

SCALING UP OBJECT DETECTION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Olga Russakovsky

August 2015

© Copyright by Olga Russakovsky 2015
All Rights Reserved

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Fei-Fei Li) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Yuanqing Lin)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Michael Bernstein)

Approved for the Stanford University Committee on Graduate Studies

Preface

Hundreds of billions of photographs are created on the web each year. An important step towards understanding the content of these photographs is to be able to understand all objects that are depicted. My research focuses on the problem of automatically naming and localizing objects in large collections of images. This is referred to as the task of *object detection*.

There are multiple challenges that I have addressed that help scale up object detection algorithms in both the number of images and the number of objects that can be recognized. First, algorithms for analyzing such large collections of images need to be very efficient. I've developed an object detection algorithm that was an order of magnitude faster than previous methods. Second, recognizing all objects in images is difficult because of the large diversity of object classes. I've studied using shareable generic object attribute descriptions to effectively describe a variety of object classes without learning individual object models. Third, extensive manual annotation is required for training object detectors at scale, which can be very time-consuming and expensive. The core of my thesis focuses on this third challenge.

I've addressed the challenge of required extensive manual annotation in three ways. First, I developed a weakly supervised learning method for localizing objects in images using just cheap binary-level image labels. This method eliminates the need for significantly more expensive bounding box annotations at a small reduction in accuracy.

However, even these simple binary labels are very expensive to obtain at the scale of hundreds of object classes and tens of thousands of images, leading to my second contribution. Therefore, my colleagues and I created the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). ILSVRC serves as a benchmark large-scale object recognition for hundreds of international research teams. I led the effort to construct the object detection benchmark, scaling up by more than an order of magnitude compared to previous dataset (e.g., the PASCAL VOC [48]). The construction of this dataset required developing novel crowd engineering techniques for reducing annotation cost. By taking on the massive labeling task ourselves on behalf of the community, we were able to eliminate the need for costly annotation by individual research teams. Further, the availability of this large-scale data enabled us to perform detailed analysis of the current state of the field of object recognition, providing insights for future research efforts.

The ILSVRC dataset and corresponding competition enabled incredible progress in object recognition accuracy over the past few years. Thinking ahead about scaling up object detection even further, I developed a framework for bringing together the state-of-the-art automatic large-scale object detection with state-of-the-art crowd engineering techniques into a principled human-in-the-loop framework for accurately and efficiently localizing objects in images.

By designing efficient object detection algorithms and using novel crowd engineering techniques to quickly label large amounts of data, we hope to be able to localize most objects in large collections of images.

To my parents Larisa and Alex Russakovsky.

Acknowledgments

First and foremost, thank you to my advisor Fei-Fei Li. She has been my mentor, my friend and my rock through these years.

Thank you to all my co-authors whose work is featured in this thesis (in alphabetical order): Alexander Berg, Michael Bernstein, Jia Deng, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Jonathan Krause, Fei-Fei Li, Jia Li, Yuanqing Lin, Sean Ma, Andrew Ng, Sanjeev Satheesh, Hao Su and Kai Yu. Thank you for great discussions that contributed to this work to: Anelia Angelova, Lamberto Ballan, Lubomir Bourdev, Timothee Cour, Alexei Efros, Derek Hoiem, Chang Huang, Justin Johnson, Quoc Le, Jitendra Malik, Serge Plotkin, Chuck Rosenberg, Ashutosh Saxena, Kevin Tang, Bangpeng Yao, Andrew Zisserman and Shenghuo Zhu. Thank you to my other co-authors during the PhD: Misha Andriluka, Paul Baumstark, Amy Bearman, Blake Carpenter, Vittorio Ferrari, Ian Goodfellow, Stephen Gould, Jenny Jin, Ellen Klingbeil, Daphne Koller, Davide Modolo, Greg Mori, Alexander Vezhnevets and Serena Yeung.

Thank you to all my research mentors over the years: Serafim Batzoglou, Sam Gross and Tom Do during my undergraduate years; Andrew Ng, Stephen Gould and Alexis Battle during my early PhD years; Fei-Fei Li, Jia Deng, Armand Joulin, Jia Li, Juan Carlos Niebles, Alex Berg and Greg Mori during my later PhD years; and Yuanqing Lin and Anelia Angelova during my collaboration with NEC Labs America.

Thank you to my thesis readers Michael Bernstein and Yuanqing Lin and to my defense committee members Mykel Kochenderfer and Bernd Girod.

Thank you to NSF, ONR MURI, NEC Labs America, UNC Chapel Hill, Google and Facebook for sponsoring parts of this research.

Thank you to my vision lab family, to my friends (who are really more like family) and to my family. Thank you to my dad for inspiring me to do a PhD in the first place, and to both my mom and my dad for helping in an impressive variety of ways throughout the program.

Contents

Preface	iv
Acknowledgments	vii
1 Introduction	1
2 A Steiner tree approach to efficient object detection	4
2.1 Introduction	4
2.2 Related work	5
2.3 Fast object detection	7
2.4 Segmentation and rectangle selection	8
2.4.1 Segmentation	8
2.4.2 Rectangle selection	8
2.4.3 Trimming parameter	9
2.5 Steiner Trees for Parameter Selection	10
2.5.1 Directed Steiner Trees	10
2.5.2 Constructing the Steiner Tree	11
2.5.3 Approximation Algorithms	12
2.6 Experiments	12
2.6.1 Training stage 1: Object classifiers	13
2.6.2 Training stage 2: Steiner trees	13
2.6.3 Evaluation	14
Indoor scenes	14
Outdoor scenes	15
2.7 Conclusions	16
3 Attribute learning in large-scale datasets	18
3.1 Introduction	18
3.2 Learning visual connections in ImageNet	19

3.3	Related work	20
3.4	Building and labeling an attribute dataset	22
3.5	Experiments	23
3.5.1	Implementation	24
3.5.2	Learning image attributes	24
3.5.3	Transfer learning using attributes	26
3.5.4	Synset-level connections	29
3.6	Conclusions	31
4	Simultaneous image classification and object localization	34
4.1	Introduction	34
4.2	Related work	36
4.3	Object-centric spatial pooling (OCP) for image classification	37
4.3.1	Classification formulation	38
4.3.2	Foreground-background feature representation	39
4.3.3	Optimization	40
4.4	Experiments	41
4.4.1	Joint classification and localization	42
4.4.2	Image classification	43
4.4.3	Weakly supervised object localization	44
4.5	Conclusions	45
5	ImageNet Large Scale Visual Recognition Challenge	47
5.1	Introduction	47
5.2	Related work	49
5.3	Challenge tasks	50
5.3.1	Image classification task	51
5.3.2	Single-object localization task	52
5.3.3	Object detection task	52
5.4	Dataset statistics	53
5.4.1	Image classification and single-object localization	53
5.4.2	Object detection	54
5.5	Evaluation at large scale	55
5.5.1	Image classification	56
5.5.2	Single-object localization	56
5.5.3	Object detection	58
5.6	Methods	60
5.6.1	Challenge entries	60

5.6.2	Large scale algorithmic innovations	62
5.7	Results and analysis	63
5.7.1	Improvements over the years	63
5.7.2	Statistical significance	65
5.8	Conclusions	65
5.8.1	Lessons learned	65
5.8.2	Criticism	67
5.8.3	The future	68
6	Large-scale object detection dataset construction	69
6.1	Introduction	69
6.2	Defining object categories	70
6.3	Collecting scene images	71
6.4	Scalable multi-label annotation	74
6.4.1	Overview	75
6.4.2	Related Work	76
6.4.3	Approach	76
6.4.4	Experiments	79
6.4.5	Discussion and Conclusion	81
6.5	Bounding box system for object detection	81
6.5.1	ILSVRC bounding box object annotation system	81
6.5.2	Object detection modifications	83
6.6	Conclusions	84
7	Analysis of large-scale object recognition accuracy	86
7.1	Introduction	86
7.2	Related work	88
7.3	Factors for analysis	89
Image-level properties	89
Intrinsic object class properties.	90
7.4	Analysis of the deep learning breakthrough	91
7.4.1	Setup	91
7.4.2	Algorithms	92
7.4.3	Recognition accuracy as a function of object hierarchy	92
7.4.4	Effect of image-level statistics on localization accuracy	93
7.4.5	Effect of intrinsic object class properties on localization accuracy	95
7.4.6	Conclusions from year 2012	99
7.5	Current state of categorical object recognition	100

7.5.1	Setup	100
7.5.2	Range of accuracy across object classes	101
7.5.3	Qualitative examples of easy and hard classes	101
7.5.4	Effect of image-level statistics on recognition accuracy	102
7.5.5	Effect of intrinsic object class properties on recognition accuracy	104
7.6	Conclusions	110
8	Human-machine collaboration for object annotation	111
8.1	Introduction	111
8.2	Related work	113
8.3	Problem formulation	113
8.4	Method	114
8.4.1	Annotation evaluation	114
8.4.2	MDP formulation for human task selection	116
8.5	Human-in-the-loop probabilistic framework	118
8.5.1	Incorporating computer vision input	120
8.5.2	Incorporating user input	121
8.6	Experiments	124
8.6.1	Setup	124
8.6.2	Human annotation design and observations	126
8.6.3	Evaluating labeling quality	128
	Reducing error rates	130
8.6.4	Satisfying requester constraints	131
8.7	Conclusions	132
9	Conclusions	133
9.1	Lessons Learned	134
9.2	Future directions	135
A	Detailed analysis of the ILSVRC localization dataset	136
B	Analysis of the top-5 ILSVRC evaluation criteria	139
C	Manually curated queries for ILSVRC detection images	141
D	Hierarchy of questions for ILSVRC detection annotation	143
E	User interfaces for human-machine object annotation	152
	Bibliography	162

List of Tables

2.1	Test set performance of each parameter selection method on the supplemented LabelMe dataset. Detection time and the average number of windows analyzed are reported per image per object. The “Average” row contains the average area under the PR curve over all 9 objects. Performance superior to the sliding windows approach is bold-faced.	15
3.1	Examples of synsets labeled positive by mining WordNet definitions, by both WordNet and AMT labelers, and just by AMT labelers.	23
3.2	Performance of attribute classifiers as measured by the area under the ROC curve.	25
3.3	On the left are the animal classes and the corresponding human attribute annotations, and on the right is the confusion table from the transfer learning experiments. The rows of the confusion table are the ground truth labels and the columns are the classifier outputs.	29
4.1	Classification AP of object-centric spatial pooling compared to the standard SPM spatial pooling on the PASCAL07 test set.	43
4.2	Comparison of detection AP on the PASCAL07-6x2 test set for our method versus [40, 136]. Both [40, 136] split up the objects by left and right viewpoint to make the models easier to learn. We do not make use of these additional labels and learn a single model for each object.	45
5.1	Overview of the provided annotations for each of the tasks in ILSVRC.	51
5.2	Scale of ILSVRC image classification task (top) and single-object localization task (bottom). The numbers in parentheses correspond to (minimum per class - maximum per class). The 1000 classes change from year to year but are consistent between image classification and single-object localization tasks in the same year. All images from the image classification task may be used for single-object localization.	53
5.3	Scale of ILSVRC object detection task. Numbers in parentheses correspond to (minimum per class - median per class - maximum per class).	55

5.4	We use bootstrapping to construct 99.9% confidence intervals around the result of up to top 5 submissions to each ILSVRC task in 2012-2014. †means the entry used external training data. The winners using the provided data for each track and each year are bolded. The difference between the winning method and the runner-up each year is significant even at the 99.9% level.	66
6.1	Correspondences between the object classes in the PASCAL VOC [48] and the ILSVRC detection task. Object scale is the fraction of image area (reported in percent) occupied by an object instance. It is computed on the validation sets of PASCAL VOC 2012 and of ILSVRC-DET. The average object scale is 24.1% across the 20 PASCAL VOC categories and 20.3% across the 20 corresponding ILSVRC-DET categories.	71
6.2	The most useful queries at the first iteration of our algorithm. Utility is the expected number of new values for object labels as a result of this query. Cost is the human time needed (in seconds) to obtain the correct answer with a minimum of 95% accuracy on expectation.	80
6.3	Our algorithm versus the naïve brute force approach. Thresh is a parameter of the algorithm (please refer to text for detail).	81
8.1	Human annotations tasks. One important property of our model is that it will automatically find the best question to pose, so there’s no harm in adding extra tasks.	117
8.2	For every event E (row) and question (column), the table reports what the <i>true</i> answer to the question if the event E happened. ✓ means “yes” is the true answer, ✗ means “no” is the true answer, and – means that event E provides no information about the true answer. – is the default when not specified. Here every question is treated as a binary question: for example, for draw-box question, the answer of drawing a box is simply “yes” and the answer of refusing to draw a box is “no”.	122
8.3	Human annotations tasks with the corresponding accuracy rates and costs. Detailed explanations of each task are in Table 8.1. FP column is the false positive probability of user answering “no” (or refusing to draw a box, or write a name) when the answer should in fact be “yes.” For the open-ended tasks, if the answer was given, we also need to estimate the probabilities of the given answer being <i>wrong</i> : these probability are draw-box 0.29, name-object 0.06, name-image 0.05. FN column is the true negative probability of the user answering “yes” when the answer should be “no.” Cost is median human time in seconds.	126
8.4	Cost of each task (in median human time) broken down by positive versus negative responses. On average, positive responses take longer than negative ones. One interesting potential extension to our human-machine object annotation framework would be to incorporate this fact.	128

A.1	Comparison of the PASCAL VOC 2012 object detection dataset and the ILSVRC 2012 single-object localization dataset on several key properties of object localization difficulty (please see Section 5.4.1 for definitions). For each property, the ILSVRC subset is chosen by selecting the largest set of ILSVRC classes with the same average difficulty as that of the PASCAL classes. The number of object classes is indicated in parentheses. Note that according to any of these measures of difficulty there is a subset of ILSVRC which is as challenging as PASCAL but more than an order of magnitude greater in size.	137
-----	---	-----

List of Figures

1.1	This thesis studies the problem of <i>object detection</i> , or automatically naming and localizing objects in images. Instead of focusing on just a small subset of object classes [48] we aim to scale up object detection to hundreds of object categories and hundreds of thousands of images.	2
2.1	Sample images from our object detection dataset with the desired objects outlined in green.	7
2.2	<i>Left.</i> Original image. <i>Center.</i> Segmentation with the parameter $k = 1600$. It works remarkably very well for ski boots (shown in blue) and contains only 88 segments, most of them too small or too big to even be considered by the classifier. However, there is no hope of recovering any of the smaller objects, such as the coffee mugs and paper cups on the table. <i>Right.</i> Segmentation with $k = 100$ ($s = 0.8$, $m = 20$). Notice that good bounding boxes around the mugs and cups can now be reconstructed by combining a small number of segments, yet ski boots are extremely over-segmented and thus very difficult to reconstruct.	9

2.3	<i>Left.</i> Illustration of the five sequential steps of our segmentation and rectangle selection pipeline described in section 2.4. (1) Smooth the original image ($s = 1$ shown). (2) Segment ($k = 150$ shown). (3) Merge small segments together ($m = 160$ shown). (4) Propose rectangular bounding boxes around groups of segments (only boxes around individual segments are shown). (5) Trim each of the bounding boxes using parameter p to eliminate long sparse tails ($p = 40$ here). The bounding box shown is obtained by merging three vertically aligned segments (purple, brown and blue, top to bottom). The new bounding box is tight enough for the classification algorithm to detect the paper cup. (Classification) Finally, evaluate each proposed rectangle with the object classifier. Note that the picture shown here is for illustration purposes only; there is no single assignment to the five segmentation and rectangle selection parameters that would allow the classifier to perform well on all 3 object types in this image. <i>Right.</i> Illustration of the directed graph G generated by the Steiner tree algorithm, described in section 2.5.	10
2.4	Accuracy of our algorithm on 355 images from the StreetScenes dataset. The blue line corresponds to the sliding windows algorithm, and the red bold line is our algorithm.	16
3.1	The goal of our work is to build visual connections between object categories. We focus on the large-scale ImageNet dataset which currently uses WordNet [128] to provide a semantic hierarchy provides a semantic hierarchy of categories. Discovering a visual hierarchy would be useful for a variety of tasks; for example, targeted retrieval.	19
3.2	Example images of synsets that are direct descendants of the edible fruit synset. First, the high variability within each of the four synsets makes classification on this dataset very challenging. Second, the four object classes are sibling synsets in WordNet since they are all children of the “edible fruit” synset; however, visually they are quite different from each other in terms of color, texture and shape.	20
3.3	Some example images labeled by the human subjects as “striped.” There is large intra-class variation, making this a challenging computer vision dataset.	24
3.4	Performance of attribute classifiers sorted by attribute type. The average performance of each type is reported in parentheses. Performance is measured by the area under the ROC curve.	25
3.5	Visualization of four of the learned attributes (for the other attributes, see Figures 3.6 and 3.7). For each attribute, the 5 rows represent the 5 training folds, and each row shows the top 8 images retrieved from among all synsets that didn’t appear in that fold’s training set. The border around each image corresponds to the human labeler annotation (green is positive, red is negative, yellow is ambiguous).	26
3.6	Continuation of Figure 3.5 visualizing the learned attributes.	27
3.7	Continuation of Figures 3.5 and 3.6 visualizing the learned attributes.	28

3.8	This figure shows the top 10 synsets that were returned by the algorithm as the most representative for a subset of the attributes (see Figures 3.9 and 3.10 for the remainder). The number in parenthesis represents the median probability assigned to images within that synset by the attribute classifier.	30
3.9	Continuation of Figure 3.8 showing the visual connections made between synsets. . .	32
3.10	Continuation of Figures 3.8 and 3.9 showing the visual connections made between synsets.	33
4.1	We present <i>object-centric spatial pooling</i> (OCP), a method which first localizes the object of interest and then pools foreground object features separately from background features. In contrast, Spatial Pyramid Matching (SPM) based pooling [112] (top), the most common spatial pooling method for object classification, results in inconsistent image features when the object of interest (here, a car) appears in different locations within images, making it more difficult to learn an appearance model of the object. For the purpose of easy illustration, circles (yellow) denote object-related local features, triangles (green) denote background-related local features, and the numbers indicate the fraction of the respective local features in each pooling region.	35
4.2	A popular image classification pipeline of state-of-the-art methods (at the time of publication of this work) [203, 227, 215]. In this chapter we focus on the pooling step and propose an object-centric spatial pooling approach which achieves superior classification accuracy compared to the SPM pooling.	36
4.3	Example images which were misclassified using just the foreground representation but correctly classified when using the foreground-background representation.	39
4.4	Bounding boxes bb_1 and bb_2 have a similar foreground-only feature representation, but they are very different under the foreground-background representation. Here, the numbers denote the count of object-related descriptors. For bb_1 , parts of object that leaked into the background will be greatly discounted by the background model.	40
4.5	Classification and detection mAP on the PASCAL07 test set over the iterations of our joint detection and classification approach. The red solid line is classification mAP, and the blue dotted line is detection mAP. We see a steady joint improvement of classification and detection accuracy.	42
4.6	Images where object-centric pooling with the foreground-background model (yellow) localizes objects more accurately than the foreground-only model (green).	44
4.7	Foreground regions detected by the object-centric pooling framework on PASCAL07 test images. The models are learned without any ground truth localization information. Yellow boxes correspond to correct detections and red boxes are failed detections. On images where multiple instances of a object class are presented, we show the top few detections after running non-maximal suppression.	46

5.1	The ILSVRC classification and single-object localization dataset contains many more fine-grained classes compared to the standard PASCAL VOC benchmark; for example, instead of the PASCAL "dog" category there are 120 different breeds of dogs in ILSVRC.	54
5.2	Tasks in ILSVRC. The first column shows the ground truth labeling on an example image, and the next three show three sample outputs with the corresponding evaluation score.	57
5.3	Images marked as "difficult" in the ILSVRC2012 single-object localization validation set. Please refer to Section 5.5.2 for details.	58
5.4	Performance of winning entries in the ILSVRC2010-2014 competitions in each of the three tasks (details about the entries and numerical results are in Section 5.6.1). There is a steady reduction of error every year in object classification and single-object localization tasks, and a 1.9x improvement in mean average precision in object detection. There are two considerations in making these comparisons. (1) The object categories used in ILSVRC changed between years 2010 and 2011, and between 2011 and 2012. However, the large scale of the data (1000 object categories, 1.2 million training images) has remained the same, making it possible to compare results. Image classification and single-object localization entries shown here use only provided training data. (2) The size of the object detection training data has increased significantly between years 2013 and 2014 (Section 5.4.2). Section 5.7.1 discusses the relative effects of training data increase versus algorithmic improvements.	63
6.1	Summary of images collected for the detection task. Images in green (bold) boxes have all instances of all 200 detection object classes fully annotated.	72
6.2	Random selection of images in ILSVRC detection validation set. The images in the top 4 rows were taken from ILSVRC2012 single-object localization validation set, and the images in the bottom 4 rows were collected from Flickr using scene-level queries.	73
6.3	Multi-label annotation becomes much more efficient when considering real-world structure of data: correlation between labels, hierarchical organization of concepts, and sparsity of labels.	75
6.4	Our algorithm dynamically selects the next query to efficiently determine the presence or absence of every object in every image. Green denotes a positive annotation and red denotes a negative annotation. This toy example illustrates a sample progression of the algorithm for one label (cat) on a set of images.	77
6.5	The Amazon Mechanical Turk interface for obtaining human annotations. Here workers are asked to select images which contain a rabbit, and are shown good and bad example images.	79

7.1	The diversity of ILSVRC along eight dimensions. Please refer to Section 7.3 for the definitions. For each dimension, we show example object categories along the range of that property.	87
7.2	(a) Confusion matrix of classification scores using the SV algorithm, with categories ordered semantically. The expected block pattern shows most confusion between similar categories. (b) Classification and (c) classification with localization accuracy as a function of moving up the ImageNet hierarchy. [36] At level 0 the algorithm is required to produce the exact class label; at level 1 any sister label is accepted, at level 10 any leaf node of generic concepts such as "living thing" is accepted.	93
7.3	The impact of several quantitative measures of localization difficulty (Section 7.3) on the localization accuracy. Each dot corresponds to one of the 1000 object categories of ILSVRC2012 and to one of the two algorithms (SV in red, VGG in blue). The difficulty measure (x-axis) is computed on the validation set; the accuracy of the algorithm (y-axis) is evaluated on the test set. The best fit linear models for each algorithm are also shown to summarize trends. The black line corresponds to the upper bound combination of SV and VGG; the accuracy on individual class of this method is not shown to reduce clutter. Please refer to Section 7.4.4 for analysis.	94
7.4	Cumulative localization of SV (red) and VGG (blue) as a function of chance performance of localization (CPL). The height of the curve corresponds to the average accuracy of the object categories with equal or smaller CPL measures. SV outperforms VGG except when considering a subset of 225 object categories with lowest CPL.	95
7.5	Localization accuracy of Upper Bound (gray), SV (red) and VGG (blue) as a function of the intrinsic properties of the ILSVRC2012 object categories. Plots (a), (c), (d) show the average localization accuracy of algorithms on subsets of ILSVRC. Plot(b) shows the cumulative localization accuracy as a function of the chance performance of localization (Section 7.3). The height of the curve corresponds to the average accuracy of the object categories with equal or smaller CPL measures. Continued in Figures 7.6 and Figure 7.7.	96
7.6	Continuation of Figure 7.5. Localization accuracy of Upper Bound (gray), SV (red) and VGG (blue) as a function of the intrinsic properties of the ILSVRC2012 object categories. All plots show the average localization accuracy of algorithms on subsets of ILSVRC. Plots (c) and (d) additionally report classification accuracy (white bars). Continued in Figure 7.7.	97

7.7	Continuation of Figures 7.5 and 7.6. Localization accuracy of Upper Bound (gray), SV (red) and VGG (blue) as a function of the intrinsic properties of the ILSVRC2012 object categories. All plots show the average localization accuracy of algorithms on subsets of ILSVRC. Plot (a) shows the average localization accuracy (colored bars) and classification accuracy (white bars) of the algorithms on subsets of ILSVRC. Plot (b) shows the cumulative localization accuracy as a function of the chance performance of localization (Section 7.3). The y-axis corresponds to the localization accuracy <i>only on images which were correctly classified</i>	99
7.8	For each object class, we consider the best performance of any entry submitted to ILSVRC2012-2014, including entries using additional training data. The plots show the distribution of these “optimistic” per-class results. Performance is measured as accuracy for image classification (left) and for single-object localization (middle), and as average precision for object detection (right). While the results are very promising in image classification, the ILSVRC datasets are far from saturated: many object classes continue to be challenging for current algorithms.	101
7.9	For each object category, we take the best performance of any entry submitted to ILSVRC2012-2014 (including entries using additional training data). Given these “optimistic” results we show the easiest and hardest classes for the classification task. The numbers in parentheses indicate classification accuracy. The 10 easiest classes are randomly selected from among 121 object classes with 100% accuracy. Results from other tasks are shown in Figures 7.10 and 7.11.	102
7.10	For each object category, we take the best performance of any entry submitted to ILSVRC2012-2014 (including entries using additional training data). Given these “optimistic” results we show the easiest and hardest classes for the localization task. The numbers in parentheses indicate localization accuracy. Results from other tasks are shown in Figures 7.9 and 7.11.	103
7.11	For each object category, we take the best performance of any entry submitted to ILSVRC2012-2014 (including entries using additional training data). Given these “optimistic” results we show the easiest and hardest classes for the object detection task. The numbers in parentheses indicate average precision. Results from other tasks are shown in Figures 7.9 and 7.10.	104

7.12	Performance of the “optimistic” method as a function of object scale in the image, on each task. Each dot corresponds to one object class. Average scale (x-axis) is computed as the average fraction of the image area occupied by an instance of that object class on the ILSVRC2014 validation set. “Optimistic” performance (y-axis) corresponds to the best performance on the test set of any entry submitted to ILSVRC2012-2014 (including entries with additional training data). The test set has remained the same over these three years. We see that accuracy tends to increase as the objects get bigger in the image. However, it is clear that far from all the variation in accuracy on these classes can be accounted for by scale alone.	105
7.13	Performance of the “optimistic” computer vision model as a function of object properties. The x-axis corresponds to object properties annotated by human labelers for each object class, described in Section 7.3 and illustrated in Figure 7.1. The y-axis is the average accuracy of the “optimistic” model. Note that the range of the y-axis is different for each task to make the trends more visible. The black circle is the average accuracy of the model on all object classes that fall into each bin. We control for the effects of object scale by normalizing the object scale within each bin (details in Section 7.5.5). The color bars show the model accuracy averaged across the remaining classes. Error bars show the 95% confidence interval obtained with bootstrapping. Some bins are missing color bars because less than 5 object classes remained in the bin after scale normalization. For example, the bar for XL real-world object detection classes is missing because that bin has only 3 object classes (airplane, bus, train) and after normalizing by scale no classes remain. Continued in Figures 7.14 and 7.15. . .	106
7.14	Continuation of Figure 7.13. Please see its caption for details.	107
7.15	Continuation of Figures 7.13 and 7.14. Please see Figure 7.13 for details.	108
8.1	This cluttered image has 100 annotated objects shown with green, yellow and pink boxes. The green boxes correspond to the 6 objects correctly detected by the state-of-the-art RCNN model [66] trained on the ILSVRC dataset [152]. (The about 500 false positive detections are not shown.) Yellow boxes loosely correspond to objects that are annotated in current object detection datasets such as ILSVRC. The majority of the objects in the scene (shown in pink) are largely outside the scope of capabilities of current object detectors. We propose a principled human-in-the-loop framework for efficiently detecting all objects in an image.	112
8.2	Overview of our system. Given a request for annotating an image, the system alternates between updating the image annotation and soliciting user feedback through human tasks. Upon satisfying the requester constraints, it terminates and returns a image with a set of bounding box annotations.	115
8.3	Three of the user interfaces for our human annotation tasks; others are in Appendix E.	118

8.4	Bounding boxes with increasing intersection over union (IOU) with the optimal tight box. Training human annotators to make binary decision on whether or not a bounding box is a good detection is quite difficult; this the primary contributor to human error rates. Guidance such as “the object occupies more than half the bounding box” is confusing since objects like corkscrews (bottom row) occupy a small area even at perfect IOU.	125
8.5	Our computer vision+human model (<i>CV+H</i>) compares favorably with others. The number of labeled objects (y-axis) is computed by averaging across multiple levels of precision on each image and then across all test images. Error bars correspond to one standard deviation across simulation runs. <i>Left.</i> The joint model handily outperforms the human-only (<i>H only</i>) and vision-only (<i>CV only</i>) baselines at low budget. <i>Right.</i> Our principled MDP is significantly better than choosing questions at random (<i>rand</i>). Variety of human interventions is critical; using only <i>verify-image</i> and <i>verify-box</i> human tasks is insufficient (<i>CV+H:vi,vb</i>). Our model also outperforms the ILSVRC-DET annotation baseline of [152].	129
8.6	Some example results from our system integrating computer vision with multiple types of user feedback to annotate objects.	130
8.7	The average number of objects labeled as a function of human error rates. Reducing human error rates can significantly increase the quality of labeling.	131
8.8	Quality of returned labeling as a function of requester constraints; details in Section 8.6.4.	132
A.1	Distribution of various measures of localization difficulty on the ILSVRC2012-2014 single-object localization (dark green) and PASCAL VOC 2012 (light blue) validation sets. The plots on top contain the full ILSVRC validation set with 1000 classes; the plots on the bottom contain 200 ILSVRC classes with the lowest chance performance of localization. All plots contain all 20 classes of PASCAL VOC.	138
B.1	(a) Classification and (b) classification with localization accuracies of the three methods as a function of the number of guesses allowed during evaluation on ILSVRC2012.	140
E.1	Overview of our annotation layout. A close-up of the instructions (bottom) is in Figure E.2. Only one type of task is shown here (top); closeups of each of the seven types are in Figures E.3-E.9).	153
E.2	General instructions for our human annotation tasks.	154
E.3	User interface for verify-image task. The correct answer is “no.”	155
E.4	User interface for verify-box task. The correct answer is “yes.”	156
E.5	User interface for verify-cover task. The correct answer is “no.”	157

E.6	User interface for draw-box task. The correct answer is “no other box can be drawn.”	158
E.7	User interface for verify-object task. The correct answer is “yes.”	159
E.8	User interface for name-object task. The correct answer is “not a good box.” . . .	160
E.9	User interface for name-image task. The correct answer is, for example, “umbrella.”	161

Chapter 1

Introduction

Hundreds of billions of photographs are created on the web each year. An important step towards understanding the content of these photographs is to be able to understand all objects that are depicted. My research focuses on the problem of automatically naming and localizing objects in large collections of images. This is referred to as the task of *object detection*. Figure 1.1 shows some examples of the types of results that we hope to obtain automatically.

There are multiple challenges that I have addressed that help scale up object detection algorithms in both the number of images and the number of objects that can be recognized. First, algorithms for analyzing such large collections of images need to be very efficient. The sliding windows method, where all image regions are progressively examined (e.g., [34, 165, 184]) is very common in the object detection literature. I developed an algorithm that was significantly more efficient than the sliding windows method. The core insight is to first generate a small set of candidate regions and then only evaluate these likely image locations instead of examining every image region. Similar insights have been utilized in both earlier [73, 157, 113] as well as later literature [186, 3, 66]. My method in particular aims to optimize the cost of generating the candidate region proposals across multiple object classes in the image. It yields an object detector that is an order of magnitude faster than sliding windows while maintaining comparable detection accuracy. More details about the algorithm and the results are provided in Chapter 2 and in [156].

The second challenge of scaling up object detection is that recognizing all objects in images is difficult because of the large diversity of object classes. I've studied using shareable generic object attribute descriptions to effectively describe a variety of object classes without learning individual object models. For example, by learning the generic attributes “striped” and “black and white” we can effectively describe the object class zebra or panda. In general, by learning N binary attributes we may in theory be able to describe up to 2^N object classes. Instead of studying specific domains such as birds [19, 207] or animals [108, 148], we focus on generic attributes such as “red” or “shiny” that are applicable to a wide range of object classes. More details about the large-scale attribute

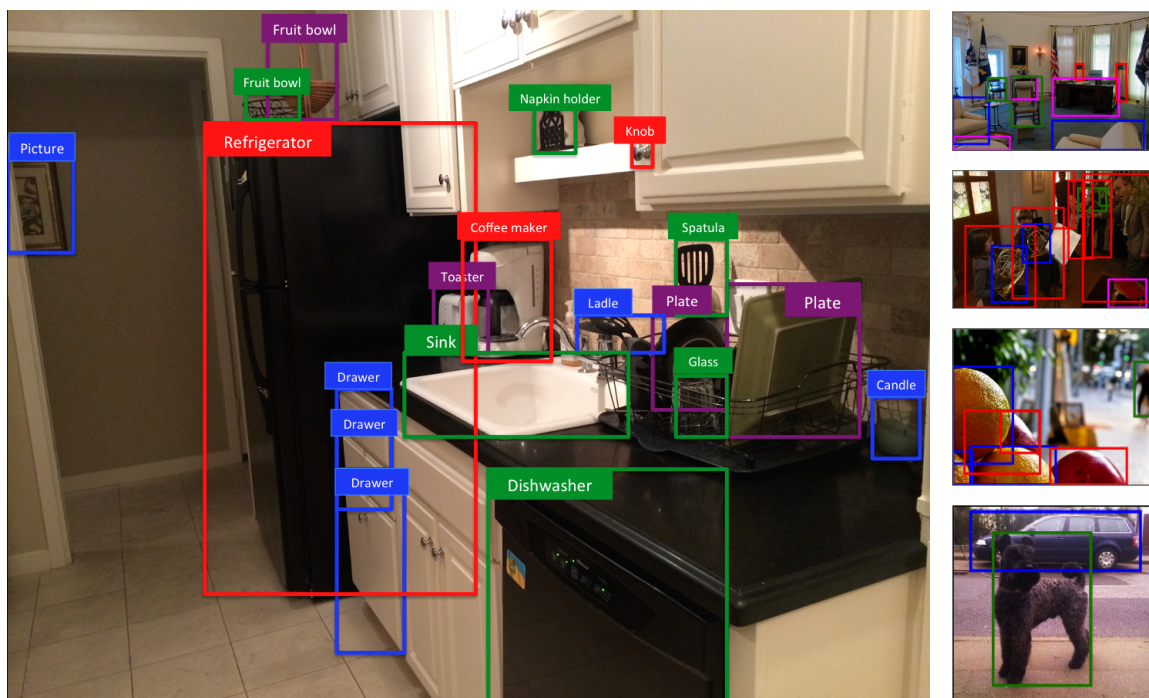


Figure 1.1: This thesis studies the problem of *object detection*, or automatically naming and localizing objects in images. Instead of focusing on just a small subset of object classes [48] we aim to scale up object detection to hundreds of object categories and hundreds of thousands of images.

dataset we collected, the algorithm and the insights into generic attribute prediction are provided in Chapter 3 and in [153].

The third challenge of scaling up object detection is that extensive manual annotation is required for training object detectors at scale, which can be very time-consuming and expensive. The core of my thesis focuses on this third challenge.

I've addressed the challenge of required extensive manual annotation in three ways. First, I developed a weakly supervised learning method for localizing objects in images that reduces the need for detailed manual annotations. Standard object detection algorithms are trained from bounding box annotations around every object in the image [195, 34, 54, 48, 152]. However, it is also possible to train object detectors using just binary image annotations for whether the object appears in the image or not [27, 129, 40, 136]. We improve upon a standard image classification pipeline [203, 227, 215] to simultaneously classify and localize objects in images. This method eliminates the need for significantly more expensive bounding box annotations at a small reduction in accuracy. Details about the algorithm and both quantitative and qualitative results are available in Chapter 4 and in [155].

However, even these simple binary annotations are very expensive to obtain at the scale of

hundreds of object classes and tens of thousands of images. For the second contribution addressing the expense of manual annotation, my colleagues and I created the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). ILSVRC serves as a benchmark large-scale object recognition for hundreds of international research teams. Chapter 5 describes the ILSVRC history, provides insights into the concrete large-scale recognition tasks we formulated, and presents some of the leading algorithms. For my thesis, I led the effort to construct the object detection benchmark, scaling up by more than an order of magnitude compared to previous dataset (e.g., the PASCAL VOC [48]). This required insights into defining an appropriate set of object categories, collecting images on a large scale, and developing novel crowd engineering techniques for reducing annotation cost. Chapter 6 goes into details of our data collection efforts. By taking on the massive labeling task ourselves on behalf of the community, we were able to eliminate the need for costly annotation by individual research teams. Further, the availability of this large-scale data enabled us to perform detailed analysis of the current state of the field of object recognition, providing insights for future research efforts. Chapter 7 presents our insights – for example, the fact that objects that are highly textured are on average easier for current algorithms to recognize than man-made untextured objects. My work on ILSVRC was published in [152, 38, 151].

The ILSVRC dataset and corresponding competition enabled incredible progress in object recognition accuracy over the past few years. [152] Thinking ahead about scaling up object detection even further, I introduce the third contribution for overcoming the challenges of time-consuming large-scale manual annotation. In [154], we design a principled framework for accurately and efficiently localizing objects in images. Instead of relying purely on manual annotation, the system brings together state-of-the-art automatic large-scale object detection with state-of-the-art crowd engineering techniques for a human-in-the-loop method. One key component of our method, in contrast to prior work [19, 91, 140, 200, 192], is the incorporation of feedback from multiple types of human input and from multiple computer vision models. Details about this work are in Chapter 8.

This thesis describes the progression of scaling up object detection along multiple dimensions: increasing the number of images, increasing the number of object classes, reducing the required detailed manual annotation and lowering annotation cost. One important lesson of this work is that scaling up a visual recognition system is not just a matter of throwing more computational resources at the problem. There is always a human cost to be paid; there is a direct tradeoff between the human effort and the computer vision capabilities. This human cost comes in many forms. Detailed large-scale training annotations opens up avenues for development of algorithms for new objects domains [152, 38]. Careful algorithmic design can compensate for the lack of detailed training annotations [155, 150]. Direct integration of human and machine efforts can allow for additional benefits [154]. Human cost should be carefully accounted for as it is inevitably present in computer vision systems.

Chapter 2

A Steiner tree approach to efficient object detection

2.1 Introduction

Object detection has seen significant advances in the last few years [48], but many algorithms are still slow and unsuitable for real-time performance. The standard sliding window approach to object detection analyzes a large number of image regions (on the order of 50,000 for a 640x480 pixel image) to see which of them may contain an object of interest. For many applications, multiple object classes need to be recognized in each scene, and so multiple binary classifiers are run over each region. Thus, if we are trying to detect any of 10 object classes, we may need about 500,000 classifications per image.

In our approach, we propose only a small subset of “promising” regions for each classifier to analyze. By sharing the computation for selecting appropriate regions across the different object classes, we show we can often achieve a 10x computational speedup, without sacrificing accuracy.

At the heart of our approach is a reduction of a feature selection problem to a directed Steiner tree optimization problem. Briefly, in a directed Steiner tree problem [221], we are given a directed graph with weights on the edges, a subset of nodes designated as special *Steiner nodes* and a root node. Our goal is to find a directed tree of minimum weight rooted at the root node that spans all Steiner nodes. This problem is NP-hard [64] but can be efficiently approximated [24, 229]. Concretely, image segmentation, which is used to select the “promising” windows, is expensive to compute. Further, different segmentations are suited for finding different sorts of objects; a segmentation into many small segments may be more suited for small objects such as coffee mugs, whereas a coarser segmentation may be better for larger objects such as computer monitors. When we are interested in detecting many object classes, we would like to find a small number of segmentations that can be

shared across multiple object classes. We show how a directed Steiner tree optimization formulation can be used to select the segmentation parameters efficiently.

