et al., 2015) by a large margin. Thereafter, object detection approaches have heavily relied on CNNs to achieve ever more impressive results (Sermanet et al., 2014; Girshick et al., 2014; He et al., 2014; Girshick, 2015; Ren et al., 2015; He et al., 2016). Besides object detection, deep neural networks have achieved indisputable hegemony in numerous computer vision tasks such as image classification (Simonyan and Zisserman, 2015; Szegedy et al., 2015), fine-grained recognition (Zhang et al., 2014a; Lin et al., 2015; Zhang et al., 2016), semantic segmentation (Hariharan et al., 2014; Long et al., 2015; Caesar et al., 2015), human pose estimation (Pishchulin et al., 2016; Pfister et al., 2015), image captioning (Vinyals et al., 2015; Karpathy and Fei-Fei, 2015), and object tracking (Hong et al., 2015; Wang et al., 2015a).

CNNs are feed-forward artificial neural networks designed to have images as inputs. As with standard neural networks, the input is processed through a series of hidden layers composed of neurons with learnable weights and biases, generally followed by a non-linear activation function. The distinctive characteristic of a CNN is the convolutional nature of its filters, which are slided convolutionally over the input. Therefore, convolutional layers act only on a local region at a time, as opposed to fully connected layers, which act on the whole input at once. The network may also contain other types of layers. For example, pooling layers are used to downsample the input, providing some translation invariance and decreasing the number of parameters, which in turn reduces overfitting. The input of every layer is the output of the previous one, except for the first layer, which takes the image as input. The output of the model is the output of the last network layer. For example, in an image classification CNN, the last layer outputs the class scores of the input image. A loss function is defined on the output layer and optimized using Stochastic Gradient Descent (SGD) with *backpropagation* (LeCun et al., 1998). Backpropagation updates the weights of each layer using a single backward pass that computes derivatives in a memory-efficient fashion.

The original model of Krizhevsky et al. (2012), commonly known as *AlexNet*, contains five convolutional layers followed by three fully connected layers. As non-linearity it uses Rectified Linear Units (ReLU), defined for input x as $\max(0,x)$. The non-saturating properties of the ReLU result in faster training compared to other activation functions, such as the hyperbolic tangent. The first two convolutional layers are also followed by local response normalization layers. Max-pooling layers follow the normalization layers as well as the last convolutional layer. This type of pooling layer summarizes each local input patch by its maximum value. More details regarding this architecture can be found in Krizhevsky et al. (2012).

Several other architectures are in use nowadays. As depth seems to play a decisive role in the network's performance, newer architectures tend to be deeper. An example of this is *VGG-16*, the successful 16-layer model of Simonyan and Zisserman (2015). VGG-16's smaller convolutional filters allow for a greater network depth, which results in significant performance gains. The current state-of-the-art network is the *Residual Net* of He et al. (2016), whose depth ranges from 100 layers to the staggering amount of 1000 layers. Residual Nets enable these extremely deep architectures by explicitly forcing its layers to learn a residual mapping, implemented by shortcut connections

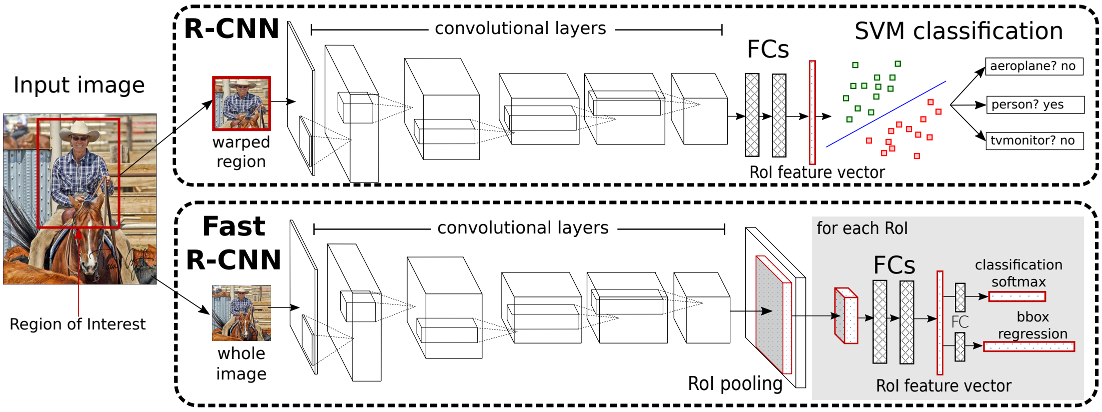


Figure 2.2: *CNN-based approaches for object detection. R-CNN (Girshick et al., 2014) processes all windows independently through all the layers in the network and uses a SVM to classify each window. Fast R-CNN (Girshick, 2015) processes the whole image once through all convolutional layers and classifies each window by pooling from the corresponding region of the convolutional map.*

that skip one or more layers. They rely on the fact that learning a residual mapping and then recasting it into the desired mapping is considerably easier to optimize than directly learning the desired mapping.

Object detection. One of the first CNN-based object detectors, *Overfeat* (Sermanet et al., 2014), proposed an approach that combined a CNN classifier with the sliding-window paradigm. However, it was Girshick et al. (2014) who implemented the first state-of-the-art CNN object detector with *Region-CNN* (R-CNN). The unprecedented performance of R-CNN can be mostly attributed to the two following factors.

First, the description of image windows using the powerful image classification CNN model of Krizhevsky et al. (2012), which is much more effective than HOG features or the BoW representation. For each proposal, the model extracts fixed-length features from intermediate network layers and uses them to train a linear SVM. These features provide an excellent visual representation (Donahue et al., 2014) that facilitate the learning of classification decisions. The use of these features was enabled by the adoption of object proposals (Selective Search (Van de Sande et al., 2011)), as such CNN model is prohibitively expensive for a sliding-window approach. R-CNN also includes a per-class bounding-box regression mechanism that refines its detections to enclose the object instances more accurately.

Second, the development of an effective two-stage network training. In a first stage called *pre-training*, the network is trained for the image classification task. Pre-training enables the network to learn a discriminative visual representation by leveraging the vast amounts of annotated data for such task (Russakovsky et al., 2015). The second stage consists in *fine-tuning* the network weights for the object detection task. This normally requires adapting the last layer to accommodate for the difference in the number of classes. Fine-tuning tailors the general visual representation learned during the pre-training stage for the actual task at hand. Moreover, this process prevents overfitting the CNN to the relatively small object detection dataset by leveraging the great amounts of data used during pre-training.

Many object detection works (He et al., 2014; Hariharan et al., 2014; Girshick, 2015; Lenc and Vedaldi, 2015a; Ren et al., 2015; Zhang et al., 2016; Huang et al., 2016; Papadopoulos et al., 2016; Shrivastava and Gupta, 2016; Li et al., 2016; Lin et al., 2017) have either successfully used R-CNN in their models or built on it as a starting point. Among the most successful extensions of R-CNN we find Fast R-CNN (Girshick, 2015) and Faster R-CNN (Ren et al., 2015).

Fast R-CNN builds on an idea initially proposed by He et al. (2014) that consists in processing all convolutional layers only once for the whole image, and then extracting window-specific features from the corresponding region of the convolutional map (fig. 2.2). This approach considerably increases the model's efficiency, as most of the computation is shared among all windows. Fast R-CNN combined this with the integration of a multi-task loss function within the network, allowing to simultaneously train for classifiying object proposals and the bounding-box regressor.

Faster R-CNN extends Fast R-CNN by introducing a Region Proposal Network (RPN) that supercedes the manually engineered object proposal generators. Instead of relying on externally generated object proposals, the RPN directly proposes candidate windows within the network. This approach reduces the total detection time, as object proposal generators are Fast R-CNN's computational bottleneck. Moreover, it enables to train the proposal generator too, which leads to better performance.

The latest leading object detection approaches go a step further and directly predict object locations instead of using object proposals. For example, SSD (Liu et al., 2016) generates adjustments for a set of default boxes in the image. The current state-of-the-art approach for real-time object detection, YOLO (Redmon et al., 2016; Redmon and Farhadi, 2017), divides the image into a regular grid of *cells* and predicts bounding-boxes and scores for each of these cells. This line of work results in an intellectual evolution step for object detection and regards the window classifier approach obsolete.

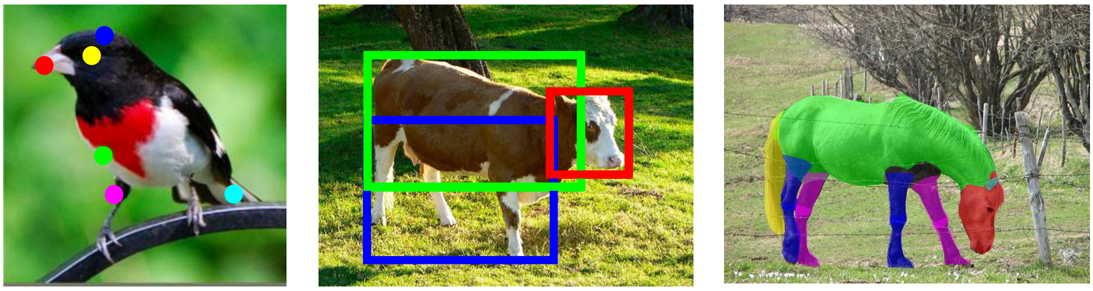
In this thesis, we use R-CNN in chapters 3 and 4, and in chapter 5 we build a new detection model based on Fast R-CNN.

2.2 Part detection

The word *part* can be associated with two different computer vision concepts. First, a part can be understood as any image patch that is discriminative for an object class (Fergus et al., 2003; Arbeláez et al., 2012; Song et al., 2014; Endres et al., 2013; Tsai et al., 2015; Zhang et al., 2014b; Juneja et al., 2013; Simon and Rodner, 2015). For example, the parts used in the DPM object detector (Felzenszwalb et al., 2010b) are of this type. Alternatively, we can refer to parts as *semantic parts*, i.e. object regions that are interpretable and nameable by humans, such as 'head', 'wheel', or 'handle' (Wah et al., 2011a; Sun and Savarese, 2011; Ukita, 2012; Azizpour and Laptev, 2012; Liu et al., 2012; Zhang et al., 2013; Chen et al., 2014; Vedaldi et al., 2014; Liu et al., 2014; Zhang et al., 2014b; Chen et al., 2014; Gkioxari et al., 2015; Wang et al., 2015b; Wang and Yuille, 2015; Yang et al., 2016). In this thesis, we address the detection of the latter type, and we evaluate part detection analogously to object detection.

Semantic part detection is an interesting and challenging task, yet only seldom explored in the detection literature. In fact, parts are generally used as support for other computer vision tasks, given their multiple benefits. First, they present lower intra-class variation than whole objects, as parts tend to be rather similar to each other. Second, visual representations based on parts are more robust to pose variation. Third, part configurations convey relevant information regarding the aspect of the objects. For example, we can infer the object viewpoint based on the part locations. Finally, parts capture subtle appearance variations at a finer granularity than objects.

For these reasons, part-based models have gained a great deal of attention and have been successfully applied to a multitude of tasks. An excellent example of a task that heavily uses part-based models is *fine-grained recognition*. The goal of fine-grained recognition is to discriminate between sub-categories of a class, such as car models or bird species. Sub-categories are very similar to each other, as they contain the same parts under similar configurations. Therefore, discrimination is only possible by focusing on subtle differences in the appearance of specific parts, e.g. the shape of a bird's beak. Fine-grained recognition methods exploit this by explicitly incorporating semantic parts in their models, enabling visual representations with higher discriminative potential (Parkhi et al., 2012; Zhang et al., 2014a; Lin et al., 2015; Shih et al., 2015; Xiao et al., 2015; Zhang et al., 2016; Akata et al., 2016).



(a) (b) (c)

Figure 2.3: *Three different precision levels for part localization: (a) keypoints (Huang et al., 2016), (b) bounding-boxes, and (c) segmentation (Cheng et al., 2014).*

Semantic parts have been used also for object class detection (Felzenszwalb et al., 2010b; Azizpour and Laptev, 2012; Chen et al., 2014; Tsai et al., 2015) and segmentation (Arbeláez et al., 2012; Wang and Yuille, 2015; Wang et al., 2015b), in which part are especially helpful when the whole appearance of the object is not very informative due to occlusions or low resolution. In human pose estimation (Liu et al., 2014; Sun and Savarese, 2011; Ukita, 2012; Yang et al., 2016), parts are crucial to determine the object's spatial configuration. In the attribute prediction task (Gkioxari et al., 2015; Vedaldi et al., 2014; Zhang et al., 2013), parts might contain particular information in relation to the object's attributes. Parts may also be used in whole scene classification (Juneja et al., 2013). Finally, parts are necessary to obtain rich object characterizations and deliver a more comprehensive image understanding, enabling reasoning about object-part interactions in semantic terms.

Methods in the literature localize part instances mainly using three different precision levels (fig. 2.3). The simplest localization level places a keypoint to indicate the part center (Wah et al., 2011a; Gavves et al., 2013; Liu et al., 2014; Simon and Rodner, 2015; Huang et al., 2016; Shih et al., 2015; Akata et al., 2016). The next level delimits the whole extent of the part up to a bounding-box (Sun and Savarese, 2011; Ukita, 2012; Zhang et al., 2014a; Gkioxari et al., 2015; Chen et al., 2014; Vedaldi et al., 2014; Lin et al., 2015; Xiao et al., 2015), as in standard object detection (Felzenszwalb et al., 2010b; Girshick et al., 2014; Girshick, 2015). Lastly, the most precise level determines the accurate area of the the part through a pixel-wise segmentation mask (Wang et al., 2015b; Hariharan et al., 2015; Liang et al., 2016a; Xia et al., 2016; Wang and Yuille, 2015; Yang et al., 2015). In chapters 4 and 5 of this thesis, we focus on the intermediate precision level and localize semantic parts up to a bounding-box.

Despite the relevance of semantic parts, not many works tackle part detection as a task in itself. In fact, it is common to simply repurpose an object detection model to detect parts instead of objects. For example, several works use HOG templates to detect semantic parts (Sun and Savarese, 2011; Azizpour and Laptev, 2012; Ukita, 2012; Vedaldi et al., 2014; Cheng et al., 2014). Zhang et al. (2014a) use R-CNN, the object detection approach of Girshick et al. (2014), but trained for part classes instead of object classes. Moreover, a few works use even simpler approaches, relying on the assumption that convolutional filters can act as part detectors without explicitly training the model to detect parts (Gkioxari et al., 2015; Xiao et al., 2015). In chapter 4 we investigate whether this assumption holds true. Moreover, we propose in chapter 5 a dedicated part detection approach that modifies a CNN architecture to leverage object information.

2.3 Context for recognition

Context is a very rich source of information for visual recognition. It is especially valuable when the sole appearance of the object is not very informative due to occlusions, small size, low resolution, etc. For this reason, the incorporation of context in object recognition has been extensively researched and is still a popular topic nowadays. Many works (Torralba, 2003; Murphy et al., 2003; Russel et al., 2007; Modolo et al., 2015; Harzallah et al., 2009; Song et al., 2011; Hoiem et al., 2008; Vu et al., 2015; Rabinovich et al., 2007; Galleguillos et al., 2008; Desai et al., 2009; Choi et al., 2010; Felzenszwalb et al., 2010b; Heitz and Koller, 2008; Dalal and Triggs, 2005; Li et al., 2011; van de Sande and Gevers, 2010; Crandall and Huttenlocher, 2007; Mottaghi et al., 2014) have attempted to leverage the numerous sources of context in an image. Divvala et al. (2009) presents an extensive analysis of different context sources as well as their impact on object recognition. This section explores the three context types most commonly used as support for the object detection task.

Generally, context approaches in the literature use image context to assist object detection. In chapter 3, we use this type context to guide an active search strategy for object detection. Furthermore, chapter 5 introduces another type of context, object context, which we use to help part detection.

2.3.1 Global image appearance

The global appearance of the image indicates which classes might be present and even where they might be located. In fact, psychological studies show how humans quickly scan the totality of a scene before focusing on specific individual objects (Navon, 1977; Biederman, 1987). Motivated by this, several works model context as the relation between global image features and the objects inside it (Torralba, 2003; Murphy et al., 2003; Russel et al., 2007; Modolo et al., 2015; Harzallah et al., 2009; Song et al., 2011; Hoiem et al., 2008; Vu et al., 2015).

A commonly used global image descriptor is *Gist* (Oliva and Torralba, 2001). Gist is a holistic representation based on low level features extracted from the whole image. Intuitively, the Gist descriptor encodes structural scene properties by summarizing gradient information from different parts of the image. Torralba (2003) employed this descriptor for inferring which object classes might be present and their approximate locations and scales.

In this same spirit, Murphy et al. (2003) extended the work of Torralba (2003) by combining spatial constraints derived from Gist with the outputs of local object detectors trained with boosting. Additionally, they explore the effectiveness of Gist for the image classification task, i.e. predicting the presence of objects *anywhere* in the image. The good results obtained on this task confirm the validity of Gist as a reliable source of information regarding the presence of the objects. Russel et al. (2007) use the Gist descriptor to align the current test scene to example scenes in a large training set of labeled images. Once similar scenes are found, they transfer the annotated object knowledge in the example scenes to the current scene.

Some other works followed a similar idea but using different descriptors or more advanced ways of mapping from global image appearance to the properties of the objects inside it. Harzallah et al. (2009) complement an object detector with an image classification approach, which integrates information about the image as a whole, by combining probabilities in a post-processing step. Similarly, Song et al. (2011) introduce contextual information from the output of an image classification method into a context-adaptive object detector, called Context-SVM. Modolo et al. (2015) use a Random Forest framework to predict likely locations for the objects in the images and remove false positive detections. Hoiem et al. (2008) first estimate the scene's 3D geometry from its global appearance and then use this estimate to predict the expected scale and location of the objects in it. Finally, the head detection work of Vu et al.

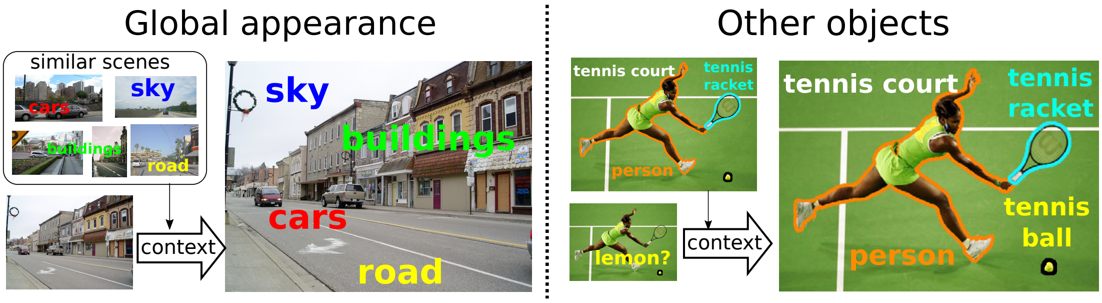


Figure 2.4: *Two types of context widely used for recognition. (Left) Global image appearance may indicate the presence and location of objects. (Right) Some methods exploit the relationships between objects that tend to appear together.*

(2015) trains a CNN model that predicts the locations and scales of heads directly from the input image.

2.3.2 Other objects

Most objects interact with other objects in the image in a variety of ways. Objects may have a semantic relationship, often appearing together in the same image. For example, tennis balls tend to appear in tennis courts, next to tennis players and rackets (fig. 2.4). Therefore, the presence of particular objects in the image may indicate the presence of related objects. Continuing with the previous example, we can expect to find tennis rackets after detecting tennis players. Furthermore, objects can be physically related, presenting stable spatial configurations across different images. For example, keyboards tend to be next to monitors because they are functionally connected. Consequently, many works exploit these phenomena by explicitly modeling the relationships between objects.

Rabinovich et al. (2007) incorporate semantic object context by reasoning about co-occurrence of object classes. This is done by optimizing label agreement through a Conditional Random Field (CRF) that integrates information from all image regions. Their method can be applied on top of any object detector that classifies disjoint image regions, as it rescores the classifiers in a post-processing step. Galleguillos et al. (2008) extend Rabinovich et al. (2007) by also reasoning about the spatial relationships between objects, modeled as pairwise terms in the CRF. Galleguillos et al. (2008) learn these relationships exclusively from annotated data, whereas Rabinovich et al. (2007) also mine them using an image search engine as external knowledge source.

Desai et al. (2009) cast multi-class object detection into a structured prediction framework. Their approach predicts a structured labeling of the whole image simultaneously. It gathers statistics regarding the spatial relationships between objects in real images and uses them to suppress or favor spatial arrangements at test time. Alternatively, Choi et al. (2010) propose a tree graphical model to encode spatial and co-occurrence dependencies between different object classes. By also incorporating global image features (Gist), their model predicts contextually coherent object locations and remove out-of-context false-positive detections. Finally, Felzenszwalb et al. (2010b)'s DPM detector rescores detections by adding image context into a quadratic SVM. They model image context as a vector with as many dimensions as object classes. Each coordinate represents the presence of a class by using the normalized score of the highest detection of that class.

Stuff classes, which are amorphous regions that are usually considered background such as road or sky, may also provide valuable context for recognition. For example, Heitz and Koller (2008) exploit the context of this type of classes by incorporating them in a probabilistic object detection model that considers the relative spatial location of objects with respect to stuff. They learn stuff classes from data in an unsupervised way, by clustering visually similar image background regions that provide a solid context for object classes.

2.3.3 Object surroundings

Image regions that are not semantically meaningful may also provide valuable contextual information for object detection. Concretely, the immediate surroundings of a particular object class may be significantly informative, as they stay rather stable across different instances. For example, cows generally appear surrounded by green color from the fields, whereas airplanes appear surrounded by blue from the sky. Moreover, when observing only a very tight window around an object instance, the object boundary becomes less explicit. Boundaries are informative recognition cues and window classifiers may benefit from including them.

Several detectors expand each window input into their window classifiers to include the region immediately surrounding the object. Dalal and Triggs (2005) includes a margin of 16 pixels around every window and shows how adding this context gives a significant boost to the object detection performance. Motivated by a human study showing how we perform better at recognizing objects when given image regions that are bigger than just the objects, Li et al. (2011) propose learning adaptive contextual regions around the objects. These regions are class-specific and provide a valuable contextual cue that help object detection. The Selective Search proposal generator of van de Sande and Gevers (2010) generates approximate locations by using rough segmentations, instead of generating precise locations that tightly delineate the objects. Therefore, Selective Search proposals include some object surroundings as local context, and thus all the works that use them (Uijlings et al., 2013; Girshick et al., 2014; Zhang et al., 2014a; Girshick, 2015; Gkioxari et al., 2015) indirectly benefit from this type of context.

Crandall and Huttenlocher (2007) add context from some scene regions by expanding a part-based model to a hierarchical two-layer model. The finer level represents the object as usual. The coarser level represents the scene, and its parts are automatically learned image patches around the object. Similarly, Mottaghi et al. (2014) also extend a deformable part model to include local context around the object. Concretely, it adds contextual 'parts' at the top, bottom, right, and left of the object's root filter. Their model, which also includes global image context, increases object detection performance, especially on objects with extreme sizes (e.g. tiny or very large).

2.4 Discussion

Context in object detection has been predominantly applied with the goal of improving detection performance. Another promising use of context more rarely exploited is as guidance for visual search strategies in images. Visual searches could benefit from information provided by image context to focus their attention on promising areas while ignoring areas unlikely to contain objects. In this spirit, we use image context to guide the search strategy of chapter 3.

CNNs are being constantly used for object detection, among other tasks. However, they are very unintuitive and the details of their internal representations are largely unknown. For this reason, more research is needed to provide a better understanding of their inner workings. Chapter 4 attemps to clarify an often assumed property of CNNs regarding the emergence of semantic parts in their learned representations.

Semantic parts are relevant visual components used for multitude of computer vision tasks, but the detection of parts as a task on its own has received little attention. We tackle this interesting and challenging task in chapter 5, where we exploit object context to provide more accurate part detections.

Chapter 3

Active search strategies for efficient object class detection

3.1 Introduction

Most object detectors first partition the input image into a set of windows and then evaluate them independently using a window classifier. The window classifier scores each window to determine whether it contains an object of a particular class or not. Finally, the detector outputs the windows with the locally highest scores. Classical approaches partition the image using sliding-window (Dalal and Triggs, 2005; Harzallah et al., 2009; Felzenszwalb et al., 2010b; Malisiewicz et al., 2011), whereas more recent object detectors (Uijlings et al., 2013; Wang et al., 2013; Girshick et al., 2014; He et al., 2014; Zhang et al., 2014a; Szegedy et al., 2015; Ouyang et al., 2015; Girshick, 2015) use object proposals (Van de Sande et al., 2011; Alexe et al., 2010). In both cases, the window classifier evaluates all windows in the set, effectively assuming that they are independent.

In this work, we propose an *active search strategy* that sequentially chooses the next window to evaluate based on previously observed windows, rather than going through the whole window set in an arbitrary order. Our search method extracts information given by the observed windows and integrates it into the search, effectively guiding future observations to windows likely to contain the object. Thereby, our method explores the window space in an intelligent fashion, where future observations depend on all the information gathered so far. This results in a more natural and elegant way of searching, avoiding wasteful computation in uninteresting areas. As a consequence, our method is able to find the objects while evaluating much fewer windows,

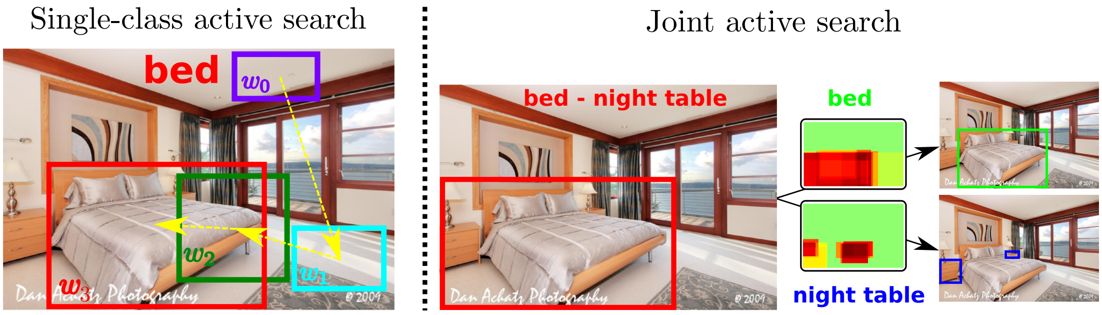


Figure 3.1: *Intuition behind the proposed active search strategies.*

typically only a few hundred (sec. 3.6.3).

Fig. 3.1 (left) shows the intuition of our method on detecting beds. We use two guiding forces: context and window classifier score. Context exploits the statistical relation between the appearance of a window and its location relative to the objects, as observed in the training set. For example, the method can learn that beds tend to be on the floor, below ceilings. Observing a window in the ceiling (w_0) in a test image suggests that there might be a bed below, whereas a window on the floor (w_1) suggests making a horizontal move instead. This contextual information may point to any area of the image. On the other hand, the classifier score of a window gives information about the score of other nearby windows, due to the smoothness of the classifier function. It guides the search to areas where we observed windows with high scores, while pushing away from those with low scores. For example, a window partially containing a bed (w_2) will attract the search to its surroundings. Our active search effectively combines these two forces to explore the image in an efficient way.

In a multi-class setting, we can further reduce the number of evaluated windows by active search. Many objects in the real world co-occur and often appear close in the image. For example, beds tend to appear next to night tables (fig. 3.1). Because of this, the detection of a particular class may assist in the detection of other classes, providing helpful cues about their location and scale. For this reason, we also propose an active search strategy that searches for two object classes jointly.

Fig. 3.1 (right) gives an overview of our joint search strategy, on the pair of classes bed and night table. We first search for both classes together as a single unit, inspired by the *visual phrase* (Sadeghi and Farhadi, 2011) work. In our case, a visual phrase is a pair of classes that co-occur frequently and present a consistent relative position and scale across images. After finding the visual phrase, the search branches off into two individual searches, one for each object. These individual searches can exploit the information provided by the visual phrase, which is an advantage over single-class active search. The visual phrase is a very informative cue about the location and scale of the objects inside it, thus providing an excellent initialization for the individual searches. Moreover, the context guiding force is typically more effective within a visual phrase, as this has a much more specific appearance and object layouts than the entire image. This results in searches able to detect the individual objects in very few iterations.

Experiments on the challenging SUN2012 dataset (Xiao et al., 2010) and PASCAL VOC10 (Everingham et al., 2010) demonstrate that our single-class active search explores the image in an intelligent way, effectively detecting objects in only a few hundred iterations. As window classifiers we use the popular R-CNN (Girshick et al., 2014) and the UvA Bag-of-Words model of Uijlings et al. (2013). For R-CNN on SUN2012, our method matches the detection accuracy of evaluating all proposals independently, while evaluating 9× fewer proposals. For the UvA window classifier, our search strategy only needs 35 proposals per image to match the performance of evaluating all of them (a reduction of 85×). By letting the search run for longer, we even *improve* accuracy while evaluating 30× fewer proposals, as it avoids evaluating some cluttered image areas that lead to false-positives. Depending on the used classifier, our active search strategy results in an actual runtime speedup (sec. 3.6.5). Additionally, we demonstrate our joint active search on SUN2012 and Microsoft COCO (Lin et al., 2014). It is especially effective for challenging classes containing small objects, such as desk lamp and tennis racket, where it needs to evaluate even fewer windows than our single-class active search.

A preliminary version of this work appeared at CVPR 2015 (Gonzalez-Garcia et al., 2015), covering only the single-class case. In this chapter, we also introduce joint active search over pairs of classes and present a more extensive experimental evaluation.

The rest of the chapter is organized as follows. Section 3.2 describes related work to our approach. We detail our method for single-class active search in section 3.3, and for joint active search in section 3.4. Section 3.5 presents all the implementation details for both approaches. In sections 3.6 and 3.7, we describe the performed experiments and discuss their results for single-class and joint active search, respectively. Finally, we conclude the chapter and propose future work in section 3.8.

3.2 Related work

Modern object detectors. Chapter 2 introduced the notion of object proposals and how modern object detectors (Cinbis et al., 2013; Uijlings et al., 2013; Wang et al., 2013; Girshick et al., 2014; He et al., 2014; Zhang et al., 2014a; Szegedy et al., 2015; Ouyang et al., 2015; Girshick, 2015) have reduced the number of window evaluations by using them, compared to the classical sliding-window paradigm (Dalal and Triggs, 2005; Harzallah et al., 2009; Felzenszwalb et al., 2010b; Malisiewicz et al., 2011). The work presented in this chapter brings even further reductions, as we only evaluate a subset of the object proposals (typically just a few hundred).

After the publication of our active search method (Gonzalez-Garcia et al., 2015), most recent object detectors have incorporated efficiency improvements that render this work less relevant computationally. Fast R-CNN (Girshick, 2015) processes the network's convolutional layers only once for the whole image, and thus most of the computation is shared among all windows. Therefore, the total cost of evaluating a big number of windows is similar to the cost of evaluating few windows. The leading object detection approaches propose even more efficient architectures, either by integrating the object proposal step into the CNN as in Faster R-CNN (Ren et al., 2015), or by directly predicting the object locations without the need of object proposals, like SSD (Liu et al., 2016) or YOLO (Redmon et al., 2016; Redmon and Farhadi, 2017). Besides the decrease in runtime, another benefit of such models is the ability to be trained end-to-end for detection. In the case of Faster R-CNN, this makes the generated windows very effective for the detection task and thus they need to evaluate much fewer windows compared to approaches that use externally generated object proposals, normally in the range of only a few hundred. SSD and YOLO achieve even smaller runtimes by avoiding the proposal generation step altogether. Huang et al. (2017) present an extensive survey of the state-of-the-art object detectors and their trade-offs between efficiency and detection performance.

Although our active search does not bring a computational benefit when applied to very recent detectors, it is still an interesting research area on its own. In the future, it is possible that the next most accurate window classifier is also computationally very expensive. If such classifier cannot easily recycle computation across windows, our active search would then still be applicable, as it is independent of the window classifier.

Improving sliding-window. Some works reduce the number of window classifier evaluations. Rowley et al. (1996) and Viola and Jones (2001) consider coarser window grids that minimize the drop in performance. Lampert et al. (2008) use a branch-and-bound scheme to efficiently find the maximum of the classifier over all windows. However, it is limited to classifiers for which tight bounds on the highest score in a subset of windows can be derived. Lehmann et al. (2011) and Sun and Batra (2015) extend Lampert et al. (2008) to some more classifiers and to detect multiple object instances in the same image, respectively. Several works perform coarse-to-fine detection for symbol recognition (Amit and Geman, 1999), face detection (Fleuret and Geman, 2001), and cat detection (Fleuret and Geman, 2008). They apply a first 'visual search' step that reduces the number of windows to be evaluated by the window classifier.

An alternative approach is to reduce the cost of evaluating the classifier on a window. For example, both Harzallah et al. (2009) and Vedaldi et al. (2009) first run a linear classifier over all windows and then evaluate a complex non-linear kernel only on a few highly scored windows. Schneiderman (2004) shares feature evaluations across different windows, which effectively reduces the window evaluation time. Several techniques are specific to certain types of window classifiers and achieve a speedup by exploiting their internal structure (e.g. DPM (Felzenszwalb et al., 2010a; Pedersoli et al., 2011; Zhu et al., 2014), CNN-based (He et al., 2014; Girshick, 2015), additive scoring functions (Wu and Zhu, 2013), cascaded boosting on Haar features (Viola and Jones, 2001; Saberian and Vasconcelos, 2014)). Our work instead can be used with any window classifier as it treats it as a black-box.

Sequential search. The 'active testing' framework was first proposed by Geman and Jedynak (1996) for the task of road tracking. It is a sequential search strategy that gets updated as it gathers information about the image. Starting with an initial set of location hypotheses for the road, the search sequentially applies tests to these hypotheses and changes their probabilities based on the test results. Every step of the search reduces the uncertainty, until it reaches a chosen threshold that makes the search finish. Sznitman and Jedynak (2010) extend this framework for face detection by adding a hierarchical model and pruning heuristics. More recently, active testing has been applied to point-could registration (Pinheiro et al., 2013) and graph matching (Serradell et al., 2015).

A few works develop techniques that make sequential fixations inspired by human perception for tracking in video (Bazzani et al., 2011), image classification (Denil et al., 2012; Larochelle and Hinton, 2010; Mnih et al., 2014) and face detection (Butko and Movellan, 2009; Tang et al., 2014). However, they only use the score of a (foveated) window classifier, not exploiting the valuable information given by context. Moreover, they experiment on simple datasets, far less challenging than SUN2012 (Xiao et al., 2010) (MNIST digits, faces).

Another line of visual search works relies on visual saliency of objects in images. Itti et al. (1998) presented a visual attention model that selects the attended locations based on image saliency maps. Walther and Koch (2006) extended this model by including an estimation of the object extent at each attented location. Sun and Fisher (2003) developed an attention framework that effectively integrates space-based attention with object-based attention. Sun et al. (2008) extended this framework by adding biologically inspired eye-movement information. Similar approaches have been followed for grasping in humanoid robots (Orabona et al., 2005) and face detection (Lee et al., 2005).

Since the publication of our paper on single-class active search (Gonzalez-Garcia et al., 2015), some works have applied reinforcement learning techniques to this task (Caicedo and Lazebnik, 2015; Mathe et al., 2016; Jie et al., 2016). Caicedo and Lazebnik (2015) localize objects using a Markov Decision Process that starts from the whole image and iteratively modifies the current window to better enclose the object (e.g. make it smaller). Mathe et al. (2016) balance exploration and exploitation by accumulating evidence from different image regions. Although these two approaches do reduce the number of windows observed, they damage the performance of the detector applied (e.g. R-CNN (Girshick et al., 2014)). In contrast, our method achieves the same performance as evaluating all possible windows. Finally, Jie et al. (2016) sequentially propose windows to be observed by exploiting the internal architecture of a CNN, similarly to the Region Proposal Network in Fast R-CNN (Ren et al., 2015). Additionally, Caicedo and Lazebnik (2015); Jie et al. (2016) do not use context to guide the search.

Context. Many works use context as an additional cue on top of object detectors, complementing the information provided by the window classifier, but without altering the search process (sec. 2.3).

The most related work to ours is Alexe et al. (2012), which proposes a search strategy driven by context. Here we go beyond in several ways: (1) We use context much more efficiently (Random Forest vs nearest-neighbors). We add only a very small overhead, resulting in an actual wall-clock speedup for some classifiers (sec. 3.6.5). (2) While Alexe et al. (2012) use only context, we guide the search also by the classifier score, and learn an optimal combination of the two forces (sec. 3.3.1). (3) Alexe et al. (2012) is only single-class, whereas we perform multi-class detection with joint search strategies and introduce two new types of context for this task. (4) While Alexe et al. (2012) perform experiments only on PASCAL VOC10, we also use SUN2012 (Xiao et al., 2010) and MS-COCO (Lin et al., 2014), which have more cluttered images with smaller objects.

Visual Phrases. Sadeghi and Farhadi (2011) introduce *visual phrases* as composites such as “person riding a bicycle”. They train visual phrase detectors from manually annotated visual phrases in PASCAL (Everingham et al., 2010), and show that they outperform individual class detectors, even when combined with spatial reasoning. Some works (Desai and Ramanan, 2012; Kong et al., 2014) extend visual phrases or combine them with other methods for articulated pose estimation (Desai and Ramanan, 2012), action classification (Desai and Ramanan, 2012) and human interaction recognition (Kong et al., 2014). Finally, Li et al. (2012) and Lan et al. (2013) automatically detect semantically meaningful visual composites, used later for object detection. To our knowledge, ours is the first work to use visual phrases to guide visual search.

3.3 Single-class active search

3.3.1 Search model

Let I be a test image represented by a set of object proposals (Van de Sande et al., 2011), $I = \{o_i\}_{i=1}^{N}$. The goal of our method is to efficiently detect objects in I, by evaluating the window classifier on only a subset of the proposals. Our method is a class-specific sequential procedure that evaluates one window at a time and then decides where to look next. At every iteration t, it selects the next window o^{t+1} according to all the observations $\{o^k\}_{k=1}^{t}$ performed so far. Throughout this paper, o_i indexes through the input set of proposals I, whereas o^t is the proposal actively chosen by our strategy in the t-th iteration. We assign a belief value $b^t(o_i, \{o^k\}_{k=1}^{t}; \Theta)$ to each object proposal o_i and update it after every iteration. This belief indicates how likely it is that o_i contains the object, given all previously observed windows $\{o^k\}_{k=1}^{t}$. Here $\Theta = \{\lambda, \sigma_S, \sigma_C\}$ are hyperparameters and t indexes the iteration.