We apply these ideas to speeding up object detection, and test our approach on eleven object classes from the LabelMe and StreetScenes datasets, along with our own collected images, in conjunction with a classification algorithm inspired by Torralba et al. [184]. We achieve significant run-time improvement without sacrificing detection accuracy. More broadly, however, we make three main contributions: (1) we present a classifier-agnostic approach to speeding up the sliding windows algorithm, (2) we address the task of efficiently recognizing multiple object classes within the same scene, and (3) we introduce a novel Steiner tree formulation for parameter selection.

2.2 Related work

Efficient detection. The sliding window approach is common in object detection [34, 48, 184], and much work has been done to improve its running time. Viola and Jones [195] (see also Wu et al. [212], Rowley et al. [149], and, recently, Harzallah et al. [78]) sped up localization by quickly rejecting many of the rectangles as not containing the object of interest. In contrast, our algorithm works by proposing only the rectangles that appear likely to contain an object, based on the segmentation, without needing to analyze each sliding window individually.

Some techniques for object localization avoid using sliding windows entirely by instead applying the generalized Hough transform [61, 124, 201]. Among the latest such tools is the work of Gall and Lempitsky on Hough Forests [61] with running times of 6 seconds per 720x576 pixel image, scaling linearly in the number of objects to be detected. Our approach runs in roughly 1.5 seconds per image per object with 9 objects and amortizes part of its running time as more objects are added. Further, our method is classifier-independent and can be used in conjunction with any type of classifier, including Hough forests-based classifiers.

Another approach to speeding up localization is to use local optimization methods, by first identifying promising regions of interest and then using iterative refinement to obtain better region boundaries [27]. Lampert et al. [109] proposed a branch-and-bound algorithm to repeatedly decrease the region of interest from the entire image to a bounding box around the desired object using a bag of words sparse feature model. While their method is highly effective, our technique applies to a much broader class of visual features; for instance, the dense responses of the location-sensitive patch-based classifiers of [184] that we consider in our experiments cannot be used within their framework.

There are methods which use low-level features to create a saliency map of the image [90, 93] and focus attention for object localization that way; their techniques could be used in our Steiner tree framework to replace the segmentation algorithm and instead propose windows at various granularities using the detected interest points. However, we believe that merging together superpixel

segmented regions is more intuitive than combining interest points which are intended to emphasize regions of high variability.

Joint segmentation and detection. There are many methods for using segmented superpixels and merging them together to form an object boundary [60, 69]. Russell et al. [157] introduced the “soup of segments” idea, where multiple segmentations of an image are obtained, and then all the segments are considered together as building blocks in tasks such as object discovery [157], spatial support [125], or joint object classification and segmentation [69, 137]. As discussed below, our framework also allows multiple combinations of segmentation parameters for each object class.

Many algorithms exist as well for simultaneously performing both image segmentation and object recognition that combine bottom-up and top-down models [111, 113, 115, 211, 218]. These methods have generally focused on improving the *accuracy* of both segmentation and object detection, rather than on minimizing the object detector running time.

Gu et al. [73] recently introduced a technique for using regions for object detection instead of the sliding-window approach, and reported significant run-time improvements. They make the assumption that each segment corresponds to a probabilistically recognizable object part, whereas our algorithm is specifically designed to compensate for imperfections in the segmentation algorithms by automatically considering a large number of schemes for merging superpixel regions to create an object. Further, their classifier directly utilizes the segmented regions while our techniques can be used in conjunction with any classifier.

Similar insights of using segmentation to propose object detection regions have also been utilized in literature published after this work, e.g., [186, 66, 3].

Multi-class classification. Sharing computation for multi-class detection has been explored by [159, 181, 184] among others; e.g., Todorovic and Ahuja [181] consider a taxonomy of object subcategories (parts) which can be detected in images and then used to construct multiple object classes.

Steiner trees. We use the algorithm due to Charikar et al. [24] in our parameter selection method to find the approximate minimal cost Steiner tree. It is impossible to do justice to all Steiner tree literature here, but briefly [221] analyzes the general directed acyclic case, [127] presents a new primal-dual approximation algorithm, [44] discusses the terminal Steiner tree problem, and [229] presents a solution using a linear program which is polynomial-time but too computationally expensive for us to use in practice.

Also, Parikh et al. [139] recently applied Steiner trees to the task of learning spatial hierarchy in images.

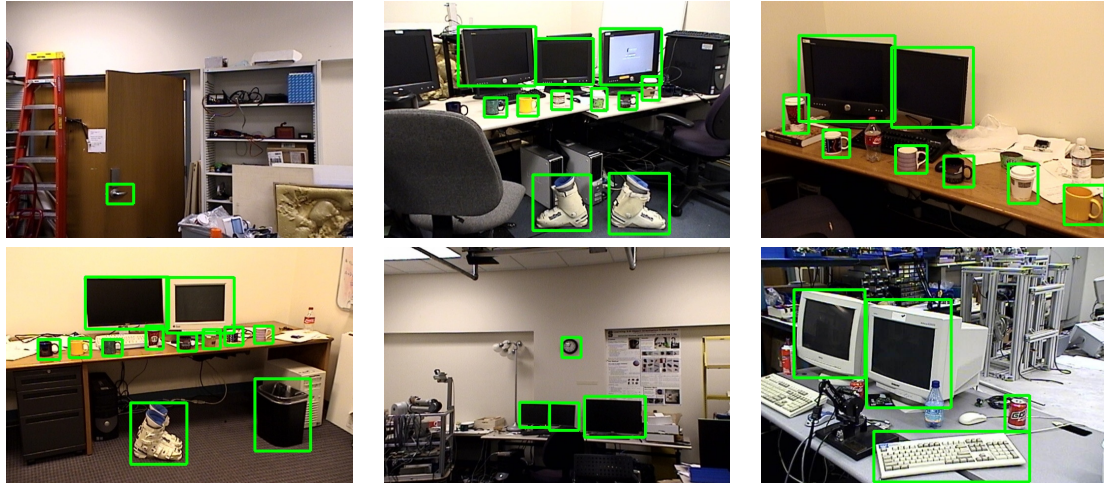


Figure 2.1: Sample images from our object detection dataset with the desired objects outlined in green.

2.3 Fast object detection

A common way to detect object begins by building a binary classifier that takes as input a small (say 32×40) rectangular image patch, and classifies that image patch as either containing a cup (or other object of interest) or not [34, 78, 184]. Given a full-size image, object detection is performed by running this classifier on every 32×40 sub-image of this larger image. To detect the same object at multiple sizes, we scale the image down and repeat.¹

Given an image to analyze, our approach consists of three main steps: (1) **Segmentation**, where we use a standard unsupervised algorithm to segment the image into small locally coherent regions; (2) **Rectangle selection**, where we take these irregular-shaped image segments, and combine/reshape them as needed to obtain a small number of rectangular windows; (3) **Classification**, where we apply a binary classifier to each of these windows to decide if an object of interest appears in it.

We may have to run the segmentation algorithm multiple times and use different rectangle selection strategies for the different objects, so as to generate the most appropriate rectangles for each. For example, we may want to generate square rectangles to detect objects with a square bounding box (like monitors, mugs and wall-clocks), and longer rectangles to detect longer objects (like keyboards). Further, a segmentation into many small segments may be necessary for finding small objects such as coffee mugs, whereas a coarser segmentation with fewer regions may be sufficient for larger coherent objects such as computer monitors.

¹To detect the same object with varying aspect ratios, a fourth search dimension is required in the sliding windows algorithm, greatly slowing down the algorithm. While for this work we restrict our attention to objects with fixed aspect ratios, our algorithm can easily be extended to the more general cases since the bounding boxes extracted from segmentation inherently represent the varying aspect ratios of the different objects.

The segmentation and rectangle selection steps are themselves computationally expensive. Thus, if we can share parts of these computations among different object classes, then their cost can be *amortized*, thus further reducing the overall running time of the system. In section 2.5, we address this optimization problem using directed Steiner trees.

2.4 Segmentation and rectangle selection

We begin by briefly describing the various parameters of the segmentation and rectangle selection algorithms. The goal is to identify promising regions of the image that may contain an object, so that the classifier can analyze only these regions. Our pipeline consists of five sequential steps, diagrammed in Figure 2.3 *left*.

2.4.1 Segmentation

We use the segmentation algorithm of Felzenszwalb and Huttenlocher [56], which has parameters s , k , and m . Briefly, s controls how much we smooth the original image, k determines roughly how many segments the image is broken into, and m controls a post-processing step that ensures that all resulting segments are of size at least m pixels. Even though [56] gives suggestions for parameter settings that produce visually pleasing segmentations, we found that it was impossible to find a single parameter setting that works well for detecting all the objects of interest.

When the image is over-segmented (into a large number of regions, corresponding to small k), ski boots, which are often very detailed in the image, tend to be broken into many individual segments (Figure 2.2). It is then difficult to automatically combine these segments together to find a rectangle that correctly bounds the entire ski boot. Conversely, it is difficult for a segmentation algorithm to detect the correct object boundaries around cups and mugs, and so unless the image is over-segmented, these tend to be merged with the background.

2.4.2 Rectangle selection

Given an image segmented into irregular regions, there are various methods of using these regions to generate rectangular windows likely to contain objects. Three simple ways are computing bounding boxes (1) around individual segments, (2) around each segment and all of its neighbors, and (3) around all pairs of adjacent segments. By analyzing the typical segmentations of wide or tall objects (such as keyboards or paper cups) we found it beneficial to also take bounding boxes around triples of segments lined up either vertically or horizontally.

These five merging schemes seem robust to a large range of object shape, size and pose variations, which makes it possible to use the same segmentation parameters for groups of objects and partially amortize the large segmentation cost. However, often no single merging scheme is sufficient to detect



Figure 2.2: *Left.* Original image. *Center.* Segmentation with the parameter $k = 1600$. It works remarkably very well for ski boots (shown in blue) and contains only 88 segments, most of them too small or too big to even be considered by the classifier. However, there is no hope of recovering any of the smaller objects, such as the coffee mugs and paper cups on the table. *Right.* Segmentation with $k = 100$ ($s = 0.8$, $m = 20$). Notice that good bounding boxes around the mugs and cups can now be reconstructed by combining a small number of segments, yet ski boots are extremely over-segmented and thus very difficult to reconstruct.

a specific object class robustly. On the other hand, employing all schemes together results in an excessive number of boxes for the classifier to analyze, increasing the running time and potentially decreasing object detection precision if the classifier makes errors.

We introduce the parameter b which takes on one of $2^5 - 1 = 31$ values and corresponds to generating all the rectangles from any possible non-empty subset of these five merging schemes. This allows the learning algorithm to automatically tradeoff speed and classification accuracy.

Note that the framework for using multiple combinations of merging scenes can be extended to the other parameters as well; for example, combinations of the segmentation parameter k could be considered, resulting in multiple segmentations of the same image used to propose regions.

2.4.3 Trimming parameter

Finally, the rectangles generated from this segmentation and merging process are often too large, in a specific and repeatable way. For example, a mug on a table may be segmented almost correctly except that the segmentation merges the mug with the edge of the table (which is very long and thin; see level 5 in Figure 2.3 *left*). This effect seems common to many segmentation algorithms, including [56, 166]. The bounding rectangle around this segment would be much larger than the object itself, making it very difficult for the classifier to recognize the object. To account for it, we introduce the final parameter p that determines how aggressively we trim down the generated boxes.²

²Specifically, consider a segment (or a few segments merged together as described above) and its bounding box. We compute the number of pixels within the left-most column of the bounding box that are also contained within the segment. If this number is smaller than $p\%$ of the bounding box height, we consider the column “sparse” and, if it helps bring the aspect ratio of the bounding box closer to the desired value, remove the column. We run the same process repeatedly on right/top/bottom edges as well.

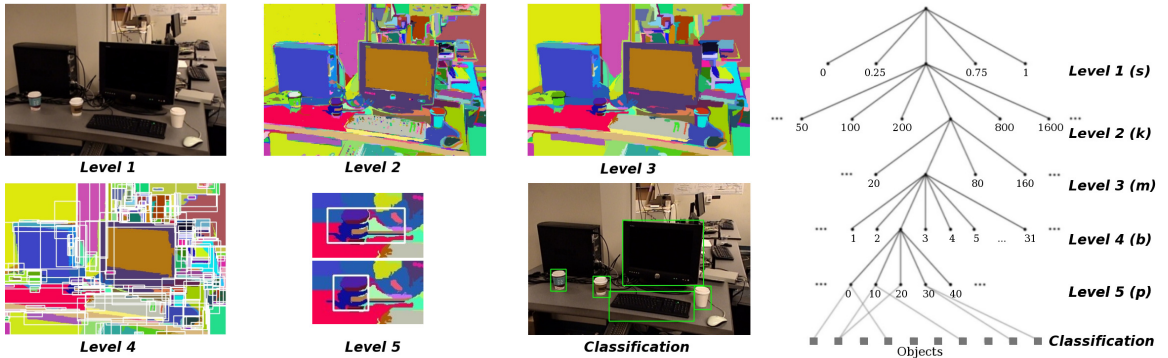


Figure 2.3: *Left.* Illustration of the five sequential steps of our segmentation and rectangle selection pipeline described in section 2.4. (1) Smooth the original image ($s = 1$ shown). (2) Segment ($k = 150$ shown). (3) Merge small segments together ($m = 160$ shown). (4) Propose rectangular bounding boxes around groups of segments (only boxes around individual segments are shown). (5) Trim each of the bounding boxes using parameter p to eliminate long sparse tails ($p = 40$ here). The bounding box shown is obtained by merging three vertically aligned segments (purple, brown and blue, top to bottom). The new bounding box is tight enough for the classification algorithm to detect the paper cup. (Classification) Finally, evaluate each proposed rectangle with the object classifier. Note that the picture shown here is for illustration purposes only; there is no single assignment to the five segmentation and rectangle selection parameters that would allow the classifier to perform well on all 3 object types in this image. *Right.* Illustration of the directed graph G generated by the Steiner tree algorithm, described in section 2.5.

2.5 Steiner Trees for Parameter Selection

Since the segmentation and rectangle selection steps described above are computationally expensive, we want to share these computations among different object classes. This sharing can occur at multiple levels; e.g., if two object detection algorithms can share the image segmentation computation but not the rectangle selection step (which is based on the segmentation), then that would still be preferable to sharing neither segmentation nor rectangle selection. We show how the problem of parameter selection reduces to a directed Steiner tree.

2.5.1 Directed Steiner Trees

In the directed Steiner tree problem [221], we are given a directed graph $G = (E, V)$ with weights $w(E)$ on the edges, a set of **Steiner nodes** $S \subseteq V$, and a **root node** $r \in V$. Our goal is to find a directed tree that is rooted at r so that the tree spans all vertices in S , while minimizing the total weight of all the edges in the tree. Note that this is a somewhat different problem from the standard Steiner tree problem [87], where the graph is undirected and there is no special “root” node.

The directed Steiner tree problem is NP-hard;³ however, there are efficient approximation algorithms. Below, we describe how our problem can be formulated as a directed Steiner tree; we then discuss approximation algorithms.

2.5.2 Constructing the Steiner Tree

We now show how to construct the Steiner tree for our problem. An illustration of the graph G that we use is shown in Figure 2.3 *right*. Recall that our segmentation and rectangle selection approach comprises five sequential steps with parameters s , k , m , b and p . The interpretation of the graph is as follows. The root node has five children, corresponding to the five possible (discretized) values of s that we will consider. Each of these edges from the root has a cost equal to performing the first step of the algorithm using the selected parameter s . Traversing the graph from the root r to a node at level 5 corresponds to assigning values to each of the parameters s , k , m , b , and p . The cost of each edge corresponds to the running time of performing the corresponding step of the pipeline using the parameter selected.

The number of Steiner nodes $|S|$ in our tree is equal to the number of object classes we are trying to recognize (i.e., the number of classification problems we would like to do well on). The Steiner nodes are square in the figure, and together comprise the bottom-most 6th level of the graph. A node at level 5 of the graph, which corresponds to a set of rectangles generated using a specific set of parameters s , k , m , b , and p , will be connected to a Steiner node at level 6 only if that set of rectangles, when analyzed with the classifier corresponding to that Steiner node, achieves a minimum desired level of performance. Further, the cost of this edge is the running time of the classifier applied to the corresponding set of rectangles (which is roughly linear in the number of rectangles examined).

By construction of the graph G and the associated costs, we see that if we are able to find a minimum cost Steiner tree, then we will have found the set of parameters that minimize the overall computational cost, while achieving the desired classification performance for each of our objects. This allows trading off between classification performance and running time; for example, by relaxing the constraints on the performance requirements of the classifiers, many more edges between level 5 and 6 nodes will be added to G , and thus the minimal Steiner tree of G will likely have smaller cost – implying that the corresponding classifiers have faster running times.

Choosing subsets of merging methods. One final detail is necessary to correctly compute the costs on the edges. For the sake of simplicity, we will describe this detail ignoring level 5 of the tree (i.e., as if there was no parameter p), and imagine that level 4 nodes were directly connected to the level 6 Steiner nodes.

³For example, when all terminals of G are exactly 2 edges away from the root r and all costs are 1, this reduces to the **minimum set cover** problem [64].

Recall the five different merging schemes for the rectangle selection step discussed in section 2.4. If one object requires taking the union of all the rectangles from merging schemes (i) and (ii), and another requires the rectangles from (i) and (iii), then we should not separately “charge” the algorithm twice for computing the rectangles for (i). Instead, we want to allow the algorithm to choose a value for parameter b that corresponds to generating all the rectangles using methods (i), (ii) and (iii). Then, having paid the cost on the incoming edge corresponding to this value of b , we want to allow it to use any subset of methods (i), (ii) and (iii)’s boxes to perform different classification tasks. To accomplish this, the level 4 node corresponding to b will be connected to a Steiner node n whenever *any* subset of b ’s merging schemes (in this example, $2^3 - 1 = 7$ subsets) results in satisfactory classifier accuracy on the corresponding object. Furthermore, the cost on this edge from b to n will be the *minimum* of the classifier running times for obtaining satisfactory accuracy on this object (where in this example the minimum is taken over the 7 possible subsets of b ’s merging schemes).

2.5.3 Approximation Algorithms

Even though the directed Steiner tree problem is NP-complete even on planar graphs [64], there exist a variety of approximation algorithms. For our application, we used the algorithm of Charikar et al. [24]. This algorithm is extremely efficient in practice for our formulation (because of our tree-like G), and gives good results for our problem sizes with running times ranging from a few seconds to just under half an hour when the total number of vertices in the graph is on the order of 20,000 (MATLAB implementation).

Briefly, the algorithm, which is parameterized by i , works as follows. For $i = 1$, it simply computes the shortest paths from the root to each of the terminals, and combines them to output a spanning tree. This gives a trivial $|S|$ -approximation (where $|S|$ is the number of objects). Because there is at most one directed path from any node v to any other node u in our input graph G , these shortest path computations can be done extremely efficiently. The algorithm is recursive, and for higher values of i it successively finds better approximations, using the output of the algorithm with parameter $i - 1$ run on different subgraphs of G . (See [24] for details.)

2.6 Experiments

Given the setup described above we test our algorithm on (1) a combination of 359 images of indoor office scenes from the LabelMe dataset [158], combined with our own collected dataset of 557 indoor images, and (2) the outdoor StreetScenes dataset [14].

2.6.1 Training stage 1: Object classifiers

We analyze the performance of our algorithm on 9 common indoor objects (cans, clocks, computer monitors, door handles, keyboards, paper cups, ski boots, and wastebaskets) and 2 outdoor objects (cars and pedestrians). For each object we wish to recognize, we train a binary classifier using the method of Torralba et al. [184]. Briefly, for each object of interest, we obtain a set of positive and negative training examples, all cropped to the same default window size chosen based on the object’s aspect ratio (e.g. mugs and clocks were scaled to fit into a 32×32 window, keyboards 96×32 , cups 32×40). We build a patch dictionary for each object by extracting a set of random patches from the positive training examples, and recording the location within the image that each patch originated from. The patches are extracted from the intensity and gradient magnitude images. For each patch and for each training image, we then compute the corresponding feature by finding the maximum normalized cross-correlation between the patch and the training example within a small window around the original location of the patch. Given these features, we use the Gentle AdaBoost algorithm to train a binary decision tree classifier.

We used the implementation of [70] for this stage of training. The indoor object detectors were trained using 214 of our collected scenes of office environments, and took around 2-4 hours each to train. For the outdoor images we used 80% of the 3547 StreetScene images, along with the INRIA dataset [34], to train the classifiers.

Run-time object detection. During test time, the standard approach is to apply the classifier to every subwindow of a full-size image to determine if this subwindow outlines the object of interest. To detect objects of multiple sizes, this is done for a discrete set of scales, e.g., by repeatedly making the image 1.2 times smaller.

As suggested by [184], in this case the feature computation step can be significantly sped up by pre-computing convolutions over the entire image. At each scale σ , one first computes, for each patch g_f , the response image $I_\sigma^f(x, y) = (I_\sigma \otimes g_f)$, where \otimes is the normalized cross-correlation and I_σ is the image at that scale. These full-sized response images are then used to analyze each subwindow (in 4 pixel shifts) within I_σ using our trained classifier.

When running detection on the sparse candidate set of regions obtained using our segmentation and rectangle selection method, full-image convolutions are no longer effective. Thus we have to compute the features individually within each promising window. Despite this, we are able to reduce the number of windows so drastically that our algorithm still shows significant run-time improvements.

2.6.2 Training stage 2: Steiner trees

In the second stage of training, we learn the best parameter settings for our object detection pipeline using the Steiner tree formulation. For every possible setting of the 5 segmentation and rectangle

selection parameters, we obtain a set of windows to analyze, and then evaluate the performance of each of the object classifiers on these regions. To provide a more controlled comparison to sliding windows, our algorithm is constrained to only return boxes which would have been considered by sliding windows (so shifts of 4 pixels, and successive changed in scales of 1.2). Each bounding box proposed by the segmentation and rectangle selection pipeline is mapped to 8 sliding window location boxes (4 at the smaller scale and 4 at the larger scale). This part of Steiner tree training takes on the order of 8 hours parallelized over 20 machines for around 500 704x480 training images.

We report results using the Steiner approximation algorithm [24] with $i = 2$. We also experimented with $i = 3$, since larger values of i give better approximations, but due to the structure of our graph, $i = 3$ typically gave identical results to $i = 2$, while increasing the training time of this stage from 1-2 minutes to up to half an hour.

2.6.3 Evaluation

During the object detection test phase, for each object we generate proposed windows in the test set images using the chosen segmentation and rectangle selection parameter, making sure to reuse computation whenever possible between objects (i.e., if the Steiner tree learned the same segmentation parameter setting for both monitors and keyboards, but different rectangle selection parameters, then we will only segment the image once but then will run multiple rectangle selection methods). These windows are then evaluated using our binary classifiers, and the results are reported below.

Indoor scenes

To analyze our approach, we use a combination of the remaining 343 images from our collected dataset (which were not used for classifier training) along with 359 images from LabelMe [158], all scaled to 704x480 resolution. 70% of these images are used for Steiner tree training and 30% for testing. We employ the evaluation criteria of Pascal VOC [48], so a detection is considered positive only if its intersection with a ground truth bounding box divided by their union is greater than 50%, and at most a single detection per groundtruth object is considered correct.

We compare the test set classification accuracy and the test set running times (Table 2.1) to those of the sliding window detector. The reported running times include feature computation, and represent the average processing time per image *per object*. As mentioned in section 2.6.1, the image features in the sliding window approach can be simultaneously computed very efficiently on the full image using convolutions and integral images. In contrast, our approach computes the features separately for each selected window.

For the first experiment, we simply choose, for each object independently, the parameter setting which achieves the best performance (area under the PR curve) on the training set. We refer to this method as “100B”, and use it as a baseline to compare against the Steiner tree-based algorithm. In this setting, there is very little sharing amongst objects (the single best performing parameter setting

Method		Sliding Windows	100B	95B	90B	80B
Detection time (seconds)		18.85	4.62	2.43	1.72	1.29
Number of windows		52398	1685	917	570	394
Detection Accuracy	Average	0.443	0.489	0.462	0.446	0.421
	Boot	0.322	0.550	0.572	0.555	0.565
	Can	0.455	0.486	0.452	0.391	0.359
	Clock	0.793	0.722	0.618	0.736	0.629
	Cup	0.514	0.532	0.470	0.421	0.383
	Handle	0.089	0.173	0.173	0.182	0.247
	Keyboard	0.452	0.535	0.526	0.457	0.456
	Monitor	0.647	0.695	0.665	0.664	0.630
	Mug	0.564	0.590	0.547	0.473	0.392
Trashcan	0.152	0.120	0.132	0.134	0.125	

Table 2.1: Test set performance of each parameter selection method on the supplemented LabelMe dataset. Detection time and the average number of windows analyzed are reported per image per object. The “Average” row contains the average area under the PR curve over all 9 objects. Performance superior to the sliding windows approach is bold-faced.

for one object is in practice very different from the best performing parameter setting on each of the other objects). Even with this simple method, which does not take advantage of the Steiner tree formulation, just from using the segmentation algorithm we obtained a 31x reduction in the number of windows considered, resulting in running 4 times faster than standard sliding windows (despite the added computational cost of computing features independently for each window). Further, because the segmentation eliminated many false positive windows which were previously considered by the classifier, this method yielded a 10% improvement in average area under the PR curve over the 9 objects we considered.

For “100B” we chose (for each object independently) the parameter setting that gives the best classification performance on the training set; in the next set of experiments, we consider any parameter setting that performs within 5%, 10%, and 20% of this optimum (referred to as methods “95B”, “90B”, and “80B” respectively). This results in more connections between the 5th and 6th levels in the graph G , and thus in minimal Steiner trees of lower cost.

With these methods, the detection accuracy slowly degrades down to 5% below the performance of sliding windows, while the reduction in the number of windows analyzed and the speedup in the detection running time increase to 133x and 14.5x respectively. Most notable is the 90B algorithm which achieves an 11x improvement in running time (92x fewer windows considered) without sacrificing average detection accuracy.

Outdoor scenes

We also evaluated the performance of our approach on the StreetScenes dataset [14]. The 710 images remaining after classifier training were split in half and used for training and evaluating our detection

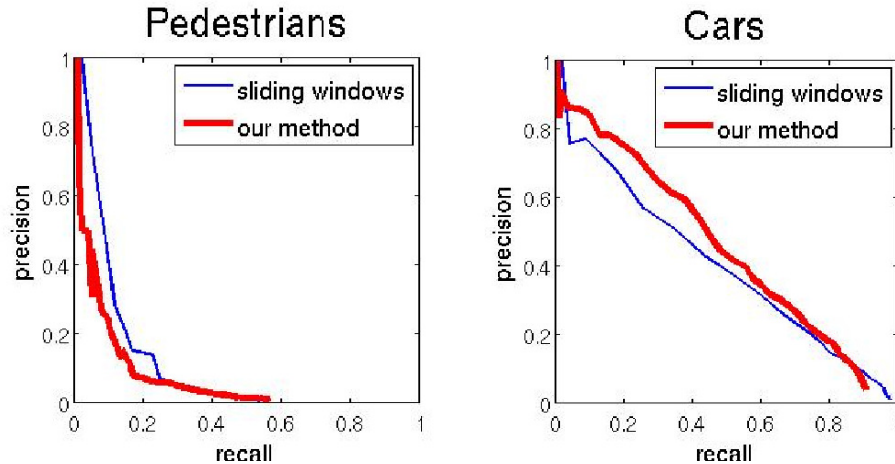


Figure 2.4: Accuracy of our algorithm on 355 images from the StreetScenes dataset. The blue line corresponds to the sliding windows algorithm, and the red bold line is our algorithm.

approach. The images were scaled down to 320x240 resolution. For each ground truth object at most one detection was considered correct. A detection was considered positive when its intersection with the ground truth bounding box divided by their union was at least 20%.⁴

We used all parameter settings that performed within 99% of optimal as measured by maximum F-score, and obtained the PR curves shown in Figure 2.4. Since objects are often occluded, it is very difficult to obtain a good segmentation of this dataset, and the ability to combine multiple segments in various ways to generate better bounding boxes was key to getting good performance.

The number of regions classified per object class on average went down from 9548 per image to 158 (60.4x). The running time of the sliding windows algorithm was 7.95 seconds per image per object; we are able to run in 0.524 seconds per object (15.2x). In this case only two object classes are being detected; our approach is designed to achieve even greater speed-ups with more classes.

2.7 Conclusions

We have described a method for speeding up object detection algorithms to enable them to be used in real-time applications. We present an approach that segments the image and uses the resulting segments to propose image windows most likely to contain the objects of interest. We then use object classifiers to analyze only these proposed regions. Central to our approach is a method for choosing a small subset of segmentation parameters to use for all object classes, so that the cost of

⁴This criteria was used e.g., by [79], and was chosen because the object size in the images is so small: the smallest car in the test set is just 7×7 pixels, and the smallest pedestrian is 7×14 pixels. Our classifiers expected the input size of 40×20 and 32×64 , which helps explain the poor baseline recognition performance on this dataset.

segmentation is amortized across multiple object classes. This is done using a directed Steiner tree formulation. Our method results in a significant (10x) speedup compared to the standard sliding window technique without sacrificing accuracy, and a 15x speedup with a slight drop in accuracy.

More generally, the directed Steiner tree formulation also applies to other multitask learning scenarios [20] in which features have different costs to compute or measure, and certain computations may be prerequisites of others, but where we would like to find the minimum cost set of the features while maximizing classification performance.

Chapter 3

Attribute learning in large-scale datasets

3.1 Introduction

Computer vision has traditionally focused on object categories: object classification, object segmentation, object retrieval, and so on. Recently, there has been some interest in transitioning from learning visual nouns (whether object categories, such as cars or pedestrians, or object parts, such as “wheel” or “head”) to visual adjectives (such as “red” or “striped” or “long”) which can be used to describe a wide range of object categories [108, 58, 50, 49, 106, 148]. Learning visual attributes has been shown to be beneficial for improving performance of detectors [50] but especially for transferring learned information between object categories. For example, learning the color “red” or the pattern “striped” from a series of training images can then be used to recognize these attributes in a variety of unseen images and object categories [108, 50].

The term “attribute” is defined in Webster’s dictionary as “an inherent characteristic” of an object, and various types of attributes have been explored in the literature: appearance adjectives (such as color, texture, shape) [108, 58, 50, 49, 106, 214], presence or absence of parts [108, 49, 148] and similarity to known object categories [108, 106, 148]. Attributes have also been broken up into (1) semantic, i.e., those that can be described using language [108, 49, 214], and (2) non-semantic but discriminative [50] or similarity-based [106, 148]. In this chapter, we focus on semantic appearance attributes.

Attributes and parts-based models are particularly important when building large-scale systems, where it is infeasible to train an object classifier independently for each object class. Given a sufficiently rich dataset of learned adjectives, new categories of objects can be recognized simply from a verbal description consisting of a list of the attributes [108, 50] or a verbal description in

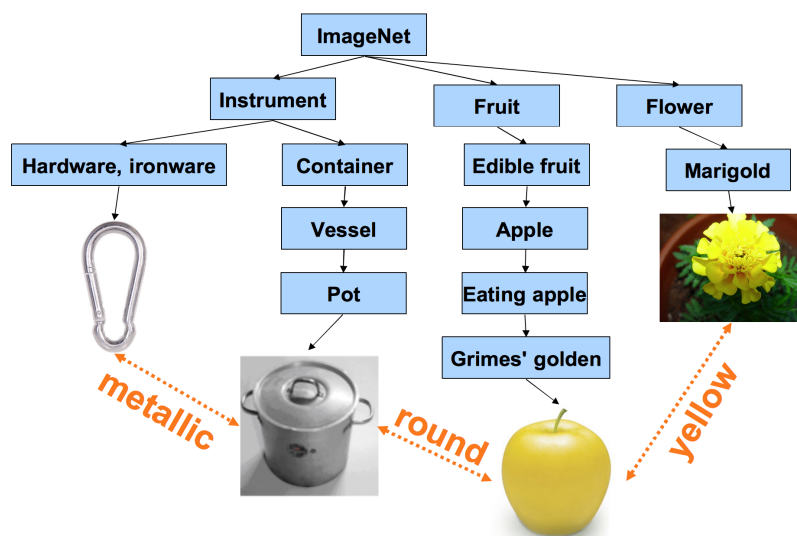


Figure 3.1: The goal of our work is to build visual connections between object categories. We focus on the large-scale ImageNet dataset which currently uses WordNet [128] to provide a semantic hierarchy provides a semantic hierarchy of categories. Discovering a visual hierarchy would be useful for a variety of tasks; for example, targeted retrieval.

combination with just a few training examples [50].

In this chapter, we consider learning multiple visual attributes on ImageNet [36], which is a large-scale ontology of images built upon WordNet [128]. It contains more than 11 million images representing more than 15 thousand concepts. While the dataset already provides useful structure and connections between object classes through the hierarchical semantic ontology of WordNet, we want to learn *visual* relationships or hierarchies between the classes (see Figure 3.1). We begin by describing the existing connections within the ImageNet dataset in Section 3.2, and discussing prior work for attribute learning in Section 3.3. In Section 3.4 we describe our approach to obtaining ground truth human labeling of attributes. We then learn 20 visual attributes on the ImageNet data and present results on a number of tasks in Section 3.5. We conclude and discuss future work in Section 3.6.

3.2 Learning visual connections in ImageNet

The ImageNet dataset [36] contains representative images for more than 15 thousand image categories, or *synsets* as they are called in WordNet.¹ Recently, bounding box annotations have been released for some of the categories, making it easier to perform object categorization or attribute learning. However, the dataset remains highly challenging, with lots of variety within the synsets, as shown in Figure 3.2.

¹We use the terms “synset” and “object category” interchangeably.

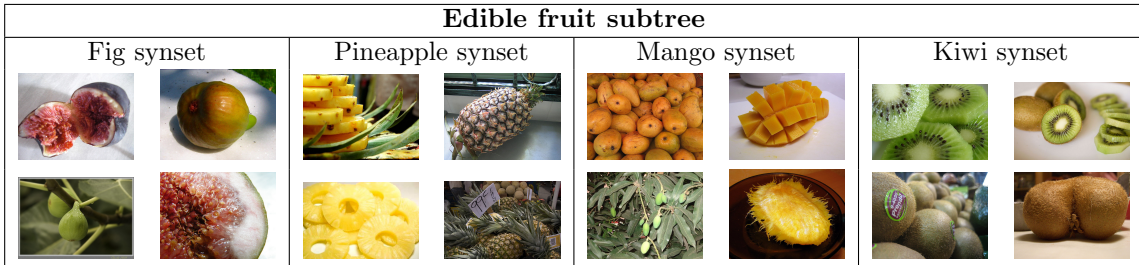


Figure 3.2: Example images of synsets that are direct descendants of the edible fruit synset. First, the high variability within each of the four synsets makes classification on this dataset very challenging. Second, the four object classes are sibling synsets in WordNet since they are all children of the “edible fruit” synset; however, visually they are quite different from each other in terms of color, texture and shape.

Noun hierarchies such as WordNet have been very successfully used in natural language processing. However, the WordNet noun hierarchy is far from visual; for example, human-made objects within ImageNet are organized by their high-level purpose and animals are organized by their evolutionary relation, and as a result the sibling synsets are often very far from each other in appearance (see Figure 3.2). Evolutionary hierarchies are fundamental in genomics and evolutionary biology, but for computer vision, it would be more useful to be able to derive a hierarchy of (or at least a set of relations between) object categories that’s based on visual adjectives or attributes of objects, rather than their evolutionary relation.

Connections based on the visual attribute such as “striped” are missing: striped animals (zebras, raccoons, tigers), striped insects (hairstreak butterfly), striped flowers (butterfly orchid, moosewood tree), striped vegetables (cushaw, watermelon), striped fish (black sea bass, lionfish) and inanimate objects such as striped fabric are not related within ImageNet. To the best of our knowledge, previous work on attributes has focused on making connections within a much more narrow set of object categories (such as animals [108, 148], cars [50, 49] or faces [106]). We are interested in discovering visual relations between all categories of ImageNet, from fruits to animals to appliances to fabrics. We show in Section 3.5.4 that our algorithm indeed manages to do that.

3.3 Related work

Ferrari and Zisserman [58] proposed learning attributes using segments as the basic building blocks. They distinguish between unary attributes (colors) involving just a single segment and binary attributes (stripes, dots and checkerboards) involving a pattern of alternating segments. Since their method relies on obtaining a near-perfect segmentation of the pattern, in practice it’s difficult to apply to challenging natural images – for example, the stripes of a tiger are very difficult to segment out perfectly, and the orange background stripes would often get merged into a single segment,

contrary to what their attribute classification algorithm expects.

Yanai and Barnard [214] learned the “visualness” of 150 concepts by performing probabilistic region selection for images labeled as positive and negative examples of a concept, and computing the entropy measure which represents how visual this concept is. They evaluated their algorithm on Google search images, and also considered each image to be a collection of regions obtained from segmentation, but didn’t consider the pairwise relationship between the regions.

Recently, Lampert et al. [108] considered the problem of object classification when the test set consists entirely of previously unseen object categories, and the transfer of information from the training to the test phase occurs entirely through attribute text labels. They introduced the Animal with Attributes dataset with 30,000 images annotated with 50 classes. They are interested in performing zero-shot object classification (where the object classes in the training and test sets are disjoint) based on attribute transfer rather than learning the attributes themselves or building an attribute hierarchy. Interestingly, some of their attributes are not even fundamentally “visual” (for example, “strong” or “nocturnal”), but were nevertheless found to be useful for classification [108]. One interesting thing to point out in relation to our work is that ImageNet already has subtrees for some of their adjectives, such as edible, living, predator/prey/scavenger, young, domestic, male/female, even insectivore/omnivore/herbivore. While many of their other attributes are animal-specific, such as “has paws,” and thus not as useful in our setting for making connections between a broad range of object categories, we were inspired by their list in creating our own.

Farhadi et al. [50] worked on describing objects by parts, such as “has head,” or appearance adjectives, such as “spotty.” They wanted to both describe unfamiliar objects (such as “hairy and four-legged”) and learn new categories with few visual examples. They distinguished between two types of attributes: semantic (“spotty”) and discriminative (dogs have it but cat don’t). Similarly, Kumar et al. [106] considered two types of attributes for face recognition: those trained to recognize specific aspects of visual appearance, such as gender or race, and “simile” classifiers which represent the similarity of faces to celebrity faces. We focus on semantic attributes in the current work, but argue that ultimately discriminative and comparative attributes are necessary because language is insufficient to precisely describe, e.g., the typical shape of a car or the texture of a fish.

Rohrbach et al. [148] use semantic relationships mined from language to achieve *unsupervised* knowledge transfer. They found that path length in WordNet is a poor indicator of attribute association (for example, the “tusk” synset is very far from the “elephant” synset in the hierarchy, making it impossible to infer that elephants would have tusks). They show that web search for part-whole relationships is a better way of mining attribute annotations for object categories. In our work, we also explore using WordNet to mine attribute associations, but consider using the WordNet synset definitions rather than path length.

Most recently, Farhadi et al. [49] discussed creating the right level of abstraction for knowledge transfer. They learned part and category detectors of objects, and described objects by spacial

arrangement of their attributes and the interaction between them. They focused on finding animal and vehicle categories not seen during training, and inferring attributes such as function and pose. They learn both the parts that are visible and not visible in each image.

Literature after the publication of this work has examined relative attributes [138], scene attributes [1], attribute calibration [164] and efficient learning with attributes [140, 15, 107] among other related topics.

3.4 Building and labeling an attribute dataset

In order to learn and evaluate attribute labels, we first need to obtain ground truth annotations of the images. [148] discusses various data mining strategies; however, it focuses on parts-based attributes, mining for relations such as “leg is a part of dog” or “dog’s leg.” WordNet provides a definition for every synset it contains; since we are instead interested in appearance-based attributes, we considered two strategies: mining these definitions directly (which is different than the path length discussed in [148]), and manual labeling (which was the approach of [108, 49]).

WordNet synset definitions are not well-suited for mining visual adjectives for several reasons. First, the mined adjectives don’t necessarily correspond to visual characteristics of the full object and require understanding of the object parts (e.g., animals with a “striped tail”). Second, the mined adjectives often need to be understood in the context of other adjectives in the definition (e.g., a flower described as “yellow or red or blue”). Also, sometimes the adjectives are extremely difficult to detect visually (e.g., a flag is defined as “rectangular” but usually doesn’t look rectangular in the image). However, since ImageNet is a very large-scale dataset, mining for attributes in this very simple way can help restrict attention to just a subset of the ImageNet data which is likely to contain a sufficient amount of positive examples for each attribute. To construct the dataset of 384 synsets that we use for our experiments, for every attribute we searched for all synsets (from among those with available bounding box annotations) which contained this attribute in either the synset name or the synset definition, and included that synset along with all of its siblings in the training set. The motivation for including the siblings was to provide a rich enough set of negative examples that are likely to differ from the positive synsets in only a few characteristics, and specifically in the characteristic corresponding to the mined attribute. For example, if a zebra is characterized as a “striped” equine, it’s reasonable to infer that other equines, such as horses, are not striped.

In order to obtain the ground truth data we use workers on Amazon Mechanical Turk (AMT) to label 25 images randomly chosen from each synset. We present each worker with 106 images (25 each from 4 different synsets plus 6 randomly injected quality control images) and one attribute, and ask to make a binary decision of whether or not this attribute applies to the image. For color attributes (black, blue, brown, gray, green, orange, pink, red, violet, white and yellow), we ask whether a significant part of the object (at least 25%) is that color. For all other attributes (furry,

Attribute	WordNet	Both	AMT
Green	salad, sukiyaki, absinthe	green lizard, grass	sunflower, bonsai
Rectang.	flag, sheet, towel	box	bench, blackboard, cabinet
Round	feline, pita, shortcake	ball, button, pot	basketball, drum, Ferris wheel
Spotted	cheetah, giraffe, pinto	jaguar	garden spider, strawberry, echidna
Striped	aardwolf, zebra		garden spider, skunk, basketball
Wooden	cross, popsicle	marimba	cabinet, pool table, ski
Yellow	grizzly, yolk, honey	sunflower	margarine

Table 3.1: Examples of synsets labeled positive by mining WordNet definitions, by both WordNet and AMT labelers, and just by AMT labelers.

long, metallic, rectangular, rough, round, shiny, smooth, spotted, square, striped, vegetation, wet, wooden), we ask if they would describe the object *as a whole* using that attribute.

Each image is labeled by 3 workers, and we consider an image to be positive (negative) if all workers agree that it’s positive (negative); otherwise, we consider it ambiguous and don’t include it in our training sets. Unfortunately, for 5 of our attributes (blue, violet, pink, square and vegetation) we did not get sufficient positive training data (at least 75 images) to include them in our experiments.

We analyze the overlap between the mined synsets and the human labeling in Table 3.1. We consider a synset to be labeled positive for an attribute by AMT workers if more than half of its labeled images are unanimously labeled as positive. Interestingly, some obvious annotations such as “green salad” or “striped zebra” were not present in the human labels. This shows that data obtained from AMT can be extremely noisy, and that better quality control and/or more annotators are needed. Currently we are only considering an image to be a positive or negative example if it is labeled unambiguously; while this gives us good precision in our training set, the recall is much lower than we would like, and thus the number of training examples for each attribute is low despite the large dataset size. Overall, we have $384 \text{ synsets} \times 25 \text{ images per synset} = 9600$ images labeled with 20 attributes, with 4% of all labels being positive, 68% negative, and 28% ambiguous.

Figure 3.3 shows some example positive examples for the attribute “striped.” This demonstrates the some of the challenges faced by computer vision classifiers that aim to detect the “striped” objects.

3.5 Experiments

We have described the procedure for obtaining 384 imageNet synsets, all of which have bounding box annotations released, with 25 images within each labeled as positive, negative or ambiguous for each of 20 attributes. In this section we show classification and retrieval performance of attribute classifiers trained using this data, as well as apply these classifiers to a simple transfer learning task following the framework of Lampert et al. [108]. Finally, we show the visual links that were



Figure 3.3: Some example images labeled by the human subjects as “striped.” There is large intra-class variation, making this a challenging computer vision dataset.

discovered between distant ImageNet synsets.

3.5.1 Implementation

We represent each image using three types of normalized histogram features: (1) color histogram of quantized RGB pixels using a codebook of size 50, (2) texture histogram of quantized SIFT descriptors at multiple levels using a codebook of size 1000 [123, 152], and (3) shape histogram of quantized shape-context features [8] with edges computed using the Pb edge detector [126, 21] using a codebook of size 500. Each of the three feature histograms was normalized independently to have L1 unit length. We use an SVM with a histogram intersection kernel [23, 176], which in our experiments significantly outperforms both the linear and RBF kernels. We use a holdout set to determine the regularization.

3.5.2 Learning image attributes

First, we train the classifiers to recognize each attribute individually and evaluate the generalization performance. All images in our training set are labeled by 3 AMT workers, and we consider an image to be a positive (negative) example of an attribute if all subjects agree that this is a positive (negative) example. We use 5-fold cross-validation, making sure that no synset appears in multiple folds. Results are shown in Table 3.2 and Figure 3.4.

Some classifiers, such as those corresponding to the color attributes, generalize quite well in this setting. We point out the two main challenges we face when training the attribute classifiers. First, the “pattern” classifiers corresponding to “striped” and “spotted” attributes perform poorly as a result of the great *variety* of the exemplars (see Figure 3.3 for examples of “striped” images). There is a lack of training data especially in light of this variety (only 99 images were labeled as

Attr.	black	brown	gray	green	orange	red	white	yellow	furry	metal	rough
ROC	0.843	0.811	0.851	0.907	0.907	0.951	0.797	0.929	0.851	0.867	0.646
Attr.	shiny	smooth	wet	wooden	spotted	striped	long	rectangular	round		
ROC	0.804	0.614	0.787	0.854	0.663	0.606	0.817	0.766	0.895		

Table 3.2: Performance of attribute classifiers as measured by the area under the ROC curve.

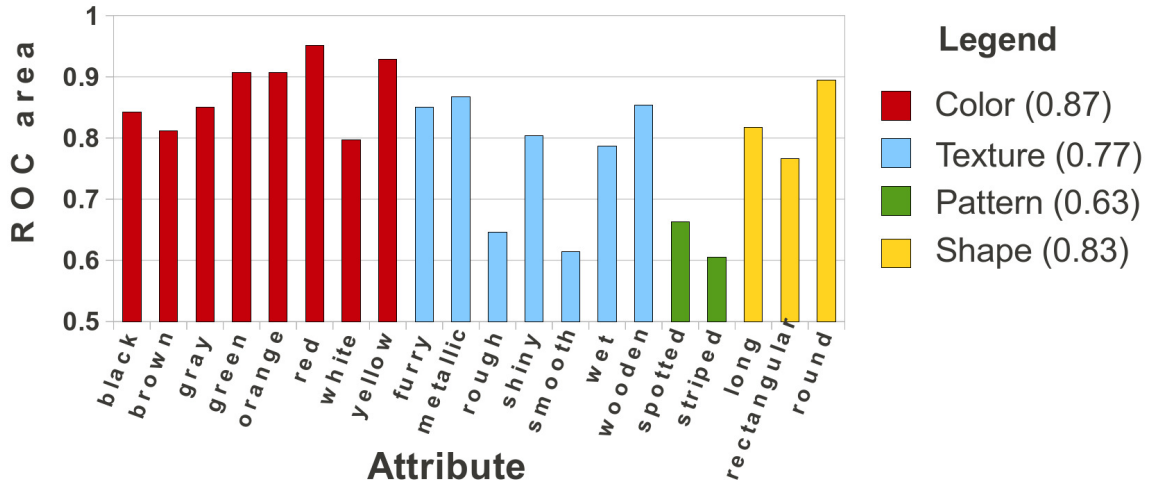


Figure 3.4: Performance of attribute classifiers sorted by attribute type. The average performance of each type is reported in parentheses. Performance is measured by the area under the ROC curve.

“striped” and 146 as “spotted”). As the number of object categories increases, so does the variety of appearances of certain attributes, and thus the amount of training data collected should be sufficient to account for this.

Second, the two texture attributes “rough” and “smooth” suffer from *ambiguity* as evidenced by the lack of labeling consensus. The labelers unanimously agreed on only 66% of the images in the dataset when labeling with the “smooth” attribute, and 72% when labeling with the “rough” attribute. In contrast, for every other attribute the annotators unanimously agreed on more than 79% of the images. As a result, whether an image was labeled as a positive or negative training example for “rough” or “smooth” was largely dependent on the specific set of labelers assigned to it. Such attributes require further refinement and/or better definitions during the labeling process.

In Figures 3.5-3.7 we show qualitative retrieval results using the trained classifiers. Note that many of the top correctly retrieved images were not used in the quantitative evaluation because they were not unanimously labeled by the labelers. This further reinforces the need for more rigorous labeling procedures.



Figure 3.5: Visualization of four of the learned attributes (for the other attributes, see Figures 3.6 and 3.7). For each attribute, the 5 rows represent the 5 training folds, and each row shows the top 8 images retrieved from among all synsets that didn’t appear in that fold’s training set. The border around each image corresponds to the human labeler annotation (green is positive, red is negative, yellow is ambiguous).

3.5.3 Transfer learning using attributes

We use the learned classifiers in a small-scale transfer learning experiment following the Direct attribute prediction (DAP) model of Lampert et al. [108]. Briefly, we are given L test classes z_1, \dots, z_L not seen during training, and M attributes, where the test classes are annotated with binary labels a_m^l for each class l and attribute m . In our experiments we consider $L = 5$ test classes: chestnut, green lizard, honey badger, zebra, and spitz, and $M = 20$ attributes described above. The synset-level annotations come from AMT human labelers.² We use 25 images per object class as above. Given an image x , the DAP model defines the probability of this image belonging to class z as

$$p(z|x) = \sum_{a \in \{0,1\}^M} p(z|a)p(a|x) = \frac{p(z)}{p(a^z)} \prod_{m=1}^M p(a_m^z|x)$$

²Out of 100 class-attribute labels, 18 were ambiguous, meaning that less than half the images within that class were unanimously annotated as either positive or negative for that attribute by all 3 workers. We manually disambiguated the annotations.



Figure 3.6: Continuation of Figure 3.5 visualizing the learned attributes.



Figure 3.7: Continuation of Figures 3.5 and 3.6 visualizing the learned attributes.











						
chestnut : brown,smooth		0.52	0.16	0.12	0.12	0.08
green lizard : green, long		0	0.84	0	0.12	0.04
honey badger : black, gray, rough, furry		0.32	0	0.60	0.04	0.04
zebra : black, white, striped, smooth		0.36	0.08	0.40	0.08	0.08
spitz : white, furry		0.08	0	0.36	0.08	0.48

Table 3.3: On the left are the animal classes and the corresponding human attribute annotations, and on the right is the confusion table from the transfer learning experiments. The rows of the confusion table are the ground truth labels and the columns are the classifier outputs.

where $p(a_m^z|x)$ is given by the learned attribute model, $p(z)$ is assumed to be a uniform class prior, and $p(a^z)$ is the prior on seeing an example with the same set of attributes as the ground truth for the target class z , computed from training data assuming a factorial distribution over attributes. Image x is assigned to class $c(x)$ using:

$$c(x) = \arg \max_{l=1,\dots,L} \prod_{m=1}^M \frac{p(a_m^{z_l}|x)}{p(a_m^{z_l})}$$

We apply this model to our learned classifiers and report our result in Table 3.3. The main source of errors is the zebra class, which relies on the poorly generalizing “striped” attribute (see results in Table 3.2 and Figure 3.4).

3.5.4 Synset-level connections

Given the attribute classifiers we can now consider making synset-level connections within ImageNet, which was the main objective of our work. For each attribute, we have 5 learned classifiers, one for each of the 5 folds. We fit a sigmoid to the output of each classifier to obtain normalized probabilities [23, 146]. We run each classifier on all images that were not part of its training set synsets. For each test synset, we compute the median confidence score of the classifier on images within that synset. Figure 3.8 shows the top returned synsets.

There are various interesting observations that could be made about the retrieved synsets. “Green” and “round” classifiers discover connections between synsets which are very far apart in the WordNet hierarchy – for example, salad, which is a node 6 levels deep under the “food, nutrient” subtree of ImageNet, green lizard, which is 13 levels deep under the “animal” subtree, and bonsai, which is 9 levels deep under the “tree” subtree. of dogs as well as Persian cats, sails, and sheets. The round classifier connects, e.g., basketball, ramekin, which is “a cheese dish made with egg and bread crumbs that is baked and served in individual fireproof dishes” [128], and egg yolk.

green	salad (.84), green lizard (.73), bonsai (.52), pesto (.43), saute (.37), daisy (.30), pot-au-feu (.12), salsa (.12), roughage (.11), cow (.11)
round	egg yolk (.75), basketball (.68), button (.63), goulash (.56), basket (.49), ramekin (.47), ball (.42), pot (.42), veloute (.39), miso (.37)
striped	barn spider (.36), daisy (.17), zebra (.17), echidna (.16), backboard (.13), drum (.12), coloring (.12), roller coaster (.12), bridge (.11), colobus (.11)
wet	rorqual (.59), sidecar (.55), orangeade (.53), flan (.52), screwdriver (.47), killer whale (.44), bowhead (.43), maraschino (.41), dugong (.40), porpoise (.40)

Figure 3.8: This figure shows the top 10 synsets that were returned by the algorithm as the most representative for a subset of the attributes (see Figures 3.9 and 3.10 for the remainder). The number in parenthesis represents the median probability assigned to images within that synset by the attribute classifier.

“Striped” and “wet” discovered some interesting connections – even though it is extremely difficult to learn the high variability of stripes in natural scenes, zebras and echidnas were retrieved, as well as “garden spiders,” which actually often do look striped upon inspection even though it is not a common example that humans would think of as a striped insect. The “wet” classifier especially was able to pick up on some very promising connections: besides just learning that the ocean tends to be wet and thus marine animals are likely wet, it also made the connection to cocktail drinks such as sidecar and screwdriver.

Figures 3.9 and 3.10 show the results on the rest of the synsets. It is interesting to look at attributes such as “long,” which are more contextual and relative, and see the kinds of synsets that were learned. It is not immediately clear that the classifier is picking up on the synsets that human would classify as “long,” although bottles and forks definitely are. It would be interesting to investigate these types of classifiers further, both through the use of a much richer training set as well as by directly visualizing the learned features.

3.6 Conclusions

In this work we began building a set of visual connections between object categories on a large scale dataset. Our ultimate goal is to automatically discover a large variety of visual connections between thousands of object categories. Discovering semantic attributes can aid in more intelligent image retrieval: for example, the user can specify exactly what he's looking for using a known dictionary of attributes instead of visual training examples. More interestingly, clustering the attributes into categories, such as shape, texture, color, and so on, and working with non-semantic attributes, can potentially lead to at least two major advantages. First, this can allow for new ways of object classification training: instead of showing the algorithm a large variety of cars during training, one can simply inject a bit of prior knowledge that cars can come in all colors but shape is the important characteristic. Second, in retrieval, instead of asking to find an image closest to the query, the user can instead specify that he's looking for something that's close in color to the query image, but round.

<p>black</p>	<p>colobus (.78), siamang (.75), guereza (.73), groenendael (.71), binturong (.69), chimpanzee (.66), schipperke (.66), silverback (.63), aye-aye (.54), gorilla (.54), skunk (.53), bowhead (.50)</p> 
<p>brown</p>	<p>puku (.82), lechwe (.73), kob (.73), steenbok (.66), sassaby (.65), redbone (.62), bushbuck (.60), ragout (.59), dhole (.57), chestnut (.56), bovid (.54), sambar (.54)</p> 
<p>furry</p>	<p>keeshond (.94), chacma (.93), macaque (.90), grivet (.90), grizzly (.88), gorilla (.88), baboon (.88), mandrill (.88), koala (.86), simian (.86), guenon (.85), kit fox (.85)</p> 
<p>gray</p>	<p>koala (.42), abrocome (.39), gorilla (.38), grivet (.33), keeshond (.29), manul (.29), schnauzer (.29), chacma (.29), viscacha (.28), vervet (.28), hominid (.27), otter (.26)</p> 
<p>long</p>	<p>kirsch (.83), sail (.77), rorqual (.74), police van (.72), fork (.69), rack (.67), killer whale (.58), window (.54), transporter (.50), pool table (.49)</p> 
<p>metallic</p>	<p>fork (.72), transporter (.56), roller coaster (.49), stick (.41), wheel (.38), police van (.37), keyboard (.34), sail (.31), bridge (.31), building (.28), ski (.25), bowhead (.25)</p> 
<p>orange</p>	<p>orangeade (.73), egg yolk (.58), sunflower (.44), strawberry (.43), fork (.42), maraschino (.42), casserole (.39), screwdriver (.37), pizza (.35), croquette (.30), vermouth (.30), moussaka (.29)</p> 
<p>rectangle</p>	<p>police van (.90), transporter (.84), cabinet (.61), marimba (.50), window (.44), varietal (.42), flag (.38), bridge (.38), kummel (.31), pot (.29), generic (.28), pool table (.26)</p> 

Figure 3.9: Continuation of Figure 3.8 showing the visual connections made between synsets.


<p>red</p>	<p>shortcake (.70), basketball (.67), catsup (.55), teriyaki (.43), salad (.42), pizza (.37), chili (.30), flan (.26), ragout (.23), slumgullion (.22), bordelaise (.20), police van (.18)</p> 
<p>rough</p>	<p>fork (.11), ski (.11), transporter (.11), sail (.11), rorqual (.11), bowhead (.11), keyboard (.11), cross (.11), killer whale (.11), roller coaster (.11), narwhal (.11), stick (.11)</p> 
<p>shiny</p>	<p>rorqual (.95), bowhead (.82), killer whale (.61), dugong (.54), narwhal (.52), manatee (.44), porpoise (.31), police van (.27), kirsch (.27), flag (.21), stick (.21), ski (.20)</p> 
<p>smooth</p>	<p>sail (.65), kirsch (.64), varietal (.63), champagne (.62), generic (.61), green lizard (.58), bottle (.56), egg yolk (.55), window (.55), mallet (.54), pool table (.53), tower (.53)</p> 
<p>spotted</p>	<p>barn spider (.37), zebra (.26), Ferris wheel (.24), cheetah (.19), insectivore (.16), badger (.15), carnivore (.15), grass (.15), kudu (.14), groundhog (.13), pesto (.12), dik-dik (.12)</p> 
<p>white</p>	<p>kuvasz (.70), Saint Bernard (.67), clumber (.65), wirehair (.62), foxhound (.60), sheet (.49), gerbil (.48), Persian cat (.48), sail (.45), bullterrier (.43)</p> 
<p>wooden</p>	<p>fork (.75), rack (.66), bridge (.54), police van (.52), pool table (.46), table (.43), kirsch (.42), marimba (.40), squash racket (.36), transporter (.35), cue (.35), slivovitz (.27)</p> 
<p>yellow</p>	<p>egg yolk (1.00), sunflower (.86), omelet (.70), kedgeree (.64), flan (.61), tostada (.48), succotash (.42), pizza (.35), zabaglione (.26), ravigote (.25), curry (.23), casserole (.21)</p> 

Figure 3.10: Continuation of Figures 3.8 and 3.9 showing the visual connections made between synsets.

Chapter 4

Simultaneous image classification and object localization

4.1 Introduction

Image object recognition has been a major research direction in computer vision. Its goal is two-fold: deciding *what* objects are in an image (classification) and *where* these objects are in the image (localization). Intuitively, if we know which objects are present, determining their location should be easier; alternatively, if we know where to look, recognizing the objects should be easier. Therefore, it is natural to think of these two tasks jointly [129, 13, 22, 102, 30, 224, 57, 78, 172].

However, in practice, classification and localization are often treated separately. Object localization is generally deemed as a harder problem than image classification even when precise object location annotations are available during training. In the purely image classification setting, it may be seen as a detour to attempt to localize objects. As a result, current state-of-the-art image classification systems don't go through the trouble of inferring object location information [203, 227, 215, 152, 48]. Most classification systems (at the time of publication of this work) are based on spatial pyramid matching (SPM) [112] which pools low-level image features over pre-defined coarse spatial bins, with little effort to localize the objects [203, 227, 215].¹

This chapter proposes a novel *object-centric spatial pooling* (OCP) approach for image classification. In contrast to SPM pooling, OCP first infers the location of the object of interest and then pools low level features separately in the foreground and background to form the image-level representation. As shown in Figure 4.1, if the location of the object of interest (a car in this case) is available, OCP tends to produce more consistent feature vectors than SPM pooling. Therefore,

¹The state of the art in image classification has improved significantly since the publication of this work. Instead of pooling low-level image features over coarse spatial bins, a unified neural network classifier is now commonly used [104, 170, 177]. There is still no explicit effort to localize the object in most modern image classification systems.

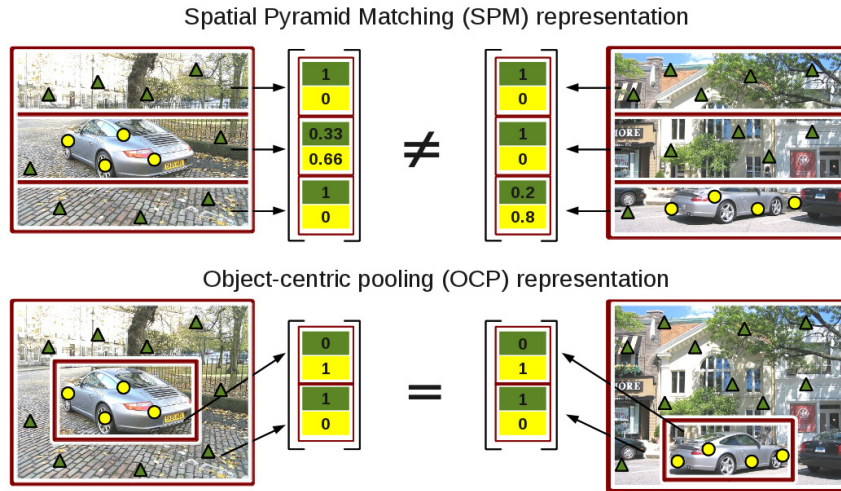


Figure 4.1: We present *object-centric spatial pooling* (OCP), a method which first localizes the object of interest and then pools foreground object features separately from background features. In contrast, Spatial Pyramid Matching (SPM) based pooling [112] (top), the most common spatial pooling method for object classification, results in inconsistent image features when the object of interest (here, a car) appears in different locations within images, making it more difficult to learn an appearance model of the object. For the purpose of easy illustration, circles (yellow) denote object-related local features, triangles (green) denote background-related local features, and the numbers indicate the fraction of the respective local features in each pooling region.

object location information can be very useful for further pushing the state-of-the-art performance of image classification.

Of course, the challenge for OCP is deriving accurate enough location information for improving classification performance. If the derived location information is not sufficiently accurate, it can end up hurting classification accuracy. There is interesting previous work on learning object detectors using only image-level class labels (or weak labels) [40, 136]. Although these methods yield impressive localization results, they are formulated as detection tasks and have not been shown to be helpful for improving image classification performance. Methods such as [129, 13, 22, 102, 30, 224, 57] attempt to localize objects to improve image classification accuracy but only demonstrate results on simple datasets such as subsets of Caltech101 classes. In contrast, we evaluate our proposed OCP method on the highly cluttered PASCAL07 data [48], where we are able to localize objects with accuracy comparable to state-of-the-art weakly supervised object localization methods [40, 136] as well as to significantly improve image classification performance. To the best of our knowledge, this work is the first to use weakly supervised object detection to improve image classification on PASCAL07, which is considered a challenging object detection dataset even when bounding box annotations are provided for training.

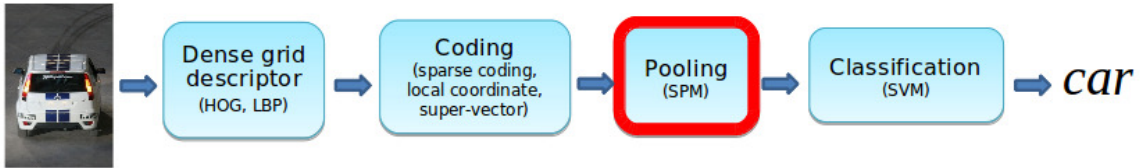


Figure 4.2: A popular image classification pipeline of state-of-the-art methods (at the time of publication of this work) [203, 227, 215]. In this chapter we focus on the pooling step and propose an object-centric spatial pooling approach which achieves superior classification accuracy compared to the SPM pooling.

4.2 Related work

Classification. Many state-of-the-art image classification systems (at time of publication of this work) follow the popular image feature extraction procedure [203, 227, 215] shown in Figure 4.2. First, for each image, low-level descriptors like DHOG [34] or LBP [2] are sampled on a dense grid. They are then coded into higher dimensions through vector quantization, local coordinate coding (LCC) [203] or sparse coding [215]. Finally the coded vectors are pooled together, typically using SPM [112] pooling, to form the image-level representation. Much research in image classification has been focused on the former two steps, namely on different types of low-level descriptors [34, 2, 123] and coding methods [203, 215, 86, 63, 144]. In this chapter we focus on the spatial pooling step, replacing the popular SPM with our object-centric pooling.

Methods such as [129, 13, 22, 102, 30, 224, 57] use localization information learned in a weakly supervised way to help boost classification accuracy by focusing on pooling low-level object features without background features. However, most of them only validate their approach on less cluttered and mostly centered datasets such as subsets of Caltech101 categories, Oxford Flowers 17 dataset, etc. For example, recently Feng et al. [57] presented a geometric pooling approach which resizes each image to the same size and learns a class-specific weighting factor for each grid position in an image. On the Caltech101 dataset, where most images are roughly aligned and centered, this method greatly improves over the previous state-of-the-art [203]. However, it has difficulty handling cluttered images like the ones of PASCAL07 [48]. Further, Nguyen et al. [129] and Bilen et al. [13] explicitly mention that some degree of context information (like road for cars) needs to be included into the detected object bounding box in order to be useful for image classification. This leads to very rough object localization even on simple datasets. In contrast, our work deals with high intra-class variability in object location and our proposed generic object-centric spatial pooling approach yields both classification improvements as well as competitive object localization results on the challenging PASCAL07 data.

If object location information is available during training, methods such as [55, 191] have been used to detect the object of interest, and [78, 172] showed how to use the output of object detectors

to boost classification performance. There are two main differences compared our approach. First, we focus on the purely classification setting where no annotations beyond image-level class labels are available during training. Second, we learn a joint model for both localization and classification instead of combining the scores of the two tasks as post-processing.

Weakly supervised localization. There is a large body of work on weakly supervised object localization [40, 136, 157, 101, 27]. Most of these methods use HOG-type low-level features [34] which are faster for detection but have been shown to be inferior than bag-of-words models for classification [203, 191]. The current state of the art is the work of Pandey and Lazebnik [136] which uses deformable parts-based models [55] trained discriminatively in a weakly supervised fashion for object localization. In contrast, our goal here is image classification (not object localization) although we do utilize localization as an intermediate step.

The state-of-the-art in weakly supervised localization has improved further since the publication of this work [81, 171, 202, 178].

4.3 Object-centric spatial pooling (OCP) for image classification

Let’s first use an empirical experiment to quantitatively see how object location information can dramatically improve image classification performance. On the PASCAL07 classification dataset [48], we trained two classifiers for each object class: one classifier using features extracted from the full image, and the other classifier using features extracted only from the provided tight bounding boxes around the objects. We followed [203] in extracting image features and training linear classifiers. Both classifiers were trained on the training set and tested on the validation set. The former classifier (trained on full images) yielded 52.0% mean average precision (mAP), whereas the latter classifier (trained and tested on tight bounding boxes) achieved an astonishing 69.7% mAP. In comparison the current state-of-the-art classification result with a single type of low-level descriptor (which used a more involved coding method as well as significant post-processing) [227] is just 59.2% mAP. Therefore, it is evident that learning to properly localize the object in the image holds great promise for improving classification accuracy.

Now, the challenge is deriving accurate enough location information to help classification. Obviously, if the location information is not reliable enough, it can easily end up hurting classification performance instead. Reliable localization becomes very challenging on generic dataset like PASCAL07 [48] where objects vary greatly in appearance and viewpoint, are often occluded, and appear in highly cluttered and unstructured scenes. In fact, most work on weakly supervised localization uses simpler datasets [129, 13, 157, 101, 27]. Recently, Deselaers et al. [40] were the first to tackle

PASCAL07. To simplify the problem, however, they trained object class models separately for different viewpoints of objects. We are interested in learning generic object detectors without any additional annotations and evaluating classification performance on the original 20 object classes. To the best of our knowledge we are the first to do so.

To this end, we introduce a novel framework of *object-centric spatial pooling* (OCP) for image classification. OCP consists of two steps: (1) inferring the location of the objects of interested; and (2) pooling low-level features from the foreground and the background separately to form the image-level representation. In order to infer the object locations, we propose an iterative procedure for learning object detectors from only image class labels (or weak labels). Very different from existing methods for learning weakly supervised object detectors [40, 136], our approach directly optimizes the classification objective function and uses object detection as an intermediate step. This is described in Section 4.3.1. More importantly, OCP enables feature sharing between classification and detection: the resulting feature representation of OCP can be seen as both a bounding box representation (for detection) and an image representation (for classification). This is described in detail in Section 4.3.2. As we show in Section 4.4, such feature sharing plays an essential role in improving classification performance.

4.3.1 Classification formulation

We assume we are dealing with the binary image classification problem since multi-class classification is often solved in practice by training one-versus-all binary classifiers. Given N data pairs, $\{\mathbf{I}_i, y_i\}_{i=1}^N$, where I_i is the i^{th} image and $y_i \in \{+1, -1\}$ is a binary label of the image, the SVM formulation for binary image classification with OCP becomes

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i^N \xi_i \quad (4.1)$$

$$\text{s.t. } y_i \max_{B \in \mathcal{BB}(i)} [\mathbf{w}^T \mathcal{P}_B(I_i) + b] \geq 1 - \xi_i \quad (4.2)$$

$$\xi_i \geq 0 \quad \forall i \quad (4.3)$$

where \mathbf{w} is SVM weight vector, b is bias term, $\mathcal{P}_B(I_i)$ is the image feature representation of image I_i using OCP with given bounding box B , and $\mathcal{BB}(i)$ is the collection of all bounding box windows within image I_i . $\mathcal{BB}(i)$ can be obtained by either densely sampling sliding windows or by using salient regions [191]. We do not require any ground truth localization information in this optimization.

Interestingly, the above formulation can also be viewed as multi-instance learning (MIL) for object detection [129]. However, as in [129], the traditional MIL formation often only uses the foreground for constructing the bounding box features and discards the background information. This has its drawbacks in both detection and classification. As a result, the method of [129] was not able to accurately localize objects even on simpler datasets such as Caltech101; it tended to choose regions

which were larger than the object of interest to encompass contextual information for classification. We fix these drawbacks by using a foreground-background representation, as described below. As a result, we are able to localize objects on the significantly more challenging PASCAL07 [48] with accuracy comparable to state-of-the-art weakly supervised object localization methods [40, 136].

4.3.2 Foreground-background feature representation

In the classification formulation in Eq. 4.3, the foreground-background feature representation of OCP provides a natural mechanism for feature sharing between classification and detection. In fact, even for standalone detection and classification, the foreground-background feature representation is advantageous compared to traditional foreground-only feature representation.

Foreground-background for classification. The foreground-background feature representation provides stronger classification performance than its foreground-only counterpart. This is not surprising since the background provides strong scene context for classification [102, 132]. For example, for the class *boat*, the surrounding water in the image may provide a strong clue that this image contains a boat; similarly, seeing road at the bottom of an image can strongly indicate that this image is likely about *cars*. Going back to the classifiers trained on the tight bounding boxes as described at the beginning of Section 3, if we replace the foreground-only feature representation with the foreground-background representation, we further improve the classification mAP from 69.7% to 71.1%. This highlights the fact that the foreground-background feature representation carries important information for classification which may be missing in the foreground-only representation. This is illustrated in Figure 4.3.

Foreground-background for detection. Object detectors trained with the foreground-background features also tend to yield more accurate bounding boxes during detection. Since the foreground and background models are learned jointly, they will prevent the object appearance features from leaking into the background, and context features from leaking into the foreground. This is illustrated in Figure 4.4. To validate the effectiveness of the foreground-background feature representation for detection, we also experimented on PASCAL07, training fully supervised object detectors using the



Figure 4.3: Example images which were misclassified using just the foreground representation but correctly classified when using the foreground-background representation.

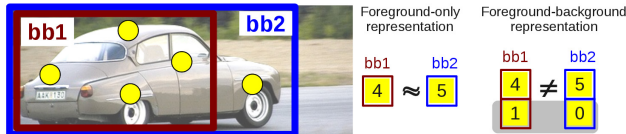


Figure 4.4: Bounding boxes bb_1 and bb_2 have a similar foreground-only feature representation, but they are very different under the foreground-background representation. Here, the numbers denote the count of object-related descriptors. For bb_1 , parts of object that leaked into the background will be greatly discounted by the background model.

foreground-only and the foreground-background feature representation respectively. It was no surprise that the foreground-background feature representation yielded significantly better detection performance. Here we skip the details of the experiments for simplicity since supervised detection is not the major focus of this chapter. In Figure 4.6 in the experimental results section, however, we show the differences in detections made with the foreground-only and the foreground-background model in our OCP framework.

With the foreground-background representation of OCP, optimizing the formulation in Eq. 4.3 can be seen as a simultaneous detection and classification procedure. This is because the foreground-background representation can be seen as both a bounding box representation (for detection) and an image-level representation (for classification).

4.3.3 Optimization

Now that we have defined our objective and our foreground-background feature representation, we discuss how to optimize this formulation. The optimization in Eq. 4.1 is non-convex because of the maximization operation in the constraints, thus we need to be careful during optimization to avoid local minima. In particular, since we are not given any localization information during training, our optimization algorithm consists of an outer loop that bootstraps the background region from the foreground and an inner loop that trains the appearance model.

Outer loop: bootstrapping background regions. In a purely classification setting, no foreground and background annotations are provided initially. We initialize the background region by cropping out a 16-pixel border of each image. Then the outer loops bootstraps the background by gradually shrinking the smallest bounding box considered in the bounding box search ($\mathcal{BB}(i)$ in Eq. 4.1). Thus we begin localizing using large windows and iteratively allow smaller and smaller windows as we learn more and more accurate models. As the background region is allowed to grow, the algorithm learns more and more accurate background models. If the algorithm goes too aggressively, it will end up in bad local minima. For example, if the localization is so inaccurate that many features from the object of interest appear in the background region, the model would learn that objects features actually belong to the background. This would lead to bad classification models

which are hard to correct in later iterations. However, as long as such bad local minima are avoided, the specific rate of shrinking the foreground region does not affect performance in our experiments.

Inner loop: learning the appearance model for detection. Given the current constraint on the background size, we need to learn the best object appearance model. This is done in two steps: (1) detection, where given the current appearance model we find the best possible object location from positive images (images that are known to contain the object of interest); and (2) classification, where given the proposed bounding boxes from positive images as positive examples and a large sample of bounding boxes from negative images as negative examples, we construct the bounding box representation using OCP and then train a binary SVM classifier for discriminating the positive bounding boxes from the negative bounding boxes. In contrast to more common treatments which would need another loop to bootstrap the difficult negative bounding boxes and iteratively improve the SVM model, here we get rid of this loop by solving an SVM optimization directly with all (often millions) negative bounding boxes.

We make use of the candidate image regions proposed in an unsupervised fashion by [191] to avoid both sampling too many negative windows for classification and running sliding windows search for detection. Since the candidate bounding boxes aim to achieve high recall rate ($> 96\%$), we ended up with 1000~3000 candidate bounding boxes per image. For PASCAL07, we have 5011 images in the training and validation sets. Therefore, for each inner loop, we need to solve for 20 binary SVMs with about 10 million data examples. Furthermore, our feature representation for OCP is very high-dimensional: we used a codebook of 8192 for LLC coding [203], pool the low-level features on the foreground region using 1×1 and 3×3 SPM pooling regions [112], and separately pool all low-level features in the background, thus resulting in a feature vector of dimension $8192 \times 11 = 90112$. Indeed, if we save all the feature vectors from the 5011 images, this would require more than 700G of space. Most off-the-shelf SVM solvers would not be able to handle such a large-scale problem. So, we developed a stochastic gradient descent algorithm with averaging using a similar idea to [119]. We were able to run an inner loop in 7~8 hours and to finish the training (inner loop and outer loop) in about 3 days on a single machine.

4.4 Experiments

We validate our approach on the challenging PASCAL07 dataset [48], containing 5011 images for training and validation, and 4952 images for testing. This dataset consists of 20 object categories, with object instances occurring in a variety of scales, locations and viewpoints.

Image representation. For low-level features, we extract DHOG [34] features with patch sizes 16×16 , 25×25 , 31×31 and 46×46 . We then run Linear Locality-Constrained (LLC) coding [203] using a codebook of size 8192 and 5 nearest neighbors. For the baseline representation, we pool the

DHOG features using 1×1 and 3×3 SPM pooling regions [112] over the full image. Thus each image is represented using a feature vector of dimension $8192 \times 10 = 81920$. For our object-centric pooling, we use the same SPM representation but on the foreground region and also pool over all low-level features in the background separately, thus giving us a feature dimension of $8192 \times 11 = 90112$.

4.4.1 Joint classification and localization

The main insight behind our approach is that object classification and detection can be mutually beneficial. In particular, as the classification accuracy improves we expect detection accuracy to improve as well, and vice versa. We begin by verifying that this is indeed the case. Figure 4.5 shows the steady improvement in mean average precision on both classification and detection over the iterations (outer loop) of our algorithms. As a baseline (iteration 0), we use a classifier trained on full images with the SPM spatial pooling representation, which is equivalent to assuming an empty background region in foreground-background representation. Interestingly, even after just one iteration, our classification mAP is already 54.8%, which is 0.5% greater than the 54.3% SPM classification result.² In the end our OCP method achieves 57.2% classification mAP, significantly outperforming the SPM representation. In fact, it significantly outperforms even a much richer 4-level SPM representation of size 8192×30 which achieves only 54.8% classification mAP. On the detection side, our approach was able to improve the baseline of 6.10% detection mAP to the final 15.0%.

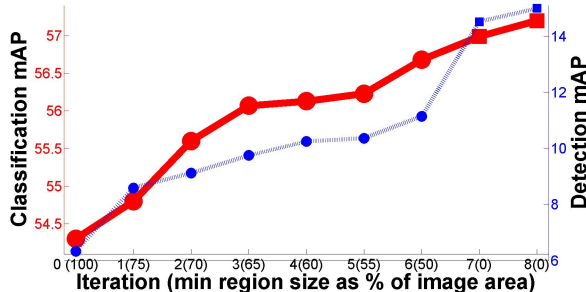


Figure 4.5: Classification and detection mAP on the PASCAL07 test set over the iterations of our joint detection and classification approach. The red solid line is classification mAP, and the blue dotted line is detection mAP. We see a steady joint improvement of classification and detection accuracy.