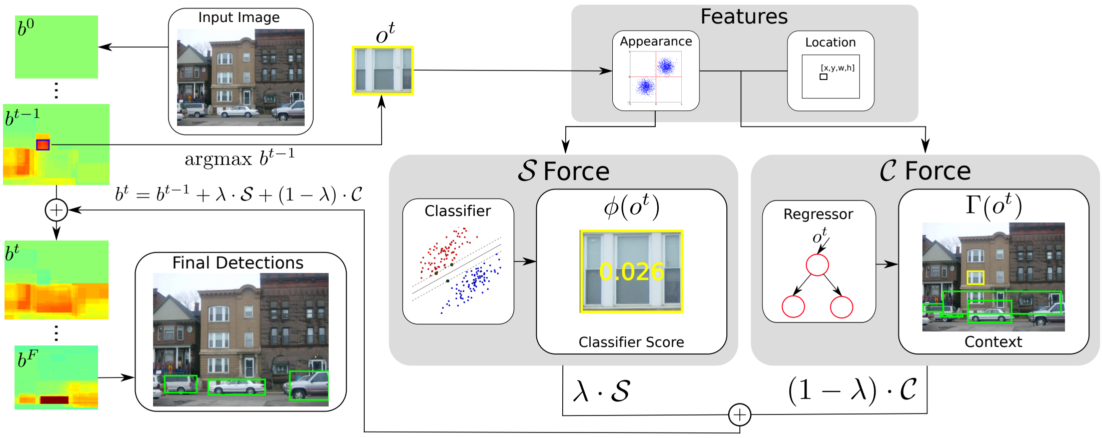


Figure 3.2: *Search model. The next observed window o^t is the maximum of the current belief map b^{t-1}. The method extracts appearance and location features for o^t, and uses them to compute its context $\mathcal{C}$ and window classifier $\mathcal{S}$ outputs. Then, it combines these outputs with the current belief map b^{t-1} into the next iteration's belief map b^t. The final belief map b^F combines all the performed observations. The output detections are the observed windows with highest scores.*

The method starts with the belief map $b^0(o_i) = 0 \ \forall o_i$, representing complete uncertainty. At iteration t, the method selects the object proposal with the highest belief

$$o^t = \underset{o_i \in I \setminus \{o^k\}_{k=1}^{t-1}}{\operatorname{argmax}} \ b^{t-1}(o_i, \{o^k\}_{k=1}^{t-1}; \Theta). \tag{3.1}$$

We avoid repetition by imposing $o^t \neq o^k, \forall k < t$. The starting window o^1 is the average of all the ground-truth bounding-boxes in the training set.

At each iteration t, the method obtains information from the new observation o^t and it updates the belief values of all windows as follows

$$\begin{aligned} b^t(o_i, \{o^k\}_{k=1}^{t}; \Theta) = b^{t-1}(o_i, \{o^k\}_{k=1}^{t-1}; \Theta) \\ + \lambda \cdot \mathcal{S}(o_i, o^t; \sigma_{\mathcal{S}}) + (1-\lambda) \cdot \mathcal{C}(o_i, o^t; \sigma_{\mathcal{C}}). \end{aligned} \tag{3.2}$$

The observation o^t, i.e. the window evaluated by the strategy at iteration t, provides two kind of information: the context $\mathcal{C}$ and the classifier score $\mathcal{S}$ (explained below). These are linearly combined with a mixing parameter $\lambda \in [0, 1]$. Fig. 3.2 illustrates our pipeline, and we summarize it in algorithm 1.

Context force $\mathcal{C}$. It points to areas of the image likely to contain the object, relative to the observation o^t. It uses the statistical relation between the appearance and location of training windows and their position relative to the objects. The context force may point to any area of the image, even those distant from o^t. For the car detection example

```
Input : Test image $I$, object proposals $\{o_i\}_{i=1}^{N}$, window classifier $\phi$,
        parameters $\Theta$
Output: Set of object detections $D$
// Set starting window
$o^1 \leftarrow$ average ground-truth;
for $t \leftarrow 2$ to $F$ do
    // Select next window
    $o^t \leftarrow \text{argmax}_{o_i \in I \setminus \{o^k\}_{k=1}^{t-1}} b^{t-1}(o_i, \{o^k\}_{k=1}^{t-1}; \Theta)$ ;
    // Evaluate window classifier
    $\phi(o^t)$;
    for $o_i \in I \setminus \{o^k\}_{k=1}^{t}$ do
        // Evaluate score and context forces
        $\mathcal{S}(o_i, o^t; \sigma_{\mathcal{S}})$, $\mathcal{C}(o_i, o^t; \sigma_{\mathcal{C}})$ ;
        // Update belief value
        $b^t(o_i) = b^{t-1}(o_i) + \lambda \cdot \mathcal{S} + (1 - \lambda) \cdot \mathcal{C}$;
    end
end
// Remove duplicate detections with NMS
$D \leftarrow \text{NMS}(\{(o^k, \phi(o^k))\}_{k=1}^{F})$;
```

Algorithm 1: Active search strategy.

depicted in fig. 3.2, if o^t contains a patch of building, $\mathcal{C}$ will point to windows far below it, as cars are below buildings (fig. 3.6). If o^t contains a patch of road, $\mathcal{C}$ will propose instead windows next to o^t, as cars tend to be on roads (fig. 3.6).

The heart of the force $\mathcal{C}$ is a *context extractor* Γ. Given the appearance and location of o^t, Γ returns a set of windows $\Gamma(o^t) = \{w_j\}_{j=1}^{J}$, which are not necessarily object proposals. These windows cover locations likely to contain objects of the class, as learned from a set of training windows and their relative position to the objects in their own images. We explain how the context extractor works in sec. 3.3.2.

We can now define $\mathcal{C}$ as

$$\mathcal{C}(o_i, o^t; \sigma_{\mathcal{C}}) = \sum_{w_j \in \Gamma(o^t)} K(w_j, o_i; \sigma_{\mathcal{C}}). \tag{3.3}$$

It gives high values to object proposals close to windows in $\Gamma(o^t)$, as we expect these windows to be near objects. The influence of the windows in $\Gamma(o^t)$ is weighted by a smoothing kernel

$$K(w, o; \sigma) = e^{-(1-\text{IoU}(w,o))^2/(2\sigma^2)}. \tag{3.4}$$

This choice of kernel assumes smoothness in the presence of an object for nearby windows. Indeed, adjacent windows to a window containing an object will also contain part of the object. The further apart the windows are, the lower the probability of containing the object is. We use the inverse overlap 1 - intersection-over-union (Everingham et al., 2010) (IoU) as distance between two windows.

Classifier score force $\mathcal{S}$. Force $\mathcal{S}$ attracts the search to the area surrounding the observation o^t if it has high classifier score, while pushing away from it if it has low score:

$$\mathcal{S}(o_i, o^t; \sigma_{\mathcal{S}}) = K(o_i, o^t; \sigma_{\mathcal{S}}) \cdot (\phi(o^t) - 0.5), \tag{3.5}$$

where $\phi(o^t) \in [0, 1]$ is the window classifier score of o^t (sec. 3.5.1 details our choice of window classifier). We translate $\phi(o^t)$ into the $[-0.5, 0.5]$ range and weight it using the smoothing kernel (3.4). Therefore, $\mathcal{S}$ operates in the surroundings of o^t, spreading the classifier score to windows near o^t. When $\mathcal{S}$ is positive, it attracts the search to the region surrounding o^t. For example, if o^t contains part of a car, it will probably have a high classifier score (fig. 3.3). Then $\mathcal{S}$ will guide the search to stay in this area, as some nearby window is likely to contain the whole car. On the other hand, when $\mathcal{S}$ values are negative, it has a repulsive effect. It pushes the search away from uninteresting regions with low classifier score, such as background (sky, buildings, etc).

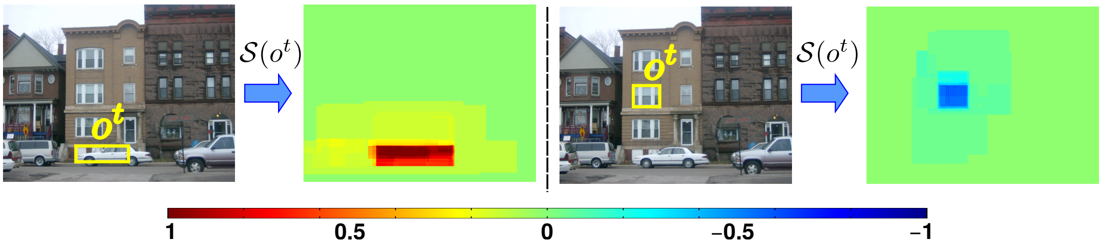


Figure 3.3: *Classifier score force $\mathcal{S}$. The input image and observation o^t are displayed on the left side of the arrow. The right side shows the belief map produced by $\mathcal{S}$. Colors correspond to belief values.*

3.3.2 Context extractor

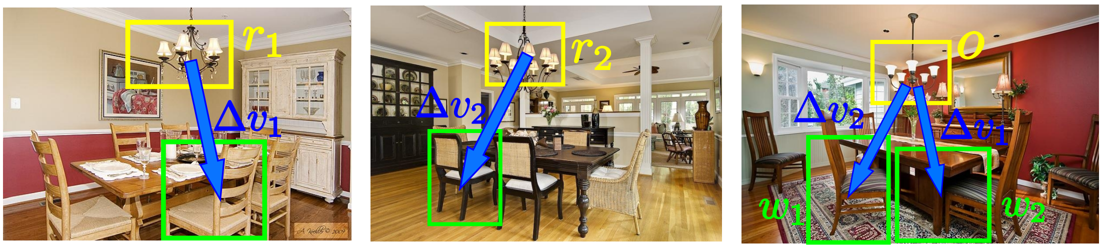
Given an input observation o in the test image, the context extractor Γ returns a set of windows $\Gamma(o) = \{w_j\}_{j=1}^{J}$ covering locations likely to contain objects.

The context extractor is trained from the same data as the window classifier, i.e. images with annotated object bounding-boxes. Hence our approach requires the same annotation as standard object detectors (Cinbis et al., 2013; Dalal and Triggs, 2005; Felzenszwalb et al., 2010b; Girshick, 2015; Uijlings et al., 2013). The training set consists of pairs $\{(r_n, \Delta v_n)\}_{n=1}^{N}$; r_n is a proposal from a training image, and Δv_n is a 4D displacement vector, which transforms r_n into the closest object bounding-box in its training image (fig. 3.4a-b). Here index n runs over all object proposals in all the training images. For 500 images and 3200 proposals per image, $N = 500 \cdot 3200 = 1'600'000$.

Given the observation o, the context extractor regresses a displacement vector Δv pointing to the object. For robustness, the context extractor actually outputs a set of displacement vectors $\{\Delta v_j\}_{j=1}^{J}$, to allow for some uncertainty regarding the object's location (fig. 3.4c). Then it applies $\{\Delta v_j\}_{j=1}^{J}$ to o, obtaining a set of displaced windows on the test image: $\Gamma(o) = \{o + \Delta v_j\}_{j=1}^{J}$. The windows in $\Gamma(o)$ indicate expected locations for the object in the test image. Note that they may be any window, not necessarily object proposals. We refer to this context extractor as *Any to Class* (A2C), as it points from any image window towards the target object class.

Random Forests. We use Random Forests (RF) (Breiman, 2001) as our context extractor. A RF is an ensemble of J binary decision trees. In our case, each tree inputs the window o and outputs a displacement vector Δv_j. The final output of the RF are all displacement vectors $\{\Delta v_j\}_{j=1}^{J}$ produced by each tree.

RF have been successfully applied in several learning problems such as classi-



(a) (b) (c)

Figure 3.4: *Context extractor. (a,b) The displacement vectors Δv_1 and Δv_2 of training samples r_1 and r_2 point to their respective ground-truth objects. (c) By applying Δv_1 and Δv_2 to test observation o, we obtain displaced windows w_1 and w_2, covering likely locations for the object with respect to o.*

fication, regression or density estimation (Criminisi et al., 2011). RF in computer vision (Gall and Lempitsky, 2009; Montillo and Ling, 2009; Criminisi et al., 2011; Fanelli et al., 2011; Girshick et al., 2011) typically use axis-aligned separators (thresholding on one feature dimension) as tests in the internal nodes. However, we found that tests on distances to training samples perform better in our case (sec. 3.6.4), as they are more informative. Hence, we build our RF based on distance tests. This is related to Proximity Forests (O'Hara and Draper, 2013), although O'Hara and Draper (2013) use RF for clustering, not geometric regression. Also, note the difference with Hough Forests (Gall and Lempitsky, 2009), which use RF within the object class model: the leaves form an implicit codebook optimized for Hough-style object detection. Instead we use RF to model context in order to guide the search.

When the test window o goes down a tree, it traverses it from the root to a leaf guided by tests in the internal nodes. At each node p the decision whether o goes left or right is taken by comparing the distance $d_p(o, r_p)$ between o and a pivot training point r_p to the threshold τ_p. The test window o proceeds to a left child if $d_p(o, r_p) \geq \tau_p$ or to the right child otherwise (fig. 3.5). This process is repeated until a leaf is reached. Each leaf stores a displacement vector, which the tree returns. The triplet (r_p, τ_p, d_p) at each internal node is chosen during training (the process can choose between two different distance functions, sec. 3.5.3). More concretely, r_p is a window represented either by location or appearance features, τ_p is a real numerical threshold, and d_p is a distance function between the location or the appearance of two windows.

Random Forest training. We train one RF with $J = 10$ trees for each class. To keep the trees diverse, we train each one on windows coming from a random sub-

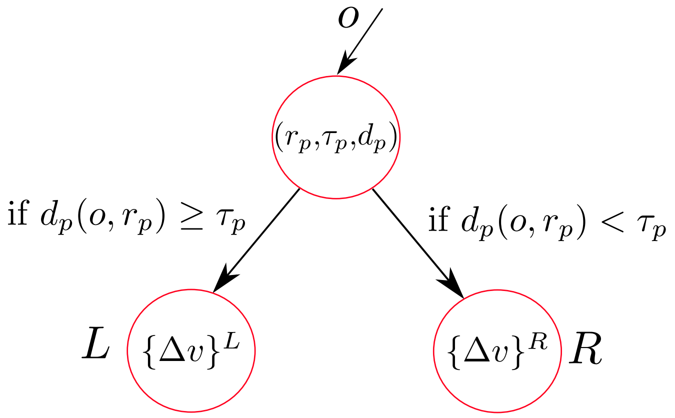


Figure 3.5: *Internal node at test time. The test compares distance $d_p(o,r_p)$ between test window o and training sample r_p with the threshold τ_p.*

sample of 40 training images. As shown in Criminisi et al. (2011), this procedure improves generalization. We construct each tree by recursively splitting the training set at each node. We want to learn tests in the internal nodes such that leaves contain samples with a compact set of displacement vectors. This way a tree learns to group windows using features that are predictive of their relative location to the object. To create an internal node p, we need to select a triplet (r_p,τ_p,d_p), which defines our test function. Following the extremely randomized forest framework (Moosman et al., 2006) we generate a random set of possible triplets. We then pick the triplet that achieves maximum information gain:

$$IG = H(S) - \frac{|S^L|}{|S|} \cdot H(S^L) - \frac{|S^R|}{|S|} \cdot H(S^R), \tag{3.6}$$

where S is the set of training samples at the node and L,R denote the left and right children with samples S^L and S^R, respectively. H is the approximated Shannon entropy of the 4D displacement vectors in S: we first compute a separate histogram per dimension, and then sum their individual entropies (Hall and Morton, 1993). Concretely, we use MATLAB's `histogram` function, which uses automatic binning with uniform width, on each of the four dimensions of the displacement vectors independently. Then, we compute the aproximated entropy as

$$H(S) = \frac{1}{N} \sum_i n_i \log\left(\frac{n_i}{Nh}\right), \tag{3.7}$$

where h is the bin width, N the total number of bins, and n_i the number of samples in the i-th bin, all returned by the `histogram` function. All samples trained to use each tree belong to only one object class. Finally, we keep in each leaf the mediod displacement

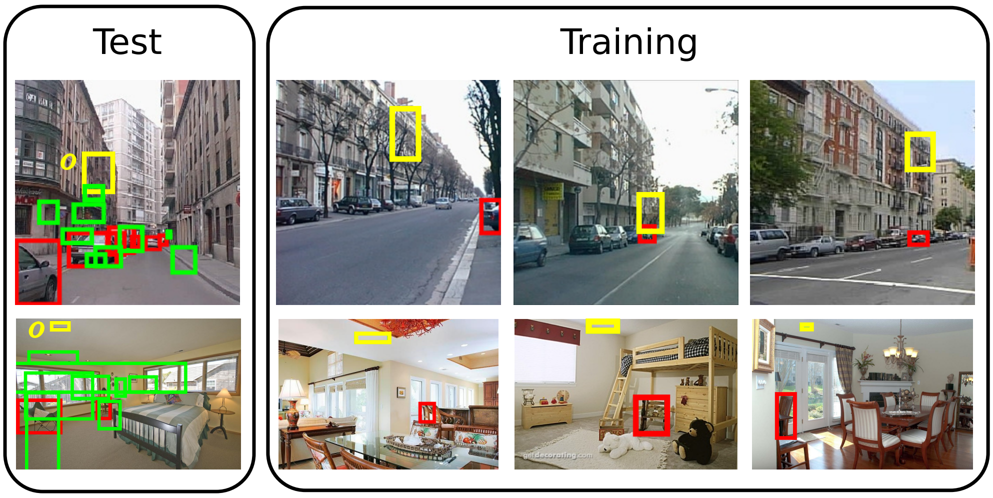


Figure 3.6: *RF examples. (Test) Example image and observation o for classes car (top) and chair (bottom). Windows displaced by the displacement vectors regressed by the RF are in green, whereas ground-truth objects are in red. (Training) Training samples (yellow) in leaves reached by o when input into our RF and associated ground-truth objects (red).*

vector of the training samples that fell into it. In practice, each tree has an average depth of 18, and thus contains around 260,000 terminal nodes. Since there are 10 trees per RF, we would need to store more than 2M displacement vectors. However, many of the displacement vectors are identical across different nodes, which substantially reduces the storage footprint.

Fig. 3.6 shows example test windows passed through the RF, along with example training windows in leaves they reach. Note how these training windows are similar in appearance to the test window, and produce displaced windows covering areas likely to contain objects of that class in the test image. This demonstrates that RF is capable of learning the relation between the appearance of a window and its position relative to the object.

Efficiency. A key benefit of RF over a simple nearest-neighbor regressor (Alexe et al., 2012) is computational efficiency. Since the observation o is only compared to at most as many pivot points as the depth of the tree, the runtime of RF grows only logarithmically with the size of the training set. In contrast, the runtime of nearest-neighbor grows linearly, since it compares o to all training samples. This makes a substantial difference to runtime in practice (sec. 3.6.5).

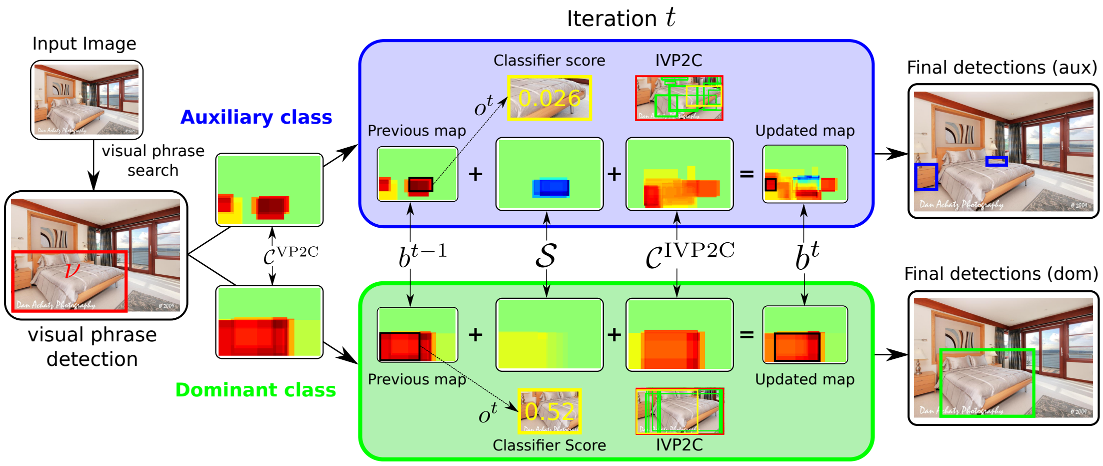


Figure 3.7: *Joint active search. We first search for the visual phrase* ν *in the image. After we find it, we branch off into two individual searches, one for the auxiliary class and one for the dominant, initialized by our VP2C context map,* $\mathcal{C}^{VP2C}$*. At iteration* t*, we evaluate score force* $\mathcal{S}$ *using the maximum window* o^t *of the previous iteration's belief map,* b^{t-1}*. We also compute the IVP2C context force,* $\mathcal{C}^{IVP2C}$*, and combine it with* $\mathcal{S}$ *to update the belief map* b^t *for the next iteration.*

3.4 Joint active search

The multi-class version of our active search performs a joint search over a pair of classes. The initial part of the search is shared by both objects (fig. 3.1). Afterwards, it branches off into individual searches for each object. We group the two objects into a *visual phrase* and we anchor our joint search around this concept. In this section, we assume we have the grouping and we explain our joint active search for one visual phrase. In section 3.7.1 we explain how we perform the grouping.

Visual phrases. In our case, each visual phrase is composed of two types of classes: dominant and auxiliary. The auxiliary is typically the smaller of the two (e.g. night table in 'bed - night table') and it is found near the dominant class. Each instance of a dominant class may be associated with one or more instances of the auxiliary class, depending on the specific visual phrase. For example, a 'night table' can only take one 'desk lamp', whereas a 'bed' can take several 'night table' instances (fig. 3.13). We manually set the number of auxiliary instances that can be associated to a dominant class depending on the visual phrase (sec. 3.7.1).

3.4.1 Search model

Figure 3.7 presents our joint active search pipeline. During the first part of the search, we obtain a visual phrase detection ν using single-class active search. This generally requires a small number of iterations, since visual phrases are easier to find than individual object classes. Then, we branch off into parallel individual searches, one for the auxiliary and one for the dominant.

The detected visual phrase provides valuable information about the objects inside it. First, it bounds their possible locations to the image region covered by the visual phrase itself. Additionally, it provides an estimation of where to expect the contained objects within it, and how big they are. For example, the bed in the 'bed - night table' example occupies the majority of the visual phrase, and is located near its center. Night tables instead are smaller and often stand near the side ends of the visual phrase.

Moreover, we can obtain even more specific information by taking into account the appearance of the detected visual phrase in the current image. A visual phrase class can exhibit different object layouts in different images. For example, night tables tend to be on the sides of beds, but depending on the viewpoint of the image, some night tables may appear towards the center of the visual phrase (fig. 3.13a). We can exploit this by relying on the different object layouts seen in the training set.

Based on the above observations, we present here a new type of context called *Visual Phrase to Class* (VP2C). This context regresses from the whole visual phrase to the objects inside. We use it to initialize the individual searches after finding the visual phrase. For simplicity, here we explain the process for just one of the two object classes (the other one is analogous). Let ν be a visual phrase detection and $\Gamma(\nu)$ a set of windows generated applying displacement vectors to ν. We create a belief map $\mathcal{C}^{\text{VP2C}}$ for the class by using displaced windows $\Gamma(\nu)$ as in (3.3), but using parameters specific to the VP2C type of context, $\sigma_{\mathcal{C}^{\text{VP2C}}}$. We initialize the search for the object class within the visual phrase detection by using the belief map $\mathcal{C}^{\text{VP2C}}$, weighted by a class-specific hyperparameter α. This parameter represents how reliable $\mathcal{C}^{\text{VP2C}}$ is for that class. The initial belief map for the individual search is then

$$b^0(o_i, \nu; \Theta) = \alpha \cdot \mathcal{C}^{\text{VP2C}}(o_i, \nu; \sigma_{\mathcal{C}^{\text{VP2C}}}). \tag{3.8}$$

After branching, the individual search explores mostly the area inside the visual phrase detection. We can exploit this further by tailoring the search to the context of the visual phrase. For example, observing a drawer inside a 'bed - night table' detection brings more information about the location of the bed and the night table than observing a drawer elsewhere in the image. In the visual phrase case, the drawer most likely belongs to a night table, whereas in the general case it may be part of another piece of furniture, such as a wardrobe.

We use this extra knowledge in a new type of context: *Inside Visual Phrase to Class* (IVP2C). This is finer, more precise than the general context of sec. 3.3.2. Given a window inside a visual phrase, the IVP2C context extractor regresses to the position of the individual objects inside it. The use of this context also results in a more local exploration, focusing on the area of the visual phrase, instead of jumping to any area of the image as the A2C context would do.

The remaining part of the individual search resembles single-class active search but using IVP2C context instead of A2C context. At every iteration t we evaluate the unvisited window with highest belief value using (3.1) and update the belief map as follows

$$\begin{aligned} b^t(o_i, \{o^k\}_{k=0}^{t}; \Theta) = b^{t-1}&(o_i, \{o^k\}_{k=0}^{t-1}; \Theta) \\ &+ (\alpha - 1) \cdot [\lambda \cdot \mathcal{S}(o_i, o^t; \sigma_{\mathcal{S}}) \\ &+ (1-\lambda) \cdot \mathcal{C}^{\text{IVP2C}}(o_i, o^t; \sigma_{\mathcal{C}^{\text{IVP2C}}})], \end{aligned} \tag{3.9}$$

where $o^0 = \nu$ and $\mathcal{C}^{\text{IVP2C}}$ follows (3.3), but using the IVP2C context extractor. The set of hyperparameters is now $\Theta = \{\lambda, \sigma_{\mathcal{S}}, \sigma_{\mathcal{C}}, \alpha, \sigma_{\mathcal{C}^{\text{VP2C}}}, \sigma_{\mathcal{C}^{\text{IVP2C}}}\}$.

Generally, the dominant class takes most of the visual phrase area. In practice, we noticed that many windows visited during the visual phrase search can be recycled for the dominant class. Therefore, we simply re-score these windows for the dominant class and avoid visiting them again during its individual search. In many cases, this suffices to find a good detection of the dominant class, although some challenging cases may require a few additional individual search iterations (sec. 3.5.4). Algorithm 2 describes our joint active search strategy.

Multiple visual phrases. Some images may contain multiple instances of the same visual phrase (fig. 3.13b). We deal with this by postponing the branching so the first part of the search finds most instances. We run τ^{VP} iterations of the visual phrase search, which results in τ^{VP} evaluated windows. We then take all those scoring higher than a threshold ϕ^{VP} (sec. 3.5.4) as visual phrase detections. This adds two hyperparameters to Θ. After that, the individual search iterations cycle over the different visual phrase detections, keeping independent belief maps for each. Hence, the search explores the most likely windows within each visual phrase early on, while going into more detail in later iterations. Overall, this strategy leads to a lower total number of window evalu-

```
Input  : Test image $I$, object proposals $\{o_i\}_{i=1}^N$, window classifier $\phi$,
         parameters $\Theta$
Output: Set of object detections for the auxiliary $D_a$ and dominant $D_d$
         classes
// Run active search to find visual phrase
$\upsilon \leftarrow$ active_search(vp) ;
for $c \in \{a,d\}$ do
    // Initialize belief map with VP2C context
    $b_c^0 \leftarrow \mathcal{C}^{\mathrm{VP2C}}(\upsilon)$ ;
    // Run active search with ICVP2C context
    $D_c \leftarrow$ active_search($c$, $\mathcal{C}^{\mathrm{IVP2C}}$);
end
```

Algorithm 2: Joint active search strategy.

ations, compared to running the search for individual objects to completion within one visual phrase detection and then moving on to the next one.

Fallbacks. There are two threatening situations for our joint active search. First, we might not detect any visual phrase in the image. In this case, we automatically fall back to running two separate single-class searches (but reusing the windows visited for the failed visual phrase search during the dominant class search). Second, there might be additional object instances located *outside* of the visual phrase. After the branching, our individual searches quickly find the objects inside the visual phrase detections. Therefore, we perform a small number of iterations τ^{IVP2C} using our $\mathcal{C}^{\mathrm{IVP2C}}$, and then we fall back to single-class context to keep exploring the rest of the image. We add τ^{IVP2C} to the final set of hyperparameters Θ.

3.4.2 Context extractors

The context extractor models used for the joint active search are identical to the single-class version (sec. 3.3.2), but they are trained from different samples.

VP2C. We train two VP2C context extractors per visual phrase, one for each of the composing object classes. Similarly to the general A2C context (sec. 3.3.2), we train each VP2C extractor using a set of pairs $\{(r_n, \Delta v_n)\}_{n=1}^N$. However, r_n is now a ground-truth bounding-box of the corresponding visual phrase, and Δv_n is a displacement vec-

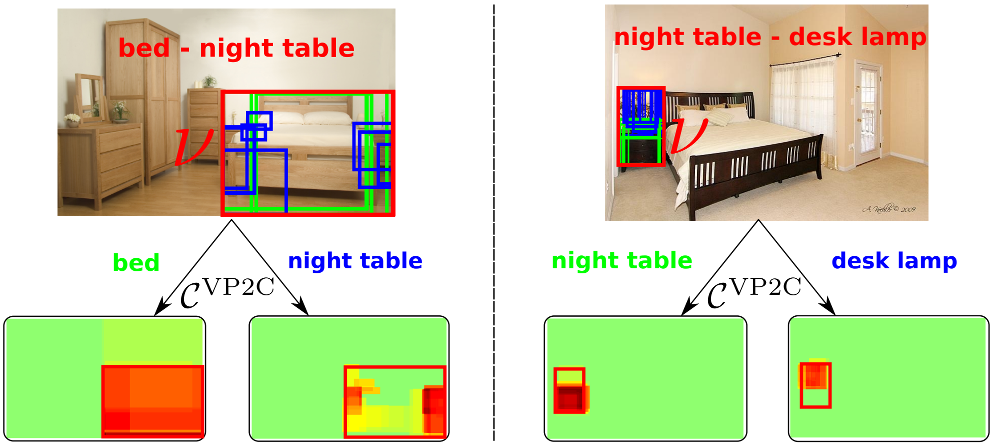
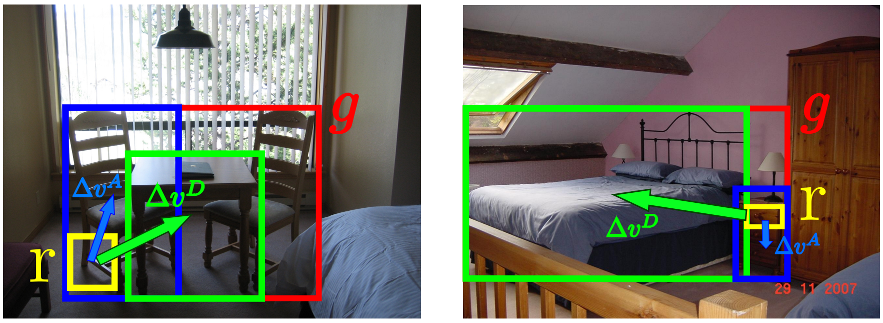


Figure 3.8: *VP2C context. (Top) Displaced windows $\Gamma^{VP2C}(\nu)$ given visual phrase detection ν, for dominant (green) and auxiliary (blue) classes. (Bottom) Belief maps $\mathcal{C}^{VP2C}$ generated by $\Gamma^{VP2C}(\nu)$. For visualization purposes, we overlay the visual phrase detection ν on the heatmaps (red box).*

tor pointing to one of the instances of the individual objects of the class inside r_n. In this case, the total number of samples N is a small value, proportional to the number of visual phrase ground-truths that contain the corresponding class. Figure 3.8 (top) shows some examples of displaced windows by VP2C context.

IVP2C. Similarly, we train two IVP2C context extractors per visual phrase. In this case, we need to restrict the windows r_n in our training pairs $(r_n, \Delta v_n)$ to be inside visual phrases. For each visual phrase ground-truth bounding-box in a training image, we take all the object proposals inside it. We normalize the location features of these proposals with respect to the visual phrase (sec. 3.5.3). Displacement vectors Δv_n point to the closest bounding-box of the composing object classes inside the visual phrase. Fig. 3.9 depicts some training samples, including the visual phrase ground-truth g and displacement vectors for both the dominant (Δv^D) and the auxiliary (Δv^A) object classes.



(a) (b)

Figure 3.9: *A few samples used to train* Γ^{IVP2C} *context extractor, for (a) chair-table and (b) bed-night table. The bounding-box in red is the visual phrase g, whereas the dominant and auxiliary classes correspond to the green and blue boxes, respectively.*

3.5 Implementation details

3.5.1 Window classifiers

We use R-CNN (Girshick et al., 2014) as main window classifier. It is based on the CNN model of Krizhevsky et al. (2012), which achieved winning results on the ILSVRC-2012 image classification competition (Russakovsky et al., 2015). The CNN is then fine-tuned into a window classifier on ground-truth bounding-boxes. Finally, a linear SVM is trained on normalized 4096-D features, obtained from the 7th layer of the CNN. We use the R-CNN implementation provided by (Girshick et al., 2014). For the single-class active search experiments we use the AlexNet architecture, as this network was state-of-the-art at the time of publishing our CVPR paper (Gonzalez-Garcia et al., 2015). For the joint active search we use the VGG16 (Simonyan and Zisserman, 2015) instead, as this superior architecture was available when we developed our joint active search.

In order to demonstrate that our method is general as it supports any window classifier, we test our single-class active search also on *UvA* (Uijlings et al., 2013). This Bag-of-Words technique was among the best detectors before CNNs. A window is described by a 3x3 spatial pyramid of bag-of-words. The codebook has 4096 words and is created using Random Forest on PCA-reduced dense RGB-SIFT (Van De Sande et al., 2010) descriptors. Overall, the window descriptor has 36864 dimensions. The window classifier is an SVM with a histogram intersection kernel on these features. We use the implementation provided to us kindly by the authors (Uijlings et al., 2013).

For both R-CNN and UvA, we fit a sigmoid to the outputs of the SVM to make the classifier score ϕ lie in $[0,1]$.

3.5.2 Object proposals

We use the fast mode of Selective Search (Van de Sande et al., 2011). We keep at most 5000 proposals per image, resulting in 3200 on average. These form the set of windows o_i available to our method (both to the context extractor and window classifier). Note how both the R-CNN and UvA detectors as originally proposed (Girshick et al., 2014; Uijlings et al., 2013) also evaluate their window classifiers on these proposals.

3.5.3 Features and distances for context extractors

We represent a window by two feature types: location and appearance. For the single-class case, we normalize window location features $[x/W, y/H, w/W, h/H]$ by image width W and height H. Here x, y, w, h are the top-left coordinates, width and height of the window. Both VP2C and IVP2C use location features normalized to the visual phrase. Let x_ν and y_ν be the center of visual phrase ν, and w_ν, h_ν its width and height. Then, the location features for a window are $[(x-x_\nu)/w_\nu, (y-y_\nu)/h_\nu, w/w_\nu, h/h_\nu]$. The distance function for both location features is the inverse overlap $1-\text{IoU}$.

The appearance features used by all context extractors match those in the window classifier. We embed the 4096-dimensional R-CNN appearance features in a Hamming space with 512 bits using Gong and Lazebnik (2011). This reduces the memory footprint by 256× (from 131072 to just 512 bits per window). It also speeds up distance computations, as the Hamming distance between these binary strings is 170× faster than L2-distance in the original space. We do the same for the Bag-of-Words features of UvA. The window classifiers work on the original features (sec. 3.5.1).

3.5.4 Hyperparameter training

Single-class active search. For each object class, we find optimal hyperparameters σ_S, σ_C, and λ by maximizing object detection performance. We use k-fold cross-validation on the training set, where $k=10$. We equally split the training set in k folds and train the network and the context extractor on $k-1$ folds. We use the last fold as validation set, performing grid search in ranges $\sigma_S, \sigma_C \in [0.01, 1]$ and $\lambda \in [0,1]$. Performance is quantified by the area under the Average Precision (AP) curve, which reports AP as a function of the number of proposals evaluated by the search (fig. 3.10).

Interestingly, the learned σ values correspond to intuition. For σ_S we obtain small values, as the classifier score informs only about the immediate neighborhood of the observed window. Values for σ_C are larger, as the context force informs about a broader region of the image. Furthermore, C produces arbitrary windows, hence the distance to proposals is generally larger.

Joint active search. We also use cross-validation and grid search to train the hyperparameter $\sigma_{C^{\text{VP2C}}}$. Unlike the other types of context, VP2C is not applied iteratively. Instead, it is used only once as initialization for the belief map of the individual searches after branching. For this reason, we optimize $\sigma_{C^{\text{VP2C}}}$ using C^{VP2C} once as the only force. On the other hand, we set $\sigma_{C^{\text{IVP2C}}} = \sigma_C$ given their similar nature.

Our focus is the auxiliary classes, as they are more challenging than the dominant classes. However, our method should still perform at least as well as single-class active search for the dominant classes. Since our joint search reuses the windows evaluated during the visual phrase search for the dominant class (sec. 3.4.1), we set τ^{VP} to be around the number of iterations that a single-class active search for the dominant class needs to match the performance of evaluating all windows.

We train hyperparameters ϕ^{VP} and α by 2D grid search, maximizing the performance of object detection for the auxiliary class, considering only instances inside visual phrases. Finally, we perform a similar grid search to train τ^{IVP2C}, but in this case considering all instances of the auxiliary class.

3.6 Experiments for single-class active search

3.6.1 Datasets

We evaluate our single-class active search on two datasets: SUN2012 (Xiao et al., 2010) and PASCAL VOC10 (Everingham et al., 2010).

SUN2012. We use all available images for the 5 most frequent object classes in the highly challenging SUN2012 dataset (Xiao et al., 2010): Chair, Person, Car, Door and Table. This amounts to 2920 training images and 6794 test images, using the official train/test split provided with the dataset (Xiao et al., 2012). Each image is annotated with bounding-boxes for these 5 classes. This dataset contains large cluttered scenes with small objects, as it was originally made for scene recognition (Xiao et al., 2010). This makes it very challenging for object detection, and also well suited to show the benefits of using context in the search strategy.

PASCAL VOC10. We use all 20 classes of PASCAL VOC10 (Everingham et al., 2010). While also challenging, on average this dataset has larger and more centered objects than SUN2012. We use the official splits, train on the `train` set (4998 images) and test on the `val` set (5105 images).

3.6.2 Protocol

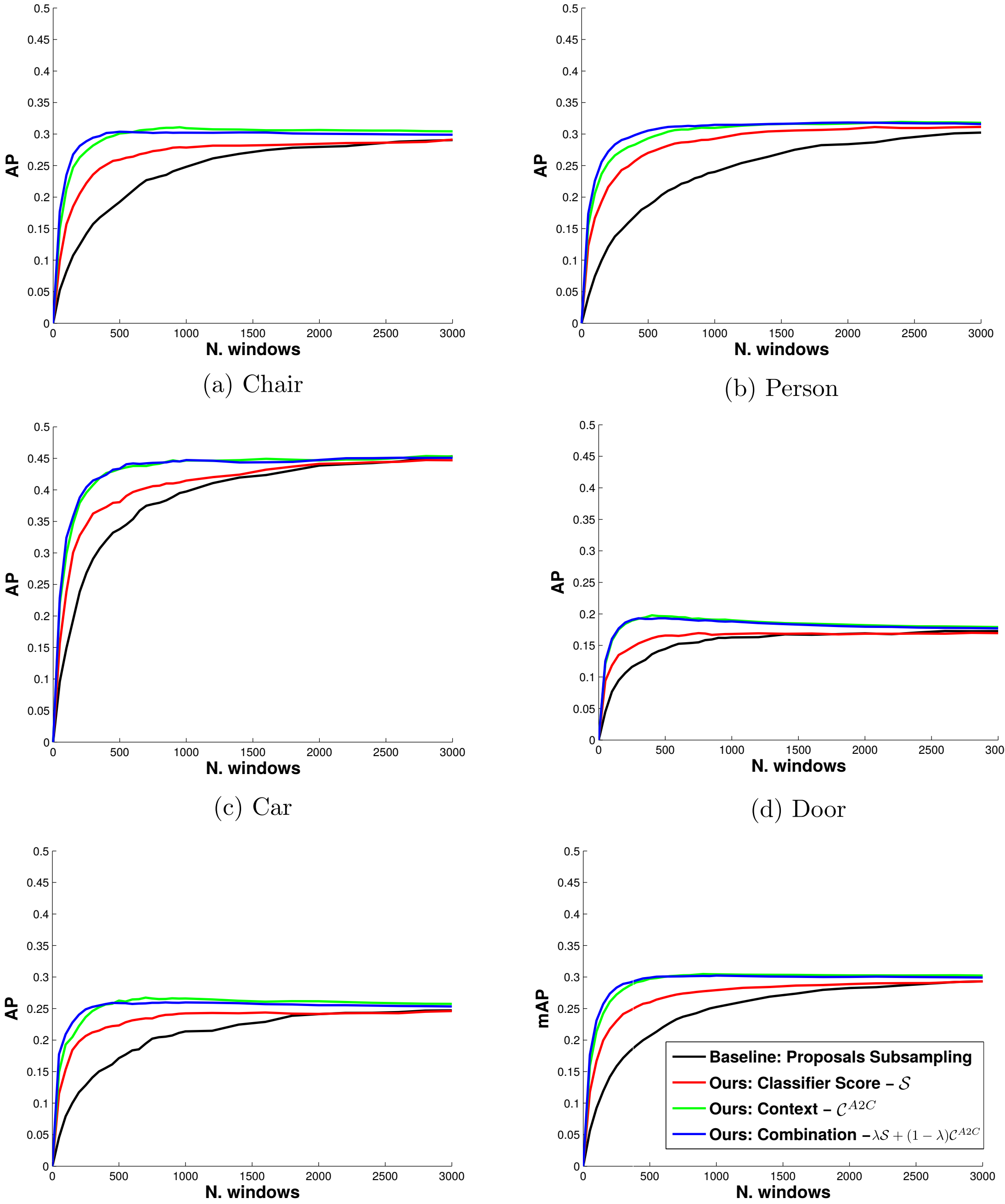
We train the window classifier (including fine-tuning for R-CNN), the Random Forest regressor and the hyperparameters Θ on the training set. We measure performance on the test set by Average Precision (AP), following the PASCAL protocol (Everingham et al., 2010), i.e. a detection is correct if it overlaps a ground-truth object > 0.5. Previous to the AP computation, we use Non-Maxima Suppression (Felzenszwalb et al., 2010b) to remove duplicate detections.