It is important to note that jointly optimizing detection and classification using OCP as in Eq. 4.3 plays an essential role in achieving the joint improvements for classification and detection. As we

²We make use of only one type of low-level image descriptor in contrast to [172, 74], and don't do any additional post-processing of the features in contrast to [203, 227]. The work of [203] gives 59.3% classification mAP on this dataset when using LLC coding, but this relied on significant post-processing of the resulting image features. To simplify the comparison, we do not involve the post-processing.

Object class	SPM method	OCP method
Aeroplane	72.5	74.2
Bicycle	56.3	63.1
Bird	49.5	45.1
Boat	63.5	65.9
Bottle	22.4	29.5
Bus	60.1	64.7
Car	76.4	79.2
Cat	57.5	61.4
Chair	51.9	51.0
Cow	42.2	45.0
Dining table	48.9	54.8
Dog	38.1	45.4
Horse	75.1	76.3
Motorbike	62.8	67.1
Person	82.9	84.4
Potted plant	20.5	21.8
Sheep	38.1	44.3
Sofa	46.0	48.8
Train	71.7	70.7
TV/monitor	50.5	51.7
Mean	54.3	57.2

Table 4.1: Classification AP of object-centric spatial pooling compared to the standard SPM spatial pooling on the PASCAL07 test set.

show below, when detection and classification are optimized separately, higher detection accuracy may not always means higher classification accuracy.

4.4.2 Image classification

OCP significantly boost of classification accuracy on most of the 20 object classes, as shown in Table 4.1. In particular, OCP achieves significant improvement on the following categories: dog (7.3% improvement), bottle (7.1%), bicycle (6.8%), sheep (6.2%), diningtable (5.9%), bus (4.6%), motorbike (4.3%) and even 1.3% on the notoriously difficult potted plant category. Noticeably, many of these categories are relatively small objects (like bottles) embedded in cluttered environments. OCP greatly improves classification accuracy on these categories by making an effort to localize the objects.

There are three categories that proved difficult for OCP to improve: chairs (-0.9%), trains (-1.0%) and birds (-4.4%). For the bird and chair categories, the objects are often occluded (e.g., birds are often occluded by trees, and chairs are often occluded by people sitting on them), which make them very challenging for detection even when bounding box annotations are available (see [55, 48]). For the slight drop in the train category, since trains are already relatively well-centered

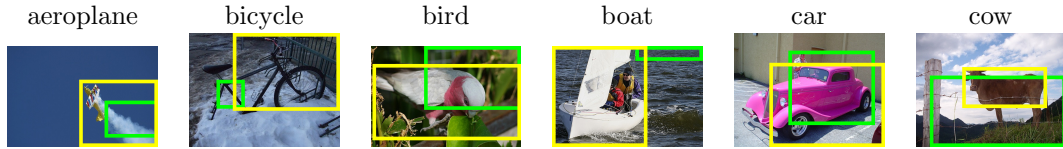


Figure 4.6: Images where object-centric pooling with the foreground-background model (yellow) localizes objects more accurately than the foreground-only model (green).

in images, SPM pooling alone yields very satisfactory classification accuracy (71.7%) and is difficult to further improve.

We also investigate using the foreground-only (instead of the foreground-background) feature representation when optimizing Eq. 4.3.³ This foreground-only representation leads to an improvement from the baseline SPM model – the mAP increases from 54.3% to 55.7%. This is a 1.4% improvement as compared to the 2.9% improvement as in the case of our foreground-background representation. Figure 4.6 illustrates some location results, showing that foreground-background representation often yields better localization.

4.4.3 Weakly supervised object localization

Even though our primary goal is image classification, the proposed object-centric spatial pooling also accurately localizes the objects of interest. PASCAL07 is a very challenging dataset for weakly supervised localization (where bounding box information is not available during training). Only a few recent works have tackled this data (Deselaers et al. [40] and Pandey and Lazebnik [136]). They focused on localizing only a handful of the object classes and use the available viewpoint annotations during training to assist learning. In contrast, we work on the full dataset without using these additional annotations to mimic the purely classification setting.

Weakly supervised localization can be evaluated directly on the training set (in our case the PASCAL07 trainval set) since only image-level class labels are available during training. Following [40, 136] we compute localization accuracy as the percentage of training image in which an instance was correctly localized by the highest-scoring detection according to the PASCAL criterion (window intersection over the union $\geq 50\%$). On the 14 classes of PASCAL07-*all*⁴ introduced by [40], our localization accuracy is 27.4%, which is comparable to 26% of [40] using additional viewpoint annotations and 30.0% of [136].

As we’re most interested in inferring object location on unseen images, we evaluate the detection accuracy on the test set as well. Table 4.2 compares our detection average precision on six PASCAL07-6x2 classes [40] evaluated on all test images with the current state-of-the-art in weakly

³This experiment is a more assertive version of the technique described in Nguyen et al. [129]: the optimization framework is similar to [129] but with significantly stronger low-level descriptors (HOG descriptors [34] with LLC coding [203] compared to vector-quantized SIFT [123]) and with much more negative training data.

⁴PASCAL07-*all* includes all classes of PASCAL07 except bird, car, cat, cow, dog and sheep. [40]

Object class		Deselaers [40]	Pandey [136]	OCP method
Aeroplane	Left	9.1	7.5	30.8
	Right	23.6	21.1	
Bicycle	Left	33.4	38.5	25.0
	Right	49.4	44.8	
Boat	Left	0.0	0.3	3.6
	Right	0.0	0.5	
Bus	Left	0.0	0.0	26.0
	Right	16.4	0.3	
Horse	Left	9.6	45.9	21.3
	Right	9.1	17.3	
Motorbike	Left	20.9	43.8	29.9
	Right	16.1	27.2	
Average		16.0	20.8	22.8

Table 4.2: Comparison of detection AP on the PASCAL07-6x2 test set for our method versus [40, 136]. Both [40, 136] split up the objects by left and right viewpoint to make the models easier to learn. We do not make use of these additional labels and learn a single model for each object.

supervised localization. We obtain 22.8%, outperforming the previous best 20.8% of [136] which used additional viewpoint annotations. On all 20 classes, we obtained 15.0% detection mAP compared to 29.1% mAP of the state-of-the-art deformable part-based model that used bounding box labels for detector training [55].

Figure 4.7 shows some examples of our detection results on PASCAL07 test set. Localization is often quite reasonable, which is amazing considering the difficulty of the dataset and the lack of any bounding box annotations during training. Even on images with multiple object instances our method is sometimes able to separate out the different instances.

Interestingly, when we used the location information derived from the deformable part-based model mentioned above [55] learned with the help of bounding box annotations, images features constructed using our image representation with the foreground-background pooling yielded a classification mAP of 56.9%. This is inferior to the aforementioned 57.2% classification mAP obtained using OCP, where our proposed approach in Eq. 4.3 did not use any bounding box annotations and only achieved 15.0% detection mAP. This strongly highlights the importance of the formulation in Eq. 4.3, which uses classification as the major optimization objective and jointly optimizes detection and classification when solving the optimization.

4.5 Conclusions

We presented an object-centric spatial pooling (OCP) approach for improving classification performance. The challenge of OCP is training reliable object detectors with no available bounding box

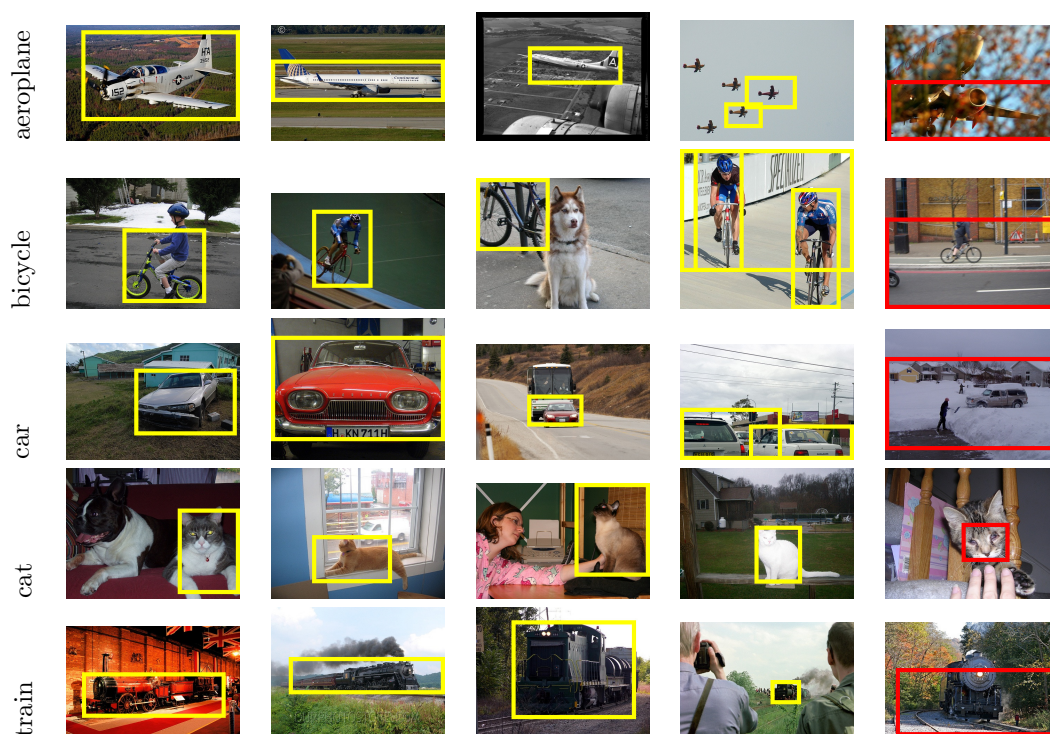


Figure 4.7: Foreground regions detected by the object-centric pooling framework on PASCAL07 test images. The models are learned without any ground truth localization information. Yellow boxes correspond to correct detections and red boxes are failed detections. On images where multiple instances of a object class are presented, we show the top few detections after running non-maximal suppression.

annotations as in a typical classification setting. We propose a framework that directly optimizes classification objective with detection being treated as an intermediate step. The key to this framework is the foreground-background feature representation by OCP that naturally enables feature sharing between detection and classification. Our results on the challenging PASCAL07 dataset show that not only is the proposed OCP approach able to improve the classification accuracy compared to using SPM pooling, but it also yields very reasonable object detection results. We believe this is an important step toward better image understanding – not only deciding *what* objects are in an image but also figuring out *where* these objects are.

Our future work includes incorporating bounding box annotations during training (from all or just a subset of images) to further improve the classification performance. We are also very interested in exploiting even more powerful visual features than the simple LLC feature as used in this chapter. As demonstrated by the motivation experiment described in the beginning of Section 3, there is much room for improving classification performance by utilizing location information. This chapter is just an initial step toward that direction.

Chapter 5

ImageNet Large Scale Visual Recognition Challenge

5.1 Introduction

Overview. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has been running annually for five years (since 2010) and has become the standard benchmark for large-scale object recognition.¹ ILSVRC follows in the footsteps of the PASCAL VOC challenge [47], established in 2005, which set the precedent for standardized evaluation of recognition algorithms in the form of yearly competitions. As in PASCAL VOC, ILSVRC consists of two components: (1) a publically available *dataset*, and (2) an annual *competition* and corresponding workshop. The dataset allows for the development and comparison of categorical object recognition algorithms, and the competition and workshop provide a way to track the progress and discuss the lessons learned from the most successful and innovative entries each year.

The publically released dataset contains a set of manually annotated *training* images. A set of *test* images is also released, with the manual annotations withheld.² Participants train their algorithms using the training images and then automatically annotate the test images. These predicted annotations are submitted to the *evaluation server*. Results of the evaluation are revealed at the end of the competition period and authors are invited to share insights at the workshop held at the International Conference on Computer Vision (ICCV) or European Conference on Computer Vision (ECCV) in alternate years.

ILSVRC annotations fall into one of two categories: (1) *image-level annotation* of a binary label

¹In this chapter, we will be using the term *object recognition* broadly to encompass both *image classification* (a task requiring an algorithm to determine what object classes are present in the image) as well as *object detection* (a task requiring an algorithm to localize all objects present in the image).

²In 2010, the test annotations were later released publicly; since then the test annotation have been kept hidden.

for the presence or absence of an object class in the image, e.g., “there are cars in this image” but “there are no tigers,” and (2) *object-level annotation* of a tight bounding box and class label around an object instance in the image, e.g., “there is a screwdriver centered at position (20,25) with width of 50 pixels and height of 30 pixels”.

Large-scale challenges and innovations. In creating the dataset, several challenges had to be addressed. Scaling up from 19,737 images in PASCAL VOC 2010 to 1,461,406 in ILSVRC 2010 and from 20 object classes to 1000 object classes brings with it several challenges. It is no longer feasible for a small group of annotators to annotate the data as is done for other datasets [51, 31, 47, 213]. Instead we turn to designing novel crowdsourcing approaches for collecting large-scale annotations [175, 36, 38].

Some of the 1000 object classes may not be as easy to annotate as the 20 categories of PASCAL VOC: e.g., bananas which appear in bunches may not be as easy to delineate as the basic-level categories of aeroplanes or cars. Having more than a million images makes it infeasible to annotate the locations of all objects (much less with object segmentations, human body parts, and other detailed annotations that subsets of PASCAL VOC contain). New evaluation criteria have to be defined to take into account the facts that obtaining perfect manual annotations in this setting may be infeasible.

Once the challenge dataset was collected, its scale allowed for unprecedented opportunities both in evaluation of object recognition algorithms and in developing new techniques. Novel algorithmic innovations emerge with the availability of large-scale training data. The broad spectrum of object categories motivated the need for algorithms that are even able to distinguish classes which are visually very similar. We highlight the most successful of these algorithms in this chapter.

Finally, the large variety of object classes in ILSVRC allows us to perform an analysis of statistical properties of objects and their impact on recognition algorithms. This type of analysis allows for a deeper understanding of object recognition, and for designing the next generation of general object recognition algorithms.

My contributions. ILSVRC is a team effort over six years from 2010-2015. My contributions are:

1. I was the lead organizer of the ILSVRC competition and workshop in years 2013 and 2014.
2. I took the lead in writing a retrospective report about the first five years of ILSVRC including the challenges we addressed when scaling up object recognition, an analysis of the dataset and lessons we learned along the way [152]. This contribution is documented in this chapter.
3. I led the effort to create the large-scale object detection dataset of ILSVRC. This work is described in Chapter 6.

4. I led the effort to provide a detailed analysis of the state of the field of object recognition over multiple years. This work is described in Chapter 7.

This chapter will focus specifically on these two contributions. The collected dataset and additional information about ILSVRC can be found in [152] and at:

<http://image-net.org/challenges/LSVRC/>

5.2 Related work

We briefly discuss some prior work in constructing benchmark image datasets.

Image classification datasets. Caltech 101 [51] was among the first standardized datasets for multi-category image classification, with 101 object classes and commonly 15-30 training images per class. Caltech 256 [72] increased the number of object classes to 256 and added images with greater scale and background variability. The TinyImages dataset [183] contains 80 million 32x32 low resolution images collected from the internet using synsets in WordNet [128] as queries. However, since this data has not been manually verified, there are many errors, making it less suitable for algorithm evaluation. Datasets such as 15 Scenes [131, 52, 112] or recent Places [225] provide a single scene category label (as opposed to an object category).

The ImageNet dataset [36] is the backbone of ILSVRC. ImageNet is an image dataset organized according to the WordNet hierarchy [128]. Each concept in WordNet, possibly described by multiple words or word phrases, is called a “synonym set” or “synset”. ImageNet populates 21,841 synsets of WordNet with an average of 650 manually verified and full resolution images. As a result, ImageNet contains 14,197,122 annotated images organized by the semantic hierarchy of WordNet (as of August 2014). ImageNet is larger in scale and diversity than the other image classification datasets. ILSVRC uses a subset of ImageNet images for training the algorithms and some of ImageNet’s image collection protocols for annotating additional images for testing the algorithms.

Image parsing datasets. Many datasets aim to provide richer image annotations beyond image-category labels. LabelMe [158] contains general photographs with multiple objects per image. It has bounding polygon annotations around objects, but the object names are not standardized: annotators are free to choose which objects to label and what to name each object. The SUN2012 [213] dataset contains 16,873 manually cleaned up and fully annotated images more suitable for standard object detection training and evaluation. SIFT Flow [122] contains 2,688 images labeled using the LabelMe system. The LotusHill dataset [216] contains very detailed annotations of objects in 636,748 images and video frames, but it is not available for free. Several datasets provide pixel-level segmentations: for example, MSRC dataset [31] with 591 images and 23 object classes, Stanford Background Dataset [68] with 715 images and 8 classes, and the Berkeley Segmentation dataset [5]

with 500 images annotated with object boundaries. OpenSurfaces segments surfaces from consumer photographs and annotates them with surface properties, including material, texture, and contextual information [7].

The closest to ILSVRC is the PASCAL VOC dataset [48, 46], which provides a standardized test bed for object detection, image classification, object segmentation, person layout, and action classification. Much of the design choices in ILSVRC have been inspired by PASCAL VOC and the similarities and differences between the datasets are discussed at length throughout the paper. ILSVRC scales up PASCAL VOC’s goal of standardized training and evaluation of recognition algorithms by more than an order of magnitude in number of object classes and images: PASCAL VOC 2012 has 20 object classes and 21,738 images compared to ILSVRC2012 with 1000 object classes and 1,431,167 annotated images.

The recently released COCO dataset [118] contains more than 328,000 images with 2.5 million object instances manually segmented. It has fewer object categories than ILSVRC (91 in COCO versus 200 in ILSVRC object detection) but more instances per category (27K on average compared to about 1K in ILSVRC object detection). Further, it contains object segmentation annotations which are not currently available in ILSVRC. COCO is likely to become another important large-scale benchmark.

Large-scale annotation. ILSVRC makes extensive use of Amazon Mechanical Turk to obtain accurate annotations [173]. Works such as [207, 167, 196] describe quality control mechanisms for this marketplace. [198] provides a detailed overview of crowdsourcing video annotation. A related line of work is to obtain annotations through well-designed games, e.g. [197]. Some of our novel approaches to crowdsourcing accurate image annotations are in Chapter 6 and [152, 38].

Standardized challenges. There are several datasets with standardized online evaluation similar to ILSVRC: the aforementioned PASCAL VOC [47], Labeled Faces in the Wild [85] for unconstrained face recognition, Reconstruction meets Recognition [187] for 3D reconstruction and KITTI [65] for computer vision in autonomous driving. These datasets along with ILSVRC help benchmark progress in different areas of computer vision. Works such as [182] emphasize the importance of examining the bias inherent in any standardized dataset.

5.3 Challenge tasks

The goal of ILSVRC is to estimate the content of photographs for the purpose of retrieval and automatic annotation. Test images are presented with no initial annotation, and algorithms have to produce labelings specifying what objects are present in the images. New test images are collected and labeled especially for this competition and are not part of the previously published ImageNet dataset [36].

Task		Image classification	Single-object localization	Object detection
Manual labeling on training set	Number of object classes annotated per image	1	1	1 or more
	Locations of annotated classes	—	all instances on some images	all instances on all images
Manual labeling on validation and test sets	Number of object classes annotated per image	1	1	all target classes
	Locations of annotated classes	—	all instances on all images	all instances on all images

Table 5.1: Overview of the provided annotations for each of the tasks in ILSVRC.

ILSVRC over the years has consisted of one or more of the following tasks (years in parentheses):³

1. **Image classification** (2010-2014): Algorithms produce a list of object categories present in the image.
2. **Single-object localization** (2011-2014): Algorithms produce a list of object categories present in the image, along with an axis-aligned bounding box indicating the position and scale of *one* instance of each object category.
3. **Object detection** (2013-2014): Algorithms produce a list of object categories present in the image along with an axis-aligned bounding box indicating the position and scale of *every* instance of each object category.

This section provides an overview and history of each of the three tasks. Table 5.1 shows summary statistics.

5.3.1 Image classification task

Data for the image classification task consists of photographs collected from Flickr⁴ and other search engines, manually labeled with the presence of one of 1000 object categories. Each image contains one ground truth label.

For each image, algorithms produce a list of object categories present in the image. The quality of a labeling is evaluated based on the label that best matches the ground truth label for the image (see Section 5.5.1).

Constructing ImageNet was an effort to scale up an image classification dataset to cover most nouns in English using tens of millions of manually verified photographs [36]. The image classification

³In addition, ILSVRC in 2012 also included a taster fine-grained classification task, where algorithms would classify dog photographs into one of 120 dog breeds [100]. Fine-grained classification has evolved into its own Fine-Grained classification challenge in 2013 [9], which is outside the scope of this paper.

⁴www.flickr.com

task of ILSVRC came as a direct extension of this effort. A subset of categories and images was chosen and fixed to provide a standardized benchmark while the rest of ImageNet continued to grow.

5.3.2 Single-object localization task

The single-object localization task, introduced in 2011, built off of the image classification task to evaluate the ability of algorithms to learn the appearance of the target object itself rather than its image context.

Data for the single-object localization task consists of the same photographs collected for the image classification task, hand labeled with the presence of one of 1000 object categories. Each image contains one ground truth label. Additionally, every instance of this category is annotated with an axis-aligned bounding box.

For each image, algorithms produce a list of object categories present in the image, along with a bounding box indicating the position and scale of one instance of each object category. The quality of a labeling is evaluated based on the object category label that best matches the ground truth label, with the additional requirement that the location of the predicted instance is also accurate (see Section 5.5.2).

5.3.3 Object detection task

The object detection task went a step beyond single-object localization and tackled the problem of localizing multiple object categories in the image. This task has been a part of the PASCAL VOC for many years on the scale of 20 object categories and tens of thousands of images, but scaling it up by an order of magnitude in object categories and in images proved to be very challenging from a dataset collection and annotation point of view (see Chapter 6).

Data for the detection tasks consists of new photographs collected from Flickr using scene-level queries. The images are annotated with axis-aligned bounding boxes indicating the position and scale of every instance of each target object category. The training set is additionally supplemented with (a) data from the single-object localization task, which contains annotations for all instances of just one object category, and (b) negative images known not to contain any instance of some object categories.

For each image, algorithms produce bounding boxes indicating the position and scale of all instances of all target object categories. The quality of labeling is evaluated by *recall*, or number of target object instances detected, and *precision*, or the number of spurious detections produced by the algorithm (see Section 5.5.3).

Image classification annotations (1000 object classes)

Year	Train images (per class)	Val images (per class)	Test images (per class)
ILSVRC2010	1,261,406 (668-3047)	50,000 (50)	150,000 (150)
ILSVRC2011	1,229,413 (384-1300)	50,000 (50)	100,000 (100)
ILSVRC2012-14	1,281,167 (732-1300)	50,000 (50)	100,000 (100)

Additional annotations for single-object localization (1000 object classes)

Year	Train images with bbox annotations (per class)	Train bboxes annotated (per class)	Val images with bbox annotations (per class)	Val bboxes annotated (per class)	Test images with bbox annotations
ILSVRC 2011	315,525 (104-1256)	344,233 (114-1502)	50,000 (50)	55,388 (50-118)	100,000
ILSVRC2012-14	523,966 (91-1268)	593,173 (92-1418)	50,000 (50)	64,058 (50-189)	100,000

Table 5.2: Scale of ILSVRC image classification task (top) and single-object localization task (bottom). The numbers in parentheses correspond to (minimum per class - maximum per class). The 1000 classes change from year to year but are consistent between image classification and single-object localization tasks in the same year. All images from the image classification task may be used for single-object localization.

5.4 Dataset statistics

Details of the construction of the large-scale image classification and single-object localization datasets are largely outside the scope of this thesis. Please refer to [36, 175, 152] for details. The details of the object detection dataset construction are in Chapter 6. In this section we present statistics of the collected datasets.

5.4.1 Image classification and single-object localization

The image classification and single-object localization tasks are closely related and thus analyzed together in this section. There are 1000 object classes and approximately 1.2 million training images, 50 thousand validation images and 100 thousand test images. As described in Section 5.3.2, each image corresponds to one target object class. Table 5.2 documents the size of the image classification datasets over the years of the challenge. Figure 5.1 demonstrates that the dataset provides an unprecedented challenge for algorithms by having many similar-looking fine-grained object classes.

For single-object localization, a subset of the image classification images are additionally annotated with bounding box labels using the annotation procedure described in [175, 152]. All 50 thousand images in the validation set and 100 thousand images in the test set are annotated with



Figure 5.1: The ILSVRC classification and single-object localization dataset contains many more fine-grained classes compared to the standard PASCAL VOC benchmark; for example, instead of the PASCAL "dog" category there are 120 different breeds of dogs in ILSVRC.

bounding boxes around all instances of the ground truth object class (one object class per image). In addition, in ILSVRC2011 25% of training images are annotated with bounding boxes the same way, yielding more than 310 thousand annotated images with more than 340 thousand annotated object instances. In ILSVRC2012 40% of training images are annotated, yielding more than 520 thousand annotated images with more than 590 thousand annotated object instances. Table 5.2 (bottom) documents the size of this dataset.

Appendix A contains additional analysis of the difficulty of these datasets compared to the standard PASCAL benchmark [48].

5.4.2 Object detection

Using the procedure described above, we collect a large-scale dataset for ILSVRC object detection task. There are 200 object classes and approximately 450K training images, 20K validation images and 40K test images. Table 5.3 documents the size of the dataset over the years of the challenge. The major change between ILSVRC2013 and ILSVRC2014 was the addition of 60,658 fully annotated training images.

Prior to ILSVRC, the object detection benchmark was the PASCAL VOC challenge [48]. ILSVRC has 10 times more object classes than PASCAL VOC (200 vs 20), 10.6 times more fully annotated

Object detection annotations (200 object classes)

Year	Train images (per class)	Train bboxes annotated (per class)	Val images (per class)	Val bboxes annotated (per class)	Test images
ILSVRC2013	395909 (417-561-66911 pos, 185-4130-10073 neg)	345854 (438-660- 73799)	21121 (23-58-5791 pos, rest neg)	55501 (31-111- 12824)	40152
ILSVRC2014	456567 (461-823-67513 pos, 42945-64614-70626 neg)	478807 (502-1008- 74517)	21121 (23-58-5791 pos, rest neg)	55501 (31-111- 12824)	40152

Table 5.3: Scale of ILSVRC object detection task. Numbers in parentheses correspond to (minimum per class - median per class - maximum per class).

training images (60,658 vs 5,717), 35.2 times more training objects (478,807 vs 13,609), 3.5 times more validation images (20,121 vs 5823) and 3.5 times more validation objects (55,501 vs 15,787). ILSVRC has 2.8 annotated objects per image on the validation set, compared to 2.7 in PASCAL VOC. The average object in ILSVRC takes up 17.0% of the image area and in PASCAL VOC takes up 20.7%; Chapter 6 has more details. Additionally, ILSVRC contains a wide variety of objects, including tiny objects such as sunglasses (1.3% of image area on average), ping-pong balls (1.5% of image area on average) and basketballs (2.0% of image area on average).

5.5 Evaluation at large scale

Once the dataset has been collected, we need to define a standardized evaluation procedure for algorithms. Some measures have already been established by datasets such as the Caltech 101 [51] for image classification and PASCAL VOC [47] for both image classification and object detection. To adapt these procedures to the large-scale setting we had to address three key challenges. First, for the image classification and single-object localization tasks only one object category could be labeled in each image due to the scale of the dataset. This created potential ambiguity during evaluation (addressed in Section 5.5.1). Second, evaluating localization of object instances is inherently difficult in some images which contain a cluster of objects (addressed in Section 5.5.2). Third, evaluating localization of object instances which occupy few pixels in the image is challenging (addressed in Section 5.5.3).

In this section we describe the standardized evaluation criteria for each of the three ILSVRC tasks. We elaborate further on these and other more minor challenges with large-scale evaluation.

5.5.1 Image classification

The scale of ILSVRC classification task (1000 categories and more than a million of images) makes it very expensive to label every instance of every object in every image. Therefore, on this dataset only one object category is labeled in each image. This creates ambiguity in evaluation. For example, an image might be labeled as a “strawberry” but contain both a strawberry and an apple. Then an algorithm would not know which one of the two objects to name. For the image classification task we allowed an algorithm to identify multiple (up to 5) objects in an image and not be penalized as long as one of the objects indeed corresponded to the ground truth label. Figure 5.2(top row) shows some examples.

Concretely, each image i has a single class label C_i . An algorithm is allowed to return 5 labels c_{i1}, \dots, c_{i5} , and is considered correct if $c_{ij} = C_i$ for some j .

Let the error of a prediction $d_{ij} = d(c_{ij}, C_i)$ be 1 if $c_{ij} \neq C_i$ and 0 otherwise. The error of an algorithm is the fraction of test images on which the algorithm makes a mistake:

$$\text{error} = \frac{1}{N} \sum_{i=1}^N \min_j d_{ij} \quad (5.1)$$

We used two additional measures of error. First, we evaluated top-1 error. In this case algorithms were penalized if their highest-confidence output label c_{i1} did not match ground truth class C_i . Second, we evaluated hierarchical error. The intuition is that confusing two nearby classes (such as two different breeds of dogs) is not as harmful as confusing a dog for a container ship. For the hierarchical criteria, the cost of one misclassification, $d(c_{ij}, C_i)$, is defined as the height of the lowest common ancestor of c_{ij} and C_i in the ImageNet hierarchy. The height of a node is the length of the longest path to a leaf node (leaf nodes have height zero).

However, in practice we found that all three measures of error (top-5, top-1, and hierarchical) produced the same ordering of results. Appendix B provides some more details. Thus, since ILSVRC2012 we have been exclusively using the top-5 metric which is the simplest and most suitable to the dataset.

5.5.2 Single-object localization

The evaluation for single-object localization is similar to object classification, again using a top-5 criteria to allow the algorithm to return unannotated object classes without penalty. However, now the algorithm is considered correct only if it both correctly identifies the target class C_i and accurately localizes one of its instances. Figure 5.2(middle row) shows some examples.

Concretely, an image is associated with object class C_i , with all instances of this object class annotated with bounding boxes B_{ik} . An algorithm returns $\{(c_{ij}, b_{ij})\}_{j=1}^5$ of class labels c_{ij} and

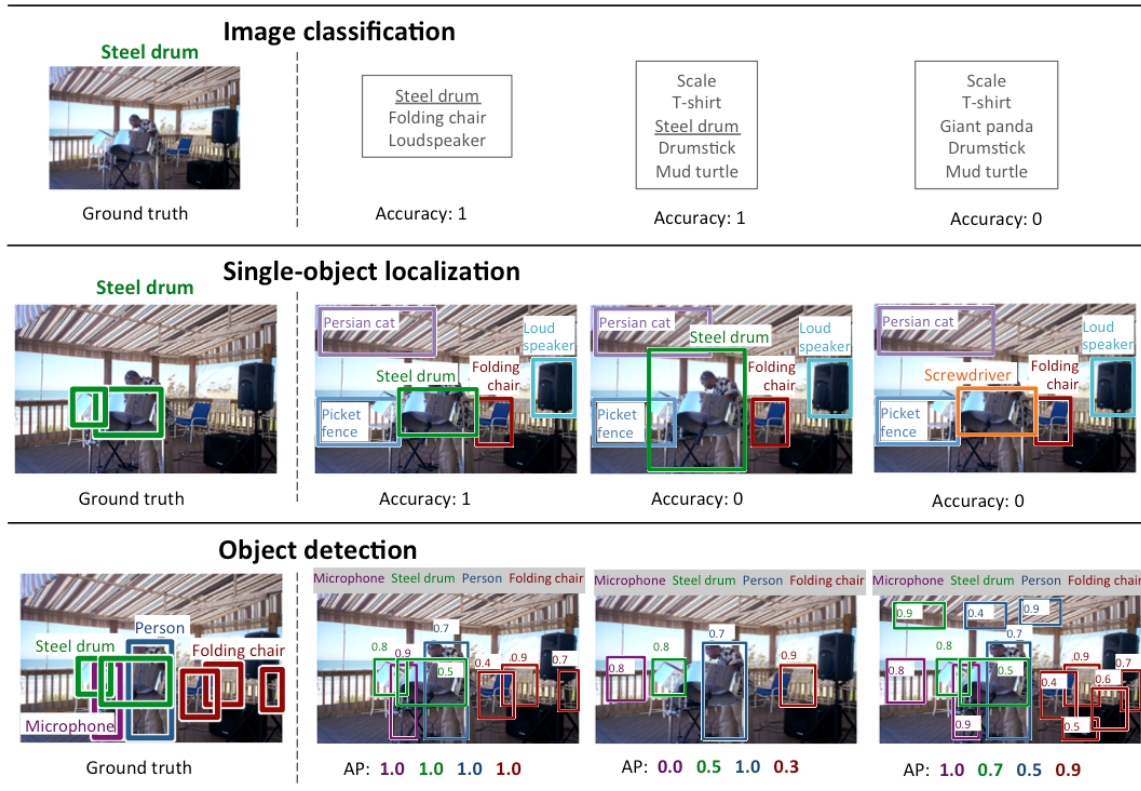


Figure 5.2: Tasks in ILSVRC. The first column shows the ground truth labeling on an example image, and the next three show three sample outputs with the corresponding evaluation score.

associated locations b_{ij} . The error of a prediction j is:

$$d_{ij} = \max(d(c_{ij}, C_i), \min_k d(b_{ij}, B_{ik})) \tag{5.2}$$

Here $d(b_{ij}, B_{ik})$ is the error of localization, defined as 0 if the area of intersection of boxes b_{ij} and B_{ik} divided by the areas of their union is greater than 0.5, and 1 otherwise. [48] The error of an algorithm is computed as in Eq. 5.1.

Evaluating localization is inherently difficult in some images. Consider a picture of a bunch of bananas or a carton of apples. It is easy to classify these images as containing bananas or apples, and even possible to localize a few instances of each fruit. However, in order for evaluation to be accurate *every* instance of banana or apple needs to be annotated, and that may be impossible. To handle the images where localizing individual object instances is inherently ambiguous we manually discarded 3.5% of images since ILSVRC2012. Some examples of discarded images are shown in Figure 5.3.



Figure 5.3: Images marked as “difficult” in the ILSVRC2012 single-object localization validation set. Please refer to Section 5.5.2 for details.

5.5.3 Object detection

The criteria for object detection was adopted from PASCAL VOC [48]. It is designed to penalize the algorithm for missing object instances, for duplicate detections of one instance, and for false positive detections. Figure 5.2(bottom row) shows examples.

For each object class and each image I_i , an algorithm returns predicted detections (b_{ij}, s_{ij}) of predicted locations b_{ij} with confidence scores s_{ij} . These detections are greedily matched to the ground truth boxes $\{B_{ik}\}$ using Algorithm 1. For every detection j on image i the algorithm returns $z_{ij} = 1$ if the detection is matched to a ground truth box according to the threshold criteria, and 0 otherwise. For a given object class, let N be the total number of ground truth instances across all images. Given a threshold t , define *recall* as the fraction of the N objects detected by the algorithm, and *precision* as the fraction of correct detections out of the total detections returned by the algorithm. Concretely,

$$Recall(t) = \frac{\sum_{ij} 1[s_{ij} \geq t]z_{ij}}{N} \quad (5.3)$$

$$Precision(t) = \frac{\sum_{ij} 1[s_{ij} \geq t]z_{ij}}{\sum_{ij} 1[s_{ij} \geq t]} \quad (5.4)$$

The final metric for evaluating an algorithm on a given object class is *average precision* over the different levels of recall achieved by varying the threshold t . The winner of each object class is then the team with the highest average precision, and then winner of the challenge is the team that wins on the most object classes.⁵

Difference with PASCAL VOC. Evaluating localization of object instances which occupy very few pixels in the image is challenging. The PASCAL VOC approach was to label such instances as “difficult” and ignore them during evaluation. However, since ILSVRC contains a more diverse set of object classes including, for example, “nail” and “ping pong ball” which have many very small instances, it is important to include even very small object instances in evaluation.

⁵In this paper we focus on the mean average precision across all categories as the measure of a team’s performance. This is done for simplicity and is justified since the ordering of teams by mean average precision was always the same as the ordering by object categories won.

Input: Bounding box predictions with confidence scores $\{(b_j, s_j)\}_{j=1}^M$ and ground truth boxes \mathcal{B} on image I for a given object class.

Output: Binary results $\{z_j\}_{j=1}^M$ of whether or not prediction j is a true positive detection

Let $\mathcal{U} = \mathcal{B}$ be the set of unmatched objects;

Order $\{(b_j, s_j)\}_{j=1}^M$ in descending order of s_j ;

for $j=1 \dots M$ **do**

 Let $\mathcal{C} = \{B_k \in \mathcal{U} : \text{IOU}(B_k, b_j) \geq \text{thr}(B_k)\}$;

if $\mathcal{C} \neq \emptyset$ **then**

 Let $k^* = \arg \max_{\{k : B_k \in \mathcal{C}\}} \text{IOU}(B_k, b_j)$;

 Set $\mathcal{U} = \mathcal{U} \setminus B_{k^*}$;

 Set $z_j = 1$ since true positive detection;

else

 Set $z_j = 0$ since false positive detection;

end

end

Algorithm 1: The algorithm for greedily matching object detection outputs to ground truth labels. The standard $\text{thr}(B_k) = 0.5$ [48]. ILSVRC computes $\text{thr}(B_k)$ using Eq. 5.5 to better handle low-resolution objects.

In Algorithm 1, a predicted bounding box b is considered to have properly localized by a ground truth bounding box B if $\text{IOU}(b, B) \geq \text{thr}(B)$. The PASCAL VOC metric uses the threshold $\text{thr}(B) = 0.5$. However, for small objects even deviations of a few pixels would be unacceptable according to this threshold. For example, consider an object B of size 10×10 pixels, with a detection window of 20×20 pixels which fully contains that object. This would be an error of approximately 5 pixels on each dimension, which is average human annotation error. However, the IOU in this case would be $100/400 = 0.25$, far below the threshold of 0.5. Thus for smaller objects we loosen the threshold in ILSVRC to allow for the annotation to extend up to 5 pixels on average in each direction around the object. Concretely, if the ground truth box B is of dimensions $w \times h$ then

$$\text{thr}(B) = \min \left(0.5, \frac{wh}{(w+10)(h+10)} \right) \quad (5.5)$$

In practice, this changes the threshold only on objects which are smaller than approximately 25×25 pixels, and affects 5.5% of objects in the detection validation set.

Practical consideration. One additional practical consideration for ILSVRC detection evaluation is subtle and comes directly as a result of the scale of ILSVRC. In PASCAL, algorithms would often return many detections per class on the test set, including ones with low confidence scores. This allowed the algorithms to reach the level of high recall at least in the realm of very low precision. On ILSVRC detection test set if an algorithm returns 10 bounding boxes per object per image this would result in $10 \times 200 \times 40K = 80M$ detections. Each detection contains an image index, a class index, 4 bounding box coordinates, and the confidence score, so it takes on the order of 28 bytes.

The full set of detections would then require 2.24Gb to store and submit to the evaluation server, which is impractical. This means that algorithms are implicitly required to limit their predictions to only the most confident locations.

5.6 Methods

The ILSVRC dataset and the competition has allowed significant algorithmic advances in large-scale image recognition and retrieval.