3.6.3 Results

R-CNN on SUN2012. Fig. 3.10 presents results for our full system ('Combination') and when using each force $\mathcal{S}$ or $\mathcal{C}$ alone. The figure shows the evolution of AP as a function of the number of proposals evaluated. As a baseline, we compare to evaluating proposals in a random sequence ('Proposals Subsampling'). This represents a naive way of reducing the number of evaluations. The rightmost point on the curves represent the performance of evaluating all proposals, i.e. the original R-CNN method. Note, however, how we only show up to 3000 window evaluations in fig. 3.10. The actual end of the curves is the maximum number of proposals considered (5000 windows), where all curves achieve the same performance.

Our full method clearly outperforms Proposals Subsampling, by selecting a better sequence of proposals to evaluate. On average over all classes, by evaluating about 350 proposals we match the performance of evaluating all proposals (fig. 3.10f). Therefore, our search strategy stops after evaluating the first 350 proposals. This corresponds to 9× fewer window classifier evaluations. Some instances might be missed due to the lower number of window evaluations. Should this be the case, such detections do not negatively affect the performance, as it is mantained. In general, 350 evaluations seem to suffice even when the image contains multiple instances (fig. 3.12).

In general, we achieve our best results by combining both forces $\mathcal{S}$ and $\mathcal{C}$. When



(a) Chair

(b) Person

(c) Car

(d) Door

(e) Table

(f)

Figure 3.10: *Results on SUN2012 for the baseline Proposals Subsampling and our method on SUN2012, using each force $\mathcal{S}$, $\mathcal{C}$ alone and in combination. The x-axis shows the number of evaluated windows. The y-axis in (a-e) shows the AP of each class, while in (f) it shows the mean AP over all classes (mAP).*

using one force alone, $\mathcal{C}$ performs better, in some cases even reaching the accuracy of our combined strategy. Nevertheless, force $\mathcal{S}$ achieves surprisingly good results by itself, providing a rather simple method to speed-up object detectors while maintaining high accuracy.

Fig. 3.12 shows our search strategy in action. After just a few iterations the belief maps are already highlighting areas containing the objects. Uninteresting areas such as the sky or ceiling are barely ever visited, hence we waste little computation. Our method detects multiple object instances and viewpoints, even in challenging images with small objects. In these examples, it finds the first instance in fewer than 50 iterations, showing its efficiency in guiding the search. After finding the first instance, it continues exploring other areas of the image looking for more instances.

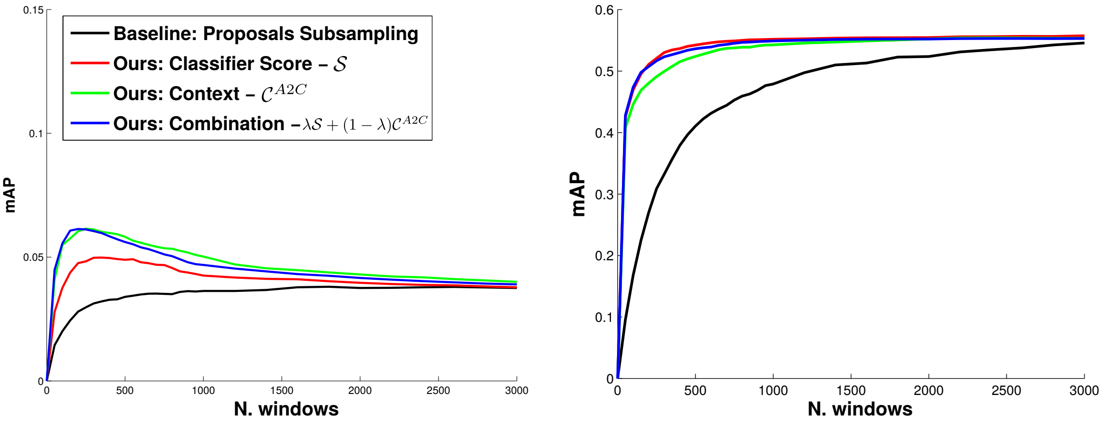
UvA on SUN2012. We re-trained all the elements of our method on the Bag-of-Words features of Uijlings et al. (2013): window classifier, RF regressor, and hyperparameters. Our method matches the performance of evaluating all proposals with 35 windows on average, a reduction of 85× (fig. 3.11a). Interestingly, the curve for our method reaches an even *higher* point when evaluating just 100 windows (+2% mAP). This is due to avoiding some cluttered areas where the window classifier would produce false-positives. This effect is less noticeable for the R-CNN window classifier, as UvA is more prone to false-positives.

R-CNN on PASCAL VOC10. As fig. 3.11b shows, our method clearly outperforms the Proposal Subsampling baseline again. In this case, active search has a very rapid growth in the first 100 windows, showing how in general PASCAL VOC10 is easier than SUN2010. This demonstrates the general applicability of our method to a variety of image types and that it still useful for less challenging datasets.

3.6.4 Context extractor

We compare here our A2C context extractor implemented using RF with one based on nearest-neighbor search, as in Alexe et al. (2012). We run our method only using the context force $\mathcal{C}$, but substituting RF with nearest-neighbors, on the same training set and input features. The results show that both ways of extracting context lead to the same performance. The AP at 500 windows, averaged over all classes, differs by only 0.006. Importantly, however, RF is 60× faster (sec. 3.6.5).

We also show that the distance tests of our RF approach are crucial for its good



(a) (b)

Figure 3.11: *(a) Results on SUN2012 using the UvA detector (Uijlings et al., 2013). (b) Results on PASCAL VOC10 using R-CNN.*

performance. For comparison, we implemented the A2C context using tests more traditionally used in RF for computer vision, i.e. axis-aligned separators (Gall and Lempitsky, 2009; Criminisi et al., 2011; Fanelli et al., 2011). An active search using only $\mathcal{C}$ with this RF barely outperforms the Proposal Subsampling baseline, thus being clearly inferior to our distance-based counterpart.

3.6.5 Runtime

We measure runtimes on an Intel Xeon E5-1620v2 CPU and a GeForce GTX 770 GPU (used by the R-CNN detector, not by UvA). We do not include the object proposal extraction time (about 4 seconds) in any reported timing, as it is performed once per image and it is common to all the methods compared.

R-CNN. The most expensive component is computing CNN features, which takes 4.5 ms per window on the GPU. Evaluating the 3200 proposals in an average image takes 14.4 seconds. The total overhead added by our method is 2.6 ms per iteration. Therefore, processing one image while maintaining the same AP performance (350 iterations on SUN2012) takes $350 \cdot (4.5 + 2.6)$ ms $= 2.5$ seconds, i.e. 6× faster than evaluating all proposals [1].

The small overhead added by our method is mainly due to the RF query performed at each iteration for the context force, which amounts to 1.9 ms including the Hamming

[1] Extracting CNN descriptors on a GPU is more efficient in batches than one at a time, and is done in R-CNN (Girshick et al., 2014) by batching many proposals in a single image. In our sequential search we can form batches by processing one window each from many different images.

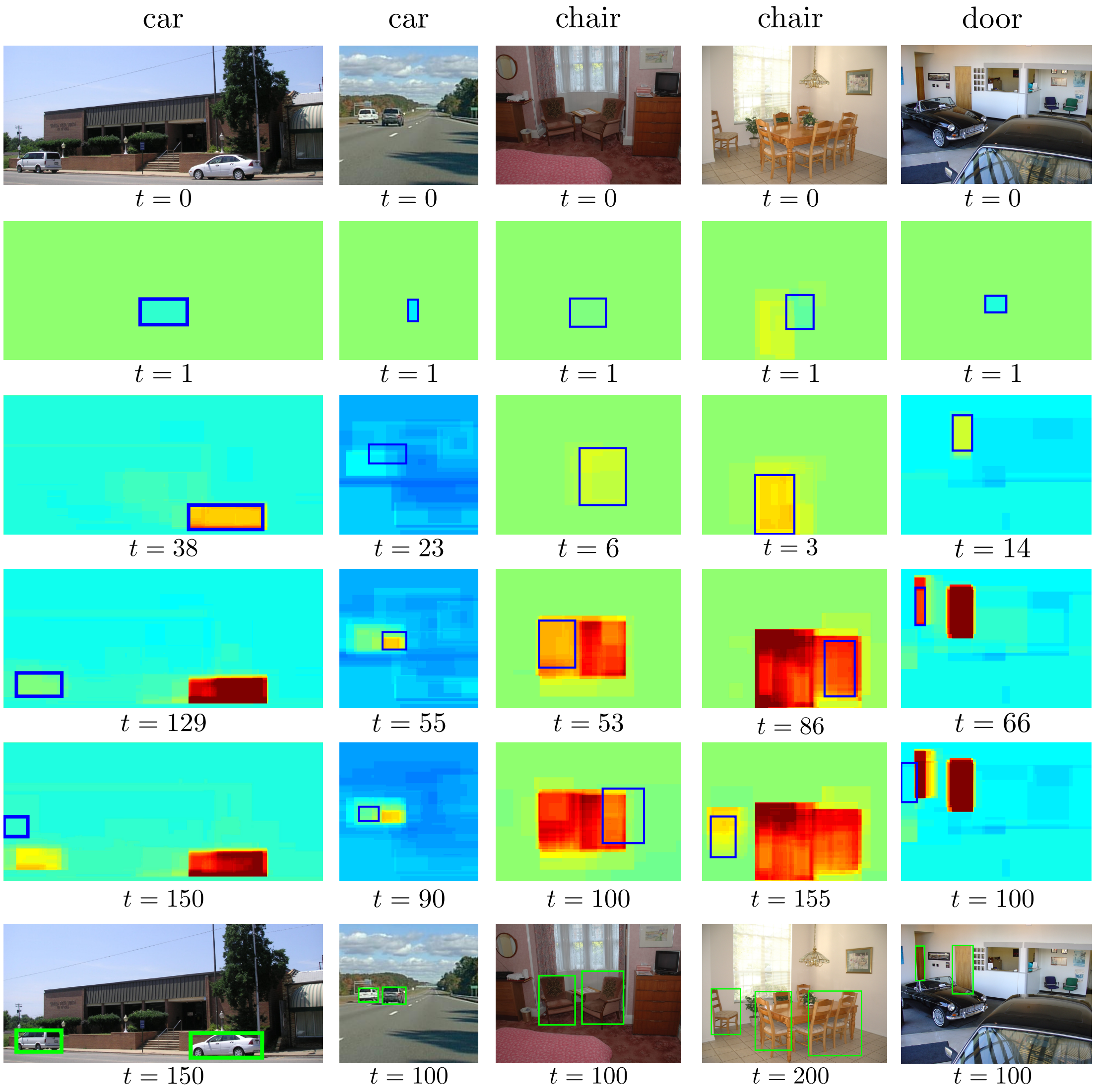
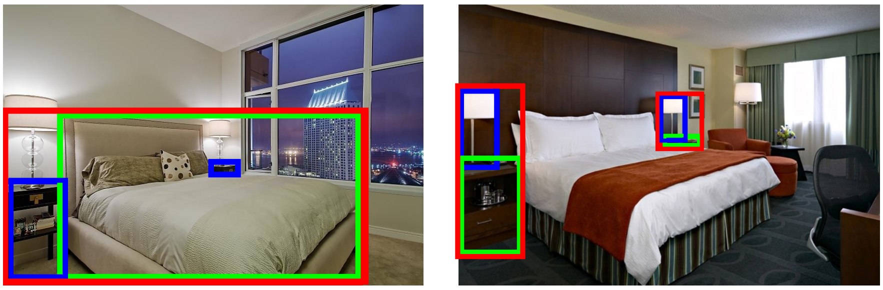

car car chair chair door
$t = 0$ $t = 0$ $t = 0$ $t = 0$ $t = 0$
$t = 1$ $t = 1$ $t = 1$ $t = 1$ $t = 1$
$t = 38$ $t = 23$ $t = 6$ $t = 3$ $t = 14$
$t = 129$ $t = 55$ $t = 53$ $t = 86$ $t = 66$
$t = 150$ $t = 90$ $t = 100$ $t = 155$ $t = 100$
$t = 150$ $t = 100$ $t = 100$ $t = 200$ $t = 100$

Figure 3.12: *Qualitative results for the single-class active search on SUN2012. (Top) Original image. (Middle) Belief maps for some iterations. The blue window indicates the observation at that iteration. (Bottom) The top scored detections.*



(a) bed - night table (b) night table - desk lamp

Figure 3.13: *Examples of visual phrases for SUN2012, automatically grouped. The final visual phrase (red) bounds both the dominant class (green), and the auxiliary ones (blue).*

embedding of the appearance features. In comparison, a context extractor implemented using nearest-neighbors as in Alexe et al. (2012) takes 57 ms per iteration, which would lead to no actual total runtime gain over evaluating all proposals.

As the runtime of the window classifier grows, the speedup made by of our method becomes more important. Extracting CNN features on the CPU takes 100 ms per window (Jia, 2013). In this regime, the overhead added by our method becomes negligible, only 3% of running the window classifier. Evaluating all 3200 proposals in an image would require 320 seconds, in contrast to just 36 seconds for evaluating 350 proposals with our method, a 9× speed-up.

UvA. This classifier has two types of processing. The first is done only once per image and thus shared among all evaluated windows. It consists of feature extraction and visual word assignment (Uijlings et al., 2013), and it takes 5 seconds on average. The second type evaluates each window by counting visual word frequencies and scoring them with an SVM. Evaluating all 3200 proposals in an image takes 6.7 seconds. Our method needs to evaluate each of the 35 windows independently, which amounts to 43 ms per window. The final runtime including our 2.6 ms overhead per iteration is $35 \cdot (44+2.6)$ ms = 1.6 seconds. Therefore, our method is about 2× faster than evaluating all proposals for this classifier, when taking into account the feature extraction and visual word assignment processing stage. It is 4× faster when detecting many object classes, as the cost of that stage is then amortized out.

3.7 Experiments for joint active search

3.7.1 Datasets

We perform experiments on SUN2012 (Xiao et al., 2010) and Microsoft COCO (Lin et al., 2014). Since these datasets do not contain annotations for visual phrases, we automatically group object instances into visual phrases, based on the overlap and distance between instances.

First, we manually select a set of object class pairs that appear often in the dataset and have consistent layouts across images. Then, we automatically process all images containing both classes and add the corresponding visual phrase annotations, as a box bounding all individual objects that belong to the visual phrase. Fig. 3.13 shows examples for SUN2012.

For fair evaluation, at test time we consider all objects in images that contain at least one visual phrase, even if they are not part of a visual phrase themselves.

SUN2012. We pick the 5 most frequently co-occurring pairs of object classes: 'bed - night table', 'bed - pillow', 'night table - desk lamp', 'table - chair', and 'keyboard - mouse'. By convention, we always express visual phrases as 'dominant - auxiliary'. Using the official train/test split, these 5 visual phrases add up to 725 training images and 1697 test images.

Microsoft COCO. MS COCO is a large-scale dataset designed to contain many different objects co-occurring in the same images. It also contains many non-iconic views of objects, which pose a challenge for independently run single-class detectors, but could be alleviated using context from other classes. We select 6 visual phrases among the most frequent: 'motorcycle - person', 'person - backpack', 'person - tennis racket', 'table - chair', 'bowl - spoon', and 'screen - keyboard'. The training set for this selection contains 12107 images, and the validation set 5863 images.

3.7.2 Protocol

We follow exactly the protocol described in section 3.6.2.

3.7.3 Context extractors

We evaluate here our VP2C and IVP2C context extractors for their specific use cases, comparing their performance to the general A2C one. To directly measure the perfor-

Context	Method	SUN2012	COCO
A2C	RF - App + Loc	0.21	0.17
VP2C	NN - Appearance	0.40	0.29
VP2C	NN - Location	0.38	0.26
VP2C	NN - App + Loc	0.39	0.27
VP2C	**RF - App + Loc**	**0.41**	**0.31**

Table 3.1: *Mean Maximum Overlap (MMO) for VP2C and A2C, using queries on visual phrases. Both context extractors use Random Forest (RF) including appearance (App) and location (Loc) features.*

mance of context extractors we use Mean Maximum Overlap (MMO). This measure is similar to Average Best Overlap, defined in Uijlings et al. (2013) to evaluate object proposals. Let $\{g_j\} \in G$ be the image ground-truth annotations for the class, and $\{w_i\} \in \Gamma(o)$ the displaced windows of context extractor Γ for observation o. MMO takes the maximum overlap for each displaced window w_i to any g_j and then averages them over $\Gamma(o)$

$$\mathrm{MMO} = \frac{1}{|\Gamma(o)|} \sum_{w_i \in \Gamma(o)} \max_{g_j \in G} \mathrm{IoU}(g_j, w_i). \tag{3.10}$$

VP2C. VPC2 uses only ground-truth bounding-boxes as training samples (as opposed to A2C, which uses all object proposals in an image). This leads to a much smaller number of samples, enabling using a simple nearest-neighbor regressor without incurring to a prohibitively expensive overhead. For this reason, we evaluate here both our RF context extractor and a nearest-neighbor version, using different combinations of feature types (location and appearance, sec. 3.5.3).

We query A2C and various versions of VP2C using windows overlapping visual phrase ground-truths (IoU > 0.7). Table 3.1 shows results for SUN2012 and MS COCO. The VP2C context extractor clearly outperforms the general A2C context in every case, almost doubling its MMO. This shows that, for the more specific use-case VP2C is trained for, it works a lot better. Moreover, the RF version achieves a slightly higher MMO than the nearest-neighbor alternative, confirming again the effectiveness of our distance RF approach.

IVP2C. Table 3.2 presents results for IVP2C compared to A2C. The query windows for this experiment come from *inside* visual phrase ground-truth bounding-boxes. In this case, we only test a RF version, because nearest-neighbors is again very expensive

Context	Method	SUN2012	COCO
A2C	RF - App + Loc	0.11	0.09
IVP2C	**RF - App + Loc**	**0.26**	**0.24**

Table 3.2: *Mean Maximum Overlap (MMO) for IVP2C and A2C. Using queries inside visual phrases. Both context extractors use Random Forest (RF) including appearance (App) and location (Loc) features.*

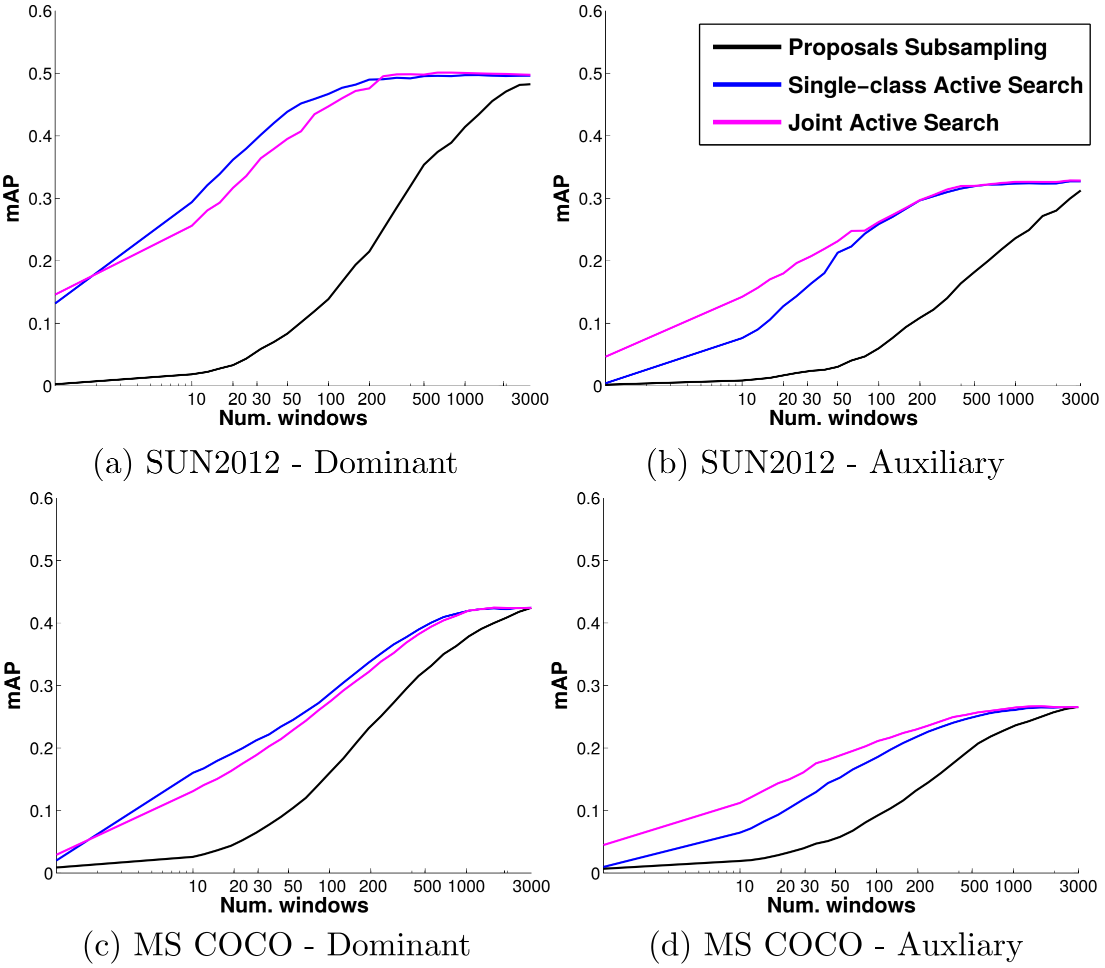
given the size of the training set. Results clearly surpass the A2C context extractor. This shows how we can benefit from the more specific IVP2C context extractor when the search takes place inside a visual phrase.

3.7.4 Results

We evaluate the effectiveness of our joint active search using the same protocol as the single-class version. Therefore, we use equivalent performance curves to those in fig. 3.10, following the method described in sec. 3.6.3. In this case, however, we use a logarithmic scale for the horizontal axis to fully appreciate the differences between methods in the very low number of windows regime. We compare to two baselines: (a) ‘Proposals Subsampling’, evaluating proposals in a random sequence, and (b) applying two times our single-class active search separately.

Figure 3.14 shows the results for R-CNN with VGG-16 on SUN2012 and MS COCO, averaged over both types of classes, dominant and auxiliary. As explained in sec. 3.4.1, we recycle windows visited during the visual phrase search for the dominant class by simply re-scoring them. This re-scoring is generally very efficient (e.g. evaluation of a linear SVM in R-CNN (Girshick et al., 2014)), as the expensive step is the feature extraction. Therefore, we include these iterations in the plots for the dominant classes (fig. 3.14a,c).We can see how our joint active search is slightly behind single-class active search for the dominant classes. This is to be expected as the early iterations are specialized to the visual phrase. However, the point at which it matches the mAP (up to 0.1%) of evaluating all windows is virtually the same.

Interestingly, our joint active search improves over the single-class version for the difficult classes, i.e. the auxiliary ones. As the iterations of the visual phrase are already accounted for in the plots for the dominant classes, here we count only the iterations for the auxiliary class search after the branching. On SUN2012, our method is consistently better than both baselines in the very low number of windows regime



(a) SUN2012 - Dominant

(b) SUN2012 - Auxiliary

(c) MS COCO - Dominant

(d) MS COCO - Auxliary

Figure 3.14: *Quantitative results for SUN2012 (a-b) and MS COCO (c-d), averaged over all dominant classes (a,c) and auxiliary classes (b,d).*

(<100). For example, at only 10 window evaluations, we double the performance of single-class active search. After the first 100 windows, our method performs closer to single-class active search, although still above by a small amount. This results in needing less than half the number of window evaluations necessary to match mAP, from 840 for the single-class search, down to 390 for the joint search. In conclusion, we match the performance of evaluating all windows with 8× fewer windows, twice as fast as single-class search.

For MS COCO, we can see an improvement over single-class active search also for greater number of windows, until about 500. For example, at 100 window evaluations, we reach 80% of the mAP of evaluating all windows, compared to 68% by single-class active search. After 500 windows, results are fairly similar to single-class active search, but matching the mAP of evaluating all windows 200 windows earlier (from 760 to 560).

Figure 3.15 presents some qualitative examples of our joint active search, focusing on auxiliary classes. When there are multiple visual phrase detections, we show their maps combined. The VP2C belief map ($t = 0$) already suggests excellent locations to start the search for the auxiliary class. Indeed, the first auxiliary object is generally found after very few iterations. Then, the IVP2C context force takes over and guides the search to other interesting areas of the visual phrase. This helps in cases where the visual phrase detection is not perfectly accurate (fig. 3.15a, right detection) or the windows with the highest belief in the VP2C map do not contain the auxiliary object (fig. 3.15b). Fig. 3.15c shows an example where there is a false-positive visual phrase detection and a missing one, due to its small size. However, the remaining visual phrase instances are correctly detected, resulting in the correct detection of the individual objects within. Finally, fig. 3.15d shows how an already good initialization by VP2C can be refined even further into a very accurate detection by IVP2C.

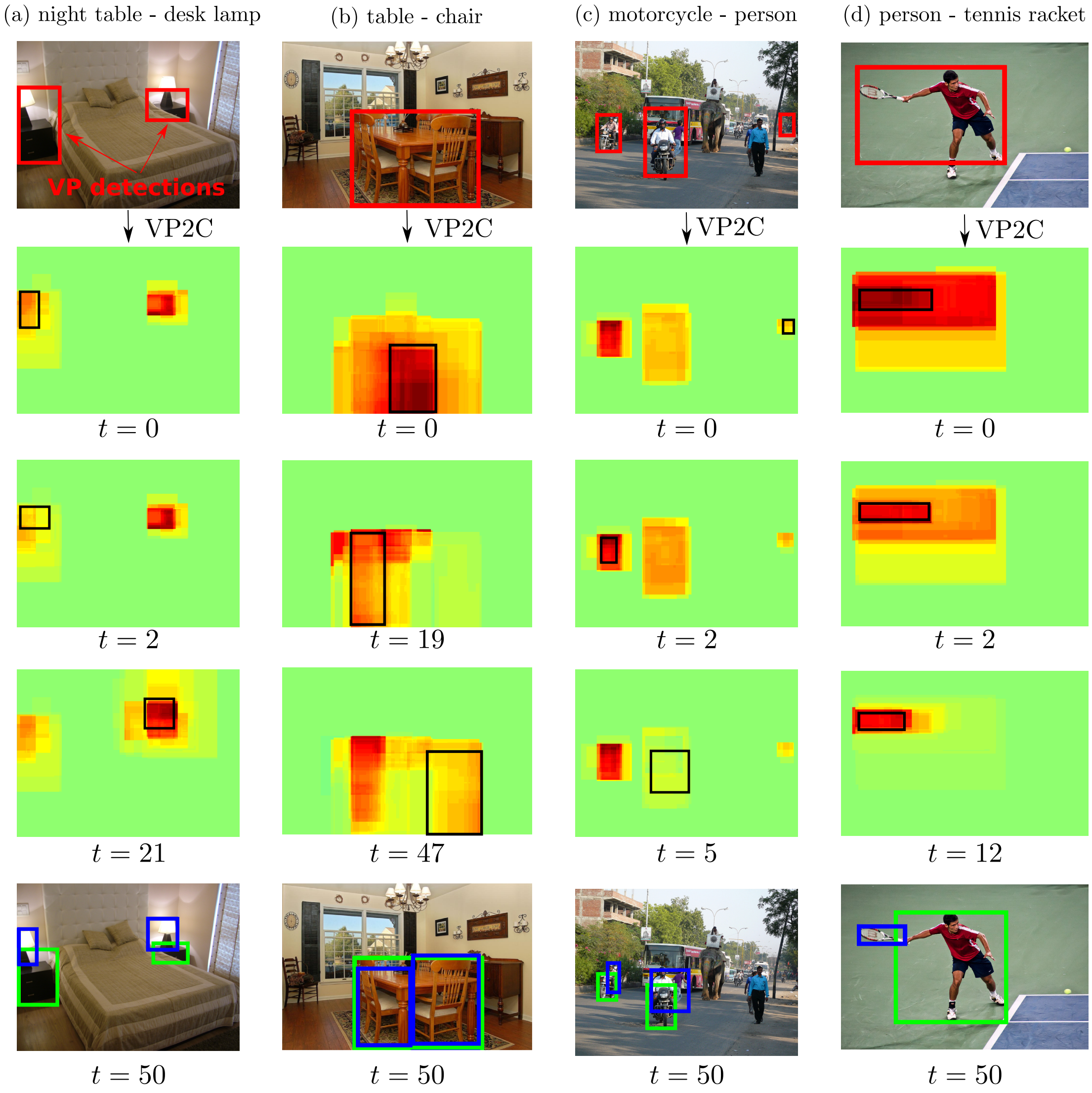


Figure 3.15: *Qualitative results for the joint active search on SUN2012. (Top) Visual phrase detections. (Middle) Belief maps for the auxiliary class for some iterations. The black window indicates the next window to be observed. (Bottom) The top scored detections for dominant class (green) and auxiliary (blue).*

3.8 Conclusions and outlook

Most object detectors independently evaluate a classifier on all windows in a large set. Instead, we presented an active search strategy that sequentially chooses the next window to evaluate based on all the information gathered before. Our search effectively combines two complementing driving forces: context and window classifier score. We also extend our method to jointly search over pairs of classes. We use visual phrases, i.e. pairs of objects that often co-occur nearby in an image. We start the search by finding the visual phrase, effectively sharing computation for both classes. Then, we branch off into searching for the individual objects within the visual phrase, exploiting the highly specific context it provides.

In experiments on several datasets, our single-class method substantially reduces the number of window classifier evaluations. Due to the efficiency of our proposed context extractor based on Random Forests, we add little overhead to the detection pipeline, obtaining significant speed-ups in actual runtime for some detectors. We also show how our joint active search is even more efficient than the single-class version for the particularly challenging object classes.

We now address some limitations of the presented active search approach and present possible solutions, along with additional future work.

Update for modern CNN architectures. The efficiency improvement brought by our active search has become rather undermined by the new CNN architectures used by recent object detectors. Approaches like Fast R-CNN (Girshick, 2015) greatly reduce the necessary computation by sharing the processing of the convolutional layers for all windows. A further progress is achieved with Faster R-CNN (Ren et al., 2015), which generates object proposals inside the network and thus dramatically reduces the total runtime and the number of windows to evaluate. Even beyond that, the CNNs of SSD (Liu et al., 2016) and YOLO (Redmon et al., 2016; Redmon and Farhadi, 2017) directly predict object locations without the need of object proposals. We suggest here two possible routes to update our active search in order to improve the efficiency of a larger number of detectors, including the most recent.

First, we can adapt our active search for CNN architectures that share convolutional layers as shown in figure 3.16. In our current version, each search step observes a window on the input image. The proposed adaptation would instead process all the convolutional layers only once for the whole image, and then search on the resulting convolutional maps. In this case, the search would observe the convolutional

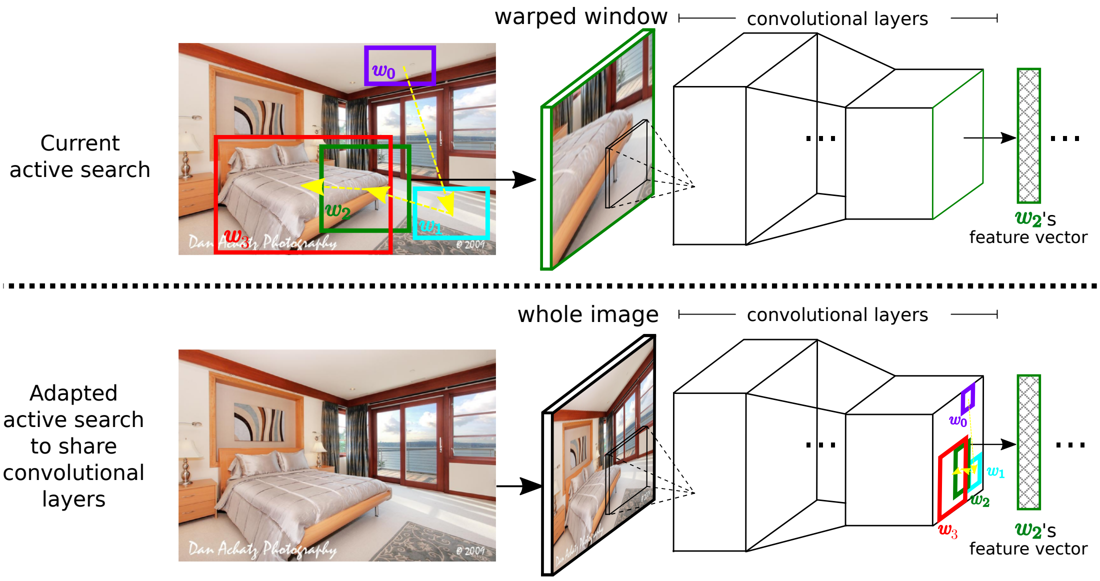


Figure 3.16: *(Top) Our current active search extracts convolutional features independently for each window. (Bottom) The proposed adaptation shares the convolutional layers for all windows. In this case, the search is performed directly on the convolutional maps of the whole image, and hence the image pixels of each window are not explicitly observed.*

features of the region corresponding to each window directly, as opposed to starting from the window pixels in the input image. Additionally, the context extractor may be implemented inside the network by using the pooled convolutional features alongside location features, thus only adding a very small overhead. Moreover, such approach enables end-to-end training, which would optimize the kind of windows suggested by the context extractor to maximize object detection performance.

An alternative adaptation of our search strategy consists in directly searching on the 4D space of all possible windows, instead of relying on externally generated object proposals. In this approach, we would model the distribution of windows according to the probability of containing an object of the class. The search would then sequentially refine this distribution with the knowledge gained from each observation, both locally around the observation and on distant areas based on contextual information. The entire method could be implemented inside the network, which would allow learning search strategies that are both efficient and effective.

Decision machine for learning a search strategy. Our current active search follows a fixed structure: it observes a window, gathers its score and context information, and decides where to look next. We could generalize our approach by giving it more freedom in the steps it can take when searching. A possible way of implementing such generalization is using a decision machine that, at each time step, actively selects an action out of a large pool of possible actions based on the information gathered so far. We consider two types of actions: queries and moves. Queries are used to obtain information and include evaluating the window classifier, applying the context extractor, or proposing unexplored areas to be visited next. Moves, on the other hand, bring the search forward, for example, by selecting the next window to be evaluated or terminating the search. Each query would have an incurred computational cost, which we could measure empirically. All measurements may be performed in a pre-search calibration phase tailored to the current circumstances, such as the computer specifications or the used window classifier.

This decision machine would be trained to learn how to choose actions that maximize object detection performance while keeping the computational cost low, possibly within a given budget. It generalizes most of the current search paradigms used in the literature, namely sliding-window, our active search, or even the branch-and-bound approach of Lampert et al. (2008).

Connections with human visual search. Eye trackers are devices designed to record the eye positions of human observers as they look at images, providing a detailed description of the image regions observed at any point in time. Using eye-tracking data, we could compare the paths followed by our active search with the paths people use when searching in images. Furthermore, we could try to learn a visual search strategy that mimics the human visual search. For example, we can exploit the data that eye-trackers provide regarding where people fixate their gaze. With this data, we can learn the appearance of interesting fixation areas that lead to quick object identification, and record their relative location with respect to the object. Our search strategy could then find these fixation regions and use them to guide the search, leveraging the type of context effectively used by humans. Moreover, we could learn additional search parameters from the human visual search, such as the number of steps needed to find the objects or the optimal step length of each search move.

Chapter 4

Do semantic parts emerge in Convolutional Neural Networks?

4.1 Introduction

Recently, Convolutional Neural Networks (CNNs) have achieved impressive results on many visual recognition tasks, and have become the state-of-the-art model for the object detection task (Girshick et al., 2014; He et al., 2014; Girshick, 2015; Ren et al., 2015; Redmon et al., 2016; Liu et al., 2016). Semantic parts, which are object regions interpretable by humans (e.g. wheel, leg) play a fundamental role in object detection and several other visual recognition tasks. For this reason, semantic part-based models have gained significant attention in the last few years. The key advantages of exploiting semantic part representations is that parts have lower intra-class variability than whole objects, they deal better with pose variation and their configuration provides useful information about the aspect of the object.

In this chapter we look into these two worlds and address the following question: *"does a CNN learn semantic parts in its internal representation?"* In order to answer it, we investigate whether the network's convolutional filters learn to respond to semantic parts of objects. Some previous works (Zeiler and Fergus, 2014; Simonyan et al., 2014) have suggested that semantic parts do emerge in CNNs, but only based on looking at some filter responses on a few images. Here we go a step further and perform two quantitative evaluations that examine the different stimuli of the CNN filters and try to associate them with semantic parts. First, we take advantage of the available ground-truth part location annotations in the PASCAL-Part dataset (Chen et al., 2014) to count how many of the annotated semantic parts emerge in a CNN. Second, we use human judgements to determine what fraction of all filters systematically fire on any semantic part (including parts that might not be annotated in PASCAL-Part).

For the first evaluation we use part ground-truth location annotations in the PASCAL-Part dataset (Chen et al., 2014) to answer the following question: *"how many semantic parts emerge in CNNs?"*. As an analysis tool, we turn filters into part detectors based on their responses to stimuli. If some filters systematically respond to a certain semantic part, their detectors will perform well, and hence we can conclude that they do represent the semantic part. Given the difficulty of the task, while building the detectors we assist the filters in several ways. The actual image region to which a filter responds typically does not accurately cover the extent of a semantic part. We refine this region by a regressor trained to map it to a part's ground-truth bounding-box. Moreover, as suggested by other works (Simon et al., 2014; Simon and Rodner, 2015; Xiao et al., 2015), a single semantic part might emerge as distributed across several filters. For this reason, we also consider filter combinations as part detectors, and automatically select the optimal combination of filters for a semantic part using a Genetic Algorithm. We present an extensive analysis on AlexNet (Krizhevsky et al., 2012) finetuned for object detection (Girshick et al., 2014). Results show that 34 out of 105 semantic parts emerge. This is a modest number, despite all favorable conditions we have engineered into the evaluation and all assists we have given to the network. This result demystifies the impressions conveyed by (Zeiler and Fergus, 2014; Simonyan et al., 2014) and shows that the network learns to associate filters to part classes, but only for some of them and often to a weak degree. In general, these semantic parts are those that are large or very discriminative for the object class (e.g., torso, head, wheel). Finally, we analyze different network layers, architectures, and supervision levels. We observe that part emergence increases with the depth of the layer, especially when using deeper architectures such as VGG16 (Simonyan and Zisserman, 2015). Moreover, emergence decreases when the network is trained for tasks less related to object parts, e.g. scene classification (Zhou et al., 2014).

Our second quantitative evaluation answers the converse question: *"what fraction of all filters respond to any semantic part?"*. As PASCAL-Part is not fully annotated (e.g. car door handle is missing), we answer it using human judgements. For each filter, we show human annotators the 10 images with the highest activations per object class. We highlight the regions corresponding to the activations and ask the annotators whether they systematically cover the same concept (e.g. a semantic part, a background, a texture, a color, etc.). In case of positive answer, we ask them to name the concept (e.g. horse hoof). In general, the majority of the filters do not seem to systematically respond to any concept. On average per object class, 7% of the filters correspond to semantic parts (including several filters responding to the same semantic part). About 10% of the filters systematically respond to other stimuli such as colors, subregions of parts or even assemblies of multiple parts. Finally, we also compare the semantic parts emerging in this evaluation with the 34 parts annotated in PASCAL-Part that emerged in the first evaluation. We find that nearly all the parts that emerge according to the detection performance criterion used in the first evaluation also emerge according to human judgements. However, more semantic parts emerge according to human judgements, including several parts that are not annotated in PASCAL-Part.

Finally, we also investigate how discriminative network filters and semantic parts are for recognizing objects. We explore the possibility that some filters respond to 'parts' as recurrent discriminative patches, rather than truly semantic parts. We find that, for each object class in PASCAL-Part, there are on average 9 discriminative filters that are largely responsible for recognizing it. Interestingly, 40% of these are also semantic according to human judgements, which is a much greater proportion than the 7% found when considering *all* filters. The overlap between which filters are discriminative and which ones are semantic might be the reason why previous works (Zeiler and Fergus, 2014; Simonyan et al., 2014) have suggested a stronger emergence of semantic parts, based on qualitative visual inspection. We also investigate to what degree the emergence of semantic parts in the network correlates with their discriminativeness for recognition. Interestingly, these are highly correlated: semantic parts that are discriminative emerge much more than other semantic parts. While this is generally assumed in the community, ours is the first work presenting a proper quantitative evaluation that turns this assumption into a fact.

The work presented in this chapter has been published in IJCV (Gonzalez-Garcia et al., 2017a).

The rest of the chapter is organized as follows. Section 4.2 discusses some related work. Section 4.3 presents our quantitative evaluation using PASCAL-Part bounding-boxes, while section 4.4 presents the evaluation using human judgements. The discriminativeness of filters is investigated in section 4.5, while section 4.6 explores the discriminativeness of semantic parts. Finally, section 4.7 summarizes the conclusions of our study and proposes future work.

4.2 Related Work

Analyzing CNNs. CNN-based representations are unintuitive and there is no clear understanding of why they perform so well or how they could be improved. In an attempt to better understand the properties of a CNN, some recent vision works have focused on analyzing their internal representations (Szegedy et al., 2014; Yosinski et al., 2014; Lenc and Vedaldi, 2015b; Mahendran and Vedaldi, 2015; Zeiler and Fergus, 2014; Simonyan et al., 2014; Agrawal et al., 2014; Zhou et al., 2015; Eigen et al., 2013). Some of these investigated properties of the network, like stability (Szegedy et al., 2014), feature transferability (Yosinski et al., 2014), equivariance, invariance and equivalence (Lenc and Vedaldi, 2015b), the ability to reconstruct the input (Mahendran and Vedaldi, 2015) and how the number of layers, filters and parameters affects the network performance (Agrawal et al., 2014; Eigen et al., 2013).

More related to this paper are (Zeiler and Fergus, 2014; Simonyan et al., 2014; Agrawal et al., 2014; Zhou et al., 2015), which look at the convolutional filters. Zeiler and Fergus (2014) use deconvolutional networks to visualize locally optimal visual inputs for individual filters. Simonyan et al. (2014) use a gradient-based visualization technique to highlight the areas of an image discriminative for an object class. Agrawal et al. (2014) show that the feature representations are distributed across object classes. Zhou et al. (2015) show that the layers of a network learn to recognize visual elements at different levels of abstraction (e.g. edges, textures, objects and scenes). Most of these works make an interesting observation: filter responses can often be linked to semantic parts (Zeiler and Fergus, 2014; Simonyan et al., 2014; Zhou et al., 2015). These observations are however mostly based on casual visual inspection of few images (Zeiler and Fergus, 2014; Simonyan et al., 2014). Zhou et al. (2015) is the only work presenting some quantitative results based on human judgements, but not focused on semantic parts. Instead, we present an extensive quantitative analysis on whether filters can be associated with semantic parts and to which degree. We transform the filters into part detectors and evaluate their performance on ground-truth part bounding-boxes from the PASCAL-Part dataset (Chen et al., 2014). Moreover, we present a second quantitative analysis based on human judgements where we categorize filters into semantic parts. We believe this methodology goes a step further than previous works and supports more conclusive answers to the quest for semantic parts.

Filters as intermediate part representations for recognition. Several works use filter responses for recognition tasks (Simon et al., 2014; Gkioxari et al., 2015; Simon and Rodner, 2015; Xiao et al., 2015; Oquab et al., 2015). Simon et al. (2014) train part detectors for fine-grained recognition, while Gkioxari et al. (2015) train them for action and attribute classification. Furthermore, Simon and Rodner (2015) learn constellations of filter activation patterns, and Xiao et al. (2015) cluster group of filters responding to different bird parts. All these works assume that the convolutional layers of a network are related to semantic parts. In this paper we try to shed some light on this assumption and hopefully inspire more works on exploiting the network's internal structure for recognition.

4.3 PASCAL-Part emergence in CNNs

Our goal is understanding whether the convolutional filters learned by the network respond to semantic parts. In order to do so, we investigate the image regions to which a filter responds and try to associate them with a particular part.

Network architecture. Standard image classification CNNs such as (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015) process an input image through a sequence of layers of various types, and finally output a class probability vector. Each layer i takes the output of the previous layer x^{i-1} as input, and produces its output x^i by applying up to four operations: convolution, nonlinearity, pooling, and normalization. The convolution operation slides a set of learned filters of different sizes and strides over the input. The nonlinearity of choice for many networks is the Rectified Linear Unit (ReLU) (Krizhevsky et al., 2012), and it is applied right after the convolution.

4.3.1 Methodology

Fig. 4.1 presents an overview of our approach. Let f^i_j be the j-th convolutional filter of the i-th layer, including also the ReLU. Each pixel in a feature map $x^i_j = f^i_j(x^{i-1})$ is the activation value of filter f^i_j applied to a particular position in the feature maps x^{i-1} of the previous layer. The resolution of the feature map depends on the layer, decreasing as we advance through the network. Fig. 4.1 shows feature maps for layers 1, 2, and 5. When a filter responds to a particular stimulus in its input, the corresponding region on

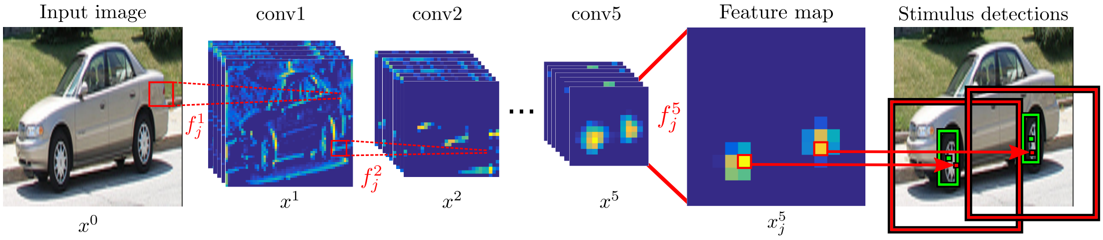


Figure 4.1: *Overview of our approach for a layer 5 filter. Each local maxima of the filter's feature map leads to a stimulus detection (red). We transform each detection with a regressor trained to map it to a bounding-box tightly covering a semantic part (green).*

the feature map has a high activation value. By studying the stimuli that cause a filter to fire, we can characterize them and decide whether they correspond to a semantic object part.

4.3.1.1 Stimulus detections from activations

The value $a_{c,r}$ of each particular activation α, located at position (c,r) of feature map x_j^i, indicates the response of the filter to a corresponding region in its input x^{i-1}. The receptive field of an activation is the region on the input image on which the filter acted, and it is determined by the network structure. By recursively back-propagating the input region of activation α down the layers, we can reconstruct the actual receptive field on the input image. The size of the receptive field varies depending on the layer, from the actual size of the filter for the first convolutional layer, up to a much larger image region on the top layer. For each feature map, we select all its local maxima activations. Each of these activations will lead to a stimulus detection in the image regardless of its activation value (i.e. no minimum threshold). Therefore, all peaks of the feature map become detections, and their detection scores are their activation values. The location of such detections is defined by the center of the receptive field of the corresponding activation, whereas its size varies depending on the layer. Fig. 4.1 shows an example, where the two local maxima of feature map x_j^5 lead to the stimulus detections depicted in red, which correspond to their receptive fields on the image.

Regressing to part bounding-boxes. The receptive field of an activation gives a rough indication about the location of the stimulus. However, it rarely covers a part tightly enough to associate the stimulus with a part instance (fig. 4.2). In general, the receptive field of high layers is significantly larger than the part ground-truth bounding-box, especially for small classes like ear. Moreover, while the receptive field is always square, some classes have other aspect ratios (e.g. legs). Finally, the response of a filter to a part might not occur in its center, but at an offset instead (e.g. on the bottom area, fig. 4.2d-e).

In order to factor out these elements, we assist each filter with a bounding-box regression mechanism that refines its stimulus detection for each part class. This regressor turns each stimulus detection, which are generally bigger than the corresponding part and have a squared aspect ratio (fig. 4.1) into more accurate detections that fit the part instances more tightly (fig. 4.2). The regressor applies a 4D transformation, i.e. translation and scaling along width and height. We believe that if a filter fires systematically on many instances of a part class at the same relative location (in 4D), then we can grant that filter a 'part detector' status. This implies that the filter responds to that part, even if the actual receptive field does not tightly cover it. For the rest of the paper, all stimulus detections include this regression step unless stated otherwise.

We train one regressor for each part class and filter. Let $\{G^l\}$ be the set of all ground-truth bounding-boxes for the part in the training set. Each instance bounding-box G^l is defined by its center coordinates (G^l_x, G^l_y), width G^l_w, and height G^l_h. We train the regressor on K pairs of activations and ground-truth part bounding-boxes $\{\alpha^k, G^k\}$. Let (c_x, c_y) be the center of the receptive field on the image for a particular feature map activation α of value $a_{c,r}$, and let w, h be its width and height ($w = h$ as all receptive fields are square). We pair each activation with an instance bounding-box G^l of the corresponding image if (c_x, c_y) lies inside it. We are going to learn a 4D transformation d_x, d_y, d_w, d_h to predict a part bounding-box G' from α's receptive field

$$G'_x = x + d_x(\gamma(\alpha)) \qquad G'_w = d_w(\gamma(\alpha))$$

$$G'_y = y + d_y(\gamma(\alpha)) \qquad G'_h = d_h(\gamma(\alpha)),$$

where $\gamma(\alpha) = (c_x, c_y, a_{c-1,r-1}, a_{c-1,r}, \ldots, a_{c+1,r+1})$. Therefore, the regression depends on the center of the receptive field and on the values of the 3x3 neighborhood of the activation on the feature map. Note that it is independent of w and h as these are fixed for a given layer. Each d_* is a linear combination of the elements in $\gamma(\alpha)$ with a weight vector w_*, where $*$ can be x, y, w, or h.

We set regression targets $(t^k_x, t^k_y, t^k_w, t^k_h) = (G^k_x - c^k_x, G^k_y - c^k_y, G^k_w, G^k_h)$ and optimize the following weighted least squares objective

$$w_* = \underset{w'_*}{\operatorname{argmin}} \sum_{k=1}^{K} a^k_{c,r} (t^k_* - w'_* \cdot \gamma(\alpha^k))^2. \tag{4.1}$$

In practice, this tries to transform the position, size and aspect-ratio of the original receptive field of the activations into the bounding-boxes in $\{G^l\}$.

Fig. 4.2 presents some examples of our bounding-box regression for 6 different parts. For each part, we show the feature map of a layer 5 filter and both the original receptive field (red) and the regressed box (green) of some activations. We can see how given a local maximum activation on the feature map, the regressor not only refines the center of the detection, but also successfully captures its extent. Some classes are naturally more challenging, like *dog-tail* in fig. 4.2f, due to higher size and aspect-ratio variance or lack of satisfactory training examples.

Evaluating filters as part detectors. For each filter and part combination, we need to evaluate the performance of the filter as a detector of that part. We take all the local maxima of the filter's feature map for every input image and compute their stimulus detections, applying Non-Maxima Suppression (Felzenszwalb et al., 2010b) to remove duplicate detections. Algorithm 3 summarizes the part detection process used. We consider a stimulus detection as correct if it has an intersection-over-union ≥ 0.4 with any ground-truth bounding-box of the part, following Chen et al. (2014). All other detections are considered false positives. A filter is a good part detector if it has high recall but a small number of false positives, indicating that when it fires, it is because the part is present. Therefore, we use Average Precision (AP) to evaluate the filters as part detectors, following the PASCAL VOC (Everingham et al., 2010) protocol.

4.3.1.2 Filter combinations

Several works (Agrawal et al., 2014; Zhou et al., 2015; Xiao et al., 2015) noted that one filter alone is often insufficient to cover the spectrum of appearance variation of an object class. We believe that this holds also for part classes. For this reason, we present here a technique to automatically select the optimal combination of filters for a part class.

For a given network layer, the search space consists of binary vectors $\mathbf{z} = [z_1, z_2, ..., z_N]$, where N is the number of filters in the layer. If $z_i = 1$, then the i-th filter is included in the combination. We consider the stimulus detections of a filter combination as the set

```
Input   : Image $I$, network filter $f$
Output: Set of part detections D
// Obtain local maxima of feature map $f(I)$
$A \leftarrow$ maxima($f(I)$) ;
for $a \in A$ do
    // Back-propagate to find stimulus
    $x \leftarrow$ receptive_field($a$) ;
    // Regress to refined stimulus
    $x' \leftarrow d(x)$ ;
end
$D \leftarrow$ NMS($\{x'\}$);
```

Algorithm 3: Filters as part detectors.

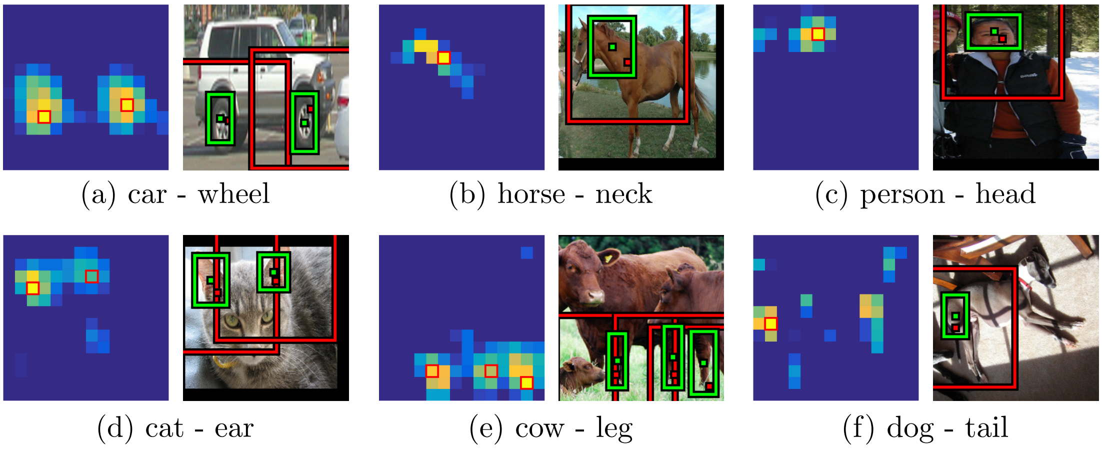

(a) car - wheel
(b) horse - neck
(c) person - head
(d) cat - ear
(e) cow - leg
(f) dog - tail

Figure 4.2: *Examples of stimulus detections for layer 5 filters. For each part class we show a feature map on the left, where we highlight some local maxima in red. On the right, instead, we show the corresponding original receptive field and the regressed box.*

union of the individual detections of each filter in it. Ideally, a good filter combination should make a better part detector than the individual filters in it. Good combinations should include complementary filters that jointly detect a greater number of part instances, increasing recall. At the same time, the filters in the combination should not add many false positives. Therefore, we can use the collective AP of the filter combination as objective function to be maximized:

$$\mathbf{z} = \operatorname*{argmax}_{\mathbf{z}'} \mathrm{AP}\Big(\bigcup_{i \in \{j | z'_j = 1\}} \mathrm{det}_i \Big), \tag{4.2}$$

where det_i indicates the stimulus detections of the i-th filter.

We use a Genetic Algorithm (GA) (Mitchell, 1998) to optimize this objective function. GAs are iterative search methods inspired by natural evolution. At every generation, the algorithm evaluates the 'fitness' of a set of search points (population). Then, the GA performs three genetic operations to create the next generation: selection, crossover and mutation. In our case, each member of the population (chromosome) is a binary vector $\mathbf{z}$ as defined above. Our fitness function is the AP of the filter combination. In our experiments, we use a population of 200 chromosomes and run the GA for 100 generations. We use Stochastic Universal Sampling (Mitchell, 1998). We set the crossover and mutation probabilities to 0.7 and 0.3, respectively. We bias the initialization towards a small number of filters by setting the probability $P(z_i = 1) = 0.02, \forall i$. This leads to an average combination of 5 filters when $N = 256$, in the initial population.

We underline that our goal is to find filter combinations that act collectively as part detectors, which is formalized in the objective (4.2). While a GA is a suitable method to maximize (4.2), other methods could be used instead.

4.3.2 AlexNet for object detection

In this section we analyze the role of convolutional filters in AlexNet and test whether some of them can be associated with semantic parts. In order to do so, we design our settings to favor the emergence of this association.

4.3.2.1 Experimental settings

Dataset. We evaluate filters on the recent PASCAL-Part dataset (Chen et al., 2014), which augments PASCAL VOC 2010 (Everingham et al., 2010) with pixel-wise se-

Class	Part	Layer 1 (96)			Layer 2 (256)			Layer 3 (384)			Layer 4 (384)			Layer 5 (256)		
		Best	GA	nFilters	Best	GA	nFilters	Best	GA	nFilters	Best	GA	nFilters	Best	GA	nFilters
aero	body	17.7	+3.4	12	23.7	+10.2	33	29.4	+9.5	62	34.0	+9.2	49	29.3	+17.0	49
	stern	10	+1.5	14	13.6	+5.1	33	21.4	+2.5	45	19.2	+5.2	32	15.0	+9.4	21
	wing	4.2	+1.2	12	5.6	+5.3	41	6.0	+7.0	63	6.9	+3.9	36	4.7	+9.6	38
	engine	2.1	+0.9	14	2.7	+1.6	5	4.2	+1.7	37	4.5	+2.5	53	1.6	+5.4	25
bike	wheel	14.2	+0.7	19	41.4	+0.0	30	49.2	+0.0	3	60.0	+0.0	10	57.1	+6.1	16
	saddle	1.0	+0.5	5	1.7	+0.5	18	1.6	+0.6	43	1.7	+0.0	4	2.1	+2.5	16
	handlebar	2.8	+0.4	17	3	+2.2	21	4.0	+2.0	37	3.2	+3.1	40	4.1	+5.8	38
	chainwheel	0.6	+0.1	4	0.6	+0.6	24	1.9	+0.0	46	1.7	+1.0	17	3.0	+0.6	6
bird	head	5.7	+0.0	1	8.0	+0.4	5	14.7	+5.0	16	24	+2.0	17	23.8	+6.5	23
	beak	0.8	+0.0	1	0.9	+0.4	35	1.1	+2.4	57	1.4	+3.2	45	2.1	+4.5	28
	torso	31.4	+0.8	8	37.5	+2.5	20	41.6	+2.7	47	44.3	+2.4	41	55.9	+5	29
	wing	4.6	+1.5	11	7.0	+7.8	38	9.7	+5.2	39	9.3	+8.3	33	7.7	+8.3	36
	tail	1.8	+0.2	9	2.4	+1.8	39	2.9	+3.7	39	4.6	+3.8	25	7.0	+4.3	17
bottle	cap	1.8	+0.6	13	4.4	+2.2	20	6.4	+0.9	21	11.2	+0.0	11	6.6	+4.6	15
	body	73.0	+0.6	4	80.9	+0.0	9	87.6	+0.0	10	83.4	+0.0	3	81.0	+6.3	25
car	mirror	0.4	+0.1	5	0.3	+0.3	14	0.4	+0.1	16	0.4	+0.2	27	0.4	+0.3	12
	door	5.3	+0.3	5	7.8	+1.4	36	11.1	+4.0	35	13.0	+4.6	55	16.5	+7.5	23
	wheel	3.5	+0.0	1	9.5	+3.2	8	27.7	+3.8	6	30.0	+4.8	15	35.4	+4.0	17
	headlight	0.3	+0.1	4	1.3	+0.0	13	1.0	+0.8	32	0.8	+0.9	32	0.6	+1.1	19
	window	5.3	+0.0	2	9.0	+1.2	12	11.8	+5.1	28	21.2	+0.0	18	19.8	+5.4	22
cat	head	16.8	+0.0	1	21.2	+0.0	8	30.6	+6.0	8	44.5	+1.1	15	53.9	+5.2	10
	eye	0.6	+0.0	4	10.4	+1.6	3	10.8	+0.0	2	3.8	+0.5	18	4.3	+1.3	4
	ear	1.9	+0.6	10	4.4	+0.3	14	4.9	+5.7	12	10.7	+5.1	13	17.5	+2.8	10
	torso	35.8	+0.7	6	40.2	+1.7	4	43.6	+2.4	32	46.7	+6.1	32	50.8	+4.2	25
	paw	0.6	+0.1	6	0.7	+0.4	18	1.9	+1.2	21	3.2	+0.0	11	1.5	+1.9	16
	tail	1.0	+0.0	1	1.3	+1.4	35	2.1	+2.8	71	2.3	+4.0	42	2.2	+3.9	24
cow	head	12.9	+0.4	12	15.8	+3.8	34	21.2	+8.5	71	22.9	+4.9	50	24.6	+17.1	34
	muzzle	3.4	+0.2	14	4.9	+2.9	37	15.4	+0.0	10	15.6	+1.9	14	16.7	+9.5	22
	torso	43.1	+0.0	1	56.6	+0.0	45	62.0	+9.0	42	63.6	+9.9	53	65.2	+13.6	42
	tail	0.9	+0.0	9	2.9	+1.1	19	2.9	+2.2	16	7.0	+2.5	30	3.7	+0.8	6
horse	head	5.4	+0.3	3	7.6	+1.8	22	10.7	+3.1	52	15.3	+5.2	27	16.1	+11.6	22
	ear	0.9	+0.3	11	1.3	+1.1	18	4.3	+1.6	9	2.7	+3.2	12	6.1	+0.5	4
	muzzle	3.2	+1.6	6	2.7	+2.3	29	4.9	+3.2	48	8.2	+5.4	30	12.1	+5.4	19
	torso	48.7	+0.9	9	52.7	+4.1	22	63.8	+0.0	11	63.0	+4.4	27	65.2	+7.1	29
	neck	3.7	+0.9	16	4.4	+5.0	51	7.3	+7.0	50	8.7	+6.7	50	12.2	+5.8	18
	leg	6.3	+0.2	10	10.7	+3.6	6	14.2	+7.5	8	23.0	+5.7	9	23.4	+9.6	14
sheep	head	7.9	+1.0	2	8.5	+2.7	20	19.4	+0.0	10	23.2	+2.4	10	23.4	+12.5	34
	ear	2.3	+0.5	5	2.8	+1.9	19	3.5	+2.4	20	2.4	+4.9	30	2.4	+5.3	22
	torso	43.6	+0.0	1	61.3	+1.6	29	74.1	+0.0	11	70.9	+0.0	7	82.2	+3.2	18
	leg	1.4	+0.0	2	1.9	+1.1	17	3.2	+1.2	5	6.0	+2.4	11	4.6	+2.6	15
person	head	6.6	+0.0	1	8.7	+0.0	3	33.8	+0.0	5	44.9	+0.0	6	58.2	+0.0	1
	hair	3.9	+0.1	4	5.1	+0.0	3	18.0	+0.0	11	28.7	+0.0	4	30.6	+0.0	1
	torso	16.1	+0.0	1	21.7	+1.8	10	23.7	+6.8	10	32.8	+1.7	9	38.3	+4.4	8
	arm	3.7	+0.0	1	4.7	+1.5	6	4.5	+3.5	13	5.4	+1.4	19	8.5	+4.7	7
	hand	0.9	+0.0	1	1.7	+0.0	3	1.4	+0.2	7	1.5	+0.0	2	0.6	+0.5	10
	foot	0.4	+0.0	1	0.7	+0.0	6	1.8	+0.0	6	1.3	+0.3	2	1.6	+1.0	6
mean (105 parts)		9.0	+0.6	6.9	12.4	+2.0	19.1	16.5	+2.6	24.0	17.8	+3.3	24.6	19.0	+5.2	18.8
absolute GA mAP			9.6			14.4			19.1			21.1			24.2	
all filters mAP		5.5		96	4.5		256	5.6		384	6.5		384	9.2		256

Table 4.1: *Part detection results in terms of AP on the* `train` *set of PASCAL-Part for AlexNet-Object.* Best *is the AP of the best individual filter whereas* GA *indicates the increment over* Best *obtained by selecting the combination of* (nFilters) *filters, hence the* $+$ *sign.*

mantic part annotations. For our experiments we fit a bounding-box to each part segmentation mask. We use the `train` subset and evaluate all parts listed in PASCAL-Part with some minor refinements: we discard fine-grained labels (e.g. 'car wheel front-left' and 'car wheel back-left' are both mapped to *car-wheel*), merge contiguous subparts of the same larger part (e.g. 'person upper arm' and 'person lower arm' become a single part *person-arm*), discard very tiny parts (average of widths and heights over the whole training set ≤ 15 pixels, like 'person eyebrow'), and discard parts with less than ≤ 10 samples in `train` (like 'bicycle headlight', which has only one annotated sample). The final dataset contains 105 parts of 16 object classes (appendix B).

AlexNet. One of the most popular networks in computer vision is the CNN model of Krizhevsky et al. (Krizhevsky et al., 2012), winner of the ILSVRC 2012 image classification challenge (Russakovsky et al., 2015). It is commonly referred to as AlexNet. This network has 5 convolutional layers followed by 3 fully connected layers. The number of filters at each of the convolutional layers L is: 96 (L1), 256 (L2), 384 (L3), 384 (L4), and 256 (L5). The filter size changes across layers, from 11x11 for L1, to 5x5 to L2, and to 3x3 for L3, L4, L5.

Training. We use the publicly available AlexNet network of Girshick et al. (2014). The network was initially pre-trained for image classification on the ILSVRC12 dataset and subsequently finetuned for object class detection (for the 20 classes in PASCAL VOC 2012 + background) using ground-truth annotations. Note how these bounding-boxes provide a coordinate frame common across all object instances. This makes it easier for the network to learn parts as it removes variability due to scale changes (the convolutional filters have fixed size) and presents different instances of the same part class at rather stable positions within the image. We refer to this network as *AlexNet-Object*. The network is trained on the `train` set of PASCAL VOC 2012. Note how this set is a superset of PASCAL VOC 2010 `train`, on which we analyze whether filters correspond to semantic parts.

Finally, we assist each of its filters by providing a bounding-box regression mechanism that refines its stimulus detections to each part class (sec. 4.3.1.1) and we learn the optimal combination of filters for a part class using a GA (sec. 4.3.1.2).

Evaluating settings. We restrict the network inputs to ground-truth object bounding-boxes. More specifically, for each part class we look at the filter responses only inside the instances of its object class and ignore the background. For example, for *cow-head* we only analyze *cow* ground-truth bounding-boxes. Furthermore, before inputting a bounding-box to the network we follow the R-CNN pre-processing procedure (Girshick et al., 2014), which includes adding a small amount of background context and warping to a fixed size. An example of an input bounding-box is shown in fig. 4.1. Finally, we intentionally evaluate on the `train` subset that was used to train the network (for object detection, without part annotations). These settings are designed to be favorable to the emergence of parts as we ignore image background that does not contain parts and, more importantly, we use object instances seen by AlexNet-Object during training.

4.3.2.2 Results

Table 4.1 shows results for a few parts of seven object classes in terms of average precision (AP). For each part class and network layer, the table reports the AP of the best individual filter in the layer ('Best'), the increase in performance over the best filter thanks to selecting a combination of filters with our GA ('GA'), and the number of filters in that combination ('nFilters'). Moreover, the last three rows of the table report the mAP over all 105 part classes. This includes the absolute performance of the GA combination in the second to last row, for easy reference, and the performance considering all filters simultaneously. Several interesting facts arise from these results.

Need for regression. In order to quantify how much the bounding-box regression mechanism of sec. 4.3.1.1 helps, we performed part detection using the non-regressed receptive fields. On AlexNet-Object layer 5, taking the single best filter for each part class achieves an mAP of 6.1. This is very low compared to mAP 19.0 achieved by assisting the filters with the regression. Moreover, results show that the receptive field is only able to detect large parts (e.g. *bird-torso, bottle-body, cow-torso*, etc.). This is not surprising, as the receptive field of layer 5 covers most of the object surface (fig. 4.2). Instead, filters with regressed receptive fields can detect much smaller parts (e.g. *cat-ear, cow-muzzle, person-hair*), as the regressor shrinks the area covered by the receptive field and adapts its aspect ratio to the one of the part. Some filters at layer 1 benefit of the opposite effect. The reason why such filters with tiny receptive fields can detect large parts such as cow-torso and horse-torso, is because regression enlarges them appropriately. Without regression, layer 1 filters have zero AP for these parts. We conclude that the receptive field alone cannot perform part detection and regression is necessary.

Differences between layers. Generally, the higher the network layer, the higher the performance (table 4.1). This is consistent with previous observations (Zeiler and Fergus, 2014; Zhou et al., 2015) that the first layers of the network respond to generic corners and other edge/color junctions, while higher levels capture more complex structures. Nonetheless, it seems that some of the best individual filters of the very first layers can already perform detection to a weak degree when helped by our regression (e.g. *bike-wheel* has 14.9 AP).

Differences between part classes. Performance varies greatly across part classes, spanning a large range of values, from near 0 mAP to very high values, over 80 mAP. Due to this large variance, combining and comparing performance values for different parts becomes a challenging task. For this reason, we focus here on different performance ranges with smaller variation. For example, some parts are clearly not represented by any filter nor filter combination, as their AP is very low across all layers e.g. 7.0 for *aeroplane-engine*, 3.6 for *bike-chainwheel*, 2.6 AP for *person-foot*, at layer 5 using the GA combinations). On other parts instead, the network achieves good detection performance e.g. 64.2 for *bike-wheel*, 59.1 for *cat-head*, and 72.3 AP for *horse-torso*). This proves that some of the filters can be associated with these parts.

Differences between part sizes. Another factor that seems to influence the performance is the average size of the part. For example, the AP achieved on *horse-torso* 72.3 on layer 5) is much higher than on the smaller *horse-ear* 6.6). In order to understand if this is common across all parts, we looked at how AP changes with respect to the average size of a part (fig. 4.3). Interestingly, these two are indeed correlated and have a Pearson product-moment correlation coefficient (PPMCC) of 0.7 (Pearson, 1895). This shows that smaller parts emerge less in the CNN than larger ones. Nonetheless, small size does not always imply low detection performance: there are some rather small parts (around 20% of the object area) which have high AP (around 60), like *bicycle-wheel*, *motorbike-wheel* and *person-head*. As we show in sec. 4.6, these are parts that are very discriminative for recognizing the objects they belong to,

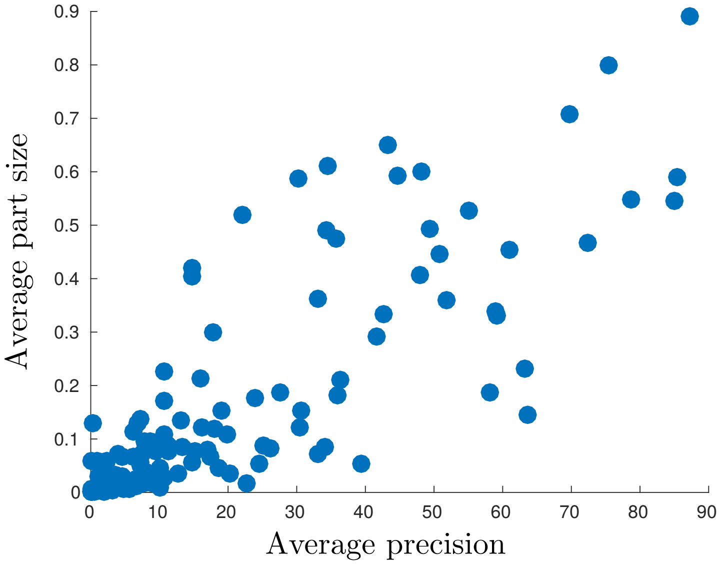


Figure 4.3: *Correlation between the size of a part class, averaged over all its instances, and part detection performance (AP for the best combination of layer 5 filters found by GA, table 4.1). The part size is normalized by the average size of the object class it belongs to. Each point corresponds to a different part class. These are highly correlated: Pearson product-moment correlation coefficient of 0.7.*

which justifies their emergence within the network.

Filter combinations by GA. Performing part detection using a combination of filters (GA) always performs better (or equal) than the single best filter. This is interesting, as it shows that different filters learn different appearance variations of the same part class. On the other hand, simultaneously using all filters brings a very low performance. This is due to irrelevant filters producing many false positives, which reduces AP. Moreover, combining multiple filters selected by the GA improves part detection performance more for deeper layers. This suggests that they are more class-specific, i.e. they dedicate more filters to learning the appearance of specific object/part classes. This can be observed by looking not only at the improvement in performance brought by the GA, but also at the number of filters that the GA selects. Clearly, filters in L1 are so far from being parts that even selecting many filters does not bring much improvement (+0.6 mAP only). Instead, in L4 and L5 there are more semantic filters and the GA combination helps more (+3.3 mAP and +5.2 mAP, respectively). Interestingly, for L5 the improvement is higher than for L4, yet fewer filters are combined. This

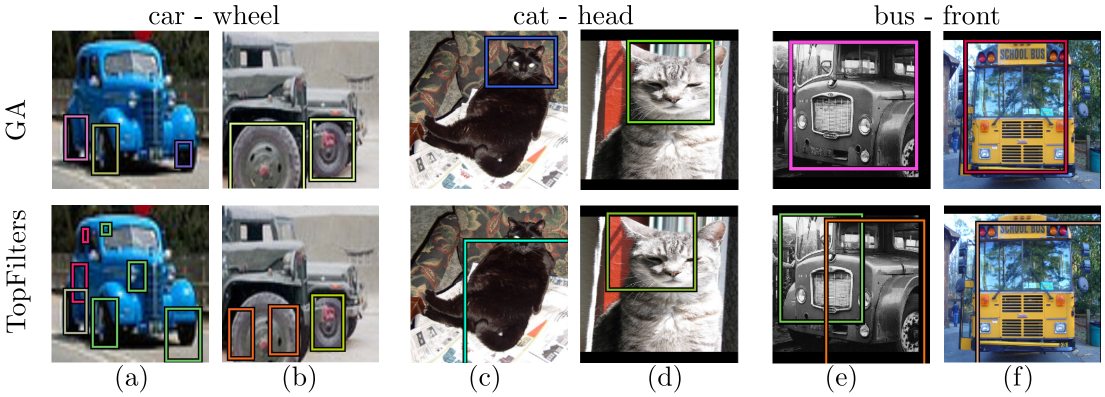


Figure 4.4: *Part detection examples obtained by combination of filters selected by our GA (top) or by TopFilters (bottom). Different box colors correspond to different filters' detections. Note how the GA is able to better select filters that complement each other.*

further shows that filters in higher layers better represent semantic parts.

TopFilters: a baseline for combining filters. The AP improvement provided by our GA for some parts is remarkable, like for *aeroplane-body* (+17.0), *horse-leg* (+9.6) and *cow-head* (+17.1). These results suggest that our GA is doing a good job in selecting filter combinations. Here we compare against a simpler method, dubbed TopFilters. It selects the top few best filters for a part class, based on their individual AP. We let TopFilters select the same number of filters as the GA. Our GA consistently outperforms TopFilters (24.2 vs 18.8 mAP, layer 5). The problem with TopFilters seems to be that often the top individual best filters capture the same visual aspect of a part. Instead, our GA can select filters that complement each other and work well jointly (indeed 57% of the filters it selects are not among those selected by TopFilters). We can see this phenomenon in fig. 4.4. On the blue car, TopFilters detects two wheels correctly, but fails to fit a tight bounding-box around the third wheel that appears much smaller (fig. 4.4a). Similarly, on the other car TopFilters fails to correctly localize the large wheel (fig. 4.4b). Instead, the GA localizes all wheels correctly in both cases. Furthermore, the GA fits tighter bounding-boxes for more challenging parts, achieving more accurate detections (fig. 4.4c-f). Finally, note how TopFilters does not even improve over selecting the single best filter (19.0), as more filters bring more false positive detections.