5.6.1 Challenge entries

This section is organized chronologically, highlighting the particularly innovative and successful methods which participated in the ILSVRC each year. We see a turning point in 2012 with the development of large-scale convolutional neural networks.

ILSVRC2010. The first year the challenge consisted of just the classification task. The winning entry from NEC team [119] used SIFT [123] and LBP [2] features with two non-linear coding representations [227, 203] and a stochastic SVM. The honorable mention XRCE team [144] used an improved Fisher vector representation [143] along with PCA dimensionality reduction and data compression followed by a linear SVM. Fisher vector-based methods have evolved over five years of the challenge and continued performing strongly in every ILSVRC from 2010 to 2014.

ILSVRC2011. The winning classification entry in 2011 was the 2010 runner-up team XRCE, applying high-dimensional image signatures [144] with compression using product quantization [161] and one-vs-all linear SVMs. The single-object localization competition was held for the first time, with two brave entries. The winner was the UvA team using a selective search approach to generate class-independent object hypothesis regions [191], followed by dense sampling and vector quantization of several color SIFT features [188], pooling with spatial pyramid matching [112], and classifying with a histogram intersection kernel SVM [124] trained on a GPU [189].

ILSVRC2012. This was a turning point for large-scale object recognition, when large-scale deep neural networks entered the scene. The undisputed winner of both the classification and localization tasks in 2012 was the SuperVision team. They trained a large, deep convolutional neural network on RGB values, with 60 million parameters using an efficient GPU implementation and a novel hidden-unit dropout trick [104, 80]. The second place in image classification went to the ISI team, which used Fisher vectors [161] and a streamlined version of Graphical Gaussian Vectors [75], along with linear classifiers using Passive-Aggressive (PA) algorithm [29]. The second place in single-object localization went to the VGG, with an image classification system including dense SIFT features

and color statistics [123], a Fisher vector representation [161], and a linear SVM classifier, plus additional insights from [4, 162]. Both ISI and VGG used [54] for object localization; SuperVision used a regression model trained to predict bounding box locations. Despite the weaker detection model, SuperVision handily won the object localization task. A detailed analysis and comparison of the SuperVision and VGG submissions on the single-object localization task can be found in [151]. The influence of the success of the SuperVision model can be clearly seen in ILSVRC2013 and ILSVRC2014.

ILSVRC2013. There were 24 teams participating in the ILSVRC2013 competition, compared to 21 in the previous three years *combined*. Following the success of the deep learning-based method in 2012, the vast majority of entries in 2013 used deep convolutional neural networks in their submission. The winner of the classification task was Clarifai, with several large deep convolutional networks averaged together. The network architectures were chosen using the visualization technique of [219], and they were trained on the GPU following [220] using the dropout technique [104].

The winning single-object localization OverFeat submission was based on an integrated framework for using convolutional networks for classification, localization and detection with a multiscale sliding window approach [165]. They were the only team tackling all three tasks.

The winner of object detection task was UvA team, which utilized a new way of efficient encoding [190] densely sampled color descriptors [188] pooled using a multi-level spatial pyramid in a selective search framework [186]. The detection results were rescored using a full-image convolutional network classifier.

ILSVRC2014. 2014 attracted the most submissions, with 36 teams submitting 123 entries compared to just 24 teams in 2013 – a 1.5x increase in participation. As in 2013 almost all teams used convolutional neural networks as the basis for their submission. Significant progress has been made in just one year: image classification error was almost halved since ILSVRC2013 and object detection mean average precision almost doubled compared to ILSVRC2013. Please refer to Section 5.7.1 for details.

In 2014 teams were allowed to use outside data for training their models in the competition, so there were six tracks: provided and outside data tracks in each of image classification, single-object localization, and object detection tasks.

The winning image classification with provided data team was GoogLeNet, which explored an improved convolutional neural network architecture combining the multi-scale idea with intuitions gained from the Hebbian principle. Additional dimension reduction layers allowed them to increase both the depth and the width of the network significantly without incurring significant computational overhead. In the image classification with external data track, CASIAWS won by using weakly supervised object localization from only classification labels to improve image classification. MCG region proposals [6] pretrained on PASCAL VOC 2012 data are used to extract region proposals,

regions are represented using convolutional networks, and a multiple instance learning strategy is used to learn weakly supervised object detectors to represent the image.

In the single-object localization with provided data track, the winning team was VGG, which explored the effect of convolutional neural network depth on its accuracy by using three different architectures with up to 19 weight layers with rectified linear unit non-linearity, building off of the implementation of Caffe [92]. For localization they used per-class bounding box regression similar to OverFeat [165]. In the single-object localization with external data track, Adobe used 2000 additional ImageNet classes to train the classifiers in an integrated convolutional neural network framework for both classification and localization, with bounding box regression. At test time they used k-means to find bounding box clusters and rank the clusters according to the classification scores.

In the object detection with provided data track, the winning team NUS used the RCNN framework [66] with the network-in-network method [117] and improvements of [84]. Global context information was incorporated following [25]. In the object detection with external data track, the winning team was GoogLeNet (which also won image classification with provided data). It is truly remarkable that the same team was able to win at both image classification and object detection, indicating that their methods are able to not only classify the image based on scene information but also accurately localize multiple object instances. Just like most teams participating in this track, GoogLeNet used the image classification dataset as extra training data.

5.6.2 Large scale algorithmic innovations

ILSVRC over the past five years has paved the way for several breakthroughs in computer vision.

The field of categorical object recognition has dramatically evolved in the large-scale setting. Section 5.6.1 documents the progress, starting from coded SIFT features and evolving to large-scale convolutional neural networks dominating at all three tasks of image classification, single-object localization, and object detection. With the availability of so much training data (along with an efficient algorithmic implementation and GPU computing resources) it became possible to learn neural networks directly from the image data, without needing to create multi-stage hand-tuned pipelines of extracted features and discriminative classifiers. The major breakthrough came in 2012 with the win of the SuperVision team on image classification and single-object localization tasks [104], and by 2014 all of the top contestants were relying heavily on convolutional neural networks.

Further, over the past few years there has been a lot of focus on large-scale recognition in the computer vision community. Best paper awards at top vision conferences in 2013 were awarded to large-scale recognition methods: at CVPR 2013 to “Fast, Accurate Detection of 100,000 Object Classes on a Single Machine” [35] and at ICCV 2013 to “From Large Scale Image Categorization to Entry-Level Categories” [133]. Additionally, several influential lines of research have emerged, such

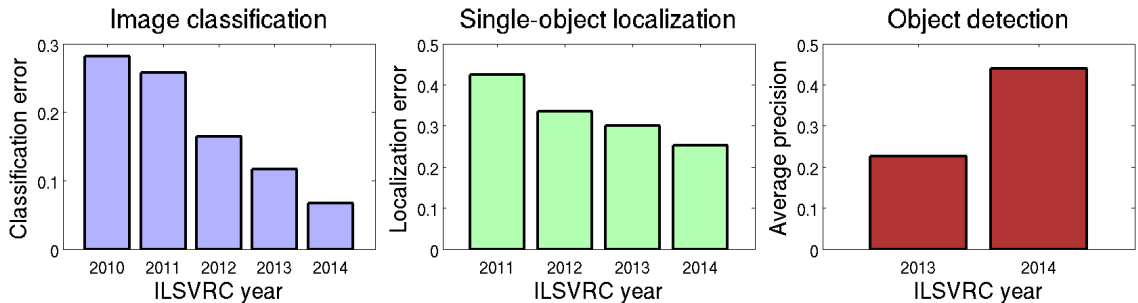


Figure 5.4: Performance of winning entries in the ILSVRC2010-2014 competitions in each of the three tasks (details about the entries and numerical results are in Section 5.6.1). There is a steady reduction of error every year in object classification and single-object localization tasks, and a 1.9x improvement in mean average precision in object detection. There are two considerations in making these comparisons. (1) The object categories used in ILSVRC changed between years 2010 and 2011, and between 2011 and 2012. However, the large scale of the data (1000 object categories, 1.2 million training images) has remained the same, making it possible to compare results. Image classification and single-object localization entries shown here use only provided training data. (2) The size of the object detection training data has increased significantly between years 2013 and 2014 (Section 5.4.2). Section 5.7.1 discusses the relative effects of training data increase versus algorithmic improvements.

as large-scale weakly supervised localization work of [105] which was awarded the best paper award in ECCV 2012 and large-scale zero-shot learning, e.g., [59].

5.7 Results and analysis

We summarize the trends over the years of ILSVRC. Chapter 7 goes into much more detailed per-class analysis of the performance of leading algorithms.

5.7.1 Improvements over the years

State-of-the-art accuracy has improved significantly from ILSVRC2010 to ILSVRC2014, showcasing the massive progress that has been made in large-scale object recognition over the past five years. The performance of the winning ILSVRC entries for each task and each year are shown in Figure 5.4. The improvement over the years is clearly visible. In this section we quantify and analyze this improvement.

Image classification and single-object localization. There has been a 4.2x reduction in image classification error (from 28.2% to 6.7%) and a 1.7x reduction in single-object localization error (from 42.5% to 25.3%) since the beginning of the challenge.⁶ Even though the exact object categories have

⁶For consistency, here we consider only teams that use the provided training data.

changed [152], the scale of the dataset has remained the same (Table 5.2), making the results comparable across the years. The dataset has not changed since 2012, and there has been a 2.4x reduction in image classification error (from 16.4% to 6.7%) and a 1.3x in single-object localization error (from 33.5% to 25.3%) in the past three years.

Object detection. Object detection accuracy as measured by the mean average precision (mAP) has increased 1.9x since the introduction of this task, from 22.6% mAP in ILSVRC2013 to 43.9% mAP in ILSVRC2014. However, these results are not directly comparable for two reasons. First, the size of the object detection training data has increased significantly from 2013 to 2014 (Table 5.3). Second, the 43.9% mAP result was obtained with extra training data (from the image classification and single-object localization tasks). Here we attempt to understand the relative effects of the training set size increase versus algorithmic improvements. All models are evaluated on the same ILSVRC2013-2014 object detection test set.

First, we quantify the effects of increasing detection training data between the two challenges by comparing the same model trained on ILSVRC2013 detection data versus ILSVRC2014 detection data. The UvA team’s framework [190] from 2013 achieved 22.6% with ILSVRC2013 data and 26.3% with ILSVRC2014 data and no other modifications.⁷ The absolute increase in mAP was 3.7%. The RCNN model achieved 31.4% mAP with ILSVRC2013 detection plus image classification data [66] and 34.5% mAP with ILSVRC2014 detection plus image classification data (Berkeley team [152]). The absolute increase in mAP by expanding ILSVRC2013 detection data to ILSVRC2014 was 3.1%.

Second, we quantify the effects of adding in the external data for training object detection models. The NEC model in 2013 achieved 19.6% mAP trained on ILSVRC2013 detection data alone and 20.9% mAP trained on ILSVRC2013 detection plus classification data [205, 152]. The absolute increase in mAP was 1.3%. The UvA team’s best entry in 2014 achieved 32.0% mAP trained on ILSVRC2014 detection data and 35.4% mAP trained on ILSVRC2014 detection plus classification data [190, 152]. The absolute increase in mAP was 3.4%.

Thus, we conclude based on the evidence so far that expanding the ILSVRC2013 detection set to the ILSVRC2014 set, as well as adding in additional training data from the classification task, all account for approximately 1 – 4% in absolute mAP improvement for the models. For comparison, we can also attempt to quantify the effect of algorithmic innovation. The UvA team’s 2013 framework achieved 26.3% mAP on ILSVRC2014 data as mentioned above, and their improved method in 2014 obtained 32.0% mAP [190, 152]. This is 5.8% absolute increase in mAP over just one year from algorithmic innovation alone.

In summary, we conclude that the absolute 21.3% increase in mAP between winning entries of ILSVRC2013 (22.6% mAP) and of ILSVRC2014 (43.9% mAP) is the result of impressive algorithmic innovation and not just a consequence of increased training data. However, increasing the

⁷Personal communication with members of the UvA team.

ISLVR2014 object detection training dataset further *is* likely to produce additional improvements in detection accuracy for current algorithms.

5.7.2 Statistical significance

One important question to ask is whether results of different submissions to ILSVRC are statistically significantly different from each other. Given the large scale, it is no surprise that even minor differences in accuracy are statistically significant; we seek to quantify exactly how much of a difference is enough.

Following the strategy employed by PASCAL VOC [46], for each method we obtain a confidence interval of its score using bootstrap sampling. During each bootstrap round, we sample N images with replacement from all the available N test images and evaluate the performance of the algorithm on those sampled images. This can be done very efficiently by precomputing the accuracy on each image. Given the results of all the bootstrapping rounds we discard the lower and the upper α fraction. The range of the remaining results represents the $1 - 2\alpha$ confidence interval. We run a large number of bootstrapping rounds (from 20,000 until convergence). Table 5.4 shows the results of the top entries to each task of ILSVRC2012-2014. The winning methods are statistically significantly better than the runner-up methods, even at the 99.9% level.

5.8 Conclusions

In this paper we described the large-scale data collection process of ILSVRC, provided a summary of the most successful algorithms on this data, and analyzed the success and failure modes of these algorithms. In this section we discuss some of the key lessons we learned over the years of ILSVRC, strive to address the key criticisms of the datasets and the challenges we encountered over the years, and conclude by looking forward into the future.

5.8.1 Lessons learned

The key lesson of collecting the datasets and running the challenges for five years is this: **All human intelligence tasks need to be exceptionally well-designed.** We learned this lesson both when annotating the dataset using Amazon Mechanical Turk workers (Section 5.4) and even when trying to evaluate human-level image classification accuracy using expert labelers [152]. The first iteration of the labeling interface was always bad – generally meaning *completely unusable*. If there was any inherent ambiguity in the questions posed (and there almost always was), workers found it and accuracy suffered. If there is one piece of advice we can offer to future research, it is to very carefully design, continuously monitor, and extensively sanity-check all crowdsourcing tasks.

The other lesson, already well-known to large-scale researchers, is this: **Scaling up the dataset**

Image classification			
Year	Codename	Error (percent)	99.9% Conf Int
2014	GoogLeNet [177]	6.66	6.40 - 6.92
2014	VGG [170]	7.32	7.05 - 7.60
2014	MSRA [77]	8.06	7.78 - 8.34
2014	AHoward [84]	8.11	7.83 - 8.39
2014	DeeperVision	9.51	9.21 - 9.82
2013	Clarifai [†] [219, 220]	11.20	10.87 - 11.53
2013	Clarifai [219, 220]	11.74	11.41 - 12.08
2013	NUS [104]	12.95	12.60 - 13.30
2013	ZF [219, 220]	13.51	13.14 - 13.87
2013	AHoward	13.55	13.20 - 13.91
2012	SuperVision [†] [104]	15.32	14.94 - 15.69
2012	SuperVision [104]	16.42	16.04 - 16.80
2012	ISI [75]	26.17	25.71 - 26.65
2012	VGG [4, 162]	26.98	26.53 - 27.43
2012	XRCE [142]	27.06	26.60 - 27.52
Single-object localization			
Year	Codename	Error (percent)	99.9% Conf Int
2014	VGG [170]	25.32	24.87 - 25.78
2014	GoogLeNet [177]	26.44	25.98 - 26.92
2013	OverFeat [165]	29.88	29.38 - 30.35
2014	Adobe [†]	30.10	29.61 - 30.58
2014	SYSU	31.90	31.40 - 32.40
2012	SuperVision [†] [104]	33.55	33.05 - 34.04
2014	MIL [96, 66]	33.74	33.24 - 34.25
2012	SuperVision [104]	34.19	33.67 - 34.69
2013	VGG [169]	46.42	45.90 - 46.95
2012	VGG [4, 162]	50.03	49.50 - 50.57
2012	ISI [75]	53.65	53.10 - 54.17
Object detection			
Year	Codename	AP (percent)	99.9% Conf Int
2014	GoogLeNet [†] [177]	43.93	42.92 - 45.65
2014	CUHK [†] [134, 135]	40.67	39.68 - 42.30
2014	DeepInsight [†]	40.45	39.49 - 42.06
2014	NUS [117, 25]	37.21	36.29 - 38.80
2014	UvA [†] [190]	35.42	34.63 - 36.92
2013	UvA [190]	22.58	22.00 - 23.82
2013	NEC [†] [205]	20.90	20.40 - 22.15
2013	NEC [205]	19.62	19.14 - 20.85
2013	OverFeat [†] [165]	19.40	18.82 - 20.61
2013	Toronto	11.46	10.98 - 12.34

Table 5.4: We use bootstrapping to construct 99.9% confidence intervals around the result of up to top 5 submissions to each ILSVRC task in 2012-2014. [†] means the entry used external training data. The winners using the provided data for each track and each year are bolded. The difference between the winning method and the runner-up each year is significant even at the 99.9% level.

always reveals unexpected challenges. From designing complicated multi-step annotation strategies (described later in Chapter 6) to having to modify the evaluation procedure (Section 5.5), we had to continuously adjust to the large-scale setting. On the plus side, of course, the major breakthroughs in object recognition accuracy (Section 5.6) and the analysis of the strength and weaknesses of current algorithms as a function of object class properties (described later in Chapter 7) would never have been possible on a smaller scale.

5.8.2 Criticism

In the past five years, we encountered three major criticisms of the ILSVRC dataset and the corresponding challenge: (1) the ILSVRC dataset is insufficiently challenging, (2) the ILSVRC dataset contains annotation errors, and (3) the rules of ILSVRC competition are too restrictive. We discuss these in order.

The first criticism is that the objects in the dataset tend to be large and centered in the images, making the dataset insufficiently challenging. In Sections 5.4.1 and 5.4.2 we tried to put those concerns to rest by analyzing the statistics of the ILSVRC dataset and concluding that it is comparable with, and in many cases much more challenging than, the long-standing PASCAL VOC benchmark [48].

The second is regarding the errors in ground truth labeling. We went through several rounds of in-house post-processing of the annotations obtained using crowdsourcing, and corrected many common sources of errors. The major remaining source of annotation errors stem from fine-grained object classes, e.g., labelers failing to distinguish different species of birds. This is a tradeoff that had to be made: in order to annotate data at this scale on a reasonable budget, we had to rely on non-expert crowd labelers. However, overall the dataset is encouragingly clean. By our estimates, 99.7% precision is achieved in the image classification dataset [152] and 97.9% of images that went through the bounding box annotation system have all instances of the target object class labeled with bounding boxes [175, 152].

The third criticism we encountered is over the rules of the competition regarding using external training data. In ILSVRC2010-2013, algorithms had to only use the provided training and validation set images and annotations for training their models. With the growth of the field of large-scale unsupervised feature learning, however, questions began to arise about what exactly constitutes “outside” data: for example, are image features trained on a large pool of “outside” images in an unsupervised fashion allowed in the competition? After much discussion, in ILSVRC2014 we took the first step towards addressing this problem. We followed the PASCAL VOC strategy and created two tracks in the competition: entries using only “provided” data and entries using “outside” data, meaning *any* images or annotations not provided as part of ILSVRC training or validation sets. However, in the future this strategy will likely need to be further revised as the computer vision field evolves. For example, competitions can consider allowing the use of any image features which

are publically available, even if these features were learned on an external source of data.

5.8.3 The future

Given the massive algorithmic breakthroughs over the past five years, we are very eager to see what will happen in the next five years. There are many potential directions of improvement and growth for ILSVRC and other large-scale image datasets.

First, continuing the trend of moving towards richer image understanding (from image classification to single-object localization to object detection), the next challenge would be to tackle pixel-level object segmentation. The recently released large-scale COCO dataset [118] is already taking a step in that direction.

Second, as datasets grow even larger in scale, it may become impossible to fully annotate them manually. The scale of ILSVRC is already imposing limits on the manual annotations that are feasible to obtain: for example, we had to restrict the number of objects labeled per image in the image classification and single-object localization datasets. In the future, with billions of images, it will become impossible to obtain even one clean label for every image. Datasets such as Yahoo's Flickr Creative Commons 100M,⁸ released with weak human tags but no centralized annotation, will become more common.

The growth of unlabeled or only partially labeled large-scale datasets implies two things. First, algorithms will have to rely more on weakly supervised training data. Second, even evaluation might have to be done *after* the algorithms make predictions, not before. This means that rather than evaluating *accuracy* (how many of the test images or objects did the algorithm get right) or *recall* (how many of the desired images or objects did the algorithm manage to find), both of which require a fully annotated test set, we will be focusing more on *precision*: of the predictions that the algorithm made, how many were deemed correct by humans.

We are eagerly awaiting the future development of object recognition datasets and algorithms, and are grateful that ILSVRC served as a stepping stone along this path.

⁸<http://webscope.sandbox.yahoo.com/catalog.php?datatype=i&did=67>

Chapter 6

Large-scale object detection dataset construction

6.1 Introduction

Chapter 5 introduced the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) and discussed the tasks, the evaluation criteria and the leading algorithms. Here we focus specifically on the object detection task, which evaluates the ability of an algorithm to name and localize *all* instances of *all* target objects present in an image. It is much more challenging than the single-object localization task because some object instances may be small/occluded/difficult to accurately localize, and the algorithm is expected to locate them all, not just the one it finds easiest.

A critical component for scaling up object detection is collecting a large-scale detection dataset. The PASCAL VOC dataset [48, 46] provides a standardized test bed for object detection with 20 object classes and 21,738 images.¹ We set out to collect an object detection benchmark dataset for ILSVRC that is more than an order of magnitude greater in size.

There are three key challenges in collecting the object detection dataset. The first challenge is selecting the set of common objects which tend to appear in cluttered photographs and are well-suited for benchmarking object detection performance. Our approach relies on statistics of the object localization dataset and the tradition of the PASCAL VOC challenge (Section 6.2).

The second challenge is obtaining a much more varied set of scene images than those used for the image classification and single-object localization datasets. Section 6.3 describes the procedure for utilizing as much data from the single-object localization dataset as possible and supplementing it with Flickr images queried using hundreds of manually designed high-level queries.

The third, and biggest, challenge is completely annotating this dataset with all the objects. This

¹Other related work is summarized in Chapter 5.

is done in two parts. Section 6.4 describes the first part: our hierarchical strategy for obtaining the list of all target objects which occur within every image. This is necessary since annotating in a straight-forward way by creating a task for every (image, object class) pair is no longer feasible at this scale. Section 6.5 describes the second part: annotating the bounding boxes around these objects, using the single-object localization bounding box annotation pipeline of Section 6.5.1 along with extra verification of Section 6.5.2 to ensure that *every* instance of the object is annotated with exactly *one* bounding box.

6.2 Defining object categories

The first step is defining the set of target object categories. To do this, we select from among the existing ImageNet [36] categories. By using WordNet as a backbone [128], ImageNet already takes care of disambiguating word meanings and of combining together synonyms into the same object category. Since the selection of object categories needs to be done only once per challenge task, we use a combination of automatic heuristics and manual post-processing to create the list of target categories appropriate for each task.

There are 200 object classes hand-selected for the detection task. These were chosen to be mostly basic-level object categories that would be easy for people to identify and label. The rationale is that the object detection system developed for this task can later be combined with a fine-grained classification model to further classify the objects if a finer subdivision is desired.² The synsets are selected such that there is no overlap as to avoid confusion: for any synsets i and j , i is not an ancestor of j in the ImageNet hierarchy.

The selection of the 200 object detection classes in 2013 was guided by the ILSVRC 2012 classification and localization dataset. Starting with 1000 object classes and their bounding box annotations we first eliminated all object classes which tended to be too “big” in the image (on average the object area was greater than 50% of the image area). These were classes such as T-shirt, spiderweb, or manhole cover. We then manually eliminated all classes which we did not feel were well-suited for detection, such as hay, barbershop, or poncho. This left 494 object classes which were merged into basic-level categories: for example, different species of birds were merged into just the “bird” class. The classes remained the same in ILSVRC2014. Appendix D contains the complete list of object categories used in ILSVRC2013-2014 (in the context of the hierarchy described in Section 6.4).

Staying mindful of the tradition of the PASCAL VOC dataset we also tried to ensure that the set of 200 classes contains as many of the 20 PASCAL VOC classes as possible. Table 6.1 shows the correspondences. The few changes were done ensure more accurate and consistent crowdsourced annotations. The object class with the weakest correspondence is “potted plant” in PASCAL VOC, corresponding to “flower pot” in ILSVRC. “Potted plant” was one of the most challenging object

²Some of the training objects are actually annotated with more detailed classes: for example, one of the 200 object classes is the category “dog,” and some training instances are annotated with the specific dog breed.

Class name in PASCAL VOC (20 classes)	Closest class in ILSVRC-DET (200 classes)	Average object scale (%)	
		PASCAL VOC	ILSVRC-DET
aeroplane	airplane	29.7	22.4
bicycle	bicycle	29.3	14.3
bird	bird	15.9	20.1
<i>boat</i>	<i>watercraft</i>	15.2	16.5
<i>bottle</i>	<i>wine bottle</i>	7.3	10.4
bus	bus	29.9	22.1
car	car	14.0	13.4
cat	domestic cat	46.8	29.8
chair	chair	12.8	10.1
<i>cow</i>	<i>cattle</i>	19.3	13.5
<i>dining table</i>	<i>table</i>	29.1	30.3
dog	dog	37.0	28.9
horse	horse	29.5	18.5
motorbike	motorcycle	32.0	20.7
person	person	17.5	19.3
<i>potted plant</i>	<i>flower pot</i>	12.3	8.1
sheep	sheep	12.2	17.3
sofa	sofa	41.7	44.4
train	train	35.4	35.1
tv/monitor	tv or monitor	14.6	11.2

Table 6.1: Correspondences between the object classes in the PASCAL VOC [48] and the ILSVRC detection task. Object scale is the fraction of image area (reported in percent) occupied by an object instance. It is computed on the validation sets of PASCAL VOC 2012 and of ILSVRC-DET. The average object scale is 24.1% across the 20 PASCAL VOC categories and 20.3% across the 20 corresponding ILSVRC-DET categories.

classes to annotate consistently among the PASCAL VOC classes, and in order to obtain accurate annotations using crowdsourcing we had to restrict the definition to a more concrete object.

6.3 Collecting scene images

The second step is collecting a diverse set of candidate images to represent the selected categories. We use both automatic and manual strategies on multiple search engines to do the image collection. Many images for the detection task were collected differently than the images in ImageNet [36] and the classification and single-object localization tasks [152]. We focus our efforts specifically on collecting scene-like images.

Figure 6.1 summarizes the types of images that were collected. Ideally all of these images would be scene images fully annotated with all target categories. However, given budget constraints our goal was to provide as much suitable detection data as possible, even if the images were drawn from a few different sources and distributions.

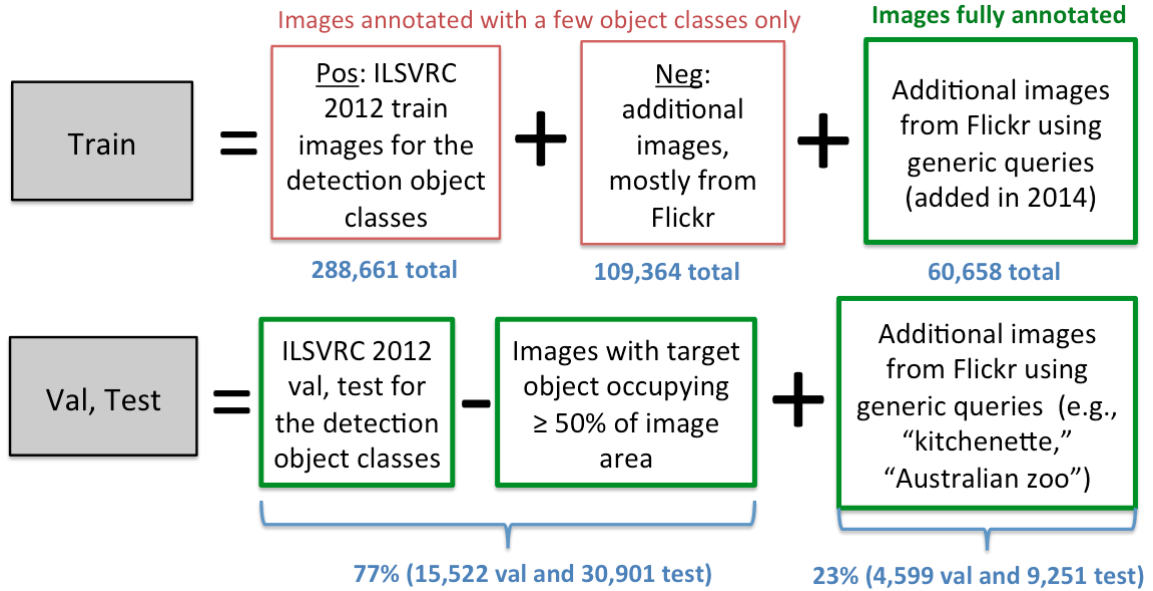


Figure 6.1: Summary of images collected for the detection task. Images in green (bold) boxes have all instances of all 200 detection object classes fully annotated.

Validation and test. The validation and test detection set images come from two sources (percent of images from each source in parentheses). The first source (77%) is images from ILSVRC2012 single-object localization validation and test sets corresponding to the 200 detection classes (or their children in the ImageNet hierarchy). Images where the target object occupied more than 50% of the image area were discarded, since they were unlikely to contain other objects of interest. The second source (23%) is images from Flickr collected specifically for detection task. We queried Flickr using a large set of manually defined queries, such as “kitchenette” or “Australian zoo” to retrieve images of scenes likely to contain several objects of interest. Appendix C contains the full list. We also added pairwise queries, or queries with two target object names such as “tiger lion,” which also often returned cluttered scenes.

Figure 6.2 shows a random set of both types of validation images. Images were randomly split, with 33% going into the validation set and 67% into the test set.³ In total there are 20,121 validation and 40,152 test images.

Training. The training set for the detection task comes from three sources of images (percent of images from each source in parentheses). The first source (63%) is all training images from ILSVRC2012 single-object localization task corresponding to the 200 detection classes (or their children in the ImageNet hierarchy). We did not filter by object size, allowing teams to take advantage

³The validation/test split is consistent with ILSVRC2012: validation images of ILSVRC2012 remained in the validation set of ILSVRC2013, and ILSVRC2012 test images remained in ILSVRC2013 test set.

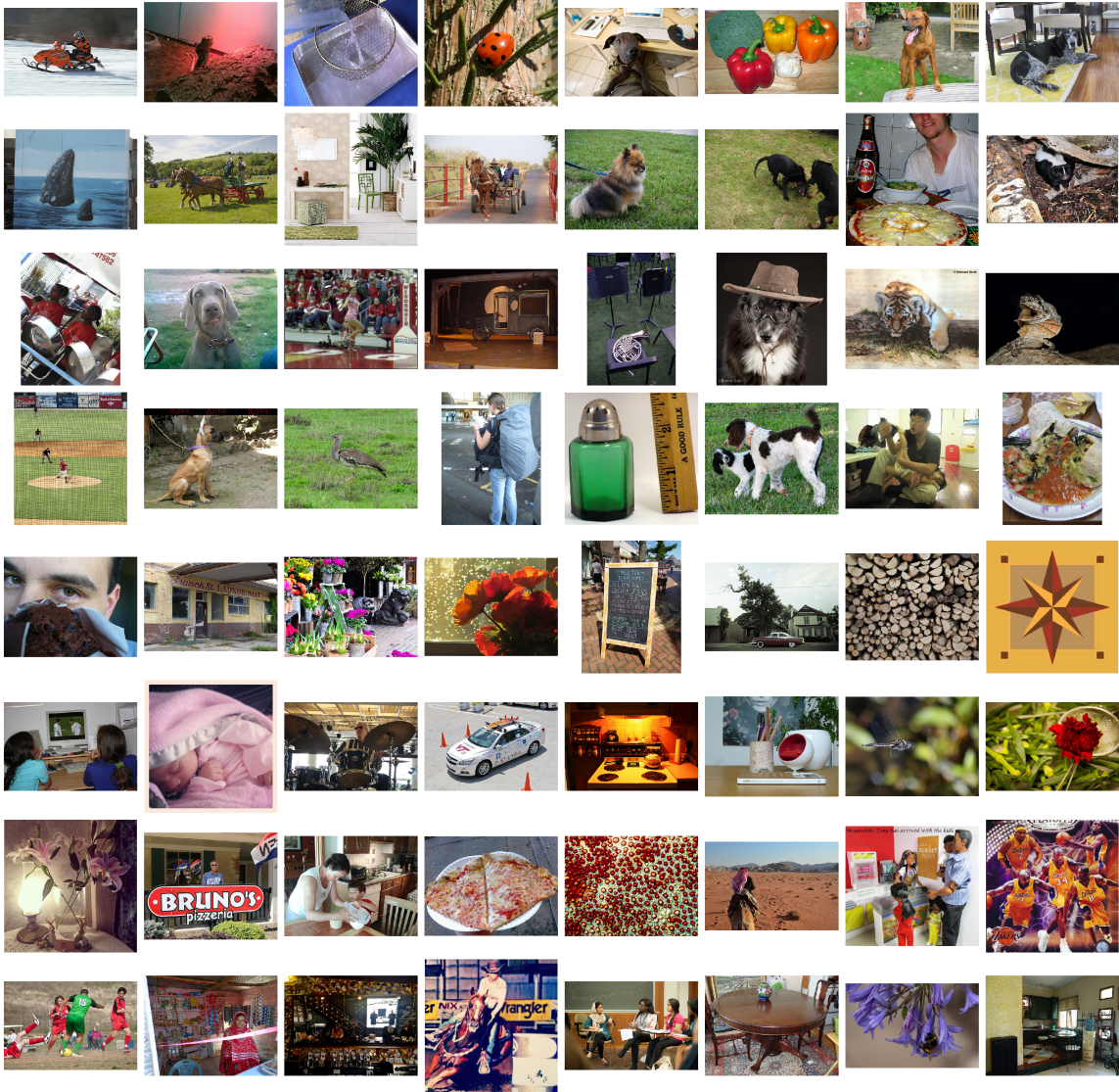


Figure 6.2: Random selection of images in ILSVRC detection validation set. The images in the top 4 rows were taken from ILSVRC2012 single-object localization validation set, and the images in the bottom 4 rows were collected from Flickr using scene-level queries.

of all the positive examples available. The second source (24%) is negative images which were part of the original ImageNet collection process but voted as negative: for example, some of the images were collected from Flickr and search engines for the ImageNet synset “animals” but during the manual verification step did not collect enough votes to be considered as containing an “animal.” These images were manually re-verified for the detection task to ensure that they did not in fact contain the target objects. The third source (13%) is images collected from Flickr specifically for the detection task. These images were added for ILSVRC2014 following the same protocol as the second type of images in the validation and test set. This was done to bring the training and testing distributions closer together.

In total there are 288,661 training images from the ILSVRC2012 single-object localization task (between 417 and 66,991 per class), 109,364 additional annotated negative training images (185-10,073 per class) and 60,658 Flickr scene images added in ILSVRC2014.

6.4 Scalable multi-label annotation

The third (and most challenging) step of constructing the object detection dataset is annotating the collected images. For building the object detection dataset, we tackle this step in two parts. First, we annotate all images with the presence or absence of target categories (i.e., we determine which objects are contained in each image). This requires developing a novel hierarchical labeling system which we describe in this section. Section 6.5 then describes the second part of annotating the object location.

The key challenge in annotating images for the object detection task is that all objects in all images need to be labeled. Suppose there are N inputs (images) which need to be annotated with the presence or absence of K labels (objects). A naïve approach would query humans for each combination of input and label, requiring NK queries. However, N and K can be very large and the cost of this exhaustive approach quickly becomes prohibitive. For example, annotating 60,000 validation and test ILSVRC detection images with the presence or absence of 200 object classes for the detection task naïvely would take 80 times more effort than annotating 150,000 validation and test images with 1 object each for the classification task – and this is not even counting the additional cost of collecting bounding box annotations around each object instance. This quickly becomes infeasible.

In this section we study general strategies for scaling up multi-label annotation, i.e. obtaining labels with a cost substantially smaller than that of the exhaustive naïve approach. This technique is important in multiple domains, such as labeling actions in videos [110], news article topics [163], functional classes of genes [45], musical attributes or emotions in songs [116], semantic classes of scenes [16], product categories customers are likely to buy [223], and categories of web pages [185]. While the problem of acquiring one label has been well studied [208, 89, 226, 207, 167, 32], to our

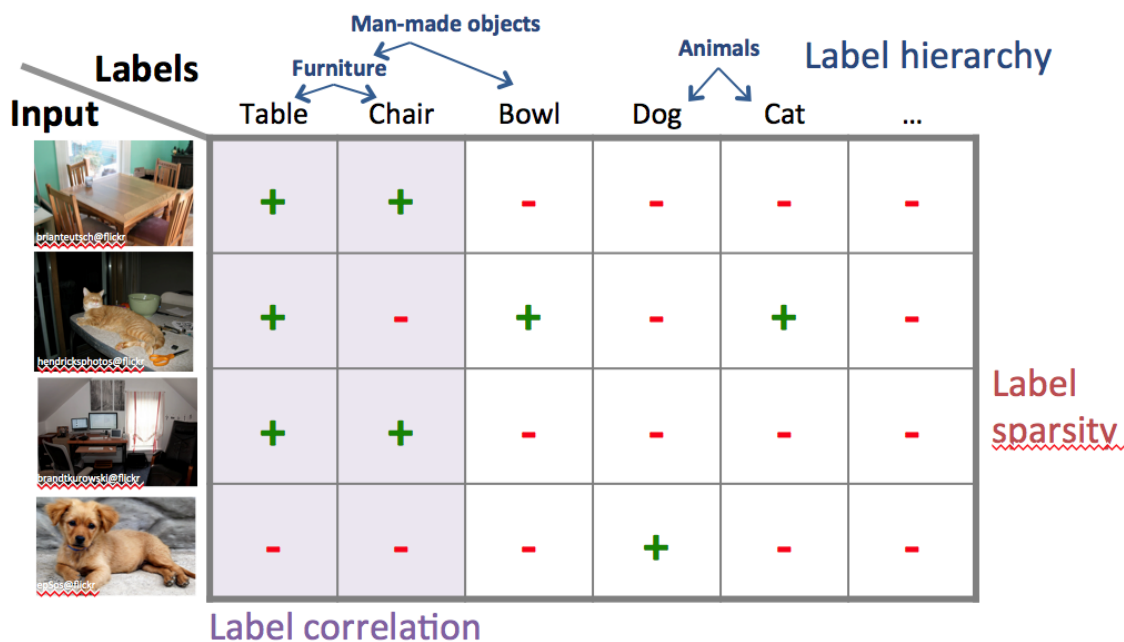


Figure 6.3: Multi-label annotation becomes much more efficient when considering real-world structure of data: correlation between labels, hierarchical organization of concepts, and sparsity of labels.

knowledge the challenge of large-scale multi-label annotation has not been addressed before.

6.4.1 Overview

We exploit three key observations for labels in real world applications (illustrated in Figure 6.3).

1. **Correlation.** Subsets of labels are often highly correlated. Objects such as a computer keyboard, mouse and monitor frequently co-occur with each other in images. Topics such as economy and finance often co-occur in news articles. Similarly, some labels tend to all be absent at the same time. For example, all objects that require electricity are usually absent in pictures taken outdoors. This suggests that we could potentially “fill in” the values of multiple labels by grouping them into only one query for humans. Instead of checking if dog, cat, rabbit etc. are present in the photo, we check them as a group animal. If the answer is no, then this implies a no for all categories in the group.
2. **Hierarchy.** The above example of grouping dog, cat, rabbit etc. into animal has implicitly assumed that labels can be grouped together and humans can efficiently answer queries about the group as a whole. This brings up our second key observation: humans organize semantic concepts into hierarchies and are able to efficiently categorize at higher semantic levels [180],

e.g. humans can determine the presence of an animal in an image as fast as every type of animal individually. This leads to substantial cost savings.