GA convergence. We present here an experiment to study the convergence of the GA by assessing the variance of its solutions. We take *car-wheel* as example part class

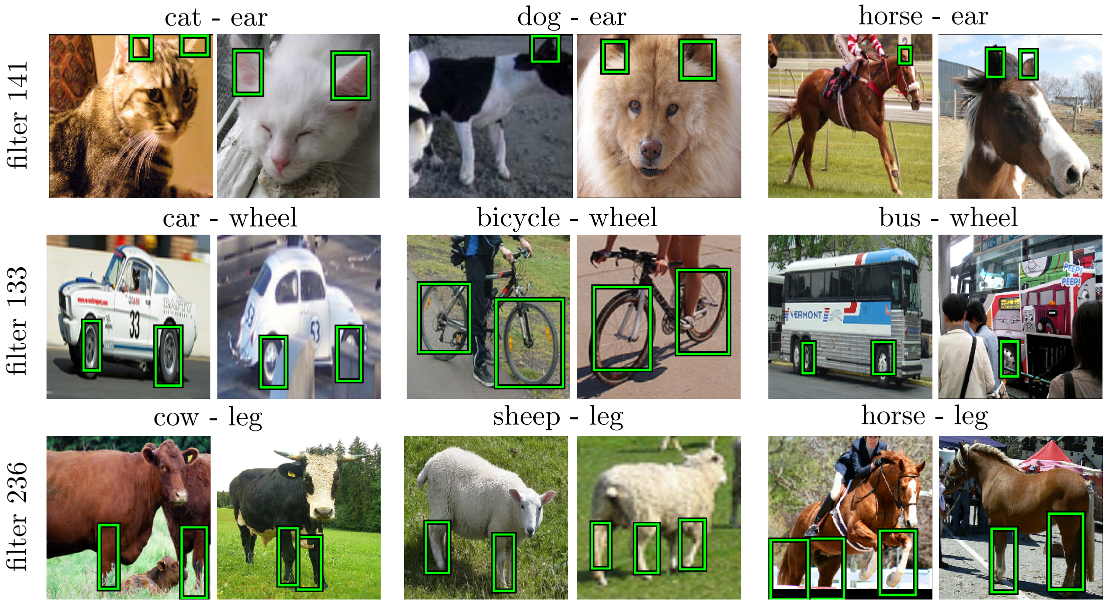


Figure 4.5: *Detections performed by filters 141, 133, and 236 of AlexNet-Object, layer 5. The filters are specific to a part and they work well on several object classes containing it.*

since it has reasonable accuracy. We ran the GA at layer 5 for 10 trials starting with different random seeds. On average, each trial selects 16 filters, with a standard deviation of 2.7. The AP remains very similar for all runs (mean 39.42, standard deviation of 0.03). This indicates that, for the task at hand, our GA is stable, converging to equivalent solutions in terms both of the number of selected filters and their collective performance.

Filter sharing across part classes. We looked into which filters were selected by our GA and noticed that some are shared across different part classes. We then confirmed that those filters have high part detection performance for equivalent part classes across different object classes (e.g. *car-wheel* and *bicycle-wheel*). Fig. 4.5 shows some examples of these filters' detections. It is clear that some filters are representative for a generic part and work well on all object classes containing it.

Instance coverage. Table 4.1 shows high AP results for several part classes, showing how some filters can indeed act as part detectors. However, as AP conflates both recall and precision, it does not reveal how many part instances the filters cover. To answer this question, fig. 4.6 shows precision vs. recall curves for several part classes. In order to create these curves, we sort detections by score and select subsets from the top until reaching a particular recall point. Then, for each recall level we compute the precision of the detections in the corresponding subset as the number of true positives

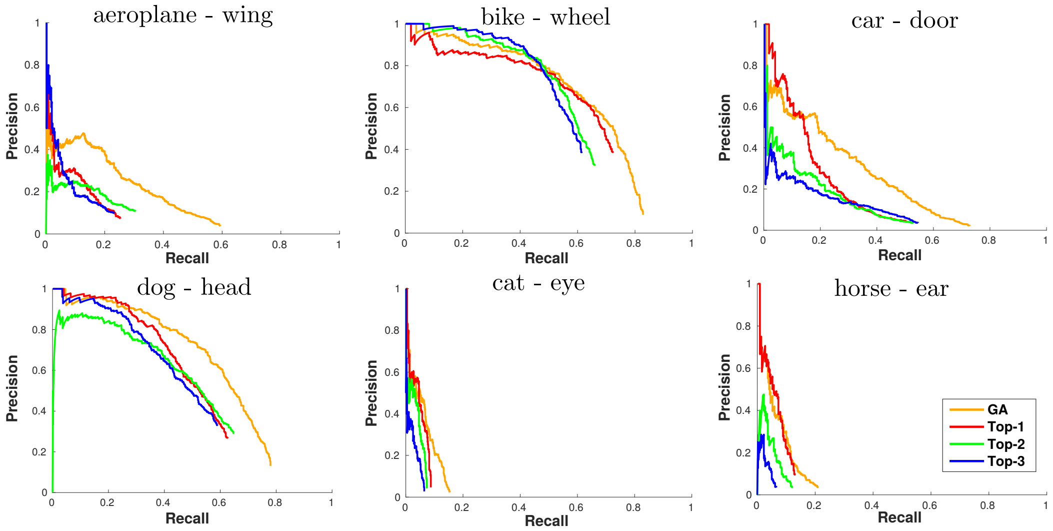


Figure 4.6: *Precision vs. recall curves for six part classes using AlexNet-Object's layer 5 filters. For each part class we show the curve for the the top three individually best filters and for the combination of filters selected by our GA.*

divided by the number of total detections in the subset. For each part class, we take the top three filters of layer 5, and compare them to the filter combination returned by the GA. We can see how the combination reaches higher AP not only by having higher precision (fewer false positives) in the low recall regime, but also by reaching considerably higher recall levels than the individual filters. For some part classes, the filter combination covers as many as 80% of its instances (e.g. *car-door, bike-wheel, dog-head*). For the more challenging part classes, neither the individual filters nor the combination achieve high recall levels, suggesting that the convolutional filters have not learned to respond to these parts systematically (e.g. *cat-eye, horse-ear*).

How many semantic parts emerge in AlexNet-Object? So far we discussed part detection performance for individual filters of AlexNet-Object and their combinations. Here we want to answer the main bottomline question: for how many part classes does a detector emerge? We answer this for two criteria: AP and instance coverage.

For AP, we consider a part to emerge if the detection performance for the best filter combination in the best layer (L5) exceeds 30 percent AP. This is a rather generous threshold, which represents the level above which the part can be somewhat reliably detected. According to this criterion, 34 out of the 105 semantic part classes emerge. This is a modest number, despite all favorable conditions we have engineered into the evaluation and all assists we have given to the network (including bounding-box regression and optimal filter combinations).

According to the instance coverage criterion, instead, results are more positive. We consider that a filter combination covers a part when it reaches a recall level above 50%, regardless of false-positives. According to this criterion, 71 out of the 105 part classes are covered, which is greater than the number of part detectors found according to AP. This indicates that, although for many part classes there is a filter combination covering many of its instances, it also fires frequently on other image regions (leading to high false positive rates).

Based on all this evidence, we conclude that the network does contain filter combinations that can cover some part classes well, but they do not fire exclusively on the part, making them weak part detectors. This demystifies the observations drawn through casual visual inspection, as in Zeiler and Fergus (2014). Moreover, the part classes covered by such semantic filters tend to either cover a large image area, such as torso or head, or be very discriminative for their object class, such as wheels for vehicles and wings for birds. Most small or less discriminative parts are not represented well in the network filters, such as headlight, eye or tail.

4.3.3 Other network architectures and levels of supervision

We now explore how the level of supervision provided during training and the network architecture affect what the filters learn.

Networks and training. We consider several additional networks with different supervision levels (*AlexNet-Image*, *AlexNet-Scenes*, and *AlexNet-Object←Scratch*), and a different architecture (*VGG16-Object*).

AlexNet-Image (Krizhevsky et al., 2012) is trained for image classification on 1.3M images of 1000 object classes in ILSVRC 2012 (Russakovsky et al., 2015). Note how this network has not seen object bounding-boxes during training. For this reason, we expect its filters to learn less about semantic parts than AlexNet-Object. On the opposite end of the spectrum, AlexNet-Scene (Zhou et al., 2014) is trained for scene recognition on 205 categories of the Places database (Zhou et al., 2014), which contains 2.5M scene-centric images. As with AlexNet-Image, this network has not seen object bounding-boxes during training. But now the training images show complex scenes composed of many objects, instead of focusing on individual objects as in ILSVRC

Network name	**Training**		**Results - Layer**		
	Pre-train	Train	**L3**	**L4**	**L5**
AlexNet-Object	ILSVRC12	VOC12	19.1	21.1	24.2
AlexNet-Image	-	ILSVRC12	20.0	21.5	23.2
AlexNet-Scene	-	Places	17.5	17.6	18.1
AlexNet-Object←Scratch	-	VOC12	14.5	16.2	18.2
VGG16-Object	ILSVRC12	VOC12	12.9	21.5	26.1
VGG16-Image	-	ILSVRC12	12.4	19.6	22.1

Table 4.2: *Part detection results (mAP). For VGG-16, L3, L4, and L5 correspond to L3_3, L4_3, and L5_3, respectively.*

2012. Moreover, while the network might learn to use objects as cues for scene recognition (Zhou et al., 2015), the task also profits from background patches (e.g. *water* and *sky* for beach). For these reasons, we expect object parts to emerge even less in AlexNet-Scene. For both AlexNet-Image and AlexNet-Scene we use the publicly available models from Jia (2013).

We introduce AlexNet-Object←Scratch in order to assess the importance of pre-training in AlexNet-Object. We directly train this network for object detection on PASCAL VOC 2012 from scratch, i.e. randomly initializing its weights instead of pre-training on ILSVRC 2012. The rest of the training process remains identical to the one for AlexNet-Object (sec. 4.3.2.1).

Finally, VGG16-Object is the 16-layer network of Simonyan and Zisserman (2015), finetuned for object detection (Girshick et al., 2014) on PASCAL VOC 2012 (like AlexNet-Object). While its general structure is similar to AlexNet, it is deeper and the filters are smaller (3x3 in all layers), leading to better image classification (Simonyan and Zisserman, 2015) and object detection (Girshick, 2015) performance. Its convolutional layers can be grouped in 5 blocks. The first two blocks contain 2 layers each, with 64 and 128 filters, respectively. The next block contains 3 layers of 256 filters. Finally, the last 2 blocks contain 3 layers of 512 filters each.

Results. Table 4.2 presents results for all networks we consider. For the AlexNet architectures, we focus on the last three convolutional layers, as we observed in sec. 4.3.2.2 that filters in the first two layers correspond poorly to semantic parts. Analogously, for

VGG16-Object we present the top layer of each of the last 3 blocks of the network (L3_3, L4_3, and L5_3). We report mAP results obtained by the GA filter combination, averaged over all part classes.

Both AlexNet-Image and AlexNet-Object present reasonable part emergence across all their layers. This shows that the network's inclination to learn semantic parts is somewhat already present even when trained for whole image classification, suggesting that object parts are useful for that task too. However, the part emergence on L5 for AlexNet-Object is higher. This indicates that parts become more important when the network is trained for object detection, affecting particularly higher layers, near the final classification layer that can use the responses of these filters to recognize the object.

Interestingly, parts emerge much less when training the network for scene recognition, as the results of AlexNet-Scene indicate (-6.1% mAP compared to AlexNet-Object). The relative performance of the three networks AlexNet-Object, AlexNet-Image, AlexNet-Scene suggest that the network seems to learn parts to the degree it needs them for the task it is trained for, a remarkable behaviour indeed.

AlexNet-Object←Scratch performs clearly worse than AlexNet-Object, which is likely due to the fact that PASCAL VOC 2012 is too small for training a complex CNN, and so pre-training on ILSVRC is necessary (Agrawal et al., 2014; Girshick et al., 2014). Finally, parts emerge more in the deeper VGG16-Object than in AlexNet-Object (L5). The higher part emergence could be due to the better performance of this network or its greater depth. In order to investigate this, we also test VGG16-Image: VGG16 architecture just pre-trained for image classification, like AlexNet-Image. This model preserves the depth but loses the performance advantage over the fine-tuned counterparts. The results are similar to AlexNet-Image for layers 4 and 5, suggesting that the better performance is indeed responsible for the higher part emergence. However, there is still correlation with depth as better networks tend to be deeper.

All the networks but AlexNet-Scene confirm the trend observed for AlexNet-Object: filters in higher layers are more responsive to semantic parts. This is especially notable for VGG16-Object. As this network has many more layers, the levels of semantic abstraction are more spread out. For example, L3_3 has very low emergence as there are six more layers above it instead of two in AlexNet. Therefore, the network can postpone the development of semantic part filters to later layers. AlexNet-Scene displays the same, rather low level of responsiveness to semantic parts in all layers considered. We hypothesize this is due to semantic object parts playing a smaller role for scene recognition.

4.4 Semantic part emergence in CNNs according to humans

Our quantitative evaluation presented in sec. 4.3 uses the semantic part annotations available in PASCAL-Part dataset (Chen et al., 2014) to determine how many semantic parts emerge in the CNNs. We address now the converse question: *"what fraction of all filters respond to any semantic part?"* Despite being the best existing parts dataset, PASCAL-Part is not complete: some semantic parts are not annotated (e.g. the door handle of a car). For this reason, we cannot answer this new question using it, as a filter might be responding to an unannotated semantic part.

We propose here a human-based experiment that goes beyond the semantic parts annotated in PASCAL-Part. For each object class we ask human annotators if activations of a filter systematically correspond to a semantic part of that object class, and, if yes, to name the part (sec. 4.4.1). This data provides a mapping from filters to semantic parts, which is only limited by the semantic parts known by the annotator. Using this mapping, we can now answer the proposed question (sec. 4.4.2). Moreover, this mapping also allows us to compare the parts emerging in this human experiment with the parts that emerged according to the PASCAL-Part annotations(sec. 4.3). This enables to discern whether some other parts emerge besides those annotated in PASCAL-Part (sec. 4.4.3).

4.4.1 Methodology

For each pair of object class and filter, we present an image like fig. 4.7 to an annotator. The image shows the top 10 activations of the filter on object instances of the class (*car*, in the figure). We show the image shaded and overlay the activation map by setting the transparency value of the shading proportionally to the activation value at a pixel. Since activation maps are smaller than the image, we interpolate the transparency value of each image pixel based on the corresponding closest activations of the map. This highlights the activation map and helps the annotator to quickly see high activation regions in the context of the rest of the image. We also indicate the maximum of the activation map with a green square to emphasize the maximum activation (which is typically in the middle of the region).

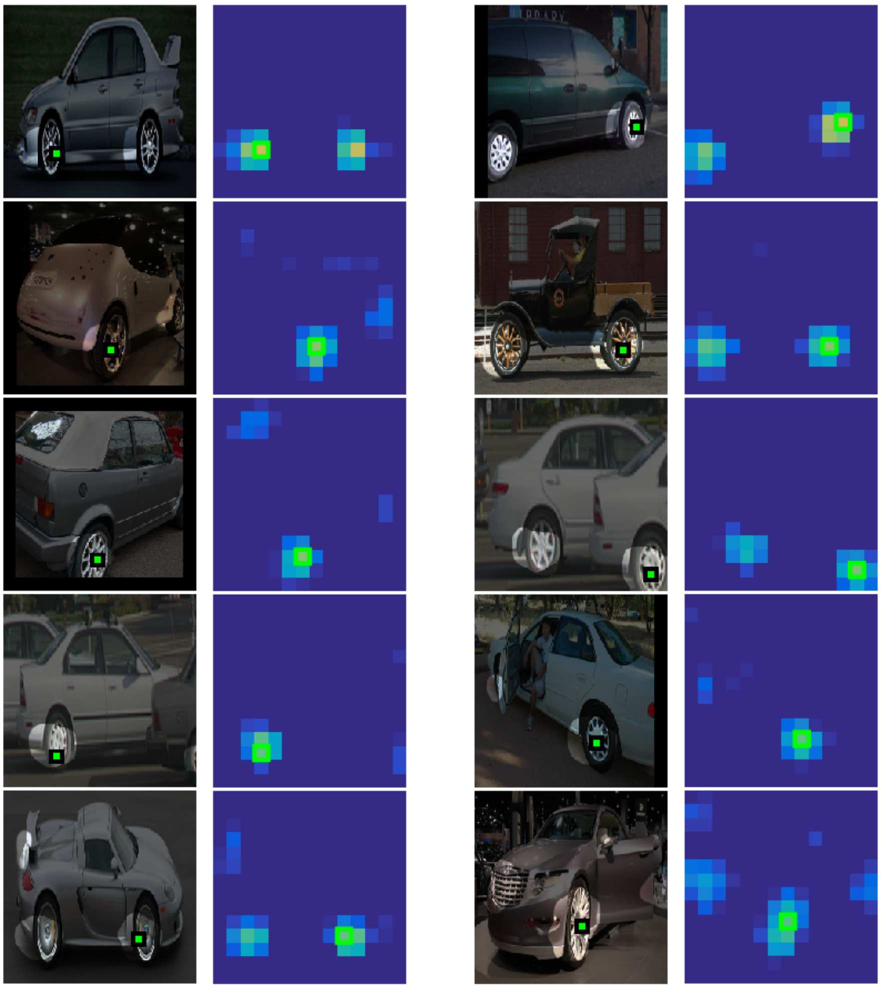


Figure 4.7: *Example of a question shown to our annotators, filter id: 186, class:* car. *We show the top 10 images with the highest activation of the filter on this object class, along with the corresponding activation maps.*

The task consists in answering the question: *"Do the highlighted areas in the images systematically cover the same concept, and if so, which?"*. This is the case if the highlighted areas cover the same concept in at least seven out of the ten images. The possible answers are the following:

1. Yes - Semantic part
2. Yes - Background
3. Yes - Other
4. No
5. Not sure

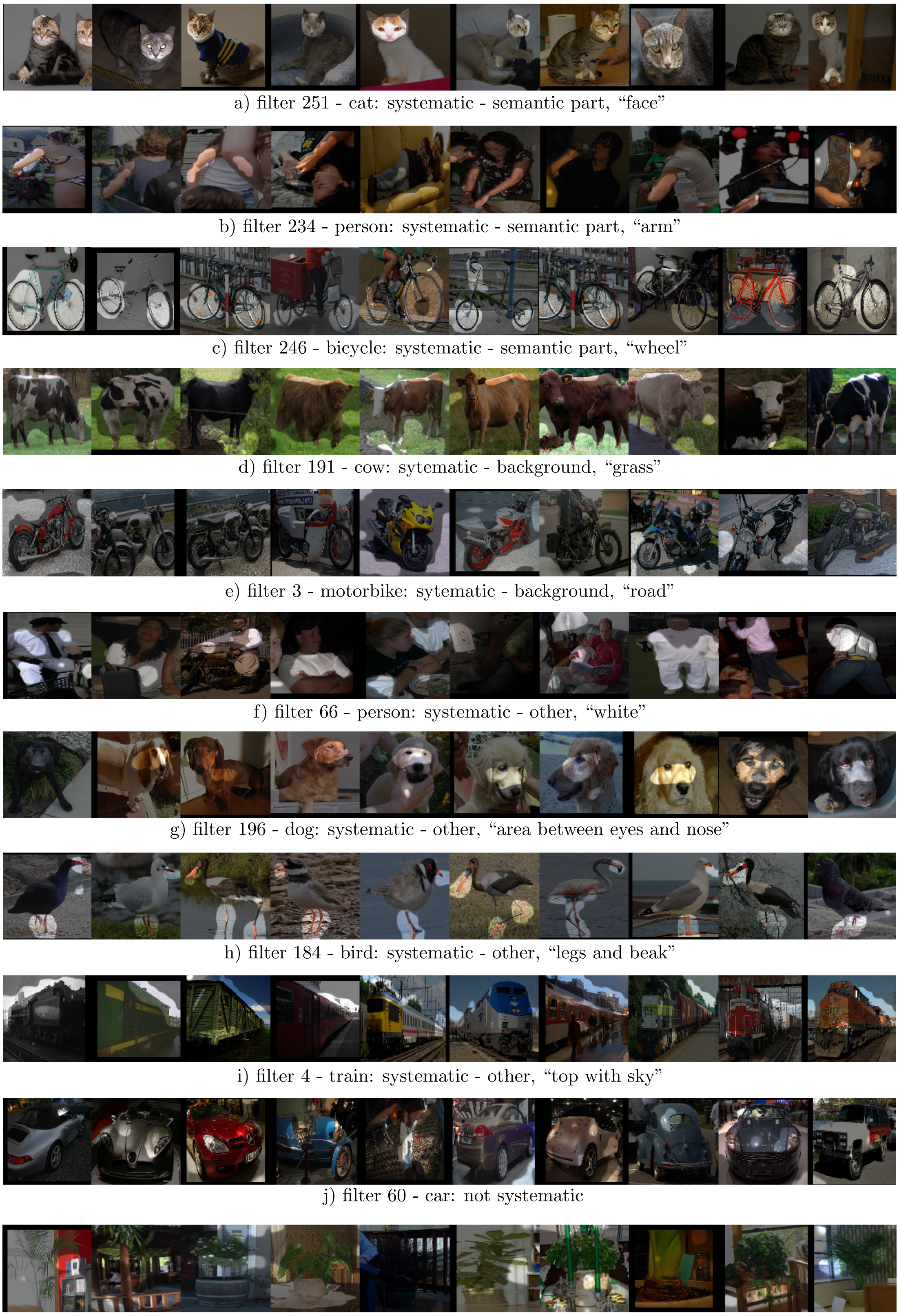
In case of an affirmative response, the annotator needs to specify one of three types of concepts: semantic part (e.g. wheel), background (e.g. grass) or anything else (e.g. white color). Additionally, we ask them to name the concept by typing it in a free-text field. The idea behind this protocol is to distinguish filters that fire on a variety of different image structures (fig. 4.8j-k), including occasionally some semantic parts, from genuine part detectors, which fire systematically on a particular part. Furthermore, we have expanded our experiment beyond semantic parts (i.e. background and other) in order to achieve a more comprehensive understanding of the filter stimuli. The last option ("Not sure") allows the annotator to skip ambiguous cases, which we later reject.

We ask a question for each combination of object class and network filter. We explore L5 filters of AlexNet-Object (256) and we consider the 16 object classes used in sec. 4.3, leading to a total of 256×16 questions. We use two expert annotators to process half of the object classes each. To measure agreement, they also process one of the object classes from the other annotator's set. Their agreement on the types of filters is high: in 79% of the questions, the two annotators clicked on the same answer (out of the 5 possible answers above).

4.4.2 Results

This experiment enables to obtain a distribution over the types of filters for each object class. Fig. 4.9 shows these distributions for three example object classes (*bird*, *car*, and *cat*) as well as the average result for all 16 object classes. Additionally, fig. 4.8 shows some example human answers. The majority of the filters do not seem to systematically respond to any identifiable concept (fig. 4.8j-k). Among the filters that do respond to concepts systematically, only an average of 7% (18 filters) correspond to semantic parts of a particular object class (fig. 4.8a-c). Furthermore, there is signficant overlap of semantic filters across different object classes, as only 131 of all filters are semantic for at least one object class. This confirms what we observed in fig. 4.5.

Only 3% of the filters respond systematically to background patches, examples include "grass", "road", and "sky"(fig. 4.8d-e). Finally, most of the systematic filters, 10% overall, respond to some other concepts. Among these, we most often find colors and textures (fig. 4.8f), but also subregions of a part (fig. 4.8g), assemblies of multiple semantic parts or their subregions (fig. 4.8h), or even regions straddling between a part and a background patch (fig. 4.8i).



a) filter 251 - cat: systematic - semantic part, “face”

b) filter 234 - person: systematic - semantic part, “arm”

c) filter 246 - bicycle: systematic - semantic part, “wheel”

d) filter 191 - cow: sytematic - background, “grass”

e) filter 3 - motorbike: sytematic - background, “road”

f) filter 66 - person: systematic - other, “white”

g) filter 196 - dog: systematic - other, “area between eyes and nose”

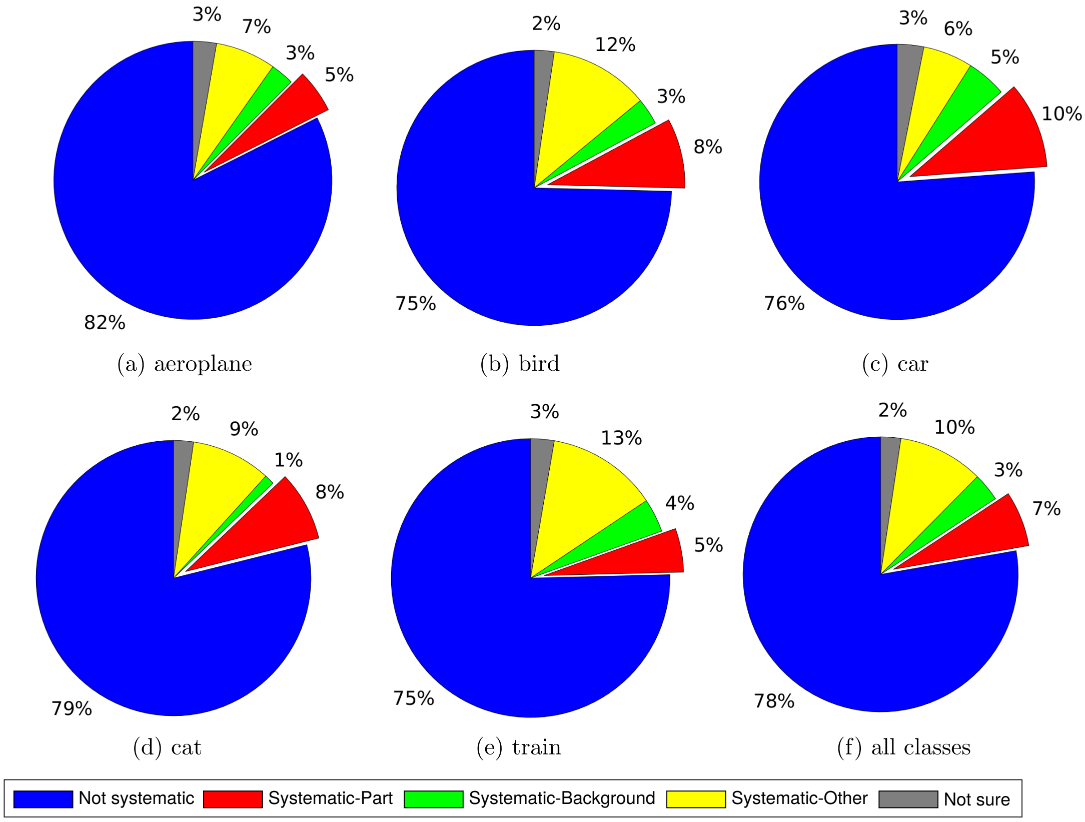
h) filter 184 - bird: systematic - other, “legs and beak”

i) filter 4 - train: systematic - other, “top with sky”

j) filter 60 - car: not systematic

k) filter 161 - pottedplant: not systematic

Figure 4.8: *Examples of human annotation for different filters and classes.*



(a) aeroplane (b) bird (c) car

(d) cat (e) train (f) all classes

Figure 4.9: *Filter distributions for bird, car, cat, and the average of all 16 classes.*

By further inspection, we found that background filters are consistent across images of different object classes. For example, a filter that systematically fires on "grass" patches does it for most classes commonly found outdoors. Similarly, some of the "other" systematic filters, especially the ones responding to colors, also exhibit the same behavior across object classes. In contrast, the situation for systematic filters responding to semantic parts is mixed. Although in some cases a filter responds to similar semantic parts of different object classes (like "wheel" or "leg", as in fig. 4.5), in some other cases this does not hold. For example, there is a filter that responds to "wheel" (in *car* images), "leg" (in *cow* images), and "paw" (in *cat* images). This indicates that the filter is responding to several types of stimuli simultaneously, possibly due to a higher order stimulus of which humans are not aware.

4.4.3 Comparison to PASCAL-Part

In this section we compare the part emergence observed in sec. 4.3 with the part emergence from this human experiment. Moreover, we also look at what semantic parts

aeroplane	nose	dog	forehead
bicycle	frame, hub, spokes, tire, tube	horse	belly, forehead, shoulder
bird	belly	motorbike	rim
bottle	base, finish, neck, shoulder	person	crotch
bus	hood	pottedplant	pot rim, soil
car	fender, grill	sheep	belly
cat	back	train	engine, headlight, window
cow	belly	tv monitor	-

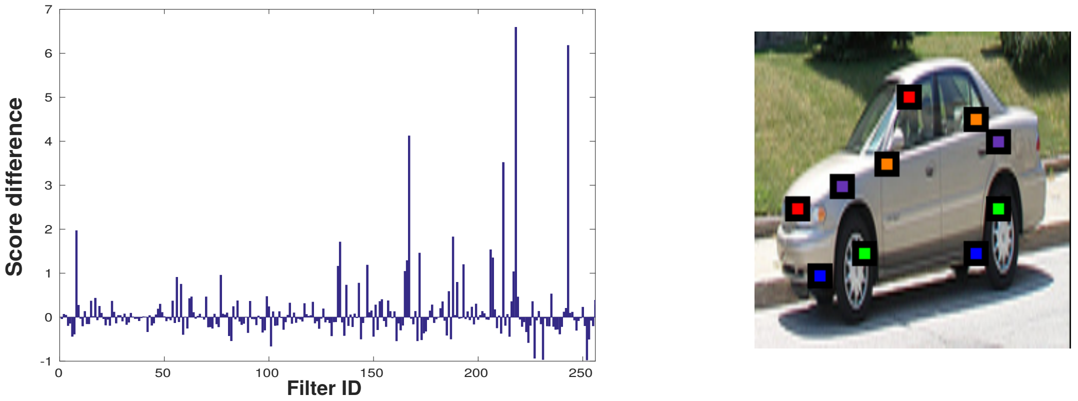
Table 4.3: *List of semantic parts that emerge in AlexNet-Object (layer 5) according to our human experiment, but that are not annotated in the PASCAL-Part dataset.*

emerge according to our annotators, but are not present in PASCAL-Part.

Emergence of PASCAL-Part classes. In sec. 4.3 we observed that 34 out of the 105 semantic parts of PASCAL-Part emerge in AlexNet-Object according to our AP criterion (layer 5, sec. 4.3.2.2). 24 out of these 34 parts also emerge according to the human judgements. Of the missing 10, four are animal *torsos*, for which humans prefer more localized names like "back" and "belly", four are vehicle viewpoints rather than actual semantic parts (e.g. *bus-leftside* and *car-frontside*). The remaining two are *aeroplane-stern* and *bottle-body*. Hence, nearly all of the actual semantic parts that emerged according to detection AP also emerge according to human judgements.

Overall, 59 of the semantic parts annotated in PASCAL-Part emerge according to human judgements. This is substantially more than the 34 that emerged according to detection AP. The reason lies on the fact that it is easier for a filter to count as a semantic part in the human experiment, because it is tested only on the 10 images with the highest activations per object class (fig. 4.7). The AP criterion is more demanding: it takes into account all instances of the part in the dataset, it also counts false-positive detections, and a detection has to be spatially quite accurate to be considered correct (IoU$\geq$ 0.4). This might be a reason why works based on looking at responses on a few images, such as (Zeiler and Fergus, 2014), claimed that filters correspond to semantic parts: they only observed a few strong activations in which this happens. Our analysis goes a step further, by examining how the filters behave over the entire dataset.

Emergence of other semantic part classes. In our human experiment, annotators are free to recognize and name *any* semantic part. Table 4.3 lists the 29 semantic parts



(a) Score difference δ

(b) Discriminative activations

Figure 4.10: *Discriminative filters for object class* car. *(a) Shows how discriminative the filters of AlexNet-Object (layer 5) are for car (higher values are more discriminative). (b) Shows the activations of the five most discriminative filters on an example image.*

that emerge in AlexNet-Object according to our annotators, but that are not annotated in PASCAL-Part. Interestingly, 9 of them concentrate on two object classes: *bicycle* (5) and *bottle* (4). This can be explained by two observations. First, the new parts of the bicycle are mostly sub-parts on the wheel, which is the most discriminative part for the detection of the object (sec. 4.5). And second, the new parts of the bottle are all sub-part of the *bottle-body* part as annotated in PASCAL-Part. As *bottle-body* is essentially the whole object, the network prefers to learn finer-grained, actual parts.

Furthermore, two semantic parts often emerging in animal classes are “belly” and “back”. Their emergence shows again how the network prefers more localized parts, rather than a larger “torso”, as in the PASCAL-Part annotations. Finally, we hypothesize that many of the remaining parts emerge because of their distinctive shapes. For example, *aeroplane-nose* resemble a cone, and *person-crotch* a triangle, *car-fender* a semi-circle and *motorbike-rim* a circle. Moreover, *car-grill* and *train-window* have a characteristic grid pattern.

4.5 Discriminativeness of filters for object recognition

The training procedure of the CNNs we considered maximizes an objective function related to recognition performance, e.g. image classification or object detection. Therefore, the network filters are likely to learn to respond to image patches discriminative for the object classes in the training set. However, these discriminative filters need not correspond to semantic parts. In this section we investigate to which degree the network learns such discriminative filters. We investigate whether layer 5 filters of AlexNet-Object respond to recurrent discriminative image patches, by assessing how discriminative each filter is for each object class. We use the following measure of the discriminativeness of a filter f_j for a particular object class. First, we record the output score s_i of the network on an input image I_i. Then, we compute a second score s_i^j using the same network but ignoring filter f_j. We achieve this by zeroing the filter's feature map x_j, which means $a_{c,r} = 0, \forall a_{c,r} \in x_j$. Finally, we define the discriminativeness of filter f_j as the score difference averaged over the set I of all images of the object class

$$\delta_j = \frac{1}{|I|} \sum_{I_i \in I} s_i - s_i^j. \tag{4.3}$$

In practice, δ_j indicates how much filter f_j contributes to the classification score of the class. Fig. 4.10a shows an example of these score differences for class *car*. Only a few filters have high δ values, indicating they are really discriminative for the class. The remaining filters have low values attributable to random noise. We consider f_j to be a discriminative filter if $\delta_j > 2\sigma$, where σ is the standard deviation of the distribution of δ over the 256 filters in L5. For the car class, only 7 filters are discriminative under this definition. Fig. 4.10b shows an example of the receptive field centers of activations of the top 5 most discriminative filters, which seem to be distributed on several locations of the car. Interestingly, on average over all classes, we find that only 9 out of 256 filters in L5 are discriminative for a particular class. The total number of discriminative filters in the network, over all 16 object classes amounts to 105. This shows that the discriminative filters are largely distributed across different object classes, with little sharing, as also observed by Agrawal et al. (2014). Hence, the network obtains its discriminative power from just a few filters specialized to each class.

In order to further study this, we now measure the collective impact of all discriminative filters, taken together as a set. To do so, we generalize the discriminativeness measure defined in eq. (4.3) to a set of filters instead of just one. This corresponds to simultaneously setting to zero the feature maps of all the filters in the set. We compute it on two different sets of filters: (1) all filters that are not discriminative and (2) all filters that are discriminative. The average discriminativness for (1) is 28.8, indicating that removing all non-discriminative filters has a significant impact on the class scores (for reference, the original average score using all filters is 69.2). This is understandable given the large number of filters removed (around 247 out of 256, as there are only

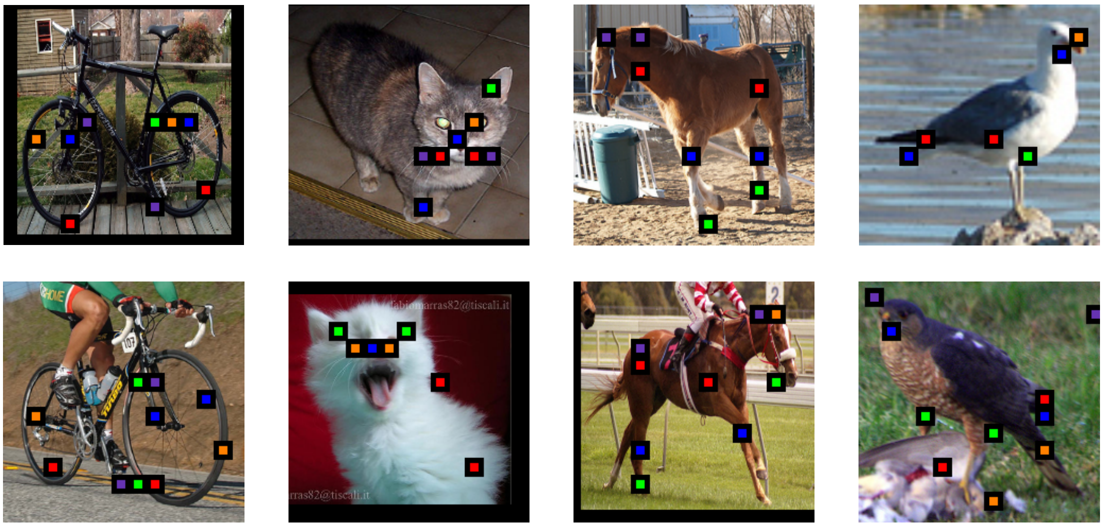
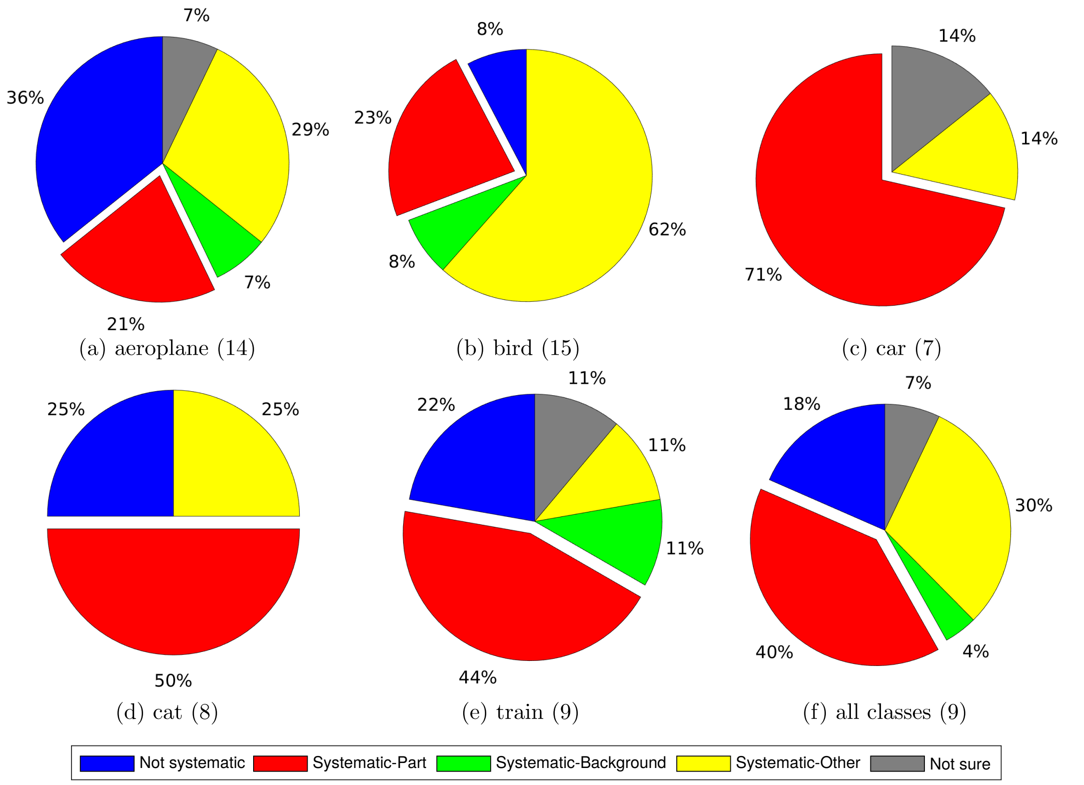


Figure 4.11: *Example activations of the five most discriminative filters for object class* bicyle, cat, horse, bird*, respectively.*

9 discriminative filters per class on average). However, for (2), the discriminativness is much higher: 48.6. Therefore, we can indeed conclude that a few discriminative filters are substantially more influential than all other filters together, as the drop in class scores is greater when these filters are removed.

Fig. 4.11 shows examples for other classes besides *car*, where we can observe some other interesting patterns. For example, wheels are extremely discriminative for class bicycle, in contrast to class car, where discriminative filters are more equally distributed across the whole surface of the object. Since wheels are generally big for bicycle images, some filters specialize to subregions of the wheel, such as its bottom area. Another interesting observation is that the discriminativeness of a semantic part might depend on the object class to which it belongs. For example, class cat accumulates most of its most discriminative filters on parts of the head. Interestingly, Parkhi et al. (2011) observed a similar phenomenon with HOG features, where the most discriminative parts of cats and dogs were found to be the heads. On the other hand, class horse tends to prefer parts of the body, such as the legs, devoting very few discriminative filters to the head. Besides firing on subregions of parts, some discriminative filters fire on assemblies of multiple parts or on a part with some neighboring region (e.g. the red filter for class bird is associated with both wing and tail).

How many discriminative filters are also semantic? We categorize now the found discriminative filters into the filter types defined in sec. 4.4, using the data collected in



(a) aeroplane (14)

(b) bird (15)

(c) car (7)

(d) cat (8)

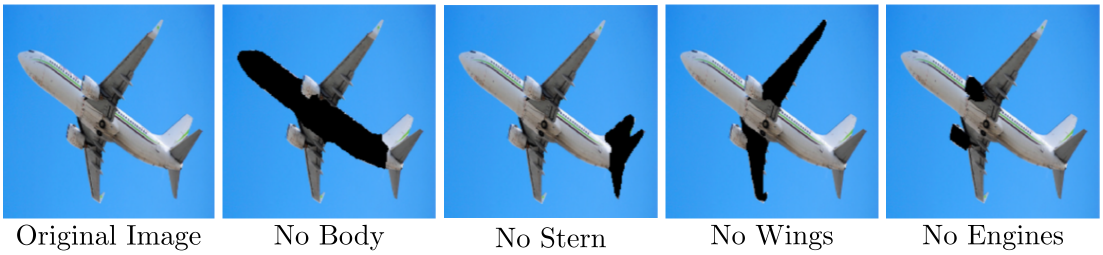
(e) train (9)

(f) all classes (9)

Figure 4.12: *Discriminative filter distributions for bird, car, cat, and for the average of all 16 classes. The number between parenthesis indicates the number of discriminative filters for each case.*

the human experiment. This enables us to determine what fraction of discriminative filters are also semantic, which in turns reveals whether semantic parts are important for recognition. Moreover, as we have defined filter types that go beyond semantic parts, we can obtain a complete list of the filter stimuli that give the network its discriminative power.

Figure 4.12 shows the distribution of discriminative filters over our filter types for three object classes, and the average for all object classes. On average, 40% of the discriminative filters are also semantic, which translates in about 4 out of the 9 filters that are discriminative for each object class, on average. This is a very high fraction, considering that we found only 7% of all filters to be semantic (fig 4.9). This clearly indicates that the network is using semantic parts as powerful discriminative cues for recognizing object classes. Additionally, about 4% of the filters systematically respond to background patches, and another 30% of the filters systematically respond to some other concept (mostly subregions or assemblies of parts). Finally, 18% of the



Original Image No Body No Stern No Wings No Engines

Figure 4.13: *Examples of input images for the network. The left-most shows the original airplane, while the others shows the same airplane, but with parts blacked out.*

filters do not correspond to any concept, a massive drop compared to the 78% statistics over all filters (fig 4.9). This distribution confirms our intuition drawn through visual inspection (fig. 4.11): filters that discriminate for the network are often stimulated by some semantic part, but also by other discriminative patches such as subregions of parts.

4.6 Discriminativeness of semantic parts for object recognition

In the previous section we investigated how discriminative each filter is for each object class. In this section, instead, we investigate the discriminativeness of semantic parts. We look at how much each part contributes to the classification score of its object class. We measure discriminativeness as in sec. 4.5, but instead of ignoring a specific filter, we now ignore a semantic part. We use the same formulation of eq. (4.3), but with different meanings for j, I and s_i^j. Given a semantic part j, I now indicates all images containing j, and s_i^j is the score given by the network to image I_i with part j blacked out. We use the segmentation masks available in PASCAL-Part to set to zero all the pixels of a part j in each input image I_i, after pre-processing by subtracting the image mean (sec. 4.3.2.1). In this way, part j is ignored and does not contribute to the classification score of the object, as all convolutional filters output 0 on blacked out regions. The network can only rely on information from the rest of the image. If it is no longer confident about the prediction of the object class, it means that the blacked out part is discriminative for it. We note that even if the part is blacked out, its boundary remains accessible to the network and can contribute some discriminative information.

We evaluate the 105 semantic parts of PASCAL-Part (sec. 4.3.2.1). Fig. 4.14 shows results for some examples parts of 9 object classes. Interestingly, similar classes do not necessarily have similar discriminative parts. For example, the most discriminative parts for class car are *door* and *wheel*. But these are not very important for class bus, which is largely discriminated by the part *window*. Moreover, *torso* is very discriminative for some animals (e.g. horse and cow), but less so for others (e.g. cat and dog). We offer two explanations for this phenomenon. First, the network seems to consider discriminative parts that are clearly visible across many instances of the object class. For example, the wheels of a car are often visible in PASCAL images, while the wheels of a bus are often occluded in the PASCAL dataset. Similarly, many images of pet animals (e.g. cat and dog) are biased towards close-ups (often occluding the body), while images of other animals typically show the whole body (e.g. horse, cow, bird). Second, the network seems to consider more discriminative parts that have lower intra-class variation. For example, the torso is very similar across all horses, while the torso of a dog varies considerably depending on breed and size.

We also observe a strong relation between these quantitative results in fig. 4.14 and the visual results of fig. 4.11. The top most discriminative filters activate on the most discriminative parts. For example, the only discriminative semantic part for bicycle is *wheel*, which is exactly where all the activations of the most discriminative filters are. Analogously, *head* is very discriminative for the class cat, *wing* is discriminative for bird and *leg* is somehow discriminative for *horse*.

Correlation to average precision and part size. In this paragraph we look at the correlation between the discriminativeness of a semantic part, the average size of a part and the detection performance results of sec. 4.3.2.2 (AP in table 4.1, GA, layer 5). Results are shown in fig. 4.15. Two interesting facts emerge. First, discriminativeness tends to increase with the average size of a part (very high PPMCC of 0.87) and second, discriminativeness correlates with how much parts emerge in the CNNs according to AP detection performance (PPMCC of 0.65). These are important correlations that support our analysis of sec. 4.3, where we observed that CNNs learn only a few of the semantic parts in their internal representation.

Finally, note how future works that use filters as intermediate part representations (as in Simon et al. (2014); Gkioxari et al. (2015); Simon and Rodner (2015); Xiao et al. (2015); Oquab et al. (2015)) will now be able to exploit these findings to create better models for recognition.

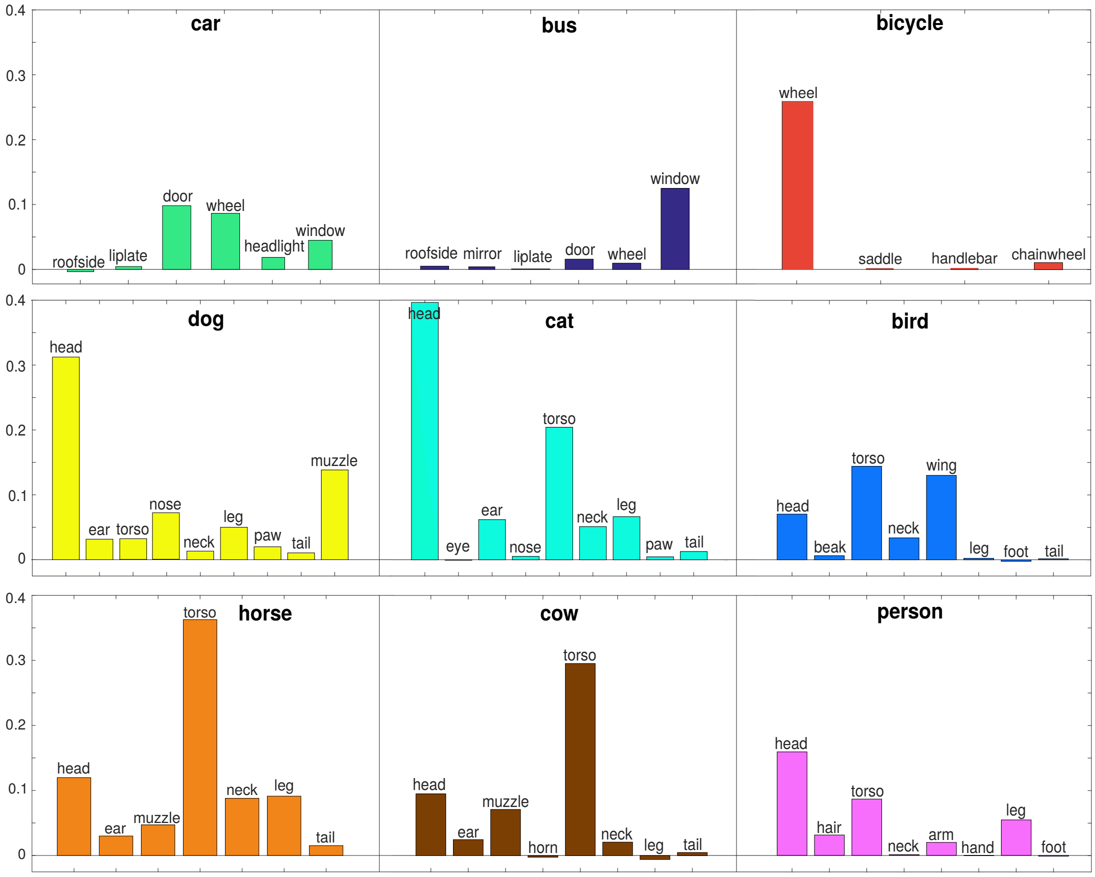


Figure 4.14: *Discriminativeness of PASCAL-Part for the classification of their objects. The vertical axis indicates the difference δ between the classification score for the original image, and the score for the image after blacking out the part. We report averages over all images in an object class (higher values mean more discriminative).*

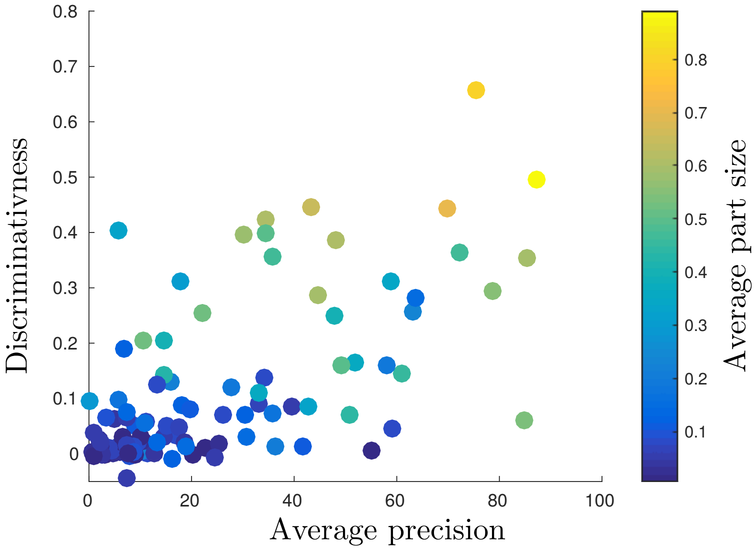


Figure 4.15: *Correlation between the discriminativeness of a semantic part, the average precision results of sec. 4.3.2.2 (mAP, GA, layer 5, table 4.1) and the average size of a part. The latter is normalized by the average size of its object. Each point corresponds to a different part. Interestingly, these measures are all correlated.*

4.7 Conclusions and outlook

We have analyzed the emergence of semantic parts in CNNs. We have investigated whether the network's filters learn to respond to semantic parts. In order to do so, we have associated filter stimuli to semantic parts, using two different quantitative evaluations. In the first one, we have used ground-truth part bounding-boxes to determine how many parts emerge in the CNN for different layers, network architectures and supervision levels. Despite promoting this emergence by providing favorable settings and multiple assists, we found that only 34 out of 105 semantic parts in PASCAL-Part dataset (Chen et al., 2014) emerge in AlexNet (Krizhevsky et al., 2012) fine-tuned for object detection (Girshick et al., 2014). This result complements previous works (Zeiler and Fergus, 2014; Simonyan et al., 2014) by providing a more accurate, quantitative assessment and shows how the network learns to associate filters only to some part classes. In the second one, we study how many filters systematically respond to semantic parts for each object class. We found that, on average, 7% of the filters respond to semantic parts, whereas 13% systematically respond to other concepts, such as subregions of parts or background patches. This filter characterization provides a more precise understanding of the internal representations learned by CNN architectures. Finally, we have studied how discriminative network filters and semantic parts are for the task of object recognition. The overlap between discriminative and semantic filters adds further insights into claims made by works based on qualitative inspection (Zeiler and Fergus, 2014; Simonyan et al., 2014).

In the following, we discuss some possible future work directions to extend or apply the analysis presented in this chapter.

Anti-discriminativeness. We have addressed the question of which filters are discriminative for the recognition of a particular object class. Using our own designed 'discriminativeness' measure, we found that most of the filters seem to have a negligible effect on the correct class score, and only a few can be considered discriminative. However, analogously to how discriminative filters affect negatively the correct class score when removed, some filters result in an increase of such score when not taken into account. This is a strange phenomenon that requires further research to be truly understood. We hypothesize that it could be due to filters firing on patterns that are shared across different classes, but with a slight preference for some of them. For example, a filter may fire on a particular ear shape that belongs to both cats and dogs, but that is much more common on cats than on dogs. When an instance of a dog with

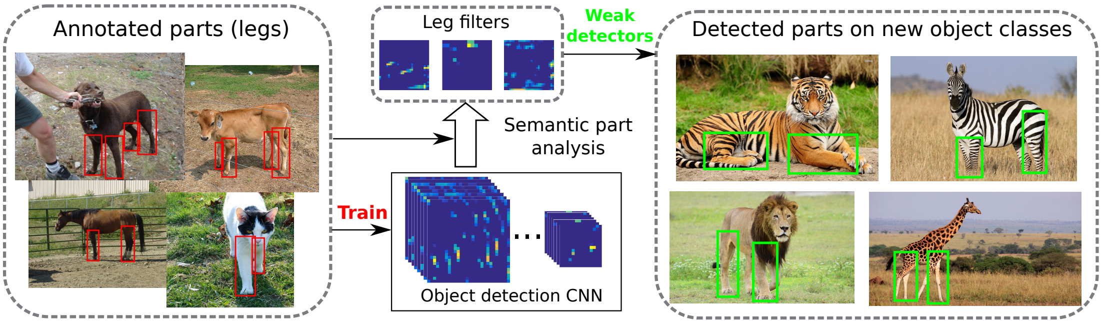


Figure 4.16: *Proposed part transfer scheme. Our semantic part analysis can identify a set of filters that respond highly to a part, such as legs in this example. Those filters may be used as weak detector to obtain rough detections of the semantic part on new object classes.*

such ears is presented, said filter obtains a high response and this causes the score of the class cat to be very high. If this filter is ignored, the network is not mislead towards believing it is a cat, and thus the score for cat decreases while the score for dog correctly increases. The analysis proposed here belongs to a recent and expanding research direction that attempts to obtain a deeper understanding of the error modes of CNNs, as well as the differences with errors committed by humans (Nguyen et al., 2015; Goodfellow et al., 2015; Moosavi-Dezfooli et al., 2016, 2017).

On an orthogonal direction, we could leverage the gained knowledge about 'anti-discriminative' filters to improve recognition accuracy. To do this, we would determine which filters are anti-discriminative for each of the object classes, and accordingly disable them at test time. Since those filters tend to negatively affect the class score, an adaptive architecture that automatically ignores them would theoretically lead to more accurate classifications. Furthermore, we could develop an even more powerful architecture that weighs how much each filter contributes to the final score based on its discriminativeness. The discriminateveness of each filter could be measured on a validation set after the network finishes training.

Transferring learned parts. Our analysis concluded that *some* filters do respond to particular semantic parts, mostly those that are big or discriminative. Moreover, some of such filters respond to equivalent semantic parts across different object classes. For example, filter 236 of layer 5 of AlexNet-Object responds to legs of cows, sheep, and horses (fig. 4.5). This provides an excellent opportunity to learn to recognize equivalent parts on similar object classes, given part annotations for a particular set of object classes. As observed in our analysis, such detectors would be rather weak. However, they would still be able to roughly localize part instances on new object classes. Continuing with our leg example, all filters that significantly respond to animal legs (including filter 236) could become weak detectors for legs of zebras, giraffes, lions, or tigers, despite the lack of part annotations for such object classes (fig. 4.16). Furthermore, the learned part detectors could be useful for other computer vision tasks, such as image retrieval. Our suggested part transfer scheme would enable answering queries such as 'retrieve all quadrupeds' or 'retrieve all objects with wheels', even if the target images contain object classes unseen at training time.

Learning effective context based on network discriminativeness. This chapter focuses on regions inside the object, as we limit the network input to a bounding-box around the object. Inevitably, such bounding-boxes often contain background regions, as the object does not occupy their whole area. Interestingly, the human experiment revealed how a significant amount of filters also fire in these background regions. In the light of the conclusions drawn from the analysis, this indicates that background must be occasionally used for the discrimination of object classes. We propose here extending the scope of our analysis to delve deeper on the discriminativeness of background regions.

This new analysis would provide characterizations of which types of background are particularly discriminative for a specific object class. For example, we can determine how much the network relies on road-like regions for car classification. This knowledge would enable the development of stronger object detectors that integrate context in a tailored way for each object class. We could learn the specific shape and extent of effective contextual regions for each class and pool it along the object appearance, analogously to how we pool the appearance of the object for our part detection approach (chapter 5). This is similar to the work of Mottaghi et al. (2014), who already implemented an object detector that integrates selected background information around the object. However, their approached is based on DPM and is very fixed, always selecting rectangular regions at somewhat stable locations around the object. The approach proposed here goes a step further by suggesting the use of class-specific free-form regions around each object instance, based on what the network relies on for correct recognition.

What do generative neural networks learn? The analysis in this chapter focuses on determining whether a CNN trained for object detection learns semantic parts. Such a CNN is a discriminative model, whose task is classifying each input into a finite set of categories. Recently, generative neural network models are gaining popularity given the success of the Generative Adversarial Networks (GAN) of Goodfellow et al. (2014). A GAN contains two networks, a generator and a discriminator. The generator tries to output images that imitate samples of the underlying distribution provided during training. The discriminator, on the other hand, tries to distinguish whether a sample belongs to the real distribution or it has been created by the generator. A possible way of expanding our deep networks analysis consists in characterizing what the generator filters learn when they generate new images. For instance, do filters still respond to semantic parts? Our conclusions indicate that the learned semantic parts were relevant for discrimination. It is then natural to ask whether semantic parts are still needed for a generative task. Given the success of GANS in various tasks such as style transfer (Li and Wand, 2016), image-to-image translation (Isola et al., 2017) or image super-resolution (Ledig et al., 2017), a deeper understanding of how these architectures work would be highly beneficial for creating more effective models, or even extending GANs to other tasks.

Chapter 5

Objects as context for part detection

5.1 Introduction

Chapter 4 presented an extensive analysis regarding the emergence of semantic parts in the internal representation of an object detection CNN, showing how only a few semantic parts actually emerge and only in a weak manner. Therefore, in order to obtain precise localizations for all part classes, an approach specialized to part detection is necessary. We address such task in this chapter, where we present a dedicated part detection approach that modifies the internal structure of a CNN to exploit the unique nature of this task.

Given the multiple benefits of semantic parts, part-based models have gained attention for tasks such as fine-grained recognition (Zhang et al., 2014a; Lin et al., 2015; Zhang et al., 2016; Parkhi et al., 2012), object class detection and segmentation (Chen et al., 2014; Wang et al., 2015b), articulated pose estimation (Liu et al., 2014; Sun and Savarese, 2011; Ukita, 2012; Yang et al., 2016), and attribute prediction (Gkioxari et al., 2015; Vedaldi et al., 2014; Zhang et al., 2013). Moreover, part localizations deliver a more comprehensive image understanding, enabling reasoning about object-part interactions in semantic terms. Nonetheless, many part-based models detect each part based only on their local appearance, using simple techniques that were originally designed for object detection (Parkhi et al., 2012; Chen et al., 2014; Zhang et al., 2013). Furthermore, some other works use even simpler approaches relying on the assumption that convolutional filters can act as part detectors (Gkioxari et al., 2015; Simon and Rodner, 2015; Xiao et al., 2015).

Parts are highly dependent on the objects that contain them. Hence, objects provide valuable cues to help detecting parts, creating an advantage over detecting them

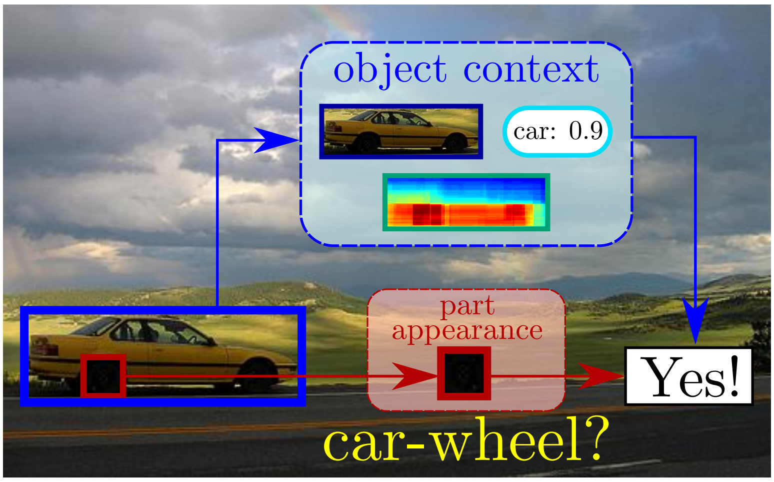


Figure 5.1: *Motivation for our model. Part appearance alone might not be sufficiently discriminative in some cases. Our model uses object context to resolve ambiguities and help part detection.*

independently. First, the class of the object gives a firm indication of what parts should be inside it, i.e. only those belonging to that object class. For example, a dark round patch should be more confidently classified as a wheel if it is on a car, rather than on a dog (fig. 5.1). Furthermore, by looking at the object appearance we can determine in greater detail which parts might be present. For example, a profile view of a car suggests the presence of a car door, and the absence of the licence plate. This information comes mostly through the viewpoint of the object, but also from other factors, such as the type of object (e.g. van), or whether the object is truncated (e.g. no wheels if the lower half is missing). Second, objects also provide information about the location and shape of the parts they contain. Semantic parts appear in very distinctive locations within objects, especially given the object appearance. Moreover, they appear in characteristic relative sizes and aspect ratios. For example, wheels tend to be near the lower corners of car profile views, often in a square aspect ratio, and appear rather small.

In this work, we propose a dedicated part detection model that leverages all of the above object information. We start from a popular CNN detection model (Girshick, 2015), which considers the appearance of local image regions only. We extend this model to incorporate object information that complements part appearance by providing context in terms of object appearance, class and the relative locations of parts within the object.

We evaluate our part detection model on all 16 object classes in the PASCAL-Part dataset (Chen et al., 2014). We demonstrate that adding object information is greatly beneficial for the difficult task of part detection, leading to considerable performance improvements. Compared to a baseline detection model that considers only the local appearance of parts, our model achieves a +5 mAP improvement. We also compare to methods that report part localization in terms of bounding-boxes (Chen et al., 2014; Gkioxari et al., 2015; Zhang et al., 2014a; Lin et al., 2015; Zhang et al., 2016) on PASCAL-Part and CUB200-2011 (Wah et al., 2011b). We outperform (Chen et al., 2014; Gkioxari et al., 2015; Zhang et al., 2014a; Lin et al., 2015) and match the performance of (Zhang et al., 2016). We achieve this by an effective combination of the different object cues considered, demonstrating their complementarity. Moreover our approach is general as it works for a wide range of object classes: we demonstrate it on 16 classes, as opposed to 1-7 in (Chen et al., 2014; Parkhi et al., 2012; Wang et al., 2015b; Gkioxari et al., 2015; Zhang et al., 2014a,b; Lin et al., 2015; Zhang et al., 2016; Liu et al., 2014; Hariharan et al., 2015; Sun and Savarese, 2011; Liang et al., 2016a; Ukita, 2012; Yang et al., 2016; Xia et al., 2016; Gavves et al., 2013; Goering et al., 2014; Zhang et al., 2013; Simon and Rodner, 2015; Xiao et al., 2015) (only animals and person). Finally, we perform fully automatic object and part detection, without using ground-truth object locations at test time (Chen et al., 2014; Wang et al., 2015b; Lin et al., 2015; Zhang et al., 2016).

This chapter will be submitted to CVPR 2018 and it is currently available as a preprint (Gonzalez-Garcia et al., 2017b). The code of our method has been released and it is available at `http://github.com/agonzgarc/parts-object-context`.

The chapter is organized as follows. We introduce related work in the next section. Section 5.3 presents our method and describes all its components. Section 5.4 specifies the implementation details. We present all results in section 5.5, including a validation of our model (sec. 5.5.1) and comparisons to other methods (sec. 5.5.2). Finally, section 5.6 summarizes the conclusions and presents an outlook.

5.2 Related work

Part-based models. Chapter 2 introduced the importance of part-based models in various tasks such as object detection (Chen et al., 2014; Endres et al., 2013; Felzenszwalb et al., 2010b; Tsai et al., 2015) and segmentation (Wang and Yuille, 2015), fine-grained recognition (Liu et al., 2012; Parkhi et al., 2012; Zhang et al., 2013; Simon and Rodner, 2015; Huang et al., 2016; Zhang et al., 2016), human pose-estimation (Liu et al., 2014;

Sun and Savarese, 2011; Ukita, 2012; Yang et al., 2016; Bulat and Tzimiropoulos, 2016), head and face detection (Yang et al., 2015; Vu et al., 2015), attribute prediction (Zhang et al., 2013, 2014b; Vedaldi et al., 2014), action classification (Gkioxari et al., 2015) and scene classification (Juneja et al., 2013). We propose here a method to improve semantic part detection by exploiting the context provided by their objects. By semantic part we refer to those object parts that are interpretable and nameable by humans.

The part-based model of Azizpour and Laptev (2012) is especially related to our work as they also simultaneously detect objects and their semantic parts. However, the focus of Azizpour and Laptev (2012) is object detection, with part detection being only a by-product. They train their model to maximize object detection performance, and thus they require parts to be located only roughly near their ground-truth box. This results in inaccurate part localization at test time, as confirmed by the low part localization results reported (Azizpour and Laptev (2012), table 5). Moreover, they only localize those semantic parts that are discriminative for the object class. Our model, instead, is trained for precise part localization and detects all object parts. Finally, Azizpour and Laptev (2012) builds on DPM (Felzenszwalb et al., 2010b), whereas our model is based on modern CNNs and offers a tighter integration of part appearance and object context.

CNN-based semantic part-based models. In recent years, CNN-based representations are quickly replacing hand-crafted features (Dalal and Triggs, 2005; Lowe, 2004) in many domains, including semantic part-based models (Bulat and Tzimiropoulos, 2016; Chen and Yuille, 2014; Gkioxari et al., 2015; Hariharan et al., 2015; Huang et al., 2016; Liang et al., 2016a; Wang and Yuille, 2015; Wang et al., 2015b; Xia et al., 2016; Zhang et al., 2014a). Our work is related to those that explicitly train CNN models to localize semantic parts using bounding-boxes (Gkioxari et al., 2015; Zhang et al., 2014a), as opposed to keypoints (Huang et al., 2016; Simon and Rodner, 2015) or segmentation masks (Hariharan et al., 2015; Liang et al., 2016a; Wang and Yuille, 2015; Wang et al., 2015b; Xia et al., 2016; Yang et al., 2015). Many of these works (Gkioxari et al., 2015; Simon and Rodner, 2015; Huang et al., 2016; Yang et al., 2015) detect the parts used in their models based only on local part appearance, independently of their objects. For example, Gkioxari et al. (2015) train independent weak part detectors that are later combined with an object instance for action and attribute recognition. Huang et al. (2016) first use a CNN to localize parts, and then feeds them to a two-stream classification network to encode both object and part cues for fine-grained recognition. Moreover, they use parts as a means for other tasks, such as object or action and attribute recognition. Therefore, they are not interested in part detection itself.

Using context for recognition. Several of the works presented in section 2.3.1 exploit global image context to help detecting objects inside it (Harzallah et al., 2009; Murphy et al., 2003; Torralba, 2003; Russell et al., 2007; Song et al., 2011; Yang et al., 2014; Modolo et al., 2015; Vu et al., 2015). In analogy, we exploit object context to help detecting parts inside them.

Some other works use relationships between parts to improve the part localization, mostly for human pose estimation. For example, Chen and Yuille (2014) estimate part keypoints and their spatial relationships with a CNN. Similarly, Tompson et al. (2014) propose a sliding-window CNN for part detection that creates heatmaps for human parts, which are later input into a spatial model using a Markov Random Field. The work of Yang et al. (2016) goes even further, creating a new type of network layer for message passing. It models the relationships between human parts and uses them to provide coherent part labelings. Alternatively, the recent work of Bulat and Tzimiropoulos (2016) refines part locations using a 2-component cascaded CNN. Finally, Yang et al. (2015) combine information from several face parts to approach the face detection task.

Several fine-grained recognition works (Gavves et al., 2013; Goering et al., 2014; Zhang et al., 2016) use nearest-neighbors to transfer part location annotations from training objects to test objects. They do not perform object detection, as ground-truth object bounding-boxes are used at both training and test time. Here, instead, at test time we jointly detect objects and their semantic parts.

The most related works to ours are Zhang et al. (2014a) and Wang et al. (2015b), which use object information to refine part detections as a post-processing step. Part-based R-CNN (Zhang et al., 2014a) refines R-CNN (Girshick et al., 2014) part detections by using nearest-neighbors from training samples. Our model, instead integrates object information also within the network, which allows us to deal with several object classes simultaneously, as opposed to only one (Zhang et al., 2014a). Additionally, we refine part detections with a new network module, Offset Net, which is more accurate and efficient than nearest-neighbors. The method of Wang et al. (2015b) is demonstrated only on 5 very similar classes from PASCAL-Part (Chen et al., 2014) (all quadrupeds), and on fully visible object instances from a manually selected subset of the test set (10% of the full test set). Instead, we show results on 105 parts over all 16 classes of PASCAL-Part, using the entire dataset. Moreover, (Wang et al., 2015b) uses manually defined object locations at test time, whereas we detect both objects and their parts fully automatically at test time.

5.3 Method

We define a new detection model specialized for parts which takes into account the context provided by the objects that contain them. This is the key advantage of our model over traditional part detection approaches, which detect parts based on their local appearance alone, independently of the objects (Gkioxari et al., 2015; Simon and Rodner, 2015; Huang et al., 2016; Yang et al., 2015). We build on top of a baseline part detection model (sec. 5.3.1) and include various cues based on object class (sec, 5.3.2.2), object appearance (sec. 5.3.2.3), and the relative location of parts on the object (sec. 5.3.3). Finally, we combine all these cues to achieve more accurate part detections (sec. 5.3.4).

Model overview. Fig. 5.2 gives an overview of our model. First, we process the input image through a series of convolutional layers. Then, the Region of Interest (RoI) pooling layer produces feature representations from two different kind of region proposals, one for parts (red) and one for objects (blue). Each part region gets associated with a particular object region that contains it (sec. 5.3.2.1). Features for part regions are passed on to the part appearance branch, which contains two Fully Connected (FC) layers (sec. 5.3.1). Features for object regions are sent to both the object class (sec. 5.3.2.2) and object appearance (sec. 5.3.2.3) branches, with three and two FC layers, respectively.

For each part proposal, we concatenate the output of the part appearance branch with the outputs of the two object branches for its associated object proposal. We pass this refined part representation (purple) on to a part classification layer and a bounding-box regression layer (sec. 5.3.4).

Simultaneously, the relative location branch (green) also produces classification scores for each part region based on its relative location within the object (sec. 5.3.3). We combine the above part classification scores with those produced by relative location (big + symbol, sec. 5.3.4), obtaining the final part classification scores. The model outputs these and regressed bounding-boxes. Algorithm 4 summarizes this procedure.

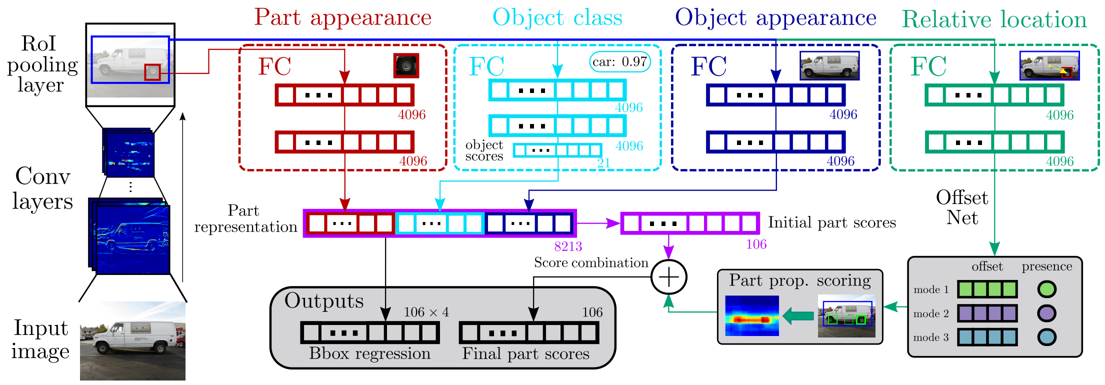


Figure 5.2: *Overview of our part detection model. The model operates on part and object region proposals, passing them through several branches, and outputs part classification scores and regressed bounding-boxes. This example depicts the relative location branch for only object class* car. *In practice, however, it processes all object classes simultaneously. When not explicitly shown, a small number next to the layer indicates its dimension for the PASCAL-Part (Chen et al., 2014) case, with a total of 20 object classes and 105 parts.*

```
Input   : Image $I$, part proposals $P$, object proposals $O$
Output: Set of part detections D
for $p \in P$ do
    // Select supporting object proposal with (5.1)
    $S_{sup}(p) \leftarrow$ supporting_proposal$(p, O)$;
    // Part representation by concatenating net features
    prt_rep$(p) \leftarrow$ prt_app$(p)$ || obj_cls$(S_{sup}(p))$ || obj_app$(S_{sup}(p))$;
    // Obtain initial part scores
    prt_scr$^I(p) \leftarrow$ FC_layer( prt_rep$(p)$) ;
    // Obtain part relative location score
    prt_RL _scr $\leftarrow$ rel_loc$(p)$
    // Combine both for final part score
    prt_scr$^F(p) \leftarrow$ prt_scr$^I(p)+$ prt_RL _scr $(p)$
end
$D \leftarrow$ NMS$(\{(p,$prt_scr$^F(p))\})$;
```

Algorithm 4: Part detection with object context.

5.3.1 Baseline model: part appearance only

As baseline model we use the popular Fast R-CNN (Girshick, 2015), which was originally designed for object detection. It is based on a CNN that scores a set of region proposals (Van de Sande et al., 2011) by processing them through several layers of different types. The first layers are convolutional and they process the whole image once. Then, the RoI pooling layer extracts features for each region proposal, which are later processed by several FC layers. The model ends with two sibling output layers, one for classifying each proposal into a part class, and one for bounding-box regression, which refines the proposal shape to match the extent of the part more precisely. The model is trained using a multi-task loss which combines these two objectives. This baseline corresponds to the part appearance branch in fig. 5.2.

We follow the usual approach (Girshick, 2015) of fine-tuning for the used dataset on the current task, part detection, starting from a network pre-trained for image classification (Krizhevsky et al., 2012). The classification layer of our baseline model has as many outputs as part classes, plus one output for a generic background class. Note how we have a single network for all part classes in the dataset, spanning across all object classes.

5.3.2 Adding object appearance and class

The baseline model tries to recognize parts based only on the appearance of individual region proposals. In our first extension, we include object appearance and class information by integrating it inside the network. We can see this as selecting an adequate contextual spatial support for the classification of each proposal into a part class.

5.3.2.1 Supporting proposal selection

Our models use two types of region proposals (sec. 5.4). *Part proposals* are candidate regions that might cover parts. Analogously, *object proposals* are candidates to cover objects. The baseline model uses only part proposals. In our models, instead, each part proposal p is accompanied by a *supporting object proposal* $S_{sup}(p)$, which must fulfill two requirements (fig. 5.3). First, it needs to contain the part proposal, i.e. at least 90% of p must be inside $S_{sup}(p)$. Second, it should tightly cover the object that contains the part, if any. For example, if the part proposal is on a wheel, the supporting proposal should be on the car that contains that wheel. To achieve this, we select the highest scored proposal among all object proposals containing p, where the score is the object classification score for any object class.

Formally, let p be a part proposal and $S(p)$ the set of object proposals that contain p. Let $\phi_{obj}^{k}(S_n)$ be the classification score of proposal $S_n \in S(p)$ for object class k. These scores are obtained by first passing all object proposals through three FC layers as in the object detector of Girshick (2015). We select the supporting proposal $S_{sup}(p)$ for p as

$$S_{sup}(p) = \underset{S_n \in S(p)}{\operatorname{argmax}} \left[\max_{k \in \{1,\ldots,K\}} \phi_{obj}^{k}(S_n) \right], \tag{5.1}$$

where K is the total number of object classes in the dataset. We give more details about how we extract both types of proposals in sec. 5.4.

5.3.2.2 Object class

The class of the object provides cues about what part classes might be inside it. For example, a part proposal on a dark round patch cannot be confidently classified as a wheel based solely on its appearance (fig. 5.1). If the corresponding supporting object proposal is a car, the evidence towards it being a wheel grows considerably. On the other hand, if the supporting proposal is a dog, the patch should be confidently classified as *not* a wheel.

Concretely, we process convolutional features pooled from the supporting object proposal through three FC layers (fig. 5.2). The third layer performs object classification and outputs scores for each object class, including a generic background class. These scores can be seen as object semantic features, which complement part appearance.

5.3.2.3 Object appearance

The appearance of the object might bring even more detailed information about what part classes it might contain. For example, the side view of a car indicates that we can expect to find wheels, but not a licence plate. We model object appearance by processing the convolutional features of the supporting proposal through two FC connected layers (fig. 5.2). These type of features have been shown to successfully capture the appearance of objects (Donahue et al., 2014; Ge et al., 2015).

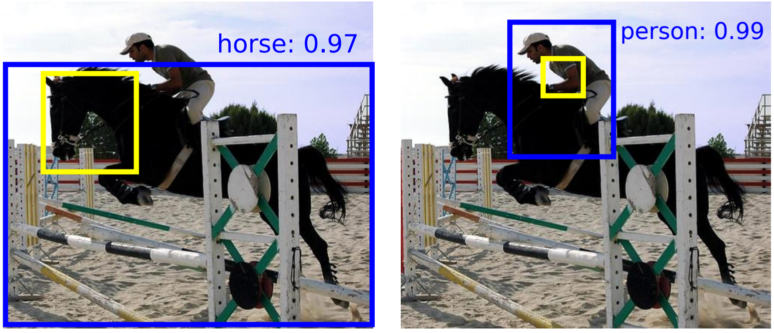


Figure 5.3: *Examples of supporting proposal selection. For each part proposal (yellow), we select as its supporting proposal (blue) the highest scored among the object proposals that contain it.*

5.3.3 Adding relative location

In this section, we add another type of information that could be highly beneficial: the relative location of the part with respect to the object. Parts appear in very distinct and characteristic relative locations and sizes within the objects. Fig. 5.4a shows examples of prior relative location distributions for some part classes as heatmaps. These are produced by accumulating all part ground-truth bounding-boxes from the training set, in the normalized coordinate frame of the bounding-box of their object. Moreover, this part location distribution can be sharper if we condition it on the object appearance, especially its viewpoint. For example, the *car-wheel* distribution on profile views of cars will only have two modes (fig. 5.4b) instead of the three shown in fig. 5.4a.