3. **Sparsity.** The values of labels for each item tend to sparse, i.e. an image is unlikely to contain more than a dozen types of objects, a small fraction of the tens of thousands of object categories. This enables a rapid elimination of many objects, filling no for many labels very quickly. With a high degree of sparsity, an efficient algorithm can have a cost which grows logarithmically with the number of objects instead of linearly.

We propose algorithmic strategies that exploit the above intuitions. The key is to select a sequence of queries for humans such that we achieve the same labeling results with only a fraction of the cost of the naïve approach. The main challenges include how to measure cost and utility, how to construct good queries, and how to order them. We present a theoretical analysis and a practical algorithm. We then demonstrate how this algorithm was used in practice to obtain the ILSVRC object detection annotations.

6.4.2 Related Work

Acquiring labels as a crowdsourcing task has been extensively studied. The key challenge is making efficient use of resources to achieve quality results. A growing body of work has studied how to estimate worker quality [89], how to combine results from multiple noisy annotators [208, 226, 207], how to model the trade-off between quality and cost [32], how to merge machine and human intelligence [94], as well as how to select the next best item to label [167]. However, they only focus on the single-label case. Multi-label annotation has been practiced in many crowd-powered systems. For example, PlateMate [130] tags all foods in each photo for nutrition estimation. VizWiz [12] labels the presence of objects in images to help blind users. These systems, however, do not address the scalability issue of a large number of labels.

Our framework of optimizing the sequence of queries to quickly fill in values relates to general strategies using iterative steps [121] to limit the search space. Bernstein et al. uses “rapid refinement” to narrow down the search space with multiple synchronized users [11]. Branson et al. investigate how to select questions posed to human users to perform multi-class image classification [19]; this is special case of our setting where only one class can be present in an image.

6.4.3 Approach

We first describe a meta algorithm for multi-label annotation, and then customize to make it more efficient. For clarity of exposition and without any loss of generality we use the task of labeling images with the presence of objects as a running example. Here each label represents the presence or absence of an object and takes a value of yes or no. We assume that all labels are binary since any multi-valued label can be represented as a set of mutually exclusive binary labels.

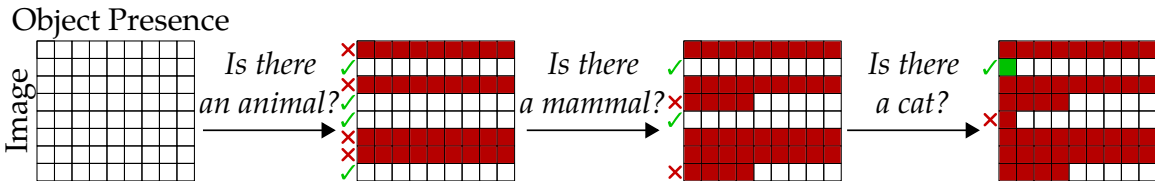


Figure 6.4: Our algorithm dynamically selects the next query to efficiently determine the presence or absence of every object in every image. Green denotes a positive annotation and red denotes a negative annotation. This toy example illustrates a sample progression of the algorithm for one label (cat) on a set of images.

Algorithm. Our meta algorithm (Algorithm 2) poses a sequence of queries to humans. Each query allows us to fill in values for some labels. We stop when all values are filled. A few sample iterations of the algorithm are shown in Figure 6.4.

Input: An item to be labeled
Output: K labels, each label $+1$ or -1
 Set values of all K labels to 0 (i.e. *missing*);
while any values are 0 **do**
 Select a query Q from possible queries \mathcal{Q} ;
 Obtain an answer A to query Q from humans;
 Set values of some labels to $+1$ or -1 given answer A ;
end

Algorithm 2: The meta algorithm for multi-label annotation

In the naïve instantiation of this meta algorithm, we issue one query for each label (i.e. is there a dog in the image). This is clearly not scalable as the cost is $O(NK)$ for N items and K labels. The key to scalability is using additional queries that may fill in multiple values (e.g. if there is no animal with four legs, we know there is no dog and no cat and no rabbit in the image). Moreover, we can exploit the fact that the meta algorithm allows *dynamic* selection of the next query based on the current available information.

A good query should fill in as many values as possible and is easy for humans to answer. In other words, we would like to pick a question with the most *utility* in filling in the values per unit of *cost*. We now make the two notions precise.

Utility. We measure the utility of a query as the expected number of new values filled in over a distribution of items to be labeled. Consider an image with k missing labels. Let $y \in \{-1, 0, +1\}^k$ represent the values of those k labels after using query Q , where -1 means “no,” 0 means “unknown” and 1 means “yes.” Thus the l_1 norm $\|y\|_1$ is the number of newly acquired labels. The utility of Q is $U(Q) = \mathbf{E}\|y\|_1$.

In practice the utility can be estimated using a “training” set, i.e. an i.i.d. sample of items with ground truth annotations.⁴ Suppose we have a set of n training images labeled with the presence or

absence of cats, dogs, and other objects of interest. The utility of the query “is there a cat present” would be 1, since on every image we would gain one new label. On the other hand, consider the utility of the query “is there an animal present.” Suppose there are n^- images which do not have an animal, and suppose there are s subcategories of animal in our set of labels. Then on each of the n^- negative images we gain s new “no” labels since we now know all subcategories of animals must be absent. On the other images we don’t gain anything since we do not know which animal is present. Thus, the estimated utility is $\hat{U}(Q) = sn^-/n$.

Seeking high utility queries exploits the correlation and sparsity of a large label set. High correlation of labels implies high utility of certain queries. For example, when annotating a diverse set of internet images for the presences of couches, desks, sofas, and chairs, designing queries with good utility (e.g., is furniture present?) is easy because the labels are correlated: most internet images that do not have couches also will not have desks. High sparsity means potentially more high utility queries because for most inputs most queries will have a no answer (e.g., most images will not have most of the objects being annotated).

Cost. We measure the cost $C(Q)$ of a query Q as the expected human time it takes to obtain a reliable answer for one item. First, we can empirically measure the average amount of time a human takes to answer a query on a small training set. Next, we might need to consult multiple humans to be confident in the answer. Here we take the majority voting approach and assume a Bernoulli process for querying multiple workers. Again on a small training set we can estimate that the average worker gives a correct answer with probability $p > 0.5$. Then the accuracy of a majority of $2n + 1$ votes is [167]: $\hat{P}_{2n+1} = \sum_{i=n+1}^{2n+1} \binom{2n+1}{i} p^i (1-p)^{2n+1-i}$. Given an acceptable accuracy threshold $1 - \epsilon$, we can find the number of votes needed to reach the threshold, which allows us to calculate $C(Q)$ as a product of the number of workers needed and the average time a worker takes to give an answer.

To be more scalable than the naive method, it is crucial to find high-utility queries that are also low cost. This is where the hierarchical structure of the label space helps. Hierarchy means many high-level or attribute-like queries (e.g. “is it red?”) have low time cost because they are not arbitrary groupings but useful shortcuts in human cognition, i.e. humans can answer is there an animal as fast as is there a dog.

Selection. In Algorithm 2, the query is selected by maximizing utility per unit cost, i.e., $Q^* = \arg \max_Q \hat{U}(Q)/C(Q)$.

⁴We could in principle estimate the utility conditioned on values of existing labels. This is beyond the scope of this paper.

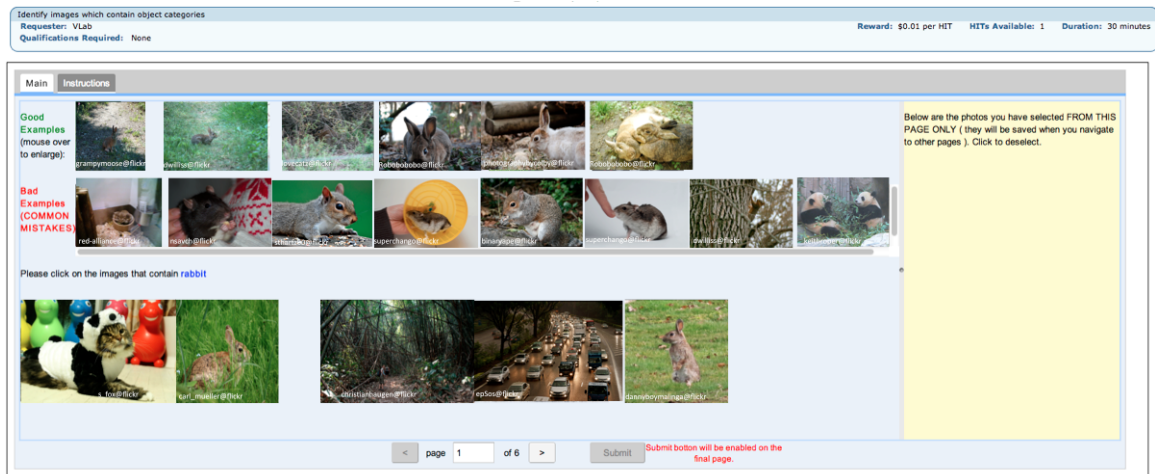


Figure 6.5: The Amazon Mechanical Turk interface for obtaining human annotations. Here workers are asked to select images which contain a rabbit, and are shown good and bad example images.

6.4.4 Experiments

Task and Implementation. We apply this algorithm to the task of labeling images with the presence or absence of many object categories. Here we evaluate the labeling on a subset of 20,000 images from the ILSVRC object detection validation set (Section 6.3). We annotate them with 200 object classes (Section 6.2). We manually create a hierarchy of these objects which contains 56 internal queries, using high-level categories such as “animals with hooves,” “electronics that play sound” or “liquid containers.” The full hierarchy is in Appendix D.

We created a user interface shown in Figure 6.5 for efficient binary labeling of images. A user is given an object category (either one of the target categories or a high-level category) along with positive and negative example images, and is then asked to click on all images from a large candidate pool which contain an instance of this category. We used this interface to query humans using Amazon Mechanical Turk.

We used an early pilot of this algorithm to obtain ground truth annotations on this data, with stringent quality control but potentially suboptimal cost. This allows us to evaluate our algorithm in a controlled setting through simulation. We estimate key simulation parameters (worker confusion matrix, worker response time per image for each query) through real AMT experiments with a sample of 100 images per category, each image labeled 3 workers. Query utility is estimated by the algorithm on the fly using the training set (we use a 10%-90% training-test split). In simulation we enforce a minimum worker accuracy of 75% after filtering of spammers.

Query construction. Before our algorithm can automatically perform query selection, we need to provide a pool of candidate queries. We can leverage general knowledge bases such as WordNet [53]),

Query: Is there a... ⁵	Utility (num labels)	Cost (secs)
mammals that have claws or fingers	12.0	3.0
living organisms	24.8	7.9
mammals	17.6	7.4
creatures without legs	5.9	2.6
land and avian creatures	20.8	9.5

Table 6.2: The most useful queries at the first iteration of our algorithm. Utility is the expected number of new values for object labels as a result of this query. Cost is the human time needed (in seconds) to obtain the correct answer with a minimum of 95% accuracy on expectation.

or specialized ones such as the product taxonomy from eBay. These databases can provide high-level concepts or attributes as candidate queries.

If manual query construction is necessary (e.g. to augment an existing pool), we provide simple heuristics. As discussed above, there are two key components of good queries: high utility and low cost. For high utility the query should be broad in scope (e.g., “is there an animal?”, “is there furniture?”, “is it sharp?”). To be low cost, the query should be easy for the average human to answer using just salient information in the input. For example, queries such as “are there school supplies?”, “motorized vehicles?”, “things used to open cans/bottles?” took up to 3 times longer on average than simple queries such as “is there a bug?”, “a canine?”, “a ball?”. Generally, queries should avoid requiring the user to do additional inference beyond the provided input.

Query construction may involve significant effort, but it is a one-time, fixed investment: the label set for a particular application is relatively static, whereas the items to label can be dynamic and infinitely many. The cost saved in labeling many items can easily outweigh the fixed, upfront cost of query construction. Moreover, our method is designed to minimize the effort of query construction as it automatically selects the most effective queries.

Some examples of highest-utility queries at the first iteration of our algorithm are shown in Table 6.2.

Large-scale evaluation. We compare our algorithm to the baseline approach that queries a human for every object in every image. We use 3 metrics: (1) accuracy, or the total percentage of correct labels, (2) F1-score, or the harmonic mean of precision and recall on labels from all categories, and (3) reduction of human annotation time of our algorithm compared to the baseline.

Results are in Table 6.3. The error bars are the result of 5 simulations. Threshold is the acceptable label accuracy that determines the number of workers to recruit for each query. Our algorithm obtains cost savings of up to 6 times compared to the naïve approach while maintaining superior label accuracy.

⁵Actual queries are longer and include detailed definitions. The full list is in Appendix D.

Thresh	Accuracy		F1 score		Cost saving
	Naïve	Ours	Naïve	Ours	
95.0	99.64	99.75±0.00	75.67	76.97±0.16	3.93±0.00
90.0	99.29	99.62±0.00	60.17	60.69±0.39	6.18±0.01
85.0	99.25	99.62±0.00	59.09	60.46±0.39	6.11±0.01

Table 6.3: Our algorithm versus the naïve brute force approach. Thresh is a parameter of the algorithm (please refer to text for detail).

6.4.5 Discussion and Conclusion

Our algorithm works well in cases where the natural distribution of labels satisfies our assumptions, i.e. when the labels are correlated, sparse, and naturally form a hierarchy. If, on the other hand, the distribution of labels is dense and independent, there is little for our algorithm to exploit. In real world scenarios, though, and as validated by our experiments, exploiting the label distribution can yield significant savings.

Coming back to the object detection image annotation task, we conclude that it is always better to stick with more general but less ambiguous questions, such as “is there a mammal in the image?” as opposed to asking overly specific but potentially ambiguous questions, such as “is there an animal that can climb trees?” False positive answers only add extra cost whereas false negatives can significantly affect the quality of labeling.

6.5 Bounding box system for object detection

Once all images are labeled with the presence or absence of all object categories (using the algorithm of Section 6.4 above), we now annotate the location of all the objects. We use the bounding box system of [175] summarized below in Section 6.5.1 along with novel modifications of Section 6.5.2 to annotate the location of every instance of every present object category.

6.5.1 ILSVRC bounding box object annotation system

We use the crowdsourced bounding box annotation system of [175] which was developed for the single-object localization task of ILSVRC and is work done prior to this thesis. We briefly summarize it here for completeness.

The goal is to build a system that is fully automated, highly accurate, and cost-effective. Given a collection of images where the object of interest has been verified to exist, for each image the system collects a tight bounding box for every instance of the object.

There are two requirements:

- **Quality** Each bounding box needs to be tight, i.e. the smallest among all bounding boxes that contains all visible parts of the object. This facilitates the object detection learning algorithms

by providing the precise location of each object instance;

- **Coverage** Every object instance needs to have a bounding box. This is important for training localization algorithms because it tells the learning algorithms with certainty what is not the object.

The core challenge of building such a system is effectively controlling the data quality with minimal cost. Our key observation is that drawing a bounding box is significantly more difficult and time consuming than giving answers to multiple choice questions. Thus quality control through additional verification tasks is more cost-effective than consensus-based algorithms. This leads to the following workflow with simple basic subtasks:

1. **Drawing** A worker draws one bounding box around one instance of an object on the given image.
2. **Quality verification** A second worker checks if the bounding box is correctly drawn.
3. **Coverage verification** A third worker checks if all object instances have bounding boxes.

The sub-tasks are designed following two principles. First, the tasks are made as simple as possible. For example, instead of asking the worker to draw all bounding boxes on the same image, we ask the worker to draw only one. This reduces the complexity of the task. Second, each task has a fixed and predictable amount of work. For example, assuming that the input images are clean (object presence is correctly verified) and the coverage verification tasks give correct results, the amount of work of the drawing task is always that of providing exactly one bounding box.

Quality control on Tasks 2 and 3 is implemented by embedding “gold standard” images where the correct answer is known. Worker training for each of these subtasks is described in detail in [175].

Empirical evaluation. The system is evaluated on 10 categories with ImageNet [36]: balloon, bear, bed, bench, beach, bird, bookshelf, basketball hoop, bottle, and people. A subset of 200 images are randomly sampled from each category. On the image level, our evaluation shows that 97.9% images are completely covered with bounding boxes. For the remaining 2.1%, some bounding boxes are missing. However, these are all difficult cases: the size is too small, the boundary is blurry, or there is strong shadow.

On the bounding box level, 99.2% of all bounding boxes are accurate (the bounding boxes are visibly tight). The remaining 0.8% are somewhat off. No bounding boxes are found to have less than 50% intersection over union overlap with ground truth.

Additional evaluation of the overall cost and an analysis of quality control can be found in [175].

Level of annotation. One final note is that every bounding box is required to be as small as possible while including all visible parts of the object instance. An alternate annotation procedure could be to annotate the *full (estimated) extent* of the object: e.g., if a person’s legs are occluded and only the torso is visible, the bounding box could be drawn to include the likely location of the legs. However, this alternative procedure is inherently ambiguous and ill-defined, leading to disagreement among annotators and among researchers (what is the true “most likely” extent of this object?). We follow the standard protocol of only annotating visible object parts [158, 48].⁶

6.5.2 Object detection modifications

The bounding box annotation system described in Section 6.5.1 is used for annotating images for both the single-object localization dataset and the object detection dataset. However, two additional manual post-processing are needed to ensure accuracy in the object detection scenario:

Ambiguous objects. The first common source of error was that workers were not able to accurately differentiate some object classes during annotation. Some commonly confused labels were seal and sea otter, backpack and purse, banjo and guitar, violin and cello, brass instruments (trumpet, trombone, french horn and brass), flute and oboe, ladle and spatula. Despite our best efforts (providing positive and negative example images in the annotation task, adding text explanations to alert the user to the distinction between these categories) these errors persisted.

In the single-object localization setting, this problem was not as prominent for two reasons. First, the way the data was collected imposed a strong prior on the object class which was present. Second, since only one object category needed to be annotated per image, ambiguous images could be discarded: for example, if workers couldn’t agree on whether or not a trumpet was in fact present, this image could simply be removed. In contrast, for the object detection setting consensus had to be reached for all target categories on all images.

To fix this problem, once bounding box annotations were collected we manually looked through all cases where the bounding boxes for two different object classes had significant overlap with each other (about 3% of the collected boxes). About a quarter of these boxes were found to correspond to incorrect objects and were removed. Crowdsourcing this post-processing step (with very stringent accuracy constraints) would be possible but it occurred in few enough cases that it was faster (and more accurate) to do this in-house.

Duplicate annotations. The second common source of error were duplicate bounding boxes drawn on the same object instance. Despite instructions not to draw more than one bounding box around the same object instance and constraints in the annotation UI enforcing at least a 5 pixel

⁶Some datasets such as PASCAL VOC [48] and LabelMe [158] are able to provide more detailed annotations: for example, marking individual object instances as being *truncated*. We chose not to provide this level of detail in favor of annotating more images and more object instances.

difference between different bounding boxes, these errors persisted. One reason was that sometimes the initial bounding box was not perfect and subsequent labelers drew a slightly improved alternative.

This type of error was also present in the single-object localization scenario but was not a major cause for concern. A duplicate bounding box is a slightly perturbed but still correct positive example, and single-object localization is only concerned with correctly localizing one object instance. For the detection task algorithms are evaluated on the ability to localize *every* object instance, and penalized for duplicate detections, so it is imperative that these labeling errors are corrected (even if they only appear in about 0.6% of cases).

Approximately 1% of bounding boxes were found to have significant overlap of more than 50% with another bounding box of the same object class. We again manually verified all of these cases in-house. In approximately 40% of the cases the two bounding boxes correctly corresponded to different people in a crowd, to stacked plates, or to musical instruments nearby in an orchestra. In the other 60% of cases one of the boxes was randomly removed.

These verification steps complete the annotation procedure of bounding boxes around every instance of every object class in the validation, test and a subset of training images.

Training set annotation. With the optimized algorithm of Section 6.4 we fully annotated the validation and test sets. However, annotating *all* training images with all target object classes was still a budget challenge. Positive training images taken from the single-object localization dataset already had bounding box annotations of all instances of one object class on each image. We extended the existing annotations to the detection dataset by making two modifications. First, we corrected any bounding box omissions resulting from merging fine-grained categories: i.e., if an image belonged to the "dalmatian" category and all instances of "dalmatian" were annotated with bounding boxes for single-object localization, we ensured that all remaining "dog" instances are also annotated for the object detection task. Second, we collected significantly more training data for the person class because the existing annotation set was not diverse enough to be representative (the only person categories in the single-object localization task are scuba diver, groom, and ballplayer). To compensate, we additionally annotated people in a large fraction of the existing training set images.

6.6 Conclusions

We described the intricate and time-consuming process of collecting a large-scale object detection dataset for ILSVRC. The selection of object classes (Section 6.2) and images (Section 6.3) was done with a lot of our manual intervention, whereas data annotation (Sections 6.4 and 6.5) requires less manual intervention but necessitates the development of novel crowd engineering techniques. The collected dataset is now used as a standard large-scale object detection benchmark by the community,

as was described in detail in Chapter 5.

Chapter 7

Analysis of large-scale object recognition accuracy

7.1 Introduction

The growth of detection datasets and the multiple directions of object detection research provide both an unprecedented need and a great opportunity for a thorough evaluation of the current state of the field of categorical object detection. In this chapter we strive to answer two key questions. Where are we now as a field: what challenges of large-scale object detection have we successfully addressed, and which ones still remain? And where should we be going in building the next generation of object detectors?

The ILSVRC dataset introduced earlier in Chapter 5 provides an excellent testbed for understanding the performance of detectors for two reasons. First, it is a commonly accepted benchmark for large-scale detection and serves as the standard testbed for novel algorithms [177, 66, 165, 205, 170, 104, 152]. Second, its scale of hundreds of object classes allows us to measure the performance of algorithms as a function of several key properties of images and object classes (illustrated Figure 7.1). Besides looking at just the average accuracy across hundreds of object categories and tens of thousands of images as in Chapter 5, we delve deeper to understand where state-of-the-art algorithms make mistakes and where researchers' efforts should be focused to expedite progress. We conduct a series of analyses looking at how different detection methods perform on a number of image-level and object-class-level properties such as texture, color, deformation, and clutter. We learn important lessons of the current object detection methods and propose a number of insights for designing the next generation object detectors.

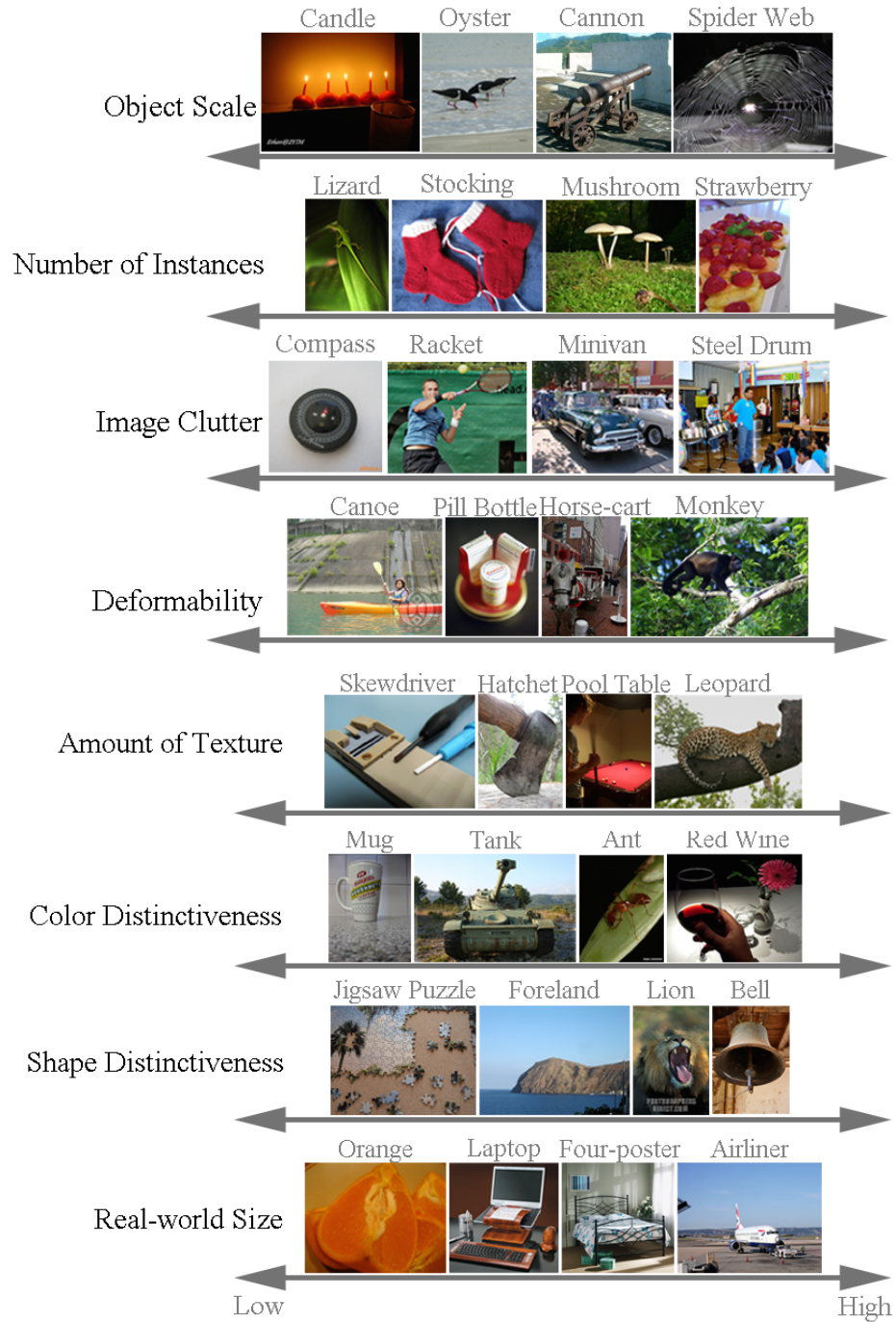


Figure 7.1: The diversity of ILSVRC along eight dimensions. Please refer to Section 7.3 for the definitions. For each dimension, we show example object categories along the range of that property.

Outline. We discuss related work in Section 7.2 and set up some preliminaries for our analysis in Section 7.3. Then we conduct analysis of detection algorithms during two time periods. Earlier in Section 5.6.1 we discussed how a deep learning approach revolutionized large-scale visual recognition in year 2012. As the first part of our analysis, we compare the performance of the first successful deep learning algorithm with the best leading method at the time back in year 2012 (Section 7.4). This is important for understanding both the importance of this algorithmic revolution as well as for the incredible rate of progress in object detection methods since then. For the second part of our analysis, we analyze the performance of the latest (deep learning-based) algorithms of ILSVRC 2014 (Section 7.5). This gives us insight into the current state of object detection.

Notation. Throughout this chapter, we use the abbreviation **ILSVRC** to refer specifically to the latest 2014 dataset of ILSVRC.¹ We use **PASCAL** to refer to the latest 2012 dataset of the PASCAL Visual Object Classes challenge [48, 46]. For consistency with the object detection metric (higher is better), in this section we will be using image classification and single-object localization *accuracy* instead of error, where $accuracy = 1 - error$.

7.2 Related work

Several works have analyzed the effects of factors such as occlusion, variations in aspect ratios and changes of viewpoints, for both specific category detection and general object detection on a small set of categories [210, 204, 83, 43, 42]. Others have provided insight into dataset design [145, 182, 48]. [228] analyzed the relative impact of adding more training data versus building better detection models.

Hoiem et al. [82] designed and performed a thorough evaluation of several state-of-the-art object detectors on the PASCAL dataset, providing much more detail than single average precision score for each category. This study highlighted some insights into where detection algorithms make mistake – e.g., false positive detections surprisingly rarely occur on the background clutter and much more often appear on objects of similar categories. This kind of analysis offers a way to examine the current state of the field of object detection, but more importantly sheds light on what should be done for designing the next generation of object detectors. While other insights can be obtained from further analyzing the PASCAL dataset, one limitation is that it contains only 20 basic object categories. Therefore, it is harder to do meta-analysis or measure the impact of object properties such as color, texture, real-world size, on the performance of object detectors. Such analysis is important in understanding when detection approaches can be expected to work or to fail, and where more research is needed.

¹Note that for image classification and object localization tasks the data in ILSVRC 2012-2014 has stayed the same (cf Chapter 5).

7.3 Factors for analysis

We will analyze the performance of algorithms as a function of two types of factors: image-level properties and intrinsic object class properties. We discuss these in detail below; please refer to Figure 7.1 for example illustrations of the properties.

Image-level properties

We introduce four metrics of localization difficulty using image-level properties and use them for our analysis in this chapter. Appendix A additionally uses these metrics to demonstrate that the ILSVRC dataset has many of the same challenges as the previously established PASCAL VOC benchmark [48]. The metrics are:

1. **Number of instances.** The number of instances of the target object per image is an important metric to differentiate between product shots and more challenging unstructured images. Real-world scenes are likely to contain multiple instances of objects. The nearby object instances are particularly difficult to delineate, making these images more challenging for object detectors.
2. **Object scale.** As described in [82], smaller objects tend to be significantly more difficult to localize. We compute the fraction of image area occupied by the bounding boxes around annotated object instances.
3. **Chance performance of localization.** Chance performance on a dataset is a common metric to consider. We introduce the chance performance of localization (CPL) metric for evaluating the difficulty of the single-object localization task (Section 5.3.2). We define the CPL measure as the expected accuracy of a detector which first randomly samples an object instance of that class and then uses its bounding box directly as the proposed localization window on all other images (after rescaling the images to the same size). Concretely, let B_1, B_2, \dots, B_N be all the bounding boxes of the object instances within a class, then

$$\text{CPL} = \frac{\sum_i \sum_{j \neq i} \text{IOU}(B_i, B_j) \geq 0.5}{N(N-1)} \quad (7.1)$$

This measure correlates strongly ($\rho = 0.9$ on ILSVRC) with the average object scale. We use CPL for analyzing accuracy on the single-object localization task and the object scale for evaluating the accuracy on the object detection task.

4. **Clutter.** Intuitively, even small objects are easy to localize on a plain background. To quantify clutter we employ the objectness measure of [3], which is a class-generic object detector evaluating how likely a window in the image contains a coherent object (of any class) as opposed to background (sky, water, grass). For every image m containing target object instances

at positions B_1^m, B_2^m, \dots , we use the publicly available objectness software to sample 1000 windows $W_1^m, W_2^m, \dots, W_{1000}^m$, in order of decreasing probability of the window containing any generic object. Let $\text{OBJ}(m)$ be the number of generic object-looking windows sampled before localizing an instance of the target category, i.e., $\text{OBJ}(m) = \min\{k : \max_i \text{IOU}(W_k^m, B_i^m) \geq 0.5\}$. For a category containing M images, we compute the average number of such windows per image and define

$$\text{CLUTTER} = \log_2 \left(\frac{1}{M} \sum_m \text{OBJ}(m) \right) \quad (7.2)$$

The higher the clutter of a category, the harder the objects are to localize according to generic cues. Like CPL, this measure is particularly suited for evaluating the difficulty of single-object localization.

We use number of instances, chance performance of localization and clutter for thoroughly evaluating single-object localization accuracy in Section 7.4.4. When evaluating the accuracy on all tasks in Section 7.5.4, we use the simpler object scale which is broadly applicable to all tasks.

Intrinsic object class properties.

We also analyze the performance of algorithms as a function of intrinsic object class properties. We first consider the differences between **natural** and **man-made** objects. The classification of classes into natural and man-made can be derived from the ImageNet hierarchy [36]. Next we define five additional properties inspired by human vision: real-world size, deformability within instance, amount of texture, distinctiveness of color and distinctiveness of shape, all visualized in Figure 7.1.

Human subjects annotated each of the 1000 image classification and single-object localization object classes from ILSVRC2012-2014 with these properties. [151]. By construction (see Section 6.2), each of the 200 object detection classes is either also one of 1000 object classes or is an ancestor of one or more of the 1000 classes in the ImageNet hierarchy. To compute the values of the properties for each object detection class, we simply average the annotated values of the descendant classes.

The domains for these properties are:

1. **Real-world size:** XS for extra small (e.g. nail), small (e.g. fox), medium (e.g. bookcase), large (e.g. car) or XL for extra large (e.g. church)
2. **Distinctiveness of color:** none (e.g. clothes), low (e.g. cleaver), medium (e.g. hay), high (e.g. tennis ball)
3. **Distinctiveness of shape:** none (e.g. chocolate sauce), low (e.g. tape player), medium (e.g. T-shirt), high (e.g. banana)
4. **Deformability within instance:** Rigid (e.g., mug) or deformable (e.g., water snake)

5. **Amount of texture:** none (e.g. punching bag), low (e.g. horse), medium (e.g. sheep) or high (e.g. honeycomb)

We will analyze the performance of different algorithms as a function of these properties and draw conclusions about which domains are easy and difficult for current algorithms.

7.4 Analysis of the deep learning breakthrough

We begin by analyzing the state of the field of large-scale object detection following the breakthrough in large-scale recognition with a deep learning method in year 2012 [104]. Additional information is available at www.image-net.org/challenges/LSVRC/2012/analysis/.

7.4.1 Setup

Data. First, we need to establish the dataset to use for our analysis. The largest object detection dataset at the time was the PASCAL VOC with 20 object classes [48, 46]; the ILSVRC object detection dataset was collected a year later in 2013. To perform a large-scale analysis of the performance of algorithms we choose the ILSVRC single-object localization dataset described in detail previously in Chapter 5. It spans 1000 object classes containing both internal nodes and leaf nodes of ImageNet. Figure 7.1 visualizes some of the diversity of the classes along several dimensions. There are 1.2 million images for training, 50K images for validation, and 100K new images for testing. There are 620K bounding box annotations on the training images (covering about 42% of the data), and an additional 230K for the validation and test images. Appendix A contains more statistics and in addition demonstrates that the dataset has many of the localization challenges of the PASCAL VOC.

Evaluation criteria We use the top-5 metric of localization accuracy presented in Section 5.5.2. Briefly, each object class C has a set of images associated with it, and each image is human annotated with bounding boxes B_1, B_2, \dots indicating the location of *all* instances of this object class. Since additional unannotated object classes may be present, the algorithm is allowed to produce up to 5 annotations per image without incurring a cost for false positive detections. The object class is considered correctly detected if for some proposed annotation (c_i, b_i) with c_i the class label and b_i the bounding box, $c_i = C$ and b_i correctly localizes one of the objects B_1, B_2, \dots according to the standard IOU measure.

¹Please see Appendix B for an analysis of the top-5 evaluation criteria.

7.4.2 Algorithms

Several object detection systems participated in the ImageNet single-object localization challenge of 2012; the winners are good candidates for our analysis (Section 5.6.1). Using two very distinct leading algorithms, we analyze the successes and weaknesses in large-scale detection. The analysis in this chapter requires only per-class accuracies and class confusion matrices of these methods.

SV system. The winning system of the challenge, named SuperVision and abbreviated as SV, uses neural networks to learn the full image representation automatically from data. It is based upon a supervised convolutional neural network with 7 hidden layers, trained using stochastic gradient descent on the GPU [104]. This system was targeted to image classification so its strong performance on object localization is particularly impressive.

VGG system. The other algorithm, OXFORD_VGG, is based on the more conventional image classification and detection pipeline. It uses an image classification system with dense SIFT features and color statistics [123], a Fisher vector representation [161], and a linear SVM classifier, plus additional insights from [4, 162], combined with the deformable parts-based model (DPM) [54] which has been the dominant model for generic object detection for many years.

Upper bound. We also consider an optimistic upper bound which combines the outputs of the VGG and SV on every image. Here the output on the image is considered correct if any of the 10 predicted (class, location) pairs are correct. Evaluating this joint algorithm helps to summarize the common trends of SV and VGG as well as to illustrate the key scenarios where SV and VGG provide complementary sources of information.

7.4.3 Recognition accuracy as a function of object hierarchy

Object localization involves both correctly predicting the class label and localizing the object; in seeking to understand the limitations of current algorithms we initially decouple these two measures. As a first step of our analysis, we look at whether the algorithms are looking for the right objects. We visualize the 1000-way confusion matrix of classification errors in Figure 7.2(a) and confirm the intuition that classes that are semantically close together get confused with each other more often [82]. Going further, the hierarchical structure of ILSVRC allows us to analyze the accuracy of the algorithm as a function of semantic depth; in other words, we can relax the requirement that a Dalmatian be classified as a Dalmatian and instead evaluate the accuracy of the algorithm when allowing any dog breed to be an acceptable label. In Figure 7.2(b,c) we plot the classification and localization accuracies as a function of semantic depth: at level 0 we require the exact label, at level 1 we accept any sister synset, and so forth. For reference, breeds of dogs are on average 4 levels removed from the "domestic dog" node and birds are 4.6 levels removed from "bird" node.

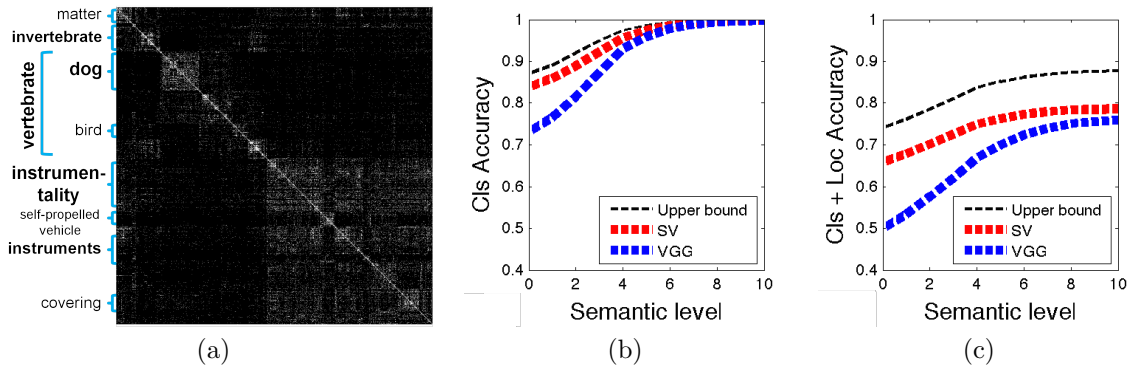


Figure 7.2: (a) Confusion matrix of classification scores using the SV algorithm, with categories ordered semantically. The expected block pattern shows most confusion between similar categories. (b) Classification and (c) classification with localization accuracy as a function of moving up the ImageNet hierarchy. [36] At level 0 the algorithm is required to produce the exact class label; at level 1 any sister label is accepted, at level 10 any leaf node of generic concepts such as "living thing" is accepted.

Three things are apparent from this analysis: (1) basic-level classification is surprisingly accurate: SV distinguishes dogs from other objects with > 0.99 accuracy, and birds with 0.98 accuracy; (2) despite this, the gap between classification and localization accuracies remains large (> 0.180 at all semantic levels); (3) SV and VGG provide complementary sources of information on localization: the upper bound is 0.083 higher than either SV or VGG alone.² The last two observations prompt more detailed investigation.

7.4.4 Effect of image-level statistics on localization accuracy

Despite high levels of classification accuracy on basic-level categories, localization is far from perfect. To better understand what scenarios are challenging for current detection algorithms, we begin by evaluating the accuracy as a function of global image statistics. Figure 7.3 visualizes the localization accuracy of the detection algorithms described in Section 7.4.2 as a function of image-level quantitative measures of localization of Section 7.3. We analyze these measures one by one.

Instances per image. SV is more strongly correlated with the number of instances of the object than VGG (correlation of -0.436 versus -0.052). *This suggests that SV would be good at, e.g., noticing that there are cars in the image but might have difficulty separating out two nearby cars.* On categories with more than 2 instances per image, the correlation between number of instances and SV accuracy is significantly weaker (-0.156), implying that multiple objects are challenging

²An interesting side note is that the probability of correctly localizing the object given that it was correctly classified remains the same for SV at all levels of the semantic hierarchy (at 78%) but increases dramatically for VGG (from 68% to 76%). This confirms the intuition that SV is well suited for fine-grained classification while VGG is best used for localizing basic object categories.

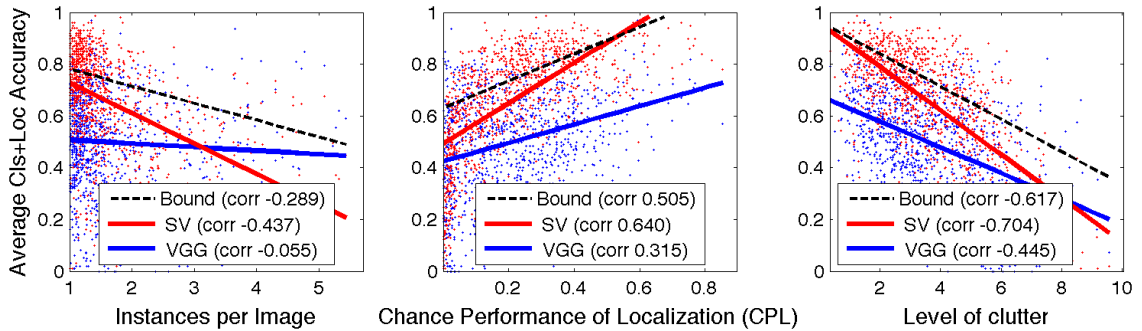


Figure 7.3: The impact of several quantitative measures of localization difficulty (Section 7.3) on the localization accuracy. Each dot corresponds to one of the 1000 object categories of ILSVRC2012 and to one of the two algorithms (SV in red, VGG in blue). The difficulty measure (x-axis) is computed on the validation set; the accuracy of the algorithm (y-axis) is evaluated on the test set. The best fit linear models for each algorithm are also shown to summarize trends. The black line corresponds to the upper bound combination of SV and VGG; the accuracy on individual class of this method is not shown to reduce clutter. Please refer to Section 7.4.4 for analysis.

regardless of the exact number. Combining the global model of SV with the strong object boundary model of VGG is a promising future direction.

Chance Performance of Localization (CPL). The correlation of CPL with SV accuracy is double that with VGG (0.640 vs 0.315) and the slope of the regression line is two times steeper (0.781 of SV, 0.356 of VGG) so SV’s accuracy degrades faster and more consistently as CPL get smaller. In fact, when considering 225 object categories with lowest CPL the localization accuracy of SV and VGG is the same at 0.404, and on smaller objects VGG outperforms SV. We plot chance performance of localization (CPL) versus the accuracy of the algorithms in Figure 7.4. CPL is highly correlated with object scale as noted in Section 7.3. *VGG is better suited than SV for localizing small objects.*

Some categories contain a bimodal distribution of images accounting for low CPL: a mixture of close-ups with one object instance occupying the whole image and images depicting a large cluster of small objects. *SV and VGG algorithms perform well on different subsets of such data and combine to create a much stronger detector.* Some example categories are screw (CPL 0.9%, upper bound accuracy 0.624, SV accuracy 0.269, VGG accuracy 0.398) or boathouse (CPL 1.1%, upper bound accuracy 0.691, SV accuracy 0.443, VGG accuracy 0.454).

Clutter. There is a strong correlation between the level of clutter and the accuracy of all the algorithms (VGG correlation -0.445 , SV -0.704). *This shows that clutter may be a useful metric for evaluating the difficulty of detection datasets.*³

³Additionally, this implies that the number of objectness windows [3] is logarithmically related to the difficulty of localization.

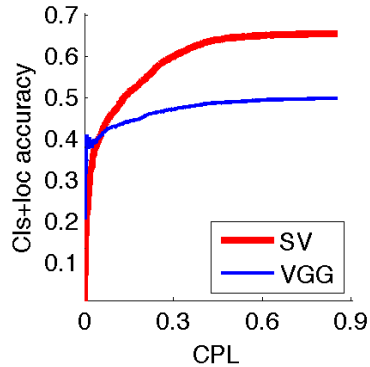


Figure 7.4: Cumulative localization of SV (red) and VGG (blue) as a function of chance performance of localization (CPL). The height of the curve corresponds to the average accuracy of the object categories with equal or smaller CPL measures. SV outperforms VGG except when considering a subset of 225 object categories with lowest CPL.

With this in mind, we consider a subset of 250 ILSVRC categories which has the same average clutter of 5.90 as the PASCAL dataset. On this subset, SV achieves localization accuracy of 0.439, still significantly outperforming VGG with accuracy of 0.374.⁴

7.4.5 Effect of intrinsic object class properties on localization accuracy

So far we have studied how image-level properties affect detection, and now turn to examine the effects of intrinsic object properties introduced in Section 7.3. Access to results for up to a thousand categories allows us to perform this analysis. Figures 7.5-7.6 visualizes the effects of these properties on the performance of object detectors, and here we summarize the key findings.

Natural vs man-made objects. *The accuracy of object detectors is strongly correlated with whether the object is natural or man-made.* Figure 7.5(a) shows that SV achieves localization accuracy of 0.768 on the 427 natural ILSVRC classes compared to only 0.570 on the 573 man-made classes; VGG achieves 0.514 on natural compared on 0.486 on man-made.

Importantly, this observation holds regardless of the image-level statistics of Section 7.4.4. Figure 7.5(b) shows the cumulative localization accuracy as a function of increasing CPL on natural and man-made objects separately. Regardless of how difficult of a subset of ILSVRC is chosen, the accuracy of SV on natural objects is always at least 0.186 higher than its accuracy of man-made objects. Interestingly, on the more challenging categories (e.g., the 330 categories with $\text{CPL} \leq 0.1$), VGG’s localization accuracy on natural objects is also 0.143 higher than on man-made ones even

⁴A similar conclusion holds true with the CPL metric as well. On the 562 hardest ILSVRC categories with the same average CPL as the PASCAL categories, SV achieves localization accuracy of 0.554, significantly outperforming VGG with localization accuracy of 0.461.

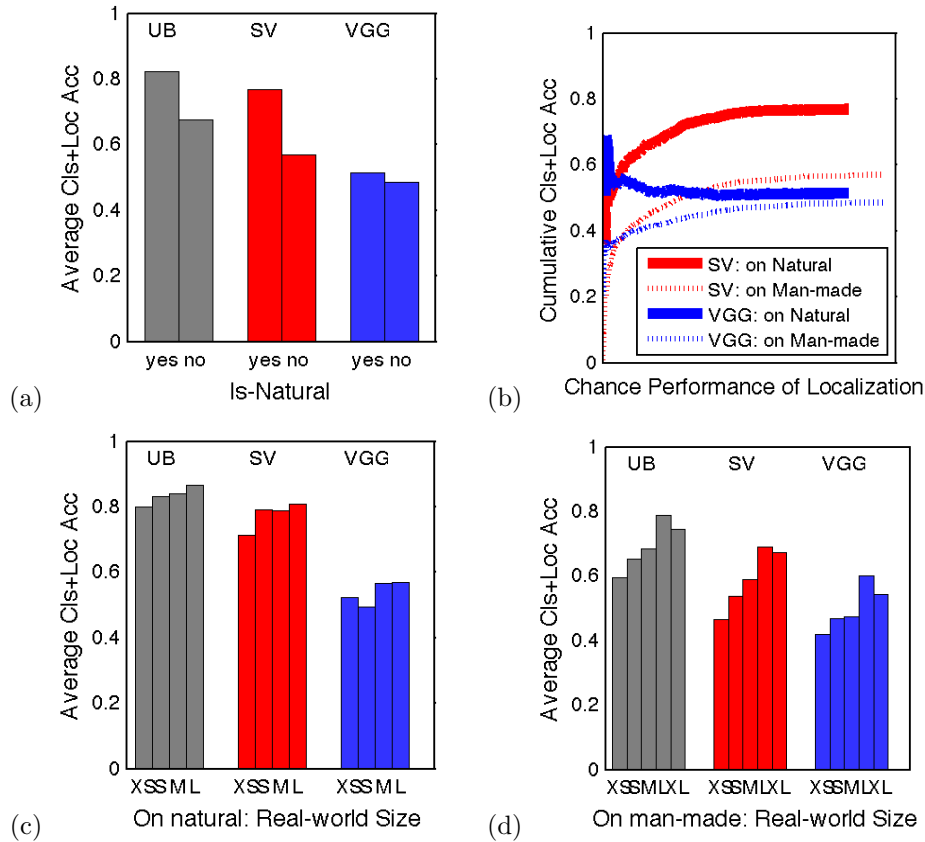


Figure 7.5: Localization accuracy of Upper Bound (gray), SV (red) and VGG (blue) as a function of the intrinsic properties of the ILSVRC2012 object categories. Plots (a), (c), (d) show the average localization accuracy of algorithms on subsets of ILSVRC. Plot (b) shows the cumulative localization accuracy as a function of the chance performance of localization (Section 7.3). The height of the curve corresponds to the average accuracy of the object categories with equal or smaller CPL measures. Continued in Figures 7.6 and Figure 7.7.

though on average over 1000 categories VGG’s accuracy is not as strongly affected by this property. Incidentally, similar patterns emerge with clutter measure instead of CPL.

Real-world size. *Algorithms tend to perform better on objects which are larger in the real-world,* both on natural objects (Figure 7.5(c)) and on man-made objects (Figure 7.5(d)). One exception is huge man-made objects which are slightly more difficult to localize than the man-made large objects: since many huge objects are buildings (e.g., church, shoe shop, prison) they may not be fully visible in the image or the picture might be taken inside the building, so localizing them may be particularly challenging. There are only a few huge *natural* objects so we omit them from the analysis.

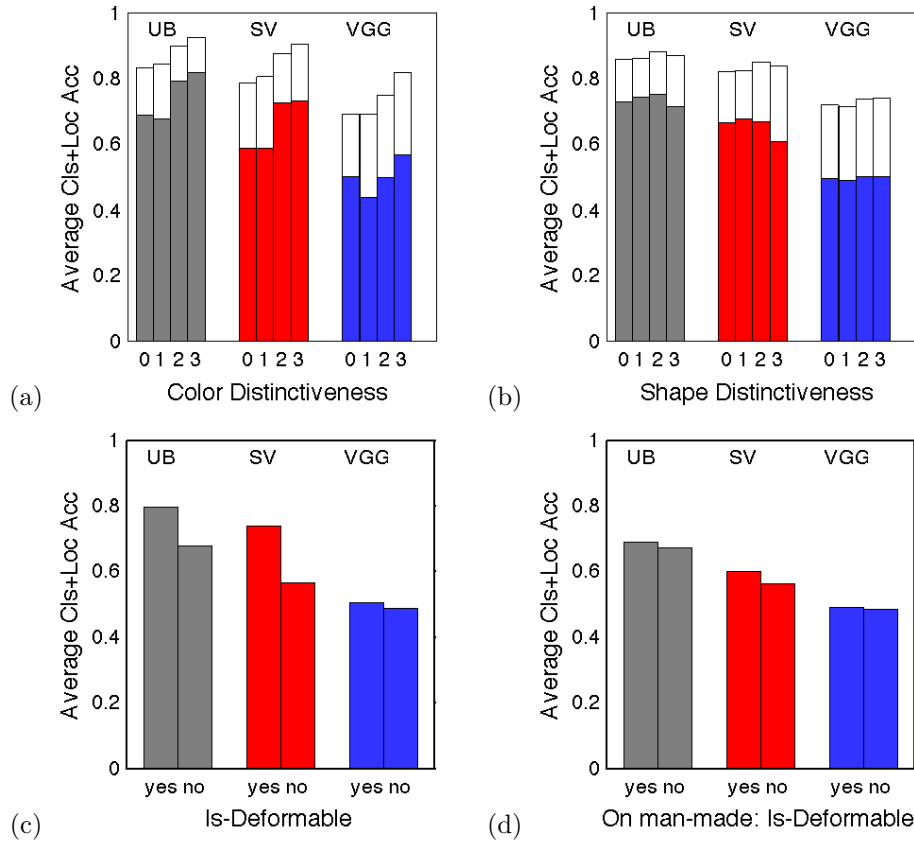


Figure 7.6: Continuation of Figure 7.5. Localization accuracy of Upper Bound (gray), SV (red) and VGG (blue) as a function of the intrinsic properties of the ILSVRC2012 object categories. All plots show the average localization accuracy of algorithms on subsets of ILSVRC. Plots (c) and (d) additionally report classification accuracy (white bars). Continued in Figure 7.7.

Tiny objects are very challenging for both SV and VGG; however, combining the algorithms yields significant improvement. The localization accuracy of the upper bound method is 0.111 higher than the accuracy of either SV or VGG on tiny objects. This is significantly better than improvements of 0.071, 0.077, 0.090 and 0.072 on the small, medium, large and huge objects respectively.

Distinctiveness of color. *Distinctiveness of color makes it easier to detect objects; however, the benefits of color distinctiveness are not as pronounced when controlling for whether the object is natural or man-made.* Figure 7.6(a) shows that objects which have distinctive colors tend to be easier to classify and localize than those that don’t. The upper bound localization accuracy is 0.117 higher on objects with distinctive color than on those without. There was significant variability in the human annotations for this attribute (color distinctiveness is difficult to precisely define), so we simplify to two cases: has distinctive color (“medium” or “high”) or does not.

Since color distinctiveness is strongly correlated with being man-made (84% of natural objects have distinctive color compared to only 21% of man-made objects), we evaluate on man-made versus natural objects separately. Restricted to man-made objects, localization accuracy of upper bound is only 0.044 higher on objects with distinctive color than on those without; restricted to natural objects, it is only 0.040 higher for objects distinctive in color.

Distinctiveness of shape. Figure 7.6(b) visualizes average classification and localization accuracies of the three algorithms across the different levels of shape distinctiveness. *There is no observed correlation between human-annotated distinctiveness of shape and the accuracy of the algorithms* (this holds true when considering subsets of the object classes, such as just man-made objects as well). This is consistent with the intuition that general object category detection algorithms tend to avoid rigid modeling of object shape and instead rely on other cues instead, such as texture or color.

Deformability within instance. One aspect of object shape that is often modeled is deformability within instance. [54] *Whether or not the object is deformable has relatively little bearing on the performance of the algorithms in the absence of other factors.* VGG is largely unaffected by deformability: localization accuracy of 0.507 on deformable versus 0.489 on non-deformable objects as shown in Figure 7.6(c). On average SV is significantly more accurate on deformable objects than non-deformable ones (localization accuracy 0.740 versus 0.566); however, when evaluating separately just on the 573 man-made classes in Figure 7.6(d) the effect becomes significantly less pronounced (localization accuracy 0.602 on 116 deformable objects versus 0.566 on 457 non-deformable ones).⁵

Amount of texture. *Both algorithms are much more accurate on textured objects in both classification and classification with localization* as shown in Figure 7.7(a)). A similar pattern appears on both man-made and natural classes independently. Man-made objects have average texture of 2.2 while natural objects are slightly more textured at 2.8.

We now consider just the images within each class which were correctly classified by VGG and compute the per-class localization accuracy conditioned on correct classification. Grouping the classes across the four levels of texture, we observe that VGG correctly localizes the object in between 65 – 68% of the correctly classified images on average per class for every level. For SV, however, the pattern is different: on untextured objects SV also accurately localizes 65% of the correctly classified images on average, but on highly textured it localized 82%!

Level of texture is correlated with chance performance of localization: going from zero to high texture, the average CPL is 12.7, 19.9, 23.7, and 26.4. In Figure 7.7(b) we show that this pattern

⁵The set of 427 natural classes contains only 32 non-deformable objects (e.g., strawberry, lemon, rose hip), so we omit it from the analysis.

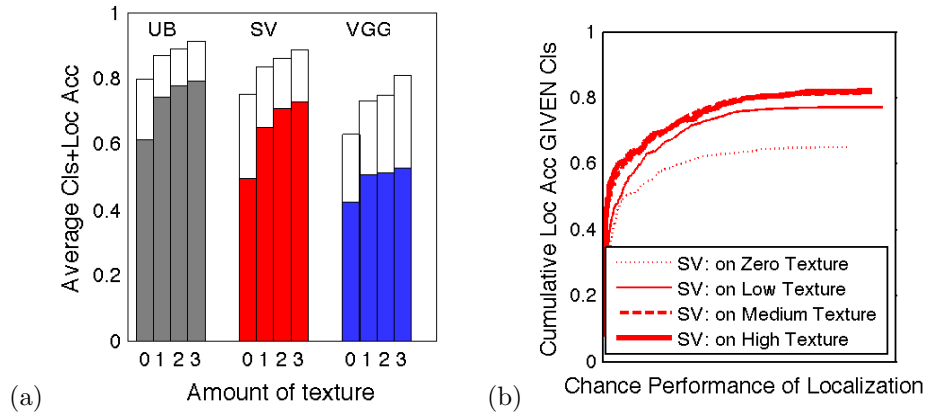


Figure 7.7: Continuation of Figures 7.5 and 7.6. Localization accuracy of Upper Bound (gray), SV (red) and VGG (blue) as a function of the intrinsic properties of the ILSVRC2012 object categories. All plots show the average localization accuracy of algorithms on subsets of ILSVRC. Plot (a) shows the average localization accuracy (colored bars) and classification accuracy (white bars) of the algorithms on subsets of ILSVRC. Plot (b) shows the cumulative localization accuracy as a function of the chance performance of localization (Section 7.3). The y-axis corresponds to the localization accuracy *only on images which were correctly classified*.

holds even across different CPL. *Once the image has been correctly classified, SV correctly localizes many more of the textured than untextured objects.*

7.4.6 Conclusions from year 2012

We summarize several key observations from the above analysis of the leading algorithms in year 2012 which provided guidance for future research.

Focus on fine-grained recognition. Classification accuracy of basic-level categories was already quite high; however, distinguishing between more fine-grained classes is much more challenging for current methods. Consistent with the recent trend in recognition literature, this reinforces the need to focus on capturing fine-grained distinction between classes in seeking to better understand the visual world.

Clutter measure for evaluating datasets and targeting algorithms. The measure of clutter using the latest techniques in unsupervised object discovery [3] defined in Section 7.3 is strongly correlated with the accuracy of current state-of-the-art algorithms. While the high-level insight that clutter is detrimental to the performance of detection algorithms is not novel, this suggests that the introduced metric is useful to consider when collecting new datasets or designing the next generation of object detectors.

Decoupling natural and man-made objects. Natural objects are significantly easier to localize than man-made objects for the current algorithms, even when controlling for factors such as scale of the object and level of clutter. This implies that (1) better modeling of man-made objects is a key direction for generic object detection, and (2) it is important to decouple the two when evaluating the performance of algorithms as a function of other properties.

Combining detection algorithms. The deep learning-based detector SV and the more traditional detector VGG are found to be complementary to each other on categories with low CPL in Section 7.4.4 and, similarly, on objects which are “tiny” in the real world in Section 7.4.5. This may be intuitive given prior knowledge about the design of the SV and VGG systems, but it is still useful to quantify. This is a key domain to consider when designing the next generation of detectors combining the benefits of the current leading systems.

7.5 Current state of categorical object recognition

After the breakthrough of year 2012, object recognition accuracy continued to improve. Many of the shortcomings identified in Section 7.4 were solved but some still remained. In this section we provide an up-to-date analysis of the latest leading algorithms of the ILSVRC challenge.

7.5.1 Setup

Data. Since the introduction of the large-scale object detection ILSVRC dataset in year 2013, we are able to analyze the performance of algorithms at scale on the standard object detection task as well. In this section, we consider all three ILSVRC tasks: image classification, object localization and object detection documented in Section 5.4. We use the latest ILSVRC2014 version.

Evaluation criteria. We use the standard evaluation metrics for each of the three tasks as described in Section 5.5.

Algorithms. As described in Section 5.6.1, the leading algorithms in ILSVRC 2013-2014 have relied on a similar deep learning framework. Thus in this section we will focus analyzing an “optimistic” measurement of state-of-the-art recognition performance instead of focusing on the differences in individual algorithms. For each task and each object class, we compute the best performance of *any* entry submitted to *any* ILSVRC2012-2014, including methods using additional training data. Since the test sets have remained the same, we can directly compare all the entries in the past three years to obtain the most “optimistic” measurement of state-of-the-art accuracy on each category.

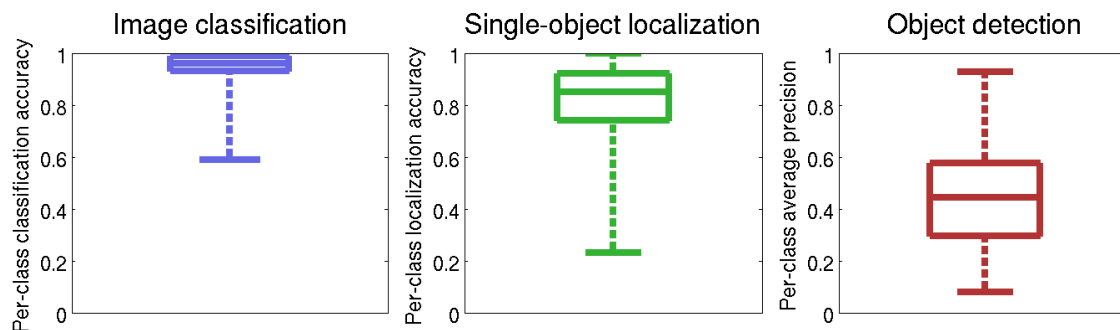


Figure 7.8: For each object class, we consider the best performance of any entry submitted to ILSVRC2012-2014, including entries using additional training data. The plots show the distribution of these “optimistic” per-class results. Performance is measured as accuracy for image classification (left) and for single-object localization (middle), and as average precision for object detection (right). While the results are very promising in image classification, the ILSVRC datasets are far from saturated: many object classes continue to be challenging for current algorithms.

7.5.2 Range of accuracy across object classes

Figure 7.8 shows the distribution of accuracy achieved by the “optimistic” models across the object categories. The image classification model achieves 94.6% accuracy on average (or 5.4% error), but there remains a 41.0% absolute difference in accuracy between the most and least accurate object class. The single-object localization model achieves 81.5% accuracy on average (or 18.5% error), with a 77.0% range in accuracy across the object classes. The object detection model achieves 44.7% average precision, with an 84.7% range across the object classes. It is clear that the ILSVRC dataset is far from saturated: performance on many categories has remained poor despite the strong overall performance of the models.

7.5.3 Qualitative examples of easy and hard classes

Figures 7.9-7.11 show the easiest and hardest classes for each task, i.e., classes with the best and worst results obtained with the “optimistic” models.

For image classification, 121 out of 1000 object classes have 100% image classification accuracy according to the optimistic estimate. Figure 7.9 shows a random set of 10 of them. They contain a variety of classes, such as mammals like “red fox” and animals with distinctive structures like “stingray”. The hardest classes in the image classification task, with accuracy as low as 59.0%, include metallic and see-through man-made objects, such as “hook” and “water bottle,” the material “velvet” and the highly varied scene class “restaurant.”

Results for single-object localization are shown in Figure 7.10. The 10 easiest classes with 99.0 – 100% accuracy are all mammals and birds. The hardest classes include metallic man-made objects such as “letter opener” and “ladle”, plus thin structures such as “pole” and “spacebar”

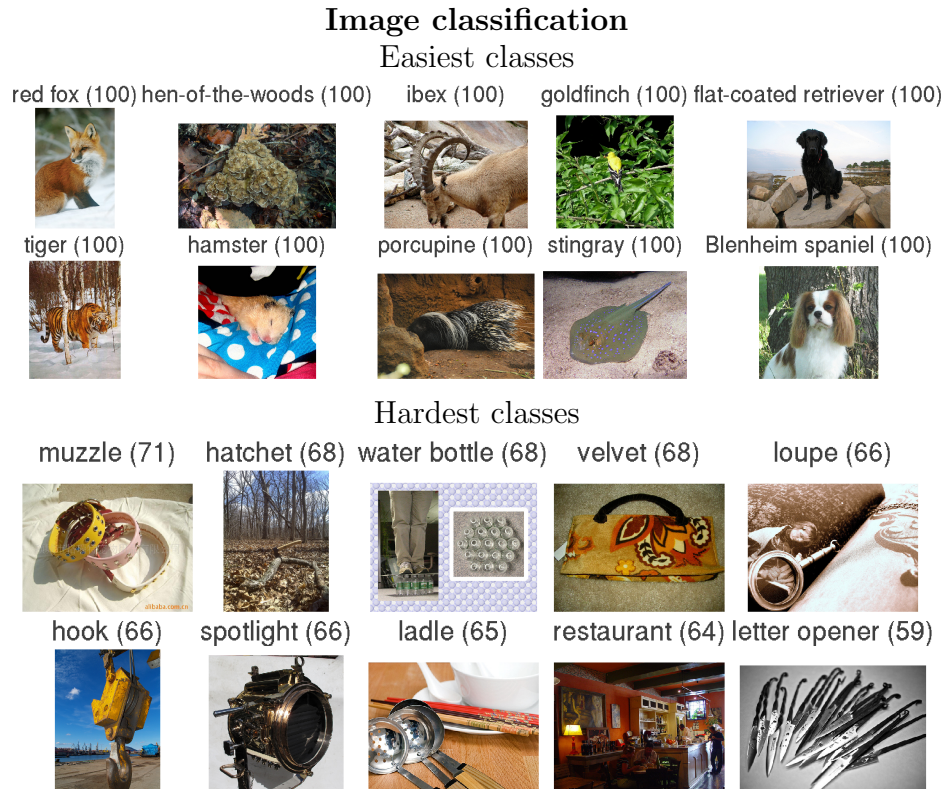


Figure 7.9: For each object category, we take the best performance of any entry submitted to ILSVRC2012-2014 (including entries using additional training data). Given these “optimistic” results we show the easiest and hardest classes for the classification task. The numbers in parentheses indicate classification accuracy. The 10 easiest classes are randomly selected from among 121 object classes with 100% accuracy. Results from other tasks are shown in Figures 7.10 and 7.11.

and highly varied classes such as “wing”. The most challenging class “spacebar” has a only 23.0% localization accuracy.

Object detection results are shown in Figure 7.11. The easiest classes are living organisms such as “dog” and “tiger”, plus “basketball” and “volleyball” with distinctive shape and color, and a somewhat surprising “snowplow.” The easiest class “butterfly” is not yet perfectly detected but is very close with 92.7% AP. The hardest classes are as expected small thin objects such as “flute” and “nail”, and the highly varied “lamp” and “backpack” classes, with as low as 8.0% AP.

7.5.4 Effect of image-level statistics on recognition accuracy

We now take a closer look at the image properties to try to understand why current algorithms perform well on some object classes but not others. In Section 7.4.4 we found chance performance of localization and image clutter are useful metrics. However, they are both designed specifically for

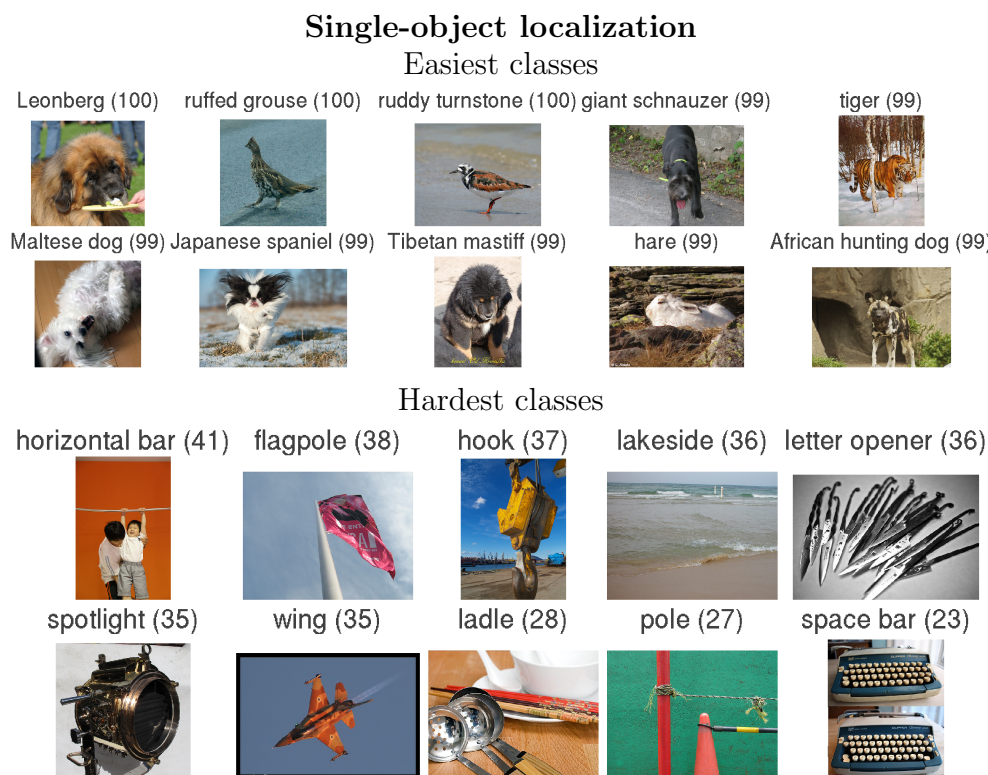


Figure 7.10: For each object category, we take the best performance of any entry submitted to ILSVRC2012-2014 (including entries using additional training data). Given these “optimistic” results we show the easiest and hardest classes for the localization task. The numbers in parentheses indicate localization accuracy. Results from other tasks are shown in Figures 7.9 and 7.11.

the image localization task (Section 7.3). In this section, since we want to evaluate the performance on all tasks, we use the average object scale as the measure of image difficulty. It is highly correlated with chance performance of localization but is more interpretable for the object detection task.

For every object class, we compute its *average scale*, or the average fraction of image area occupied by an instance of the object class on the ILSVRC2012-2014 validation set. Since the images and object classes in the image classification and single-object localization tasks are the same, we use the bounding box annotations of the single-object localization dataset for both tasks. In that dataset the object classes range from “swimming trunks” with scale of 1.5% to “spider web” with scale of 85.6%. In the object detection validation dataset the object classes range from “sunglasses” with scale of 1.3% to “sofa” with scale of 44.4%.

Figure 7.12 shows the performance of the “optimistic” method as a function of the average scale of the object in the image. Each dot corresponds to one object class. We observe a very weak positive correlation between object scale and image classification accuracy: $\rho = 0.14$. For

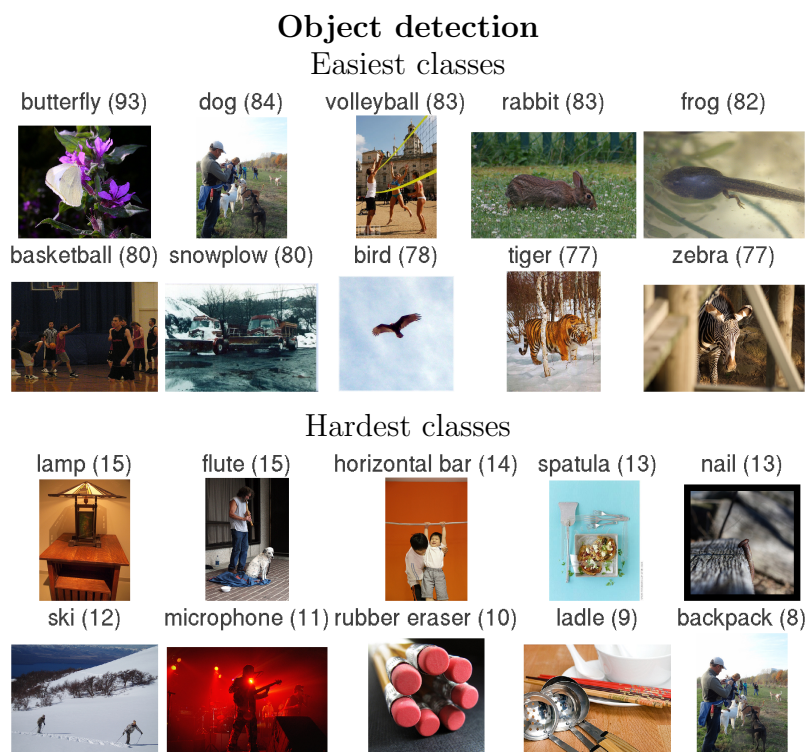


Figure 7.11: For each object category, we take the best performance of any entry submitted to ILSVRC2012-2014 (including entries using additional training data). Given these “optimistic” results we show the easiest and hardest classes for the object detection task. The numbers in parentheses indicate average precision. Results from other tasks are shown in Figures 7.9 and 7.10.

single-object localization and object detection the correlation is stronger, at $\rho = 0.40$ and $\rho = 0.41$ respectively. *We conclude that accuracy is correlated with object scale in the image, not all variation in accuracy can be accounted for by scale alone.* In the next section we will normalize for object scale to ensure that this factor is not affecting our conclusions.

7.5.5 Effect of intrinsic object class properties on recognition accuracy

Besides considering image-level properties we can also observe how accuracy changes as a function of intrinsic object properties. introduced in Section 7.3. Here we only analyze real-world size, deformability within instance and amount of texture since the other two attributes (distinctiveness of color and distinctiveness of shape) tended to have insufficient inner-annotator agreement and did not produce statistically significant conclusions.

In this section we draw the following conclusions about state-of-the-art recognition accuracy as a function of these object properties:

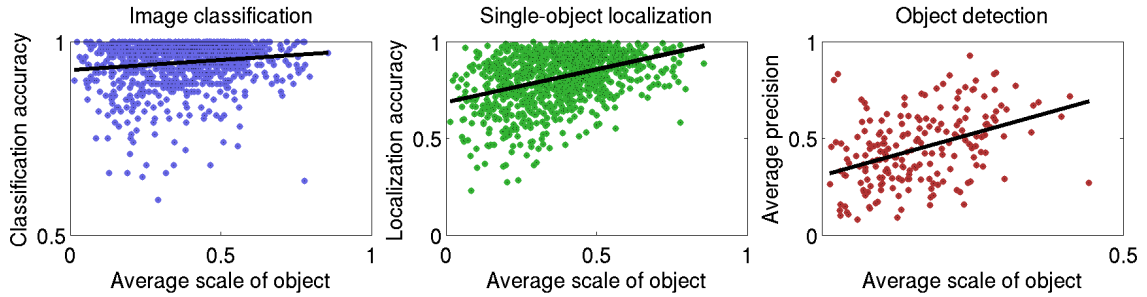


Figure 7.12: Performance of the “optimistic” method as a function of object scale in the image, on each task. Each dot corresponds to one object class. Average scale (x-axis) is computed as the average fraction of the image area occupied by an instance of that object class on the ILSVRC2014 validation set. “Optimistic” performance (y-axis) corresponds to the best performance on the test set of any entry submitted to ILSVRC2012-2014 (including entries with additional training data). The test set has remained the same over these three years. We see that accuracy tends to increase as the objects get bigger in the image. However, it is clear that far from all the variation in accuracy on these classes can be accounted for by scale alone.

- **Real-world size:** The image classification and single-object localization “optimistic” models performs better on large and extra large real-world objects than on smaller ones. The “optimistic” object detection model surprisingly performs better on extra small objects than on small or medium ones.
- **Deformability within instance:** The “optimistic” model on each of the three tasks performs statistically significantly better on deformable objects compared to rigid ones. However, this effect disappears when analyzing natural objects separately from man-made objects.
- **Amount of texture:** The “optimistic” model on each of the three tasks is significantly better on objects with at least low level of texture compared to untextured objects.

These and other findings are justified and discussed in detail below.

Experimental setup. We observed in Section 7.5.4 that objects that occupy a larger area in the image tend to be somewhat easier to recognize. To make sure that differences in object scale are not influencing results in this section, we normalize each bin by object scale. We discard object classes with the largest scales from each bin as needed until the average object scale of object classes in each bin across one property is the same (or as close as possible). For real-world size property for example, the resulting average object scale in each of the five bins is 31.6% – 31.7% in the image classification and single-object localization tasks, and 12.9% – 13.4% in the object detection task.⁶

⁶For rigid versus deformable objects, the average scale in each bin is 34.1%–34.2% for classification and localization, and 13.5%–13.7% for detection. For texture, the average scale in each of the four bins is 31.1%–31.3% for classification and localization, and 12.7% – 12.8% for detection.

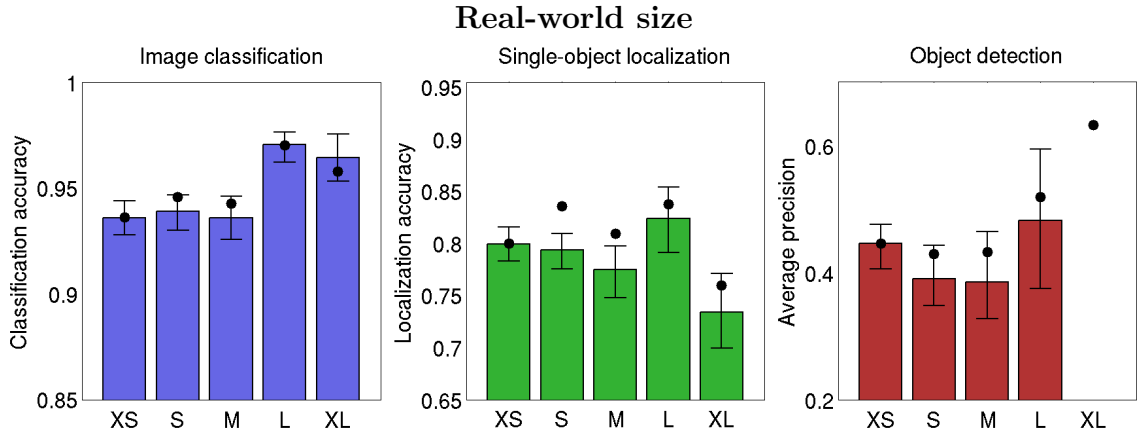


Figure 7.13: Performance of the “optimistic” computer vision model as a function of object properties. The x-axis corresponds to object properties annotated by human labelers for each object class, described in Section 7.3 and illustrated in Figure 7.1. The y-axis is the average accuracy of the “optimistic” model. Note that the range of the y-axis is different for each task to make the trends more visible. The black circle is the average accuracy of the model on all object classes that fall into each bin. We control for the effects of object scale by normalizing the object scale within each bin (details in Section 7.5.5). The color bars show the model accuracy averaged across the remaining classes. Error bars show the 95% confidence interval obtained with bootstrapping. Some bins are missing color bars because less than 5 object classes remained in the bin after scale normalization. For example, the bar for XL real-world object detection classes is missing because that bin has only 3 object classes (airplane, bus, train) and after normalizing by scale no classes remain. Continued in Figures 7.14 and 7.15.

Figures 7.13-7.15 show the average performance of the “optimistic” model on the object classes that fall into each bin for each property. We analyze the results in detail below. Unless otherwise specified, the reported accuracies below are after the scale normalization step.

To evaluate statistical significance, we compute the 95% confidence interval for accuracy using bootstrapping: we repeatedly sample the object classes within the bin with replacement, discard some as needed to normalize by scale, and compute the average accuracy of the “optimistic” model on the remaining classes. We report the 95% confidence intervals (CI) in parentheses.