We incorporate this cue into our model by scoring each part proposal using its relative location with respect to the object. This indicates the probability that a proposal belongs to a certain part class, based purely on its location (it does not depend on part appearance).

Our relative location model is specific to each part class within each object class (e.g. a model for car-wheel, another for cat-tail). Below we explain the model for one particular object and part class. Given a proposal o of that object class, our model suggests a set of windows $\Lambda(o) = \{w_i\}_{i=1}^{I}$ inside o where instances of that part class are likely to be. Naturally, these windows will also depend on the appearance of o. For example, given a car profile view, our model suggests square windows on the lower corners as likely to contain wheels (fig.5.4b top). Instead, an oblique view of a car will also suggest wheels towards the lower central region, as well as a more elongated aspect ratio for the wheels on the side (fig. 5.4b bottom).

Let O be a set of object detections for a particular image, i.e. object proposals

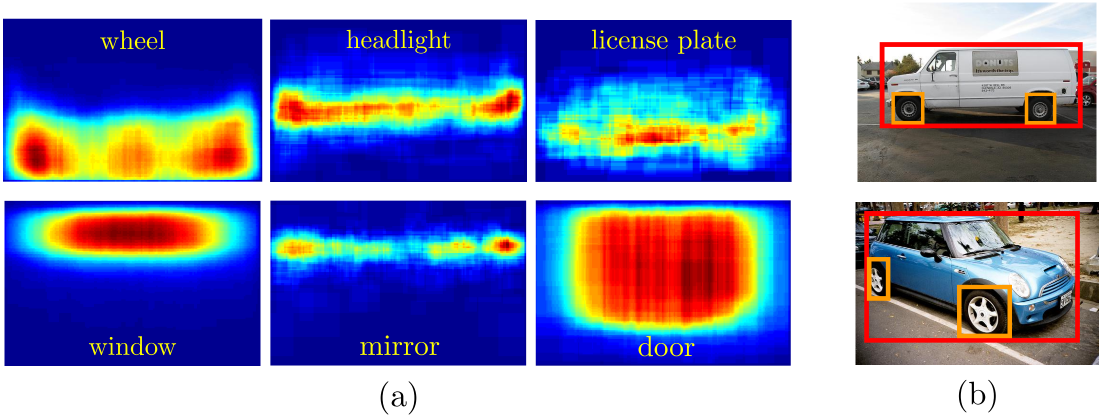


Figure 5.4: *Prior distributions and examples of part relative locations. (a) Heatmaps created using part ground-truth bounding-boxes normalized to the object bounding-box, using all* car *training samples. (b) Examples of part ground-truth bounding-boxes inside object bounding-boxes of class* car*, in different viewpoints. The oblique view in the second row has a wheel in the bottom-middle of the bounding-box.*

with high score after being processed through non-maxima suppression (Felzenszwalb et al., 2010b). We produce them automatically using standard Fast R-CNN (Girshick, 2015). Let $\phi_{obj}(o)$ be the score of detection $o \in O$ for the considered object class. We compute the relative location score $\phi_{rl}(p)$ for part proposal p using its overlap with all suggested windows

$$\phi_{rl}(p) = \max_{w_i \in \Lambda(o), o \in O} (\text{IoU}(p, w_i) \cdot \xi_i \cdot \phi_{obj}(o)), \tag{5.2}$$

where we use Intersection-over-Union (IoU) to measure overlap. Here, ξ_i is a confidence weight associated to each individual window w_i, and $\phi_{obj}(o)$ weights how much all windows $\Lambda(o)$ suggested by detection o should be trusted. Consequently, detections with higher scores provide stronger cues through more suggested windows. Fig. 5.5 shows an example, where we show only one *car* detection for clarity.

Below we explain two alternative ways to generate the suggested windows w_i and their confidence ξ_i. The first uses nearest-neighbors (sec. 5.3.3.1), whereas the second uses a new CNN architecture we dub *Offset Net* (sec. 5.3.3.2).

5.3.3.1 Nearest-neighbors (NN)

Parts of similar-looking objects tend to occupy similar relative locations within them. We exploit this phenomenon: given an object detection o in a test image, we retrieve part ground-truth bounding-boxes of objects similar to it in the training set and warp them into the coordinate frame of o. Ideally, these suggested windows will cover part instances on the test object.

Training database. For each object and part class combination, we create a database of pairs $\{(\psi(o_n), \Delta v_n)\}_{n=1}^{N}$ from the training set. Here, $\psi(o_n)$ represents the appearance of object proposal o_n. Concretely, we use L2-normalized CNN features. Some works (Donahue et al., 2014; Ge et al., 2015) have used FC features to represent appearance for clustering purposes, as they are more robust to pose than convolutional features. However, we prefer using features from the last convolutional layer, as we are interested in preserving the internal spatial structure of the object. The second element Δv_n is an offset: a 4D vector that maps object proposal o_n to a part ground-truth bounding-box inside it. If o_n contains multiple instances of the part inside (e.g. two wheels in a car), Δv_n represents a set of vectors instead, one per part instance.

Testing. At test time, given a query object detection o, our method retrieves its L nearest-neighbors from the training database and their set of offset vectors $\{\Delta v_l\}_{l=1}^{L}$. Note that the total number of vectors might be greater than L, as one Δv_l could comprise several vectors. We then obtain the set of suggested windows at test time by applying the retrieved vectors to o: $\Lambda(o) = \{o + \Delta v_l\}_{l=1}^{L}$. When $|\Delta v_l| > 1$, the offset operator $+$ is applied multiple times to o, resulting in $|\Delta v_l|$ windows. The suggested windows $\Lambda(o)$ cover likely locations for the part class, given the appearance of object o in the test image (fig. 5.5). In the nearest-neighbor case, all confidence weights ξ in (5.2) are set to 1.

5.3.3.2 Offset Net

Using nearest-neighbors has two main drawbacks. First, it is cumbersome since we need to store the database of training objects and part bounding-boxes. Second, it is slow since at test time we have to compute many distances to retrieve the nearest-neighbors of detection o.

We propose here a more elegant and faster approach: generate suggested windows $\Lambda(o)$ using a special kind of CNN, which we dub *Offset Net* (see fig. 5.2, Relative location branch). Offset Net directly learns to regress from the appearance of o to the relative location of a part class within it (i.e. learns to produce the offset that needs to be applied to o to generate $\Lambda(o)$). Nearest-neighbor instead mimics this effect by copying offset vectors from similar objects in the training set. Intuitively, a CNN is a good framework to learn this regressor, as the activation maps of the network contain

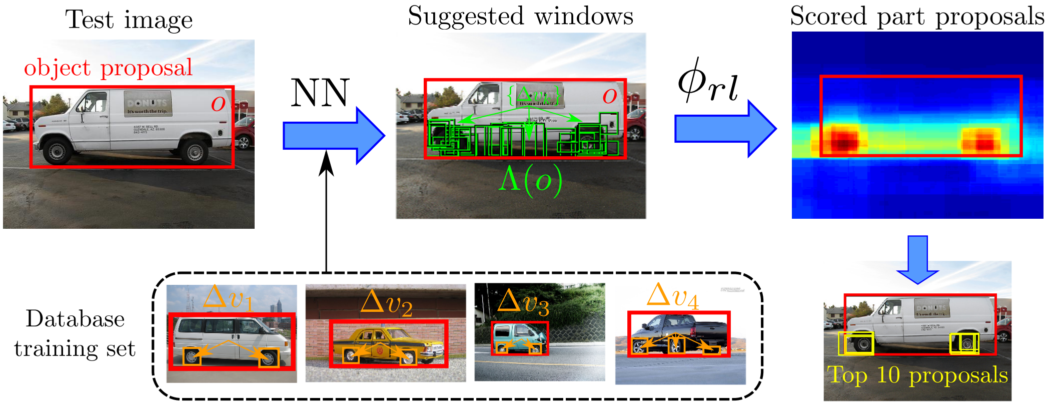


Figure 5.5: *Example of scoring part proposals based on relative location. Part class:* car-wheel. *Each object detection suggests windows likely to contain car-wheel within it. We score part proposals by computing their max IoU with any suggested window, and weighting them by confidence and object detection score. The top scored proposals, shown here as an example, nicely cover part instances. The suggested windows can be provided by nearest-neighbors (sec. 5.3.3.1), as in here, or by Offset Net (sec. 5.3.3.2).*

localized information about the parts of the object (Simon et al., 2014; Zeiler and Fergus, 2014). We confirmed this observation with our extensive analysis of chapter 4.

Model. Formally, the input to Offset Net is the test image and the set of object detections O. For each detection $o \in O$, the network outputs a set of 4D offset vectors Δv for each part class. It can contain multiple vectors as some objects have multiple instances of the same part in them (e.g. cars with multiple wheels).

Offset Net generates each offset vector in Δv through a regression layer. The nearest-neighbor handled the case $|\Delta v| > 1$ automatically by storing multiple vectors per object instance of the training set. To enable Offset Net to output multiple vectors we build multiple parallel regression layers. We set the number of parallel layers to the number of modes of the prior distribution for each part class (fig. 5.4). For example, the prior *car-wheel* has three modes, leading to three offset regression layers in Offset Net (fig. 5.2). On the other hand, Offset Net only has one regression layer for *person-head*, as its prior distribution is unimodal.

In some cases, however, not all modes are active for a particular object instance (e.g. profile views of cars only have two active modes out of the three, fig. 5.4b). For this reason, each regression layer in Offset Net has a sibling layer that predicts the presence of that mode in the input detection o, and outputs a presence score ρ. This way, even if the network outputs multiple offset vectors, only those with a high presence score will be taken into account. This construction effectively enables Offset Net to produce a variable number of output offset vectors, depending on the input o.

Training. We train the offset regression layers using a smooth-L1 loss, as in the bounding-box regression of Fast R-CNN (Girshick, 2015). We train the presence score layer using a logistic log loss: $L(x,c) = \log(1+e^{-cx})$, where x is the score produced by the network, and c is a binary label indicating whether the current mode is present ($c = +1$) or not ($c = -1$). We generate c using annotated ground-truth bounding-boxes (sec. 5.4). This loss implicitly normalizes score x using the sigmoid function. After training, we add a sigmoid layer to explicitly normalize the output presence score: $\rho = 1/(1+e^{-x}) \in [0,1]$.

Testing. At test time, given an input detection o, Offset Net generates M pairs $\{(\delta v_i, \rho_i)\}_{i=1}^{M}$ of offset vectors $\delta v_i \in \Delta v$ and presence scores ρ_i for each part class, where M is the number of modes in the prior distribution. We apply the offset vectors δv_i to o, producing a set of suggested windows $\Lambda(o) = \{o + \delta v_i\}_{i=1}^{M} = \{w_i\}_{i=1}^{M}$. Therefore, suggested windows more likely to be present have a higher confidence. We then compute the relative location score $\phi_{rl}(p)$ with eq. (5.2), using the presence scores ρ_i as confidence weights: $\xi_i = \rho_i$.

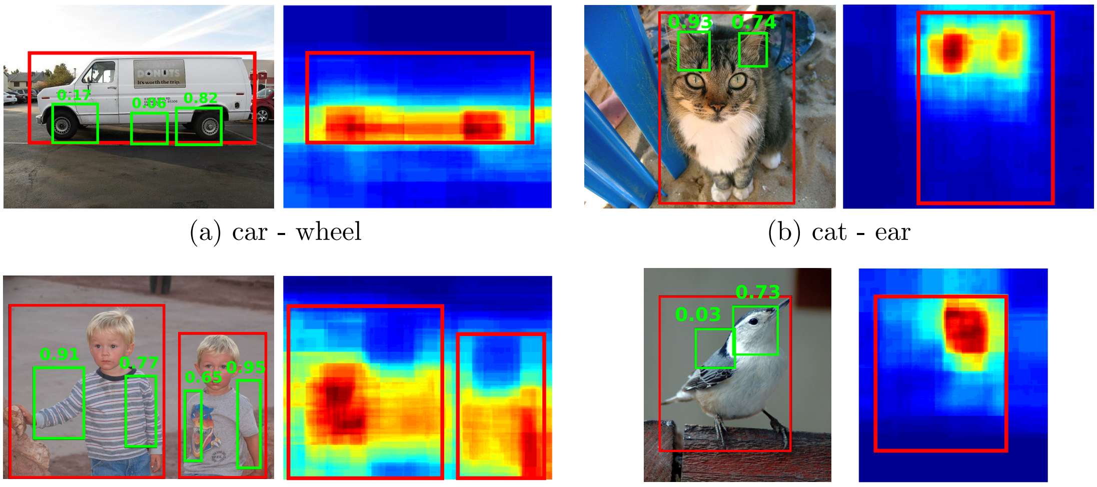
Fig. 5.6 shows examples of windows suggested by Offset Net, along with their presence score and a heatmap generated by scoring part proposals using eq. (5.2). We can see how the suggested windows cover very likely areas for part instances on the input objects, and how the presence scores are crucial to decide which windows should be relied on.

5.3.4 Cue combination

We have presented multiple cues that can help part detection. These cues are complementary, so our model needs to effectively combine them.

We concatenate the output of the part appearance, object class and object appearance branches and pass them on to a part classification layer that combines them and produces initial part classification scores (purple in fig. 5.2). Therefore, we effectively integrate object context into the network, resulting in the automatic learning of object-aware part representations. We argue that this type of context integration has greater potential than just a post-processing step (Wang et al., 2015b; Zhang et al., 2014a).

The relative location branch, however, is special as its features have a different



(a) car - wheel

(b) cat - ear

(c) person - arm

(d) bird - head

Figure 5.6: *Offset Net results. Examples of windows suggested by Offset Net (green) for different part classes, given the input object detections (red). We also show the heatmap generated by scoring all part proposals, showing how highly scored proposals occupy areas likely to contain part instances. The presence scores clearly indicate which suggested windows should be relied on (e.g. the second head of the bird and the middle wheel of the van have very low scores and are discounted in ϕ_{rl}).*

nature and much lower dimensionality (4 vs 4096). To facilitate learning, instead of directly concatenating them, this branch operates independently of the others and computes its own part scores. Therefore, we linearly combine the initial part classification scores with those delivered by the relative location branch (big + in fig. 5.2). For some part classes, the relative location might not be very indicative due to high variance in the training samples (e.g. *cat-nose*). In some other cases, relative location can be a great cue (e.g. the position of *cow-torso* is very stable across all its instances). For this reason, we learn a separate linear combination for each part class. We do this by maximizing part detection performance, using k-fold cross-validation on the training set ($k = 10$). We equally split the training set in k folds and train all networks on $k-1$ folds. We use the last fold as validation set, using grid search on the mixing weight in the $[0,1]$ range. We defined the measure of performance in sec. 3.6.3.

5.4 Implementation details

Proposals. Object proposals (Alexe et al., 2010; Dollar and Zitnick, 2014; Van de Sande et al., 2011) are designed to cover whole objects, and might fail to acknowledge separations between parts lacking changes in texture or color, such as the wing and the torso of a bird. To alleviate this issue, we changed the standard settings of Selective Search (Van de Sande et al., 2011), by decreasing the minimum box size to 10. This results in adequate proposals even for parts: reaching 71.4% recall with around 3000 proposals (IoU $>$0.5). For objects, we keep the standard Selective Search settings (minimum box size 20), resulting in around 2000 proposals.

Training the part detection network. Our networks are pre-trained for image classification on ILSVRC12 (Russakovsky et al., 2015) and fine-tuned on PASCAL-Part (Chen et al., 2014) for part detection, or on PASCAL VOC 2010 (Everingham et al., 2010) for object detection, using MatConvNet (Vedaldi and Lenc, 2015).

Fine-tuning for object detection follows the Fast R-CNN procedure (Girshick, 2015). We compute scores for all 20 object classes of PASCAL VOC 2010, although in practice we use only 16 scores, corresponding the object classes with parts of PASCAL-Part. We keep the original 20 object classes to provide a more complete object detection output, equivalent to the original Fast R-CNN. For the part detection fine-tuning we changed the following settings. Positive samples for parts overlap any part ground-truth > 0.6 IoU, whereas negative samples overlap < 0.3. We trained for 16 epochs, with learning rate 10^{-3} for the first 12 and 10^{-4} for the remaining 4.

We jointly train part appearance, object appearance, and object class branches for a multi-task part detection loss. We modify the Region of Interest (RoI) pooling layer to pool convolutional features from both the part proposal and the selected object proposal, using the object scores output by the object class branch. Backpropagation through this layer poses a problem, as (5.1) is not differentiable. We address this by backpropagating the gradients only through the area of the convolutional map covered by the object proposal selected by the argmax in (5.1). The object detection layers of the object class branch are previously initialized using the standard Fast R-CNN object detection loss, in order to provide reliable object proposal scores when joint training starts.

Training Offset Net. We need object samples and part samples to train Offset Net. Our object samples are all object ground-truth bounding-boxes and object proposals with IoU ≥ 0.7 in the training set. Our part samples are only part ground-truth bounding-boxes. We split the horizontal axis in M regions, where M is the number of modes in the part class prior relative location distribution. We assign each part ground-truth bounding-box in the object to the closest mode. If a mode has more than one part bounding-box assigned, we pick one at random. All layers except the top ones are initialized with a Fast R-CNN network trained for object detection. Similarly to the other networks, we train it for 16 epochs, but with learning rates 10^{-4} and 10^{-5}. For clarity, in fig. 5.2 we use the same convolutional features for all branches. In practice, Offset Net uses its own convolutional layers, as they also are fine-tuned for its task.

5.5 Results

5.5.1 Validation of our model

Dataset. We present results on PASCAL-Part (Chen et al., 2014), which has pixel-wise part annotations for the images of PASCAL VOC 2010 (Everingham et al., 2010). For our experiments we fit a bounding-box to each part segmentation mask. We pre-process the set of part classes as follows. We discard additional information on semantic part annotations, such as 'front' or 'left' (e.g. both "car wheel front left" and "car wheel back right" become *car-wheel*). We merge continuous subdivisions of the same semantic part ("horse lower leg" and "horse upper leg" become *horse-leg*). Finally, we discard tiny parts, with average width and height over the training set $\leq$15 pixels (e.g. "bird eye"), and rare parts that appear $<$ 10 times (e.g. "bicycle headlight"). After this pre-processing, we obtain a total of 105 part classes for 16 object classes (appendix B). We train our methods on the `train` set and test them on the `val` set (the `test` set is not annotated in PASCAL-Part). We note how we are the first work to present fully automatic part detection results on the whole PASCAL-Part dataset.

Performance measure. Just before measuring performance we remove duplicate detections using non-maxima suppression (Felzenszwalb et al., 2010b). We measure part detection performance using Average Precision (AP), following the PASCAL VOC protocol (Everingham et al., 2010). We consider a part detection to be correct if its IoU with a ground-truth part bounding-box is > 0.5.

Baseline results. As base network in all models we use AlexNet (Krizhevsky et al., 2012), unless stated otherwise (convolutional layers on the leftmost column of fig. 5.2). Tab. 5.1 presents part detection results. The baseline model achieves only 22.1 mAP without bounding-box regression (sec 5.3.1). As a reference, the same model, when trained and evaluated for object class detection, achieves 48.5 mAP on PASCAL VOC 2010 (Everingham et al., 2010), which contains the same object classes as PASCAL-

Model	Obj. App	Obj. Class	Rel Loc	mAP
Baseline (Girshick, 2015)				22.1
Obj. appearance	✓			25.1
Obj. class		✓		23.0
Obj. app + class	✓	✓		25.7
All-NN (no app)			✓	23.5
5-NN			✓	23.9
Offset Net ($M = 1$)			✓	24.3
Offset Net			✓	24.7
Obj. app + 5-NN	✓		✓	26.2
Obj. app + Offset Net	✓		✓	26.8
Full (Obj. app + class + Offset Net)	✓	✓	✓	**27.4**
Baseline (Girshick, 2015) (bbox-reg)				24.5
Full (bbox-reg)	✓	✓	✓	**29.5**

Table 5.1: *Part detection results on PASCAL-Part with AlexNet. The baseline model uses only part appearance. All other models include it too.*

Part. This massive difference in performance demonstrates the inherent difficulty of the part detection task.

Adding object appearance and class. By adding object appearance (sec. 5.3.2.3), performance increases by +3 mAP, which is a significant improvement. Adding object class (sec. 5.3.2.2) also helps, albeit less so (+0.9 mAP). This indicates that the appearance of the object contains extra knowledge relevant for part discrimination (e.g. viewpoint), which the object class alone cannot provide. Furthermore, the combination of both types gives a small additional boost (+0.6 mAP compared to using only object appearance). Although in principle object appearance subsumes its class, having a more explicit and concise characterization of the class is beneficial for part discrimination.

Adding relative location. Our relative location models (sec. 5.3.3) also bring improvements. To better understand the effects of our design choices, we evaluate various ablated versions of our models. We start with *All-NN*, a version of our nearest-neighbor approach (sec. 5.3.3.1) that uses all training instances, instead of only the ones most similar to the test object. This is essentially a location prior, independent of the object appearance (fig. 5.4a). All-NN already brings +1.4 mAP improvement, showing that location priors can indeed be helpful. Now we set $L = 5$, effectively conditioning on object appearance. The 5-NN model brings a larger improvement (+1.8 mAP), demonstrating the role of object appearance in modulating the relative location cue.

We present results for two versions of our Offset Net model (sec. 5.3.3.2). In the first one, we fix the number of modes M to just 1 for all part classes, regardless of the complexity of the prior distribution. In this simplistic setting, Offset Net already surpasses the 5-NN approach, and outperforms the baseline by +2.2 mAP. This indicates that even with only 1 suggested window, Offset Net provides a strong relative location cue. When setting M based on the prior distribution as explained in sec. 5.3.3.2, the improvement further rises to +2.6 mAP.

Both our relative location models capture the spirit of this work and help part discrimination. Given Offset Net offers better performance, higher computational efficiency and lower memory footprint than nearest-neighbors, it is our method of choice for our final model.

Combining cues. Finally, we now combine all our cues as in sec. 5.3.4 (always using also part appearance). First, we combine each of our relative location models with object appearance. Both combinations are beneficial and surpass each cue alone. In this setting, Offset Net still brings higher improvements than 5-NN.

Our best model (full) combines all cues and achieves +5.3 mAP over the baseline. We regard this as a substantial improvement, especially considering the high difficulty of the task, as demonstrated by the rather low baseline performance. This result shows that all cues we propose are indeed complementary to part appearance and to each other; when combined, all contribute to the final performance.

We also tested the baseline and our model using bounding-box regression (bbox-reg in tab. 5.1). This is beneficial in all cases. Importantly, the improvement brought by our full model over the baseline (+5 mAP) is similar to the one without bounding-box regression (+5.3 mAP).

Example detections. Fig. 5.7 shows some part detection examples for both the baseline and our full model (without bounding-box regression). In general, our model localizes parts more accurately, fitting the part extent more tightly (fig. 5.7a,5.7g). Moreover, it also finds some part instances missed by the baseline (fig. 5.7b-f, 5.7h-i). Our method uses object detections automatically produced by Fast R-CNN (Girshick, 2015). When these are inaccurate, our model can sometimes produce worse part detections than the baseline (fig. 5.7j). In some other cases, however, even inaccurate object

Model	Obj. App	Obj. Class	Rel Loc	mAP
Baseline (Girshick, 2015) (VGG16, bbox-reg)				35.8
Full (VGG16, bbox-reg)	✓	✓	✓	**40.1**

Table 5.2: *Part detection results on PASCAL-Part with VGG-16.*

detections lead to better part detections (fig. 5.7k).

Runtime. We implement our model in MatConvNet (Vedaldi and Lenc, 2015) and report runtimes on a Titan X GPU. Both our method and the baseline start by generating Selective Search proposals, taking 3.3 seconds per image (s/im). After this, the baseline takes 0.96 s/im and our model 3.79 s/im. Therefore, the total runtime for the baseline is 4.3 s/im, whereas for our model it is 7.1 s/im. Note how we also output object detections, which the baseline does not.

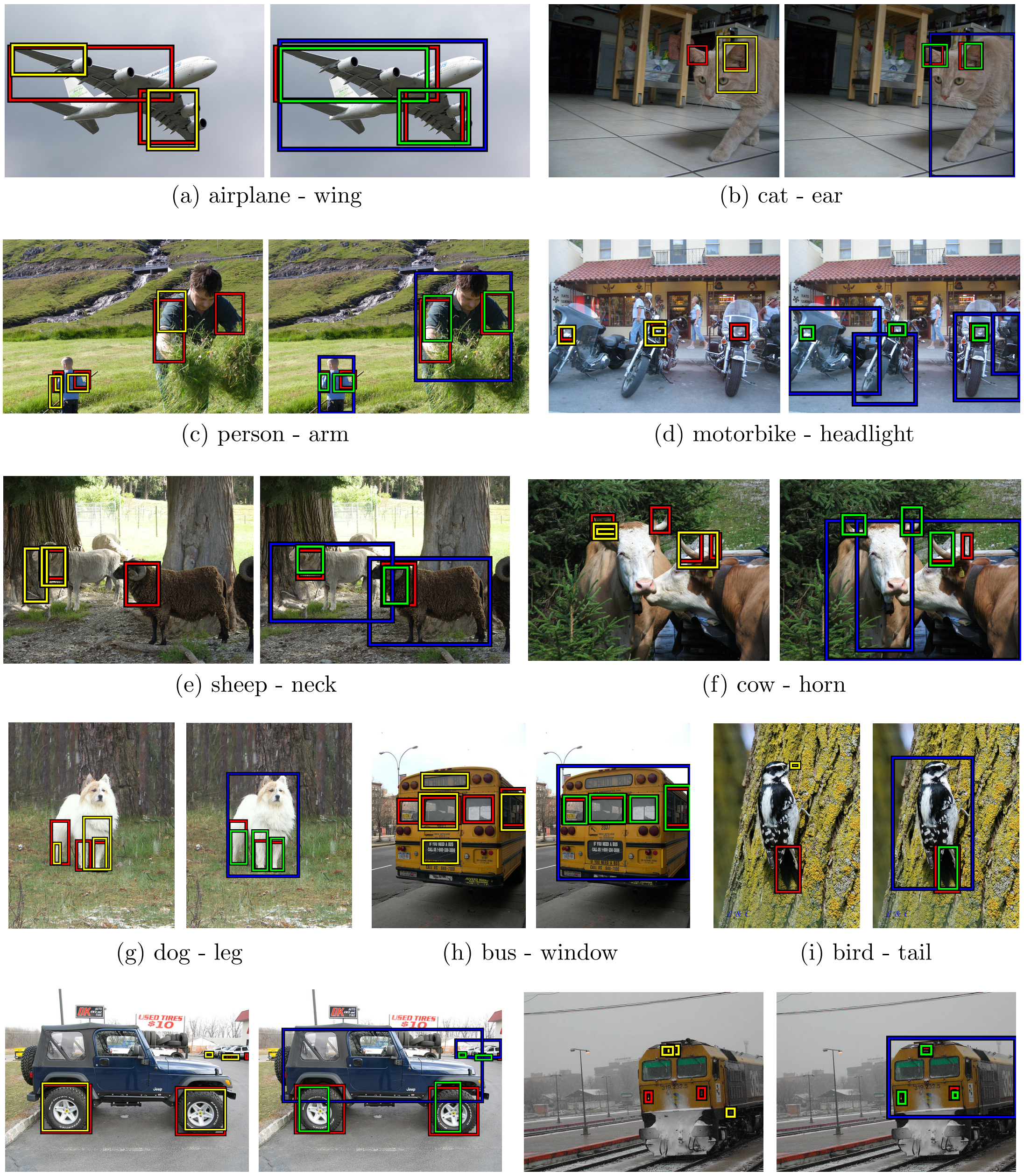
Results for VGG16. We present results for the deeper VGG16 network (Simonyan and Zisserman, 2015) in table 5.2. The relative performance of our model and the baseline is analog to the AlexNet case, but with much higher mAP values. The baseline achieves 35.8 mAP with bounding-box regression. Our full model achieves 40.1 mAP, which is a substantial improvement of magnitude comparable to the AlexNet case (+4.3 mAP).

5.5.2 Comparison to other methods

5.5.2.1 Part detection up to a bounding-box

We compare here our full (bbox-reg) model (tab. 5.1) to several prior works on detecting parts up to a bounding-box (Chen et al., 2014; Zhang et al., 2014a; Gkioxari et al., 2015) . We use AlexNet, which is equivalent to the networks used in (Zhang et al., 2014a, 2016; Gkioxari et al., 2015). Tab. 5.3 summarizes all results.

Chen et al. (2014). We compare to Chen et al. (2014) following their protocol (sec. 4.3.3 of Chen et al. (2014)). They evaluate on 3 parts (head, body, and legs) of the 6 animal classes of PASCAL-Part, using Percentage of Correctly estimated Parts as measure (PCP). They also need an extra measure called Percentage of Objects with Part estimated (POP), as they compute PCP only over object instances for which their system outputs a detection for that part class. Additionally, they use ground-truth object bounding-boxes at test time. More precisely, for each ground-truth box, they retain the best overlapping object detection, and evaluate part detection only within it. As ta-



(a) airplane - wing

(b) cat - ear

(c) person - arm

(d) motorbike - headlight

(e) sheep - neck

(f) cow - horn

(g) dog - leg

(h) bus - window

(i) bird - tail

(j) car - wheel

(k) train - headlight

Figure 5.7: *Qualitative results. Example part detections for the baseline model (yellow) and our model combining all cues (green). We also show part ground-truth bounding-boxes (red), and object detections output by our method (blue).*

Comparison to	Dataset	Measure	Obj GT at test	Theirs	Ours
Chen et al. (2014)	PASCAL-Part	POP	✓	44.5	**51.3**
		PCP	✓	70.5	**72.6**
Fine-grained	CUB200-2011	PCP		66.1 (Zhang et al., 2014a)	**91.9**
			✓	74.0 (Zhang et al., 2014a) 85.0 (Lin et al., 2015) **94.2** (Zhang et al., 2016)	92.7
Gkioxari et al. (2015)	PASCAL VOC09	AP (0.3)		38.7	**53.6 (65.5)**
		AP (0.5)	✓	17.1	**21.6 (44.7)**

Table 5.3: *Comparison to other methods. We compare to methods that report bounding-box part detection results, using their settings and measures.*

ble 5.3 shows, we outperform Chen et al. (2014) on PCP, and our POP is substantially better, demonstrating the higher recall reached by our method. Moreover, note how these measures can only be used in the specific case of one part instance per object class (e.g. all legs of the animal as one instance), whereas the standard AP detection measure we use in sec. 5.5.1 is more general. Also, Chen et al. (2014) only report results in this easier setting, whereas we report results in a fully automatic setting without using any ground-truth at test time (sec. 5.5.1).

Fine-grained (Zhang et al., 2014a; Lin et al., 2015; Zhang et al., 2016). These fine-grained recognition works report part detection results on the CUB200-2011 (Wah et al., 2011b) bird dataset for the head and body. They all evaluate using PCP and including object ground-truth bounding-boxes at test time. Our model outperforms Zhang et al. (2014a) and Lin et al. (2015) by a large margin and is comparable to Zhang et al. (2016). Only Zhang et al. (2014a) report results without using object ground-truth at test time. In this setting, our method performs almost as well as with object ground-truth at test time, achieving a very remarkable improvement (+25.8 PCP) compared to Zhang et al. (2014a). Furthermore, we note that CUB200-2011 is an easier dataset than PASCAL-Part, with typically just one, large, fully visible bird instance per image.

Gkioxari et al. (2015). This action and attribute recognition work reports detection results on three person parts (head, torso, legs) on PASCAL VOC 2009 images (tab. 1 in Gkioxari et al. (2015)). As these do not have part ground-truth bounding-boxes, they construct them by grouping the keypoint annotations of (Bourdev et al., 2010) (sec. 3.2.2 of Gkioxari et al. (2015)). For an exact comparison, we train and test our full model using their keypoint-derived bounding-boxes and use their evaluation measure (AP at various IoU thresholds). We also report (in parenthesis) results using the standard part ground-truth bounding-boxes of PASCAL-Part during both training and testing (as PASCAL VOC 2009 is a subset of PASCAL-Part). We outperform Gkioxari et al. (2015) using their bounding-boxes, and obtain even better results using the standard bounding-boxes of PASCAL-Part. Moreover, we note how their part detectors have been trained with more expensive annotations (on average 4 keypoints per part, instead of one bounding-box).

5.5.2.2 Part segmentation

Several works (Wang et al., 2015b; Hariharan et al., 2015; Liang et al., 2016a; Xia et al., 2016; Chen et al., 2016) perform part segmentation, i.e. they output pixel-wise masks instead of the bounding-boxes output by detection approaches. The segmentation architectures used in these works tend to naturally include ample regions around each classified pixel. Hence, their feature representation already encodes object context to a certain degree. We investigate here how our detection model performs in comparison to a recent part segmentation model.

We use the state-of-the-art part segmentation approach of Chen et al. (2016), termed DeepLab. Chen et al. (2016) adapt CNN models originally trained for image classification (Simonyan and Zisserman, 2015; He et al., 2016) for semantic segmentation. They use *atrous* convolutional filters, which substitute the usual approach for segmentation of downsampling and upsampling feature maps within the network (Long et al., 2015). Atrous convolutions enable a finer control of the resolution in which the network extracts its features as well as a larger field of view, which in turn increases the amount of context included. Moreover, DeepLab arranges parallel atrous filters in a pyramid to extract features at multiple scales, facilitating the network's response to objects of different sizes.

We train DeepLab [1] with ResNet-101 (He et al., 2016) architecture on all 105 parts of PASCAL-Part `train` set. We keep the original pixel-wise annotations given in the dataset. We follow the training protocol of Chen et al. (2016), which uses multi-scale input images, pre-training on Microsoft COCO (Lin et al., 2014), and extensive data augmentation. The only missing component of their model is the CRF used for post-processing, which in any case only gives a marginal gain for this task. As a sanity check, we compute the segmentation performance of the DeepLab model trained by us, and compare it to Chen et al. (2016)'s reported results on a subset of PASCAL-

[1] We use the TensorFlow implementation provided in http://github.com/DrSleep/tensorflow-deeplab-resnet.

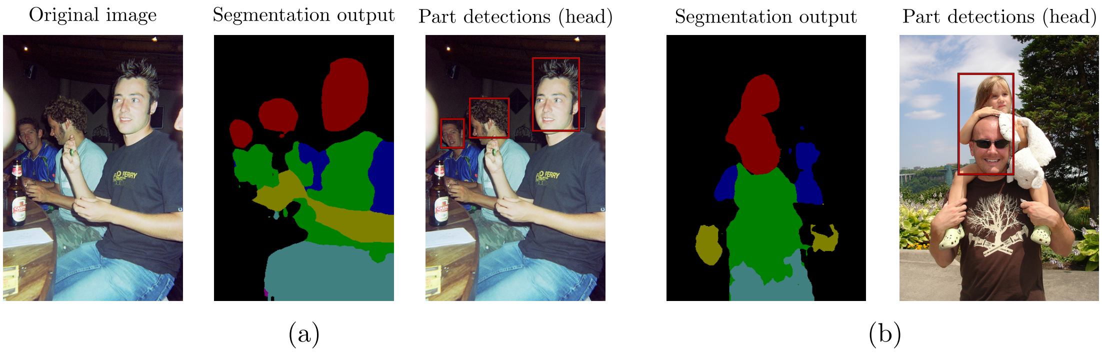


Figure 5.8: *Part segmentation method converted to part detection up to a bounding-box. (a) We place a bounding-box around each connected component of the segmentation method output. Each bounding box is a detection of score 1. (b) Parts spatially close to each other might form a common connected component, leading to detections that include several part instances.*

Part (e.g. head, torso, arms, and legs of person). We achieve comparable results, proving the validity of our trained DeepLab model.

In order to compare DeepLab with our part detection model, we need to convert DeepLab's segmentation outputs to part detections up to a bounding-box. To do so, at test time we run the trained DeepLab model on the `val` set of PASCAL-Part and place a bounding-box around each connected component of the output segmentation, as shown in fig. 5.8a. All detections have score 1, and hence there is no need to process them with NMS. DeepLab for detection achieves 11.6 mAP, averaged over all 105 part classes of PASCAL-Part. This is much lower than the result obtained by our full model, e.g. 40.1 mAP with VGG16, despite the superior performance of ResNet-101 with respect to VGG16 (He et al., 2016).

In fairness, we note that their results could be improved by preventing detections that cover multiple part instances, mostly caused by objects that are close to each other (fig. 5.8b). This issue could be alleviated by, for example, adding instance reasoning. In any case, the detection performance of DeepLab is dramatically behind our part detection model. Moreover, our method requires a lower lever of supervision, as we use bounding-boxes instead to pixel-wise masks. We conclude that, although segmentation architectures provide good segmentation results on a very restricted set of classes (humans and quadrupeds), they do not perform well for the part detection task of *general* object classes, and more research is necessary to make them reach a comparable performance level with our method.

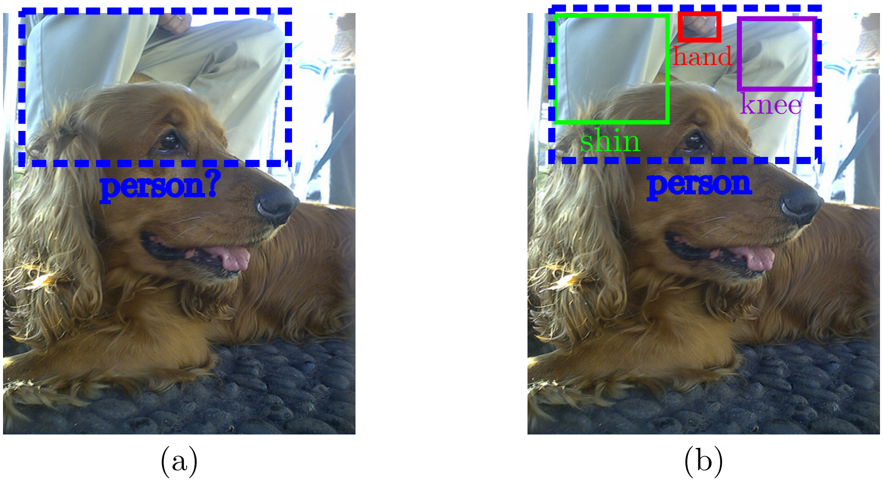


Figure 5.9: *(a) Classic object detection model, where classification is based only on the appearance of the object proposal. When the object is truncated or occluded, classification is very challenging. (b) Object detection model that uses help from the object parts. The detection of several semantic parts, which may be easier to detect than the whole object, assist in the correct categorization of challenging instances such as the one shown.*

5.6 Conclusion and outlook

We presented a semantic part detection model that detects parts in the context of their objects. Our model includes several types of object information. We incorporate object class and appearance as indicators of what parts lie inside. We also model relative location information conditioned on the object appearance. All these complementary cues are effectively combined, achieving more accurate part detections. Our model leads to a substantially better performance than detecting parts based only on their local appearance, improving by +5 mAP on the baseline model on the PASCAL-Part dataset.

We now discuss possible future work to extend the part detection model presented in this chapter and apply insights learned regarding context integration for recognition.

Sequential part detection. Our part detection model computes initial part scores that are later refined by incorporating object information. The current model only consists of only one refinement step, but it may be extended to several sequential steps that iteratively output increasingly more accurate part detections. We could accomplish this by adopting either of the two following approaches. First, we could transform our model into a sequential network architecture, similar to the successful Convolutional Pose

Machines (CPM) of Wei et al. (2016). A CPM refines a structured output, concretely for articulated pose estimation, by using several convolutional networks sequentially arranged. The input of each network is the prediction output by the previous network, except for the first one, which takes the image as input. In our case, we would use part detections predicted by the previous network as well as additional object contextual information, which could also be iteratively refined. Second, we could use a Recurrent Neural Network (RNN) to sequentially detect every part of the object. Our RNN would output each part detection automatically taking into account previous part detections and combining this information with object context. This architecture gives great flexibility when learning model parameters such as the best ordering in which parts should be detected, or how many refinement steps are necessary for a particular performance level. Furthermore, training RNNs for object and part detection in real-world images is a very challenging task and further research is required to achieve reasonable performance levels. The proposed extension would explore this exciting and potentially prosperous research avenue.

Parts to help object detection. Part-based models have shown excellent performance in the object detection task (Felzenszwalb et al., 2010b; Azizpour and Laptev, 2012; Endres et al., 2013; Chen et al., 2014; Tsai et al., 2015). Nonetheless, in the last few years the state-of-the-art in object detection has been dominated by CNN approaches that do not explicitly model object parts (Girshick et al., 2014; Girshick, 2015; He et al., 2016; Ren et al., 2015; Redmon et al., 2016; Liu et al., 2016). Despite the outstanding results of such models, they are bound to fail when objects appear under particular challenging conditions, such as severe occlusions or extreme viewpoints.

We investigated in chapter 4 how CNNs trained for recognition automatically learn discriminative visual patterns. When such patterns are not clearly visible at test time, CNNs become gravely crippled for this task. A possible solution consists in explicitly integrating part context into the network's internal representation. By actively forcing the network to learn the parts alongside the objects, we provide it with a fallback mechanism when object appearance as a whole is not discriminative enough. For example, the person in figure 5.9 is severely truncated and occluded, the most discriminative parts on which the network relies for person recognition (e.g. head, torso) are missing. However, some parts of the person might still be easy to detect, such as the hand or the knee. The effective network integration of information about those parts would provide a valuable cue for recognizing the person in this image.

The CNN model presented in this chapter offers an excellent starting point for part-aware object detection. We could first detect parts in the image and then use them as support for object recognition. For example, the feature representation for each object proposal could be enhanced by including the appearance and scores of a few parts inside it. We could automatically learn how many part classes should be integrated in the representation, or even which part classes are the most suitable for each object class. Our relative location model can be modified to suggest windows that contain the input, as opposed to windows contained in it. By combining all the part cues, this object detection model would most likely help to recognize challenging object instances. Moreover, it might also improve the precision for easier cases, for example, by outputting finer localizations based on the locations of the parts.

Part segmentation with object instance context. Several recent approaches (Wang et al., 2015b; Hariharan et al., 2015; Liang et al., 2016a; Xia et al., 2016; Chen et al., 2016) address part segmentation instead of detection, and thus their outputs are more precise than bounding-boxes. Most of these works are based on Fully Convolutional Networks (FCN) (Long et al., 2015) or atrous convolutions (Chen et al., 2016). Therefore, they automatically take into account some of the object appearance in their predictions as they classify each pixel by observing a large region around it. However, none of these approaches are aware of the object instances in the image since the general segmentation task is defined without the notion of instance.

This extension proposes coupling a part segmentation architecture with object instance reasoning. A way of modeling object instances is using object segment proposals, such as those provided by SharpMask (Pinheiro et al., 2016). With such proposals, we could assign instance-specific object context information to each pixel to be classified, including the object appearance, its class, and even the relative location of the parts inside it. The latter could be implemented analogously to our OffsetNet, i.e. using a separate network branch. In this case, instead of suggesting windows on expected locations as in OffsetNet, we would directly output heatmaps highlighting expected part locations, taking as input the region belonging to the object proposal. Current works are restricted to either person or quadruped classes for which parts can be shared amongst them. Our approach would pave the way for more general part segmentation, as it can be applied to arbitrary object classes. Moreover, instance-aware semantic segmentation is currently an engaging research topic that gathers notable interest in the community (Dai et al., 2016; Liang et al., 2016b; Li et al., 2017).

Chapter 6

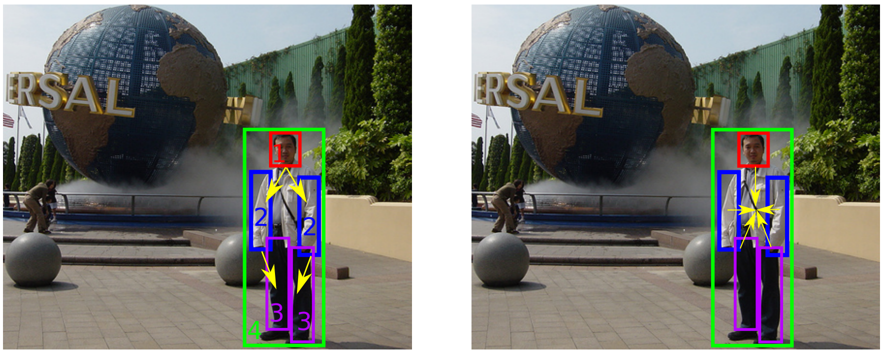
Conclusions

In this thesis, we have addressed the computer vision tasks of object detection and part detection. We have leveraged the use of different context types to advance on both tasks. The presented work can be summarized in four main contributions. The first contribution is an active search strategy that uses image context for efficient object class detection (chapter 3). The second contribution extends this active search to jointly detect pairs of object classes (chapter 3). Our third contribution presents an analysis on the emergence of semantic parts in CNNs trained for object detection (chapter 4). Finally, our last contribution is a part detection CNN model that integrates object context to obtain more accurate part detections (chapter 5).

Throughout this thesis, context has proven to be a valuable source of information that complements visual detection tasks. We have shown how adequate integration of contextual knowledge leads to effective improvements in both detection efficiency and performance. Each chapter of this thesis has presented a specific method focused on a particular task. This final chapter explores the interplay between insights gained in the different chapters of this thesis and provides additional future work.

An active search strategy guided by part context. The search strategy presented in chapter 3 is guided by generic image context. It matches the appearance of observed image regions to those recorded from the training set, and then reasons using their relative location with respect to the object. In the latest stages of the multi-class version, we employ a more specific context by restricting the origin of such regions to the visual phrase area. Nonetheless, all of the used context regions lack a semantic interpretation as they are simply arbitrary regions extracted from a particular area of the image.

In this extension, we propose guiding the visual search for objects by using a specific type of meaningful region, i.e. semantic parts. Some object parts are fairly easy



(a) (b)

Figure 6.1: *Two alternative strategies to guide the object search by using part context. (a) Sequential part search that terminates with the object. (b) Active search that integrates location cues by independently detected parts.*

to detect, for example, those that are big and have a rather stable appearance across different instances, such as car wheels. If we first detect these easily localizable parts, we can limit the object search to the area around them, as the object that contains them must be in their surroundings. Moreover, parts tend to appear at particular relative locations inside the object (fig. 5.4). Therefore, the detected parts provide excellent cues regarding the location of the object or even other parts. For example, a search strategy for cars could first localize car wheels. Wheel detection is a reasonably accessible task, as wheels exhibit smaller complexity than cars, have lower intra-class variation and are relatively big. Once we localize the wheels, the detection of the corresponding car should readily follow by exploring the area immediately above the wheels at a specific offset learned from training data.

Figure 6.1 presents two alternative approaches to implement object search strategies guided by part context. We could follow a sequential search strategy to localize the parts (fig. 6.1a), possibly by implementing the first extension suggested in section 5.6. The search should start from the easiest part. In the depicted example, where person is the object class, the starting role corresponds to the part head. The search would then find part after part, possibly selecting which part to locate next based on the parts located so far. The last 'part' of the search would be special, as it corresponds to the whole object. Alternatively, we could independently find as many parts as possible (fig. 6.1b) and use their locations to gather clues about the object location. Note how this differs from the second extension in section 5.6, since in this case parts would be solely used to guide the search, instead as context to enhance the representation used for recognition. As a by-product of these search strategies, we would also obtain part detections.

Part-based models for fine-grained recognition using truly discriminative parts. Fine-grained recognition models heavily rely on semantic parts since many sub-categories can only be recognized by subtle changes in the appearance of some parts (Parkhi et al., 2012; Zhang et al., 2014a; Lin et al., 2015; Zhang et al., 2016; Akata et al., 2016). For example, the recent work of Zhang et al. (2016) considers fixed size boxes around annotated keypoints for 16 different semantic parts of birds. Although it might seem that human experts rely on semantic parts for recognition, most likely they do not use *all* semantic parts, since some might be uninformative. Moreover, it is also possible that they are actually employing other distinctive features, such as a small subregion of a part (e.g. tip of the bird's beak). For this reason, forcing the network to explicitly take into account the totality of all the semantic parts might actually be counterproductive for recognition, as it overloads the model with irrelevant information.

Our analysis in chapter 4 showed how discriminative image regions for object recognition do not always perfectly align with semantic parts. Among others, they include part subregions (e.g. top half of the face) and assemblies of multiple semantic parts (e.g. head and neck). We propose complementing this analysis by shifting the focus to fine-grained recognition. For instance, we can identify which object regions are truly discriminative for this task. One possibility is to systematically block each filter as in section 4.5, study the effect on the class scores, and inspect the regions on which the relevant filters fire. Alternatively, we could track score changes when ignoring arbitrary image regions, similarly to how section 4.6 ignores regions occupied by semantic parts. Based on this, we could develop fine-grained models that explicitly incorporate such regions in their models for better recognition, similarly to our incorporation of object context for part detection in chapter 5. The incorporated regions may be adaptive, depending on several factors such as the overall object appearance or the object class.

As an additional interesting outcome, we would be able to compare the regions the model finds discriminative for fine-grained recognition with existing expert knowledge. For example, ornithologists might easily recognize a particular bird species by the shape of their eyes. Do our CNNs for fine-grained recognition learn this characteristic to classify such birds? If not, what regions does it use for such task? Furthermore, we could explicitly integrate very specific expert knowledge in our model, for example, by encouraging the network to focus on known discriminative areas and then determine whether this results in superior recognition performance compared to automatically learned discriminative regions.

Appendix A

Acronyms

AP - Average Precision

BoW - Bag-of-words

CNN - Convolutional Neural Network

CPM - Convolutional Pose Machine

CRF - Conditional Random Field

DPM - Deformable Part Model

GAN - Generative Adversarial Network

HOG - Histogram of Oriented Gradients

IoU - Intersection-over-Union

IVP2C - Inside Visual Phrase to Class

NMS - Non-Maxima Supression

PCA - Principal Component Analsysis

RNN - Recurrent Neural Network

SGD - Stochastic Gradient Descent

SVM - Support Vector Machine

ReLU - Rectified Linear Unit

RPN - Region Proposal Network

PPMCC - Pearson Product-Moment Correlation Coefficient

VP2C - Visual Phrase to Class

Appendix B

Object and part classes

Class	**Part**	**Number of samples**		**Class**	**Part**	**Number of samples**	
		Train	Test			Train	Test
aero	body	359	363	cow	head	188	202
	stern	330	323		ear	236	157
	wing	505	321		muzzle	157	169
	engine	364	204		horn	103	48
bike	wheel	552	295		torso	194	214
	saddle	206	216		neck	74	102
	handlebar	262	289		leg	466	157
	chainwheel	211	199		tail	48	66
bird	head	448	451	dog	head	694	696
	beak	379	371		nose	612	597
	torso	464	469		torso	678	683
	neck	249	21		neck	310	308
	wing	313	174		leg	1558	565
	leg	476	272		paw	1055	453
	foot	282	160		tail	632	635
	tail	321	333	horse	head	289	301
bottle	cap	443	452		ear	439	261
	body	551	507		muzzle	262	255
bus	frontside	162	179		torso	306	309
	leftside	99	119		neck	223	229
	rightside	117	97		leg	847	260
	backside	40	40		tail	163	179
	roofside	19	12	motorbike	wheel	497	291
	mirror	258	176		handlebar	12	18
	liplate	127	138		headlight	156	127
	door	109	100	person	head	3627	3805
	wheel	450	204		hair	2650	2776
	window	816	242		torso	3621	3819
car	frontside	425	417		neck	1854	1930
	leftside	388	364		arm	5237	3481
	rightside	348	338		hand	3626	2597
	backside	298	319		leg	3992	2425
	roofside	137	158		foot	2177	1435
	liplate	298	313	pottedplant	pot	380	395
	door	324	234		plant	401	415
	wheel	1231	642	sheep	head	307	298
	headlight	535	314		ear	431	251
	window	1272	693		muzzle	246	231
cat	head	558	564		horn	40	32
	eye	869	471		torso	329	328
	ear	1030	548		neck	209	215
	nose	486	497		leg	242	78
	torso	546	553		tail	97	96
	neck	365	397	train	head	173	161
	leg	1040	447		frontside	144	141
	paw	783	389		leftside	96	77
	tail	234	224		rightside	70	74
tvmonitor	screen	328	310		coach	321	187

Table B.1: *List of used part classes in PASCAL-Part along with the number of samples in the training and test set.*

Bibliography

Agrawal, P., Girshick, R., and Malik, J. (2014). Analyzing the performance of multilayer neural networks for object recognition. In *Proceedings of the European Conference on Computer Vision*.

Akata, Z., Malinowski, M., Fritz, M., and Schiele, B. (2016). Multi-cue zero-shot learning with strong supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Alexe, B., Deselaers, T., and Ferrari, V. (2010). What is an object? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Alexe, B., Heess, N., Teh, Y., and Ferrari, V. (2012). Searching for objects driven by context. In *Advances in Neural Information Processing Systems*.

Amit, Y. and Geman, D. (1999). A computational model for visual selection. *Neural Computation*, 11(7):1691–1715.

Arbeláez, P., Hariharan, B., Gu, C., Gupta, S., Bourdev, L., and Malik, J. (2012). Semantic segmentation using regions and parts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Azizpour, H. and Laptev, I. (2012). Object detection using strongly-supervised deformable part models. In *Proceedings of the European Conference on Computer Vision*.

Bazzani, L., de Freitas, N., Larochelle, H., Murino, V., and Ting, J. (2011). Learning attentional policies for tracking and recognition in video with deep networks. In *International Conference on Machine Learning*.

Biederman, I. (1987). Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147.

Bourdev, L., Maji, S., Brox, T., and Malik, J. (2010). Detecting people using mutually consistent poselet activations. In *Proceedings of the European Conference on Computer Vision*.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Bulat, A. and Tzimiropoulos, G. (2016). Human pose estimation via convolutional part heatmap regression. In *Proceedings of the European Conference on Computer Vision*. Springer.

Butko, N. J. and Movellan, J. R. (2009). Optimal scanning for faster object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Caesar, H., Uijlings, J., and Ferrari, V. (2015). Joint calibration for semantic segmentation. In *Proceedings of the British Machine Vision Conference*.

Caicedo, J. C. and Lazebnik, S. (2015). Active object localization with deep reinforcement learning. In *Proceedings of the International Conference on Computer Vision*, pages 2488–2496.

Carreira, J. and Sminchisescu, C. (2012). CPMC: Automatic object segmentation using constrained parametric min-cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1312–1328.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2016). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv*.

Chen, X., Mottaghi, R., Liu, X., Fidler, S., Urtasun, R., and Yuille, A. (2014). Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Chen, X. and Yuille, A. L. (2005). A time-efficient cascade for real-time object detection: With applications for the visually impaired. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

Chen, X. and Yuille, A. L. (2014). Articulated pose estimation by a graphical model with image dependent pairwise relations. In *Advances in Neural Information Processing Systems*.

Cheng, M.-M., Zhang, Z., Lin, W.-Y., and Torr, P. (2014). Bing: Binarized normed gradients for objectness estimation at 300fps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Choi, M., Lim, J., Torralba, A., and Willsky, A. (2010). Exploiting hierarchical context on a large database of object categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Cinbis, R., Verbeek, J., and Schmid, C. (2013). Segmentation driven object detection with fisher vectors. In *Proceedings of the International Conference on Computer Vision*.

Collet, A., Berenson, D., Srinivasa, S. S., and Ferguson, D. (2009). Object recognition and full pose registration from a single image for robotic manipulation. In *Proceedings of IEEE International Conference on Robotics and Automation*. IEEE.

Crandall, D. J. and Huttenlocher, D. P. (2007). Composite models of objects and scenes for category recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Criminisi, A., Shotton, J., and Konukoglu, E. (2011). Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. *Microsoft Research Cambridge, Tech. Rep. MSRTR-2011-114*.

Csurka, G., Bray, C., Dance, C., and Fan, L. (2004). Visual categorization with bags of keypoints. In *ECCV Workshop on Statistical Learning in Computer Vision*.

Dai, J., He, K., and Sun, J. (2016). Instance-aware semantic segmentation via multitask network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Dalal, N. and Triggs, B. (2005). Histogram of Oriented Gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Denil, M., Bazzani, L., Larochelle, H., and de Freitas, N. (2012). Learning where to attend with deep architectures for image tracking. *Neural Computation*, 24(8):2151–184.

Desai, C. and Ramanan, D. (2012). Detecting actions, poses, and objects with relational phraselets. In *Proceedings of the European Conference on Computer Vision*.

Desai, C., Ramanan, D., and Folkess, C. (2009). Discriminative models for multi-class object layout. In *Proceedings of the International Conference on Computer Vision*.

Divvala, S. K., Hoiem, D., Hays, J. H., Efros, A., and Hebert, M. (2009). An empirical study of context in object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Dollar, P. and Zitnick, C. (2014). Edge boxes: Locating object proposals from edges. In *Proceedings of the European Conference on Computer Vision*.

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 *International Conference on Machine Learning*.

Eigen, D., Rolfe, J., Fergus, R., and LeCun, Y. (2013). Understanding deep architectures using a recursive convolutional network. In *International Conference on Learning Representations*.

Endres, I., Shih, K., Jiaa, J., and Hoiem, D. (2013). Learning collections of part models for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2012). The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

Everingham, M., Van Gool, L., Williams, C. K. I., and Zisserman, A. (2006). The PASCAL Visual Object Classes Challenge 2006 (VOC2006).

Fanelli, G., Gall, J., and Van Gool, L. (2011). Real time head pose estimation with random regression forests. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Fei-Fei, L., Fergus, R., and Torralba, A. (2005). Recognizing and learning object categories. *Internationl Conference on Computer Vision Tutorial*.

Fei-Fei, L. and Perona, P. (2005). A Bayesian hierarchical model for learning natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Felzenszwalb, P., Girshick, R., and McAllester, D. (2010a). Cascade Object Detection with Deformable Part Models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Felzenszwalb, P., Girshick, R., McAllester, D., and Ramanan, D. (2010b). Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.

Felzenszwalb, P. and Huttenlocher, D. (2005). Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1).

Fergus, R., Perona, P., and Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Fergus, R., Perona, P., and Zisserman, A. (2005). A sparse object category model for efficient learning and exhaustive recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*.

Ferrari, V., Tuytelaars, T., and Van Gool, L. (2004). Simultaneous object recognition and segmentation by image exploration. In *Proceedings of the European Conference on Computer Vision*.

Fischler, M. and Elschlager, R. (1973). The representation and matching of pictorial structures. *IEEE Transactions on Computer*, c-22(1):67–92.

Fleuret, F. and Geman, D. (2001). Coarse-to-fine face detection. *International Journal of Computer Vision*, 41(1):85–107.

Fleuret, F. and Geman, D. (2008). Stationary features and cat detection. *Journal of Machine Learning Research*, 9(Nov):2549–2578.

Freund, Y. and Schapire, R. (1997). A decision theoretic generalisation of online learning. *Computer and System Sciences*, 55(1):119–139.

Gall, J. and Lempitsky, V. (2009). Class-specific hough forests for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Galleguillos, C., Rabinovich, A., and Belongie, S. (2008). Object categorization using co-occurrence, location and appearance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Gavves, E., Fernando, B., Snoek, C., Smeulders, A. W., and Tuytelaars, T. (2013). Fine-grained categorization by alignments. In *Proceedings of the International Conference on Computer Vision*.

Ge, Z., McCool, C., Sanderson, C., and Corke, P. (2015). Subset feature learning for fine-grained category classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Geman, D. and Jedynak, B. (1996). An active testing model for tracking roads in satellite images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1).

Girshick, R. (2015). Fast R-CNN. In *Proceedings of the International Conference on Computer Vision*.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Girshick, R., Shotton, J., Kohli, P., Criminisi, A., and Fitzgibbon, A. (2011). Efficient regression of general-activity human poses from depth images. In *Proceedings of the International Conference on Computer Vision*.

Gkioxari, G., Girshick, R., and Malik, J. (2015). Actions and attributes from wholes and parts. In *Proceedings of the International Conference on Computer Vision*.

Goering, C., Rodner, E., Freytag, A., and Denzler, J. (2014). Nonparametric part transfer for fine-grained recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Gong, Y. and Lazebnik, S. (2011). Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Gonzalez-Garcia, A., Modolo, D., and Ferrari, V. (2017a). Do semantic parts emerge in convolutional neural networks? *International Journal of Computer Vision*.

Gonzalez-Garcia, A., Modolo, D., and Ferrari, V. (2017b). Objects as context for part detection. *arXiv preprint arXiv:1703.09529*.

Gonzalez-Garcia, A., Vezhnevets, A., and Ferrari, V. (2015). An active search strategy for efficient object class detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*.

Goodfellow, I., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Hall, P. and Morton, S. C. (1993). On the estimation of entropy. *Annals of the Insititute of Statistical Mathematics*, 45(1):69–88.

Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. (2014). Simultaneous detection and segmentation. In *Proceedings of the European Conference on Computer Vision*.

Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. (2015). Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Harzallah, H., Jurie, F., and Schmid, C. (2009). Combining efficient object localization and image classification. In *Proceedings of the International Conference on Computer Vision*.

He, K., Zhang, X., Ren, S., and Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Proceedings of the European Conference on Computer Vision*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Heitz, G. and Koller, D. (2008). Learning spatial context: Using stuff to find things. In *Proceedings of the European Conference on Computer Vision*.

Hoiem, D., Efros, A. A., and Hebert, M. (2008). Putting objects in perspective. *International Journal of Computer Vision*, 80(1):3–15.

Hong, S., You, T., Kwak, S., and Han, B. (2015). Online tracking by learning discriminative saliency map with convolutional neural network. In *International Conference on Machine Learning*.

Hosang, J., Benenson, R., Dollár, P., and Schiele, B. (2016). What makes for effective detection proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):814–830.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., and Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Huang, S., Xu, Z., Tao, D., and Zhang, Y. (2016). Part-stacked cnn for fine-grained visual categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Hub, A., Diepstraten, J., and Ertl, T. (2004). Design and development of an indoor navigation and object identification system for the blind. In *ACM Sigaccess Accessibility and Computing*, number 77-78, pages 147–152.

Huttenlocher, D. P. and Ullman, S. (1987). Object recognition using alignment. In *Proceedings of the International Conference on Computer Vision*.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259.

Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

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

Jie, Z., Liang, X., Feng, J., Jin, X., Lu, W., and Yan, S. (2016). Tree-structured reinforcement learning for sequential object localization. In *Advances in Neural Information Processing Systems*.

Juneja, M., Vedaldi, A., Jawahar, C., and Zisserman, A. (2013). Blocks that shout: Distinctive parts for scene classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Karpathy, A., Shetty, S., Toderici, G., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Kong, Y., Jia, Y., and Fu, Y. (2014). Interactive phrases: Semantic descriptions for human interaction recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(9):1775–1788.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*.

Lampert, C. H., Blaschko, M. B., and Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Lan, T., Raptis, M., Sigal, L., and Mori, G. (2013). From subcategories to visual composites: A multi-level framework for object detection. In *Proceedings of the International Conference on Computer Vision*.

Larochelle, H. and Hinton, G. E. (2010). Learning to combine foveal glimpses with a third-order Boltzmann machine. In *Advances in Neural Information Processing Systems*.

Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

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

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., and Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Lee, J., Wang, J., Crandall, D., Šabanović, S., and Fox, G. (2017). Real-time, cloud-based object detection for unmanned aerial vehicles. In *Proceedings of IEEE International Conference on Robotic Computing*.

Lee, K., Buxton, H., and Feng, J. (2005). Cue-guided search: a computational model of selective attention. *IEEE transactions on neural networks*, 16(4):910–924.

Lehmann, A., Gehler, P. V., and Van Gool, L. J. (2011). Branch&rank: Non-linear object detection. In *Proceedings of the British Machine Vision Conference*.

Lenc, K. and Vedaldi, A. (2015a). R-cnn minus r. In *Proceedings of the British Machine Vision Conference*.

Lenc, K. and Vedaldi, A. (2015b). Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Li, C., Parikh, D., and Chen, T. (2011). Extracting adaptive contextual cues from unlabeled regions. In *Proceedings of the International Conference on Computer Vision*.

Li, C., Parikh, D., and Chen, T. (2012). Automatic discovery of groups of objects for scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Li, C. and Wand, M. (2016). Precomputed real-time texture synthesis with markovian generative adversarial networks. In *Proceedings of the European Conference on Computer Vision*.

Li, Y., He, K., and Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*.

Li, Y., Qi, H., Dai, J., Ji, X., and Wei, Y. (2017). Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Liang, X., Shen, X., Feng, J., Lin, L., and Yan, S. (2016a). Semantic object parsing with graph lstm. In *Proceedings of the European Conference on Computer Vision*.

Liang, X., Wei, Y., Shen, X., Jie, Z., Feng, J., Lin, L., and Yan, S. (2016b). Reversible recursive instance-level object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Lin, D., Shen, X., Lu, C., and Jia, J. (2015). Deep lac: Deep localization, alignment and classification for fine-grained recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

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 *Proceedings of the European Conference on Computer Vision*.

Liu, J., Kanazawa, A., Jacobs, D., and Belhumeur, P. (2012). Dog breed classification using part localization. In *Proceedings of the European Conference on Computer Vision*.

Liu, J., Li, Y., and Belhumeur, P. N. (2014). Part-pair representation for part localization. In *Proceedings of the European Conference on Computer Vision*.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Lowe, D. (1999). Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Lowe, D. G. (1987). Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31(3):355–395.

Mahamud, S. and Hebert, M. (2003). The optimal distance measure for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Mahendran, A. and Vedaldi, A. (2015). Understanding deep image representations by inverting them. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Malisiewicz, T., Gupta, A., and Efros, A. (2011). Ensemble of exemplar-svms for object detection and beyond. In *Proceedings of the International Conference on Computer Vision*.

Manen, S., Guillaumin, M., and Van Gool, L. (2013). Prime object proposals with randomized prim's algorithm. In *Proceedings of the International Conference on Computer Vision*.

Marin-Jimenez, M. J., Zisserman, A., Eichner, M., and Ferrari, V. (2014). Detecting people looking at each other in videos. *International Journal of Computer Vision*.

Mathe, S., Pirinen, A., and Sminchisescu, C. (2016). Reinforcement learning for visual object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Mikolajczyk, K. and Schmid, C. (2004). Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 1(60):63–86.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Mnih, V., Heess, N., and Graves, A. (2014). Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*.

Modolo, D., Vezhnevets, A., and Ferrari, V. (2015). Context forest for object class detection. In *Proceedings of the British Machine Vision Conference*.

Montillo, A. and Ling, H. (2009). Age regression from faces using random forests. In *Proceedings of the IEEE International Conference on Image Processing*.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. (2017). Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Moosman, F., Triggs, B., and Jurie, F. (2006). Fast discriminative visual codebook using randomized clustering forests. In *Advances in Neural Information Processing Systems*.

Moreels, P., Maire, M., and Perona, P. (2004). Recognition by probabilistic hypothesis construction. In *Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic*.

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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Murase, H. and Nayar, S. (1995). Visual learning and recognition of 3D objects from appearance. *International Journal of Computer Vision*, 14(1):5–24.

Murphy, K., Torralba, A., and Freeman, W. T. (2003). Using the forest to see the trees: A graphical model relating features, objects, and scenes. In *Advances in Neural Information Processing Systems*.

Navon, D. (1977). Forest before trees: The precedence of global features in visual perception. *Cognitive psychology*, 9(3):353–383.

Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

O'Hara, S. and Draper, B. A. (2013). Are you using the right approximate nearest neighbor algorithm? In *Proceedings of the IEEE Workshop on Applications of Computer Vision*.

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

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. (2015). Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Orabona, F., Metta, G., and Sandini, G. (2005). Object-based visual attention: a model for a behaving robot. In *Computer Vision and Pattern Recognition-Workshops*.

Ouyang, W., Wang, X., Zeng, X., Qiu, S., Luo, P., Tian, Y., Li, H., Yang, S., Wang, Z., Loy, C.-C., et al. (2015). Deepid-net: Deformable deep convolutional neural networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Parkhi, O., Vedaldi, A., Jawahar, C. V., and Zisserman, A. (2011). The truth about cats and dogs. In *Proceedings of the International Conference on Computer Vision*.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. (2012). Cats and dogs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Pearson, K. (1895). Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58:240–242.

Pedersoli, M., Vedaldi, A., and Gonzales, J. (2011). A coarse-to-fine approach for fast deformable object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Perronnin, F. and Dance, C. (2007). Fisher kernels on visual vocabularies for image categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Perronnin, F., Sánchez, J., and Mensink, T. (2010). Improving the fisher kernel for large-scale image classification. In *Proceedings of the European Conference on Computer Vision*.

Pfister, T., Charles, J., and Zisserman, A. (2015). Flowing convnets for human pose estimation in videos. In *Proceedings of the International Conference on Computer Vision*.

Pinheiro, M. A., Sznitman, R., Serradell, E., Kybic, J., Moreno-Noguer, F., and Fua, P. (2013). Active testing search for point cloud matching. In *Proceedings of the Information Processing in Medical Imaging*, pages 572–583. Springer.

Pinheiro, P. O., Lin, T.-Y., Collobert, R., and Dollár, P. (2016). Learning to refine object segments. In *Proceedings of the European Conference on Computer Vision*.

Pishchulin, L., Insafutdinov, E., Tang, S., Andres, B., Andriluka, M., Gehler, P. V., and Schiele, B. (2016). Deepcut: Joint subset partition and labeling for multi person pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Prest, A., Leistner, C., Civera, J., Schmid, C., and Ferrari, V. (2012). Learning object class detectors from weakly annotated video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., and Belongie, S. (2007). Objects in context. In *Proceedings of the International Conference on Computer Vision*.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Redmon, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*.

Rothganger, F., Lazebnik, S., Schmid, C., and Ponce, J. (2003). 3D object modeling and recognition using affine-invariant patches and multi-view spatial constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Rothwell, C., Zisserman, A., Mundy, J., and Forsyth, D. (1992). Efficient model library access by projectively invariant indexing functions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Rowley, H. A., Baluja, S., and Kanade, T. (1996). Human face detection in visual scenes. In *Advances in Neural Information Processing Systems*.

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. *International Journal of Computer Vision*.

Russel, B., Torralba, A., Liu, C., and Fergus, R. (2007). Object recognition by scene alignment. In *Advances in Neural Information Processing Systems*.

Russell, B., Torralba, A., Liu, C., Ferugs, R., and Freeman, W. (2007). Object recognition by scene alignment. In *Advances in Neural Information Processing Systems*.

Saberian, M. and Vasconcelos, N. (2014). Boosting algorithms for detector cascade learning. *Journal of Machine Learning Research*, 15(1):2569–2605.

Sadeghi, M. A. and Farhadi, A. (2011). Recognition using visual phrases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Schmid, C. and Mohr, R. (1996). Combining greyvalue invariants with local constraints for object recognition. Technical report, INRIA Rhône-Alpes, Grenoble, France.

Schneiderman, H. (2004). Feature-centric evaluation for efficient cascaded object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Schneiderman, H. and Kanade, T. (2004). Object detection using the statistics of parts. *International Journal of Computer Vision*, 56(3):151–177.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations*.

Serradell, E., Pinheiro, M. A., Sznitman, R., Kybic, J., Moreno-Noguer, F., and Fua, P. (2015). Non-rigid graph registration using active testing search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):625–638.

Shih, K. J., Mallya, A., Singh, S., and Hoiem, D. (2015). Part localization using multi-proposal consensus for fine-grained categorization. In *Proceedings of the British Machine Vision Conference*.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in parts from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Shrivastava, A. and Gupta, A. (2016). Contextual priming and feedback for faster r-cnn. In *Proceedings of the European Conference on Computer Vision*.

Simon, M. and Rodner, E. (2015). Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *Proceedings of the International Conference on Computer Vision*.

Simon, M., Rodner, E., and Denzler, J. (2014). Part detector discovery in deep convolutional neural networks. In *Proceedings of the Asian Conference on Computer Vision*.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR workshop*.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Sivic, J., Russell, B. C., Efros, A. A., Zisserman, A., and Freeman, W. T. (2005). Discovering object categories in image collections. In *Proceedings of the International Conference on Computer Vision*.

Song, H., Lee, Y., Jegelka, S., and Darell, T. (2014). Weakly-supervised discovery of visual pattern configurations. In *Advances in Neural Information Processing Systems*.

Song, Z., Chen, Q., Huang, Z., Hua, Y., and Yan, S. (2011). Contextualizing object detection and classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Sun, M. and Savarese, S. (2011). Articulated part-based model for joint object detection and pose estimation. In *Proceedings of the International Conference on Computer Vision*.

Sun, Q. and Batra, D. (2015). Submodboxes: Near-optimal search for a set of diverse object proposals. In *Advances in Neural Information Processing Systems*.

Sun, Y. and Fisher, R. (2003). Object-based visual attention for computer vision. *Artificial intelligence*, 146(1):77–123.

Sun, Y., Fisher, R., Wang, F., and Gomes, H. M. (2008). A computer vision model for visual-object-based attention and eye movements. *Computer Vision and Image Understanding*, 112(2):126–142.

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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Sznitman, R. and Jedynak, B. (2010). Active testing for face detection and localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1914–1920.

Tang, Y., Srivastava, N., and Salakhutdinov, R. (2014). Learning generative models with visual attention. In *Advances in Neural Information Processing Systems*.

Tompson, J. J., Jain, A., LeCun, Y., and Bregler, C. (2014). Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in Neural Information Processing Systems*.

Torralba, A. (2003). Contextual priming for object detection. *International Journal of Computer Vision*, 53(2):153–167.

Tsai, Y.-H., Hamsici, O. C., and Yang, M.-H. (2015). Adaptive region pooling for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Tuytelaars, T. and Van Gool, L. (2004). Matching widely separated views based on affine invariant regions. *International Journal of Computer Vision*, 59(1):61–85.

Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. (2013). Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171.

Ukita, N. (2012). Articulated pose estimation with parts connectivity using discriminative local oriented contours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Van de Sande, K., Uijlings, J., Gevers, T., and Smeulders, A. (2011). Segmentation as selective search for object recognition. In *Proceedings of the International Conference on Computer Vision*.

van de Sande, K. E. A. and Gevers, T. (2010). *University of Amsterdam at the Visual Concept Detection and Annotation Tasks*, volume 32: ImageCLEF, chapter 18, pages 343–358. Springer.

Van De Sande, K. E. A., Gevers, T., and Snoek, C. G. M. (2010). Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596.

Vedaldi, A., Gulshan, V., Varma, M., and Zisserman, A. (2009). Multiple kernels for object detection. In *Proceedings of the International Conference on Computer Vision*.

Vedaldi, A. and Lenc, K. (2015). Matconvnet – convolutional neural networks for MATLAB. In *ACM Multimedia*.

Vedaldi, A., Mahendran, S., Tsogkas, S., Maji, S., Girshick, R., Kannala, J., Rahtu, E., Kokkinos, I., Blaschko, M. B., Weiss, D., et al. (2014). Understanding objects in detail with fine-grained attributes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Vogel, J. and Schiele, B. (2002). Query-dependent performance optimization for vocabulary-supported image retrieval. In *Joint Pattern Recognition Symposium*, pages 600–608. Springer.

Vu, T.-H., Osokin, A., and Laptev, I. (2015). Context-aware cnns for person head detection. In *Proceedings of the International Conference on Computer Vision*.

Wah, C., Branson, S., Perona, P., and Belongie, S. (2011a). Multiclass recognition and part localization with humans in the loop. In *Proceedings of the International Conference on Computer Vision*.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011b). The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Walther, D. and Koch, C. (2006). Modeling attention to salient proto-objects. *Neural networks*, 19(9):1395–1407.

Wang, J. and Yuille, A. (2015). Semantic part segmentation using compositional model combining shape and appearance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Wang, L., Ouyang, W., Wang, X., and Lu, H. (2015a). Visual tracking with fully convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Wang, P., Shen, X., Lin, Z., Cohen, S., Price, B., and Yuille, A. L. (2015b). Joint object and part segmentation using deep learned potentials. In *Proceedings of the International Conference on Computer Vision*.

Wang, X., Yang, M., Zhu, S., and Lin, Y. (2013). Regionlets for generic object detection. In *Proceedings of the International Conference on Computer Vision*.

Weber, M., Welling, M., and Perona, P. (2000). Unsupervised learning of models for recognition. In *Proceedings of the European Conference on Computer Vision*, pages 18–32.

Wei, S.-E., Ramakrishna, V., Kanade, T., and Sheikh, Y. (2016). Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Wu, B., Iandola, F., Jin, P. H., and Keutzer, K. (2017). Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Wu, T. and Zhu, S. (2013). Learning near-optimal cost-sensitive decision policy for object detection. In *Proceedings of the International Conference on Computer Vision*.

Xia, F., Wang, P., Chen, L.-C., and Yuille, A. L. (2016). Zoom better to see clearer: Human and object parsing with hierarchical auto-zoom net. In *Proceedings of the European Conference on Computer Vision*.

Xiao, J., Hays, J., Ehinger, K., Oliva, A., and Torralba, A. (2010). SUN database: Large-scale scene recognition from Abbey to Zoo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Xiao, J., Hays, J., Ehinger, K., Oliva, A., and Torralba, A. (2012). SUN database. http://groups.csail.mit.edu/vision/SUN/.

Xiao, T., Xu, Y., Yang, K., Zhang, J., Peng, Y., and Zhang, Z. (2015). The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Yang, J., Price, B., Cohen, S., and Yang, M.-H. (2014). Context driven scene parsing with attention to rare classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Yang, S., Luo, P., Loy, C.-C., and Tang, X. (2015). From facial parts responses to face detection: A deep learning approach. In *Proceedings of the International Conference on Computer Vision*.

Yang, W., Ouyang, W., Li, H., and Wang, X. (2016). End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Ye, M. and Yang, R. (2014). Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*.

Yu, C.-N. and Joachims, T. (2009). Learning structural svms with latent variables. In *International Conference on Machine Learning*.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Proceedings of the European Conference on Computer Vision*.

Zhang, H., Xu, T., Elhoseiny, M., Huang, X., Zhang, S., Elgammal, A., and Metaxas, D. (2016). Spda-cnn: Unifying semantic part detection and abstraction for fine-grained recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Zhang, N., Donahue, J., Girshick, R., and Darrell, T. (2014a). Part-based r-cnns for fine-grained category detection. In *Proceedings of the European Conference on Computer Vision*.

Zhang, N., Farrell, R., Iandola, F., and Darrell, T. (2013). Deformable part descriptors for fine-grained recognition and attribute prediction. In *Proceedings of the International Conference on Computer Vision*.

Zhang, N., Paluri, M., Ranzato, M., Darrell, T., and Bourdev, L. (2014b). Panda: Pose aligned networks for deep attribute modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2015). Object detectors emerge in deep scene cnns. In *International Conference on Learning Representations*.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014). Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*.

Zhu, M., Atanasov, N., Pappas, G., and Daniilidis, K. (2014). Active Deformable Part Models Inference. In *Proceedings of the European Conference on Computer Vision*.

Zitnick, C. L. and Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In *Proceedings of the European Conference on Computer Vision*.