Real-world size. In Figure 7.13(left) we observe that in the image classification task the “optimistic” model tends to perform significantly better on objects which are larger in the real-world. The classification accuracy is 93.6% – 93.9% on XS, S and M objects compared to 97.0% on L and 96.4% on XL objects. Since this is after normalizing for scale and thus can’t be explained by the objects’ size in the image, we conclude that either (1) larger real-world objects are easier for the model to recognize, or (2) larger real-world objects usually occur in images with very distinctive backgrounds.

Deformability within instance

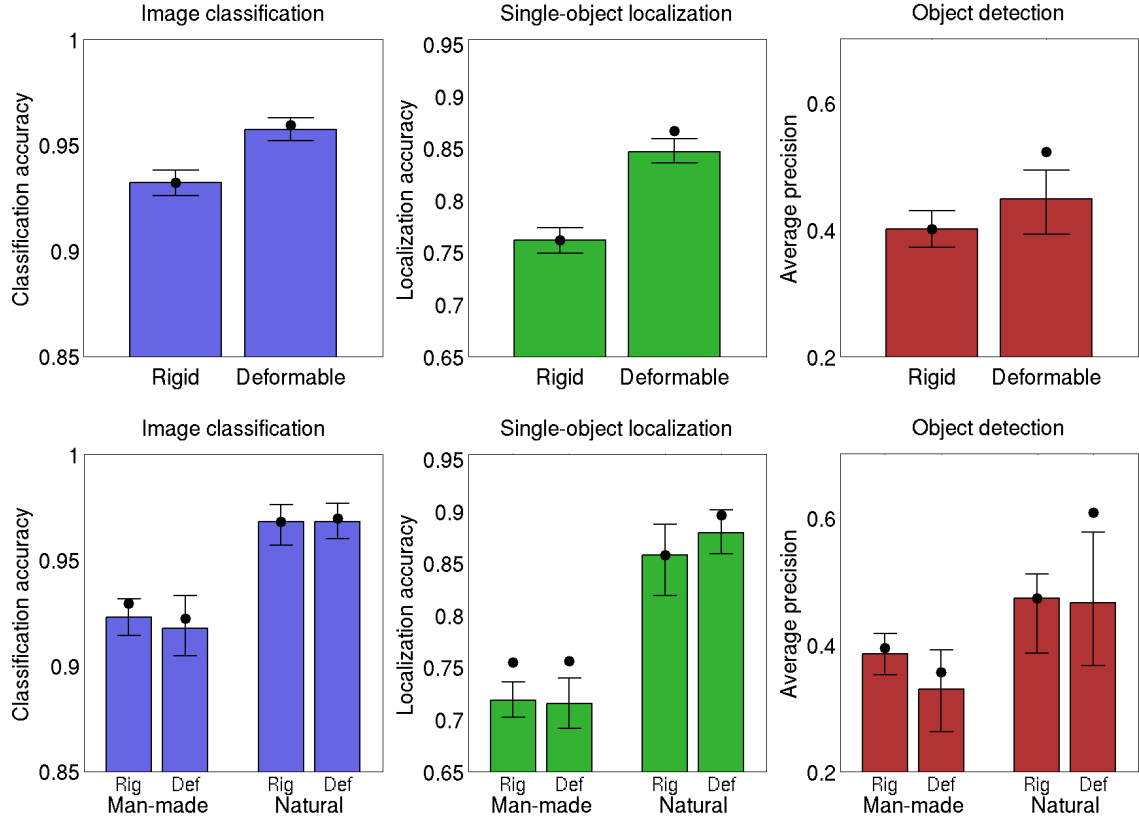


Figure 7.14: Continuation of Figure 7.13. Please see its caption for details.

To distinguish between the two cases we look at Figure 7.13(middle). We see that in the single-object localization task, the L objects are easy to localize at 82.4% localization accuracy. XL objects, however, tend to be the hardest to localize with only 73.4% localization accuracy. We conclude that the appearance of L objects must be easier for the model to learn, while XL objects tend to appear in distinctive backgrounds. The image background make these XL classes easier for the image-level classifier, but the individual instances are difficult to accurately localize. Some examples of L objects are “killer whale,” “schooner,” and “lion,” and some examples of XL objects are “boathouse,” “mosque,” “toyshop” and “steel arch bridge.”

In Figure 7.13(right) corresponding to the object detection task, the influence of real-world object size is not as apparent. One of the key reasons is that many of the XL and L object classes of the image classification and single-object localization datasets were removed in constructing the detection dataset (Section 6.2) since they were not basic categories well-suited for detection. There were only 3 XL object classes remaining in the dataset (“train,” “airplane” and “bus”), and none after scale normalization. We omit them from the analysis. The average precision of XS, S, M objects

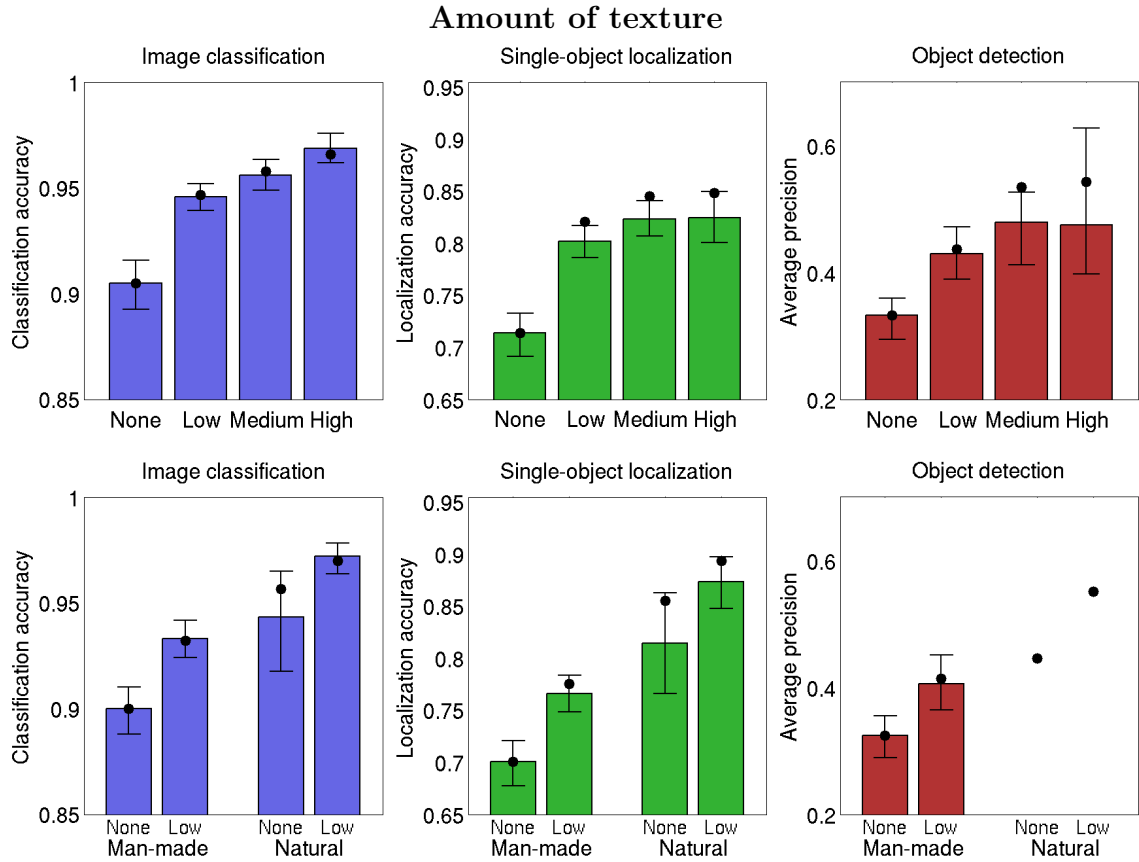


Figure 7.15: Continuation of Figures 7.13 and 7.14. Please see Figure 7.13 for details.

(44.5%, 39.0%, and 38.5% mAP respectively) is statistically insignificant from average precision on L objects: 95% confidence interval of L objects is 37.5% – 59.5%. This may be due to the fact that there are only 6 L object classes remaining after scale normalization; all other real-world size bins have at least 18 object classes.

Finally, it is interesting that performance on XS objects of 44.5% mAP (CI 40.5% – 47.6%) is statistically significantly better than performance on S or M objects with 39.0% mAP and 38.5% mAP respectively. Some examples of XS objects are “strawberry,” “bow tie” and “rugby ball.”

Deformability within instance. In Figure 7.14(top) it is clear that the “optimistic” model performs statistically significantly worse on rigid objects than on deformable objects. Image classification accuracy is 93.2% on rigid objects (CI 92.6% – 93.8%), much smaller than 95.7% on deformable ones. Single-object localization accuracy is 76.2% on rigid objects (CI 74.9% – 77.4%), much smaller than 84.7% on deformable ones. Object detection mAP is 40.1% on rigid objects (CI 37.2% – 42.9%), much smaller than 44.8% on deformable ones.

We can further analyze the effects of deformability after separating object classes into “natural” and “man-made” bins based on the ImageNet hierarchy. Deformability is highly correlated with whether the object is natural or man-made: 0.72 correlation for image classification and single-object localization classes, and 0.61 for object detection classes. Figure 7.14(bottom) shows the effect of deformability on performance of the model for man-made and natural objects separately.

Man-made classes are significantly harder than natural classes: classification accuracy 92.8% (CI 92.3% – 93.3%) for man-made versus 97.0% for natural, localization accuracy 75.5% (CI 74.3% – 76.5%) for man-made versus 88.5% for natural, and detection mAP 38.7% (CI 35.6 – 41.3%) for man-made versus 50.9% for natural. However, whether the classes are rigid or deformable within this subdivision is no longer significant in most cases. For example, the image classification accuracy is 92.3% (CI 91.4% – 93.1%) on man-made rigid objects and 91.8% on man-made deformable objects – not statistically significantly different.

There are two cases where the differences in performance *are* statistically significant. First, for single-object localization, natural deformable objects are easier than natural rigid objects: localization accuracy of 87.9% (CI 85.9% – 90.1%) on natural deformable objects is higher than 85.8% on natural rigid objects – falling slightly outside the 95% confidence interval. This difference in performance is likely because deformable natural animals tend to be easier to localize than rigid natural fruit.

Second, for object detection, man-made rigid objects are easier than man-made deformable objects: 38.5% mAP (CI 35.2% – 41.7%) on man-made rigid objects is higher than 33.0% mAP on man-made deformable objects. This is because man-made rigid objects include classes like “traffic light” or “car” whereas the man-made deformable objects contain challenging classes like “plastic bag,” “swimming trunks” or “stethoscope.”

Amount of texture. Finally, we analyze the effect that object texture has on the accuracy of the “optimistic” model. Figure 7.15(top) demonstrates that the model performs better as the amount of texture on the object increases. The most significant difference is between the performance on untextured objects and the performance on objects with low texture. Image classification accuracy is 90.5% on untextured objects (CI 89.3% – 91.6%), lower than 94.6% on low-textured objects. Single-object localization accuracy is 71.4% on untextured objects (CI 69.1% – 73.3%), lower than 80.2% on low-textured objects. Object detection mAP is 33.2% on untextured objects (CI 29.5% – 35.9%), lower than 42.9% on low-textured objects.

Texture is correlated with whether the object is natural or man-made, at 0.35 correlation for image classification and single-object localization, and 0.46 correlation for object detection. To determine if this is a contributing factor, in Figure 7.15(bottom) we break up the object classes into natural and man-made and show the accuracy on objects with no texture versus objects with low texture. We observe that the model is still statistically significantly better on low-textured object

classes than on untextured ones, both on man-made and natural object classes independently.⁷

7.6 Conclusions

With the growth of object recognition datasets, we are able to analyze the performance of algorithms with an unprecedented level of details. While looking at average accuracy across hundreds or thousands of object categories is certainly a good way to track improvement in generic object recognition, analyzing the performance in more detail can also be very enlightening.

For example, we conclude that object detectors perform well on natural textured objects. However, man-made untextured objects remain very challenging for current state-of-the-art algorithms. On the other hand, a lot of attention has been spent on building models for deformable objects – the “deformable parts model” has been the state-of-the-art for multiple years prior to 2012 [54]. It now appears that deformability within instance is no longer a challenge for current algorithms.

These insights can serve multiple purposes. First, we can focus our object recognition research efforts specifically on the more challenging objects. It might require developing some novel insights into better object representation or improved modeling specifically for recognizing some specific classes of objects. Combined with existing models, this can be a significant step forward for object detection. Second, we can focus our dataset collection efforts. For example, there may not be a need to annotate more cats in images whereas creating a new large-scale dataset for man-made tools could greatly benefit the community. Finally, we can be aware that object detection models are not very accurate on certain types of objects and can build in safeguards (for example, with a human-in-the-loop verification system) to ensure that detection accuracy remains sufficiently high when building applications.

⁷Natural object detection classes are removed from this analysis because there are only 3 and 13 natural untextured and low-textured classes respectively, and none remain after scale normalization. All other bins contain at least 9 object classes after scale normalization.

Chapter 8

Human-machine collaboration for object annotation

8.1 Introduction

The field of large-scale object detection has leaped forward in the past few years [66, 152, 35, 165, 205, 81, 190], with significant progress both in techniques [66, 152, 190, 165] as well as scale: hundreds of thousands of object detectors can now be trained directly from web data [26, 41, 35]. The object detection models are commonly evaluated on benchmark datasets [152, 48], and achievements such as 1.9x improvement in accuracy between year 2013 and 2014 on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [152] are very encouraging. However, taking a step back, we examine the performance of the state-of-the-art RCNN object detector trained on ILSVRC data [66] on the image of Figure 8.1: only the 6 green objects out of the 100 annotated objects have been correctly detected.

The question we set out to address is: what can be done to efficiently and accurately detect all objects in an image given the current object detectors? One option is by utilizing the existing models for total scene understanding [115, 209, 114] or for modeling object context [213, 39, 160, 172]. However, this is still currently not enough to go from detecting 6 to detecting 100 objects.

Our answer is to put humans in the loop. The field of crowd engineering has provided lots of insight into human-machine collaboration for solving difficult problems in computing such as protein folding [141, 28], disaster relief distribution [62] and galaxy discovery [120]. In computer vision with human-in-the-loop approaches, human intervention has ranged from binary question-and-answer [19, 199, 200] to attribute-based feedback [140, 138, 107] to free-form object annotation [198]. For understanding all objects in an image, one important decision is which questions to pose to humans. Binary questions are not sufficient. Asking humans to draw bounding boxes is expensive:

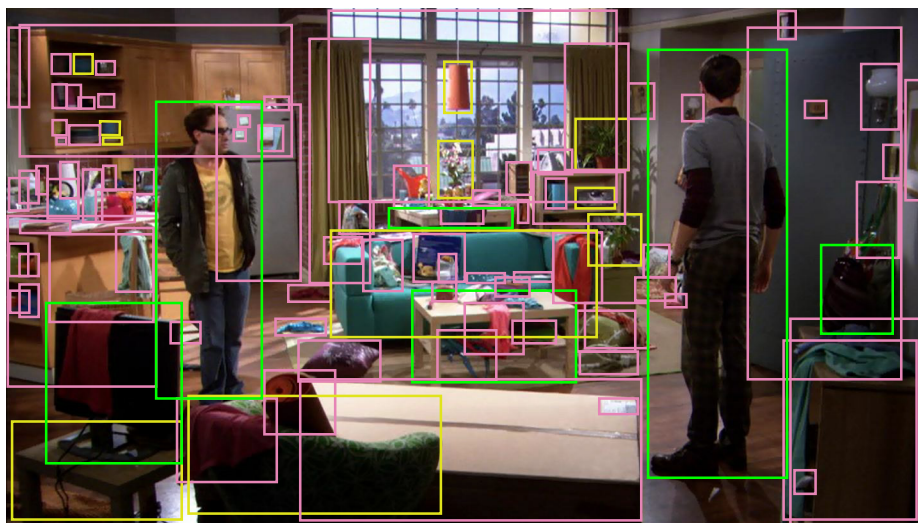


Figure 8.1: This cluttered image has 100 annotated objects shown with green, yellow and pink boxes. The green boxes correspond to the 6 objects correctly detected by the state-of-the-art RCNN model [66] trained on the ILSVRC dataset [152]. (The about 500 false positive detections are not shown.) Yellow boxes loosely correspond to objects that are annotated in current object detection datasets such as ILSVRC. The majority of the objects in the scene (shown in pink) are largely outside the scope of capabilities of current object detectors. We propose a principled human-in-the-loop framework for efficiently detecting all objects in an image.

obtaining an accurate box around a single object takes between 7 seconds [91] to 42 seconds [175], and with 23 objects in an average indoor scene [71] the costs quickly add up. Based on insights from object detection dataset construction [118, 152], it is best to use a variety of human interventions; however, trading off accuracy and cost of annotation becomes a challenge.

We develop a principled framework integrating state-of-the-art scene understanding models [66, 104, 3, 71] with state-of-the-art crowd engineering techniques [152, 118, 33, 167, 95] for detecting objects in images. We formulate the optimization as a Markov Decision Process. Our system:

1. **Seamlessly integrates computer and human input**, accounting for the imperfections in both. [19, 91] One key component, in contrast to prior work, is the incorporation of feedback from multiple types of human input and from multiple computer vision models.
2. **Automatically trades off density, precision and cost of annotation** in a principled framework.
3. **Is open-world**, by integrating novel types of scenes and objects instead of relying only on information available in a limited training set.
4. **Is light-weight and easily extensible**. The framework is able to continuously incorporate the latest computer vision and crowd engineering innovations.

We provide insights into seven types of human interventions tasks using data collected from Amazon Mechanical Turk, and experimentally verify that our system effectively takes advantage of multiple sources of input for localizing objects in images while accurately self monitoring.

8.2 Related work

Recognition with humans in the loop. Among the most similar works to ours is the approaches which combine computer vision with human-in-the-loop collaboration for tasks such as fine-grained image classification [19, 199, 37, 200], image segmentation [91], attribute-based classification [103, 140, 15], image clustering [107], image annotation [192, 193, 168], and human interaction [99] and object annotation in videos [198]. Methods such as [19, 199, 37, 200] jointly model human and computer uncertainty and characterize human time versus annotation accuracy, but only incorporate a single type of human response. Works such as [91, 38, 192] use multiple modalities of human feedback, with varying costs, and accurately model and predict the success of each modality. However, they do not incorporate iterative improvement in annotation. We build upon these approaches to integrate multiple human annotation modalities with state-of-the-art computer vision models in an iterative framework for the challenging object annotation task.

Better object detection. Methods have been developed for training better object detection models with weakly supervised data [147, 76, 179, 26, 81, 41]. Active learning approaches has been developed to improve object detectors with minimal human annotation cost during training [103, 194]. Some object detection frameworks even automatically mine the web for object names and exemplars [26, 35, 41]. All of these approaches can be plugged into our framework to reduce the need for human annotation by substituting more accurate automatic detections.

Cheaper manual annotation. Manual annotation is becoming cheaper and more effective through the development of crowdsourcing techniques such as annotation games [197, 37, 98], tricks to reduce the annotation search space [38, 17], more effective user interface design [175, 198], making use of existing annotations [18], making use of weak human supervision [91, 25] and accurately computing the number of required workers [167]. These innovations are important in our framework for minimizing the cost of human annotation when it is needed to augment computer vision. Approaches such as [33, 167, 95, 206] use iterative improvement to perform a task with accuracy per unit of human cost. We draw upon these works to provide human feedback in the most effective way.

8.3 Problem formulation

We present a policy for efficiently and accurately detecting objects in a given image. The input to the system is an image to annotate and a set of annotation constraints. The output is a set

of bounding box annotations with object names. For the rest of this chapter, we distinguish the requester (the person who wants the image annotated) from the users (the people doing the human annotation tasks).

The requester may specify up to two of three constraints:

1. **Utility.** In the simplest case, utility of a labeling corresponds to the number of objects. However, since some objects in the image may be more important than others [174, 88, 10], the requester may optionally specify a function mapping each image region and class label to a real value indicating importance. The requester then specifies the minimum total utility of the labels.
2. **Precision.** When the system returns N bounding box annotations with object names, if N_C of them are correct detections, then precision is $\frac{N_C}{N}$. The requester can specify the minimum required level of precision.
3. **Budget.** In our formulation, budget corresponds to cost of human time although methods such as [198] can be applied to also incorporate CPU cost.

On one end of the spectrum the requester can set the maximum budget to zero, and obtain the best automatic annotation of the image. On the other end she can set an infinite budget but specify 100% desired precision and 17 annotated objects per image, which will produce a policy for detailed annotation similar to that of the SUN dataset [213].

8.4 Method

The system uses both computer vision and user input to annotate objects in images subject to provided constraints (Figure 8.2). It alternates between getting user feedback and updating the image probabilities. Section 8.4.1 formalizes the requester constraints. Section 8.4.2 presents the core of our system: the selection of optimal human questions.

The later Section 8.5 describes the probabilistic framework for combining computer vision with human input.

8.4.1 Annotation evaluation

Let $\mathcal{Y} = (B_i, C_i, p_i)\}_{i=1}^N$ be the set of N object detections, each with bounding box B_i , class label c_i , and probability of detection being correct p_i . We now explain how our human-in-the-loop system evaluates \mathcal{Y} and outputs the final annotation according to requester constraints.

The **expected precision** of any labeling $Y \subseteq \mathcal{Y}$ is

$$\mathbb{E}[\text{Precision}(Y)] = \frac{\mathbb{E}[\text{NumCorrect}(Y)]}{|Y|} = \frac{\sum_{i \in Y} p_i}{|Y|} \quad (8.1)$$

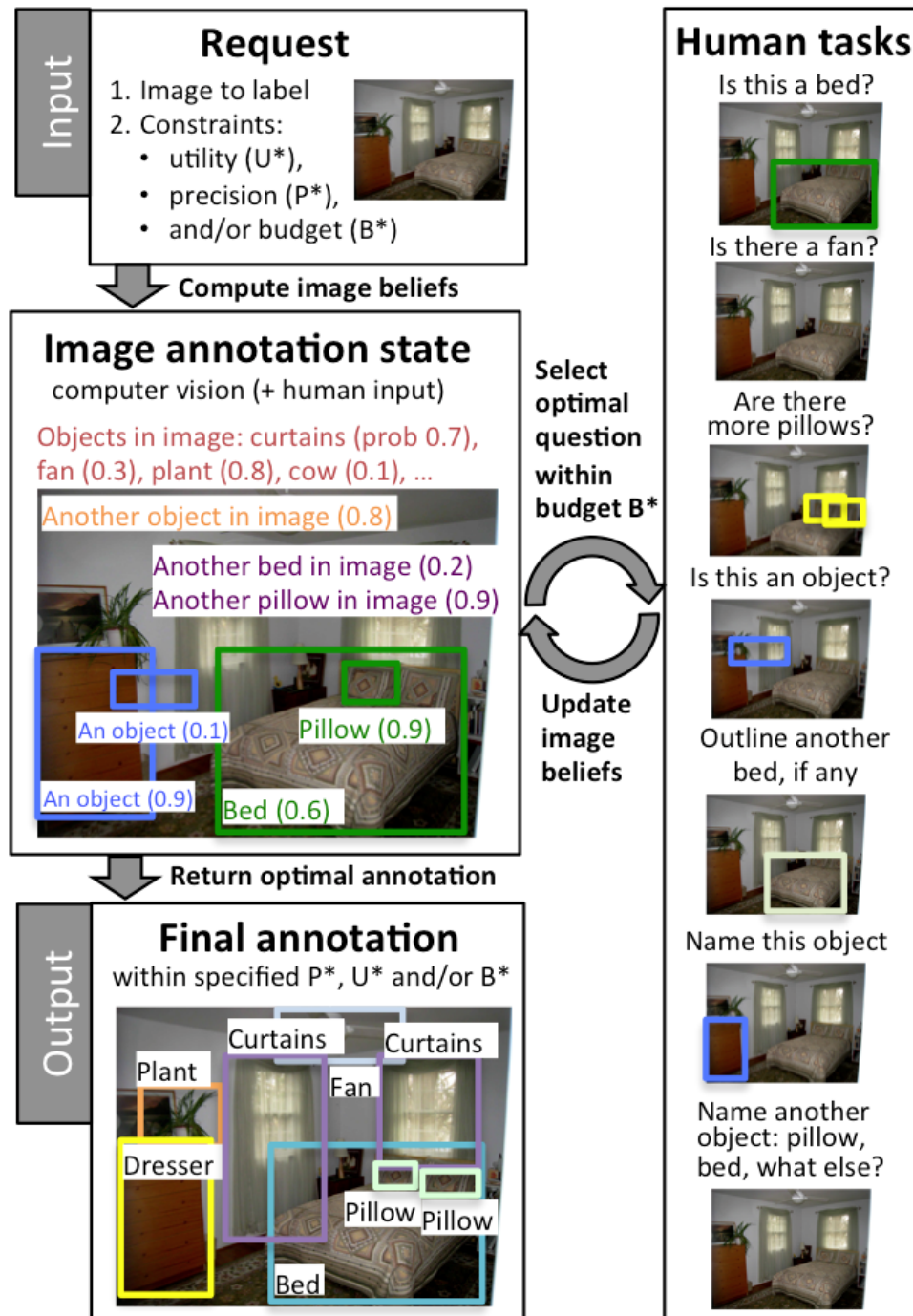


Figure 8.2: Overview of our system. Given a request for annotating an image, the system alternates between updating the image annotation and soliciting user feedback through human tasks. Upon satisfying the requester constraints, it terminates and returns a image with a set of bounding box annotations.

using the linearity of expectation. Similarly, given the requester provided utility function $f : \mathcal{B} \times \mathcal{C} \rightarrow [0, 1]$ mapping the set of bounding boxes with class labels to how much the requester cares about this label, the **expected utility** is

$$\mathbb{E}[\text{Utility}(Y)] = \sum_{i \in Y} p_i f(B_i, C_i) \quad (8.2)$$

The simplest case (used in this chapter) is valuing all detections equally at $f(B, C) = 1 \forall B, C$, making utility equal to the number of correct detections.

Annotation given constraints. Given the available set \mathcal{Y} , the system tries to output a labeling Y that satisfies the requester constraints. Recall that requester specified at most two of the three constraints of utility, precision and budget. If both target utility U^* and precision P^* are requested, the system samples detections from \mathcal{Y} into Y in decreasing order of probability while $\mathbb{E}[\text{Precision}(Y)] \geq P^*$. We define $\text{Precision}(\emptyset) = 1$ so this is always achievable. Since expected utility increases with every additional detection, this will correspond to the highest utility set Y under precision constraint P^* . If $\mathbb{E}[\text{Utility}(Y)] \geq U^*$, the constraints are satisfied. If not, we continue the labeling system.

If target precision P^* (or utility U^*) and budget B^* are specified, then we run the annotation system of Section 8.4.2 until budget is depleted, and produce the set Y as above under the precision constraint P^* (or utility constraint U^*).

Approximation of annotation quality. As the annotation system progresses, it needs to evaluate the quality of annotation set \mathcal{Y} . One option is to directly evaluate how closely \mathcal{Y} satisfies requester constraints: for example, by producing the set $Y \subseteq \mathcal{Y}$ which satisfies the requested level of precision P^* and using $\mathbb{E}[\text{Utility}(Y)]$ as the objective. However, this measure is discontinuous and difficult to optimize. Since precision and utility are closely related, we directly use $\mathbb{E}[\text{Utility}(\mathcal{Y})]$ as the objective.

8.4.2 MDP formulation for human task selection

The main component of our approach is automatically selecting the right human question to best improve the image annotation state. We quantify the tradeoff between cost and accuracy of annotation by formulating it as a Markov decision process (MDP). [97, 33, 167, 95, 67] An MDP consists of states \mathcal{S} , actions \mathcal{A} , conditional transition probabilities \mathcal{P} , and expected rewards of actions \mathcal{R} .

States. At each time period of the MDP, the environment is in some state $S \in \mathcal{S}$. In our case, a state S is our set of current beliefs about the image I , computed by combining computer vision

Human tasks (MDP actions)
Verify-box: is box B tight around an instance of class C ?
Verify-image: does the image contain an object of class C ?
Verify-cover: are there more instance of class C not covered by the set of boxes \mathcal{B} ?
Draw-box: draw a new instance of class C not already in set of boxes \mathcal{B} .
Name-image: Name an object class in the image besides the known classes \mathcal{C} .
Verify-object: is box B tight around <i>some</i> object?
Name-box: If box B is tight around an object other than the objects in \mathcal{C}_B , name the object.

Table 8.1: Human annotations tasks. One important property of our model is that it will automatically find the best question to pose, so there’s no harm in adding extra tasks.

models with user input. For simplicity, in this work we don’t update the computer vision models as annotation progresses on a single image, so the only dynamic part of \mathcal{S} in the user input U .

Actions. In an MDP, the system takes an action $a \in \mathcal{A}$ from state s , which causes the environment to transition to state s' with probability $\mathcal{P}(s'|s, a)$. In our setting, the set of actions \mathcal{A} correspond to the set of human questions that the system can ask. The types of human tasks are listed in Table 8.1. Each question is one of the tasks grounded to the image: for example, “verify-box: is box at (10, 50, 37, 89) an instance of class cat?” or “draw-box: draw a box around another instance of table besides (83, 119, 74, 281) and (281, 470, 46, 24)”. Figure 8.3 shows some example UIs.

Transition probabilities. As a result of an action a from state s , the system moves into a new state s' ; in other words, the current beliefs about the image get updated by the addition of a new user response u_t to \mathcal{U} . Transition probabilities correspond to our expectations on the outcome (user response) of the question a (Section 8.5).

Rewards. After transitioning from state s to s' through action a , the agent in an MDP receives a reward with expected value $\mathcal{R}_a(s, s')$. In our case, the states contain object detection annotations $\mathcal{Y}(s)$ and $\mathcal{Y}(s')$ respectively (Section 8.5) with detection probabilities computed in Section 8.5. Using the definition of Section 8.4.1, the reward is

$$\mathcal{R}_a(s, s') = \frac{\mathbb{E}[\text{Utility}(\mathcal{Y}(s'))] - \mathbb{E}[\text{Utility}(\mathcal{Y}(s))]}{\text{cost}(a)} \quad (8.3)$$

We treat budget constraint as rigid, so $\mathcal{R}_a(s, s') = -\text{inf}$ if the $\text{cost}(a)$ is less than the remaining budget. The system terminates once $\mathcal{Y}(s)$ satisfies the requester constraints.



Figure 8.3: Three of the user interfaces for our human annotation tasks; others are in Appendix E.

Optimization. Given the transition probabilities and expected rewards, at each step the system chooses the action $a^*(s)$ that maximizes $V(s)$, computed recursively as

$$\begin{aligned}
 a^*(s) &= \arg \max_a \left\{ \sum_{s'} P_a(s, s') (R_a(s, s') + V(s')) \right\} \\
 V(s) &= \sum_{s'} P_{a^*(s)}(s, s') (R_{a^*(s)}(s, s') + V(s'))
 \end{aligned}
 \tag{8.4}$$

We optimize Equations 8.4 with 2 steps of lookahead to choose the next action. [33] This is often sufficient in practice and dramatically reduces the computational cost.¹

8.5 Human-in-the-loop probabilistic framework

Our system is based on combining computer vision with human input into one probabilistic framework. Consider the MDP on image I at time step T , after $T - 1$ actions $a_1 \dots a_{T-1}$ were taken and user responses $\mathcal{U}_{T-1} = \{u_t\}_{t=1}^{T-1}$ obtained. We need to compute two closely related quantities: MDP transition probabilities and object detection probabilities for the labeling. We begin by describing the former, and then show how the latter is a special case.

MDP transition probabilities. The MDP is now in state s and we set out to compute the transition probabilities to new states. The next state is uniquely determined by the action (i.e., question) a_T that the system chooses to ask and by the user response u_T . Thus for each a_T we need to compute the probability of each response u_T given the image I and user responses so far \mathcal{U}_{T-1} . Let $E_1^T \dots E_K^T$ be the set of possible answers to this question a_T . By law of total probability and

¹Doing 1 step of lookahead is not sufficient because some tasks in Table 8.1 (e.g., name-image) do not directly influence the labeling.

Bayes' rule:

$$P(u_T|I, \mathcal{U}_{T-1}) = \sum_{k=1}^K P(u_T|E_k^T)P(E_k^T|I, \mathcal{U}_{T-1}) \quad (8.5)$$

The first term of Eqn. 8.5 is $P(u_T|E_k^T)$, which corresponds to the probability of user giving an answer u_T if E_k^T were the correct answer to a_T .² We simplified this term from $P(u_T|E_k^T, I, \mathcal{U}_{T-1})$ by making two assumptions following [19]: (1) noise in user responses is independent of image appearance I if correct answer E_k^T is given, and (2) user responses are independent of each other. To compute $P(u_T|E_k^T)$ we will use the empirical error rates of Section 8.6.2 for each question type in Table 8.1.

The second term of Eqn. 8.5 is $P(E_k^T|I, \mathcal{U}_{T-1})$, which corresponds to the probability of answer E_k^T to a_T in fact being correct. Applying Bayes' rule again:

$$P(E_k^T|I, \mathcal{U}_{T-1}) \propto P(E_k^T|I) \prod_{t=1}^{T-1} P(u_t|E_k^T, I, \mathcal{U}_{t-1}) \quad (8.6)$$

Here, $P(E_k^T|I)$ is the computer vision model for E_k^T (described in Section 8.5.1). $P(u_t|E_k^T, I, \mathcal{U}_{t-1})$ is very similar to the first term in Eqn. 8.5 with one important exception: it unifies user response u_t to action a_t at time t with potential answer E_k^T to action a_T at time T . We consider two cases.

Case 1: The correct response to question a_t at time t can be inferred from E_k^T at time T . For example, suppose a_t is “is there an object of class c_i in the image?” and E_k^T is “box B_i is tight around an instance of class c_i .” Then the correct response “yes” to a_t can be inferred from E_k^T . Section 8.5.2 provides a complete list of these relationships. Let the inferred correct answer be E_m^t . In this case, we again apply the model of [19] to simplify $P(u_t|E_k^T, I, \mathcal{U}_{t-1}) = P(u_t|E_m^t)$ as above.

Case 2: The correct response to question a_t at time t is independent of E_k^T at time T . For example, suppose a_t is as above “is there an object of class c_i in the image?” but E_k^T is “box B_j is tight around an instance of class c_j .” Since E_k^T does not provide any information regarding the correct response to a_t , we know $P(u_t|E, I, \mathcal{U}_{t-1}) = P(u_t|I, \mathcal{U}_{t-1})$.³ To compute this we apply Eqn. 8.5.

This concludes the transition probability computation for every action a_T and every possible user response u_T . New actions can seamlessly be added to the MDP framework assuming the set of possible answers E_1, \dots, E_k , the computer vision probabilities $P(E_k|I)$ and the user error probabilities $P(u|E_k)$ can be computed for each new action type.

Object detection probabilities. In addition to transition probabilities, we also need to compute the object detection probabilities to be used in the image labeling. For a detection with bounding

²There are only a few possible answers to each question which allows us to enumerate them. For example, if the user is asked to perform the draw-box task at time T , the only possible answers are E_1^T : the user draws a box, or E_2^T : the user stated that no box can be drawn.

³Alternatively, our model can be extended to include object-object co-occurrence information here.

box B and class label c , let \hat{E} be “ B is a tight box around an instance of class c .” The probability of the detection being correct given the information available at time T is $P(\hat{E}|I, \mathcal{U}_{T-1})$. This is computed directly with Eqn. 8.6, with $P(\hat{E}|I)$ from an object detector.

One extra consideration is that (B, c) can be automatically proposed by the object detector or manually by the users.⁴ If the box B was manually drawn at some previous time \hat{T} in response to the draw-box task, then we omit the computer vision model and modify Eqn. 8.6 slightly to

$$P(\hat{E}|I, \mathcal{U}_{T-1}) \propto P(\hat{E}|u_{\hat{T}}) \prod_t P(u_t|\hat{E}, I, \mathcal{U}_{t-1}) \quad (8.7)$$

with the product ranging over $t \in \{1, \dots, T-1\} - \hat{T}$. $P(\hat{E}|u_{\hat{T}})$ is the empirical user accuracy rate for the drawn bounding box to actually be *correct* (Section 8.6.2).

8.5.1 Incorporating computer vision input

We incorporate multiple computer vision models into our system to compute the above transition probabilities for all actions of Table 8.1:

(1) Detection. Computing transition probabilities corresponding to the *verify-box* action with box B and class C requires computing $P(\text{det}(B, C)|I)$: the probability that B is tight around an instance of class C on image I . Standard object detectors can be used here e.g., [66, 54, 81].

(2) Classification. Similarly, the *verify-image* action for object class C require computing $P(\text{cls}(C)|I)$ that C is present in the image. Models such as [104, 177] can be used.

(3) Another instance. Computing transition probabilities for *verify-cover* and *draw-box* actions for class C and set of boxes \mathcal{B} require computing $P(\text{more}(\mathcal{B}, C)|I)$ that there are other instances of C in the image beyond those contained in \mathcal{B} . We compute this probability using an empirical *distribution on number of object instances* in images. It provides the probability $P(\text{more}|n)$ of there being more instances of an object class given the image is known to contain at least n instances of this class. Let $\text{nc}(\mathcal{B}, C)$ be the number of boxes \mathcal{B} that are correct for class C , then $\mathbb{E}[\text{nc}(\mathcal{B}, C)] = \sum_{B \in \mathcal{B}} P(\text{det}(B, C)|I)$. Rounding $n := \mathbb{E}[\text{nc}(\mathcal{B}, C)]$ to the nearest integer, we compute

$$P(\text{more}(\mathcal{B}, C)|I) = \begin{cases} P(\text{cls}(C)|I) & \text{if } n = 0 \\ P(\text{more}|n) & \text{else} \end{cases} \quad (8.8)$$

(4) Another class. Similarly, a *name-image* action requires computing the probability $P(\text{morecls}(C|I))$ that another object class is present in the image beyond the classes \mathcal{C} . An empirical *distribution on number of object classes* is used as above.

⁴We use the same reasoning for name-image and name-box tasks.

(5) Objectness. A *verify-object* action requires computing $P(\text{obj}(B))$ that bounding box B is tight around *some* object. Models such as [3] can be used.

(6) New object. Finally, transition probabilities for a *name-box* action requires $P(\text{new}(B, \mathcal{C})|I)$ for a bounding box B as the probability that there is an object in this box which has not yet been named in the current set of classes \mathcal{C} . Assuming independence:

$$P(\text{new}(B, \mathcal{C})|I) = P(\text{obj}(B)|I) \prod_{C \in \mathcal{C}} (1 - P(\text{det}(B, C)|I)) \quad (8.9)$$

Adding new human tasks in Table 8.1 would likely require the addition of new computer vision models.

8.5.2 Incorporating user input

Our set of user inputs \mathcal{U} contains multiple types of information. Our goal is to estimate $P(u_t|E_k^T)$ where u_t is a user response to some question a_t and E_k^T is a fact about the image related to some other question a_T (in particular Eqn. 8.5).

Types of events E_k^T . Based on the tasks described in Table 8.1, we consider 5 types of E_k^T :

1. $\text{det}(B, C)$ for whether box B is correct around an instance of class C (**verify-box** task)
2. $\text{cls}(C)$ for whether class C is present in the image (**verify-image** task)
3. $\text{more}(\mathcal{B}, C)$ for whether there are more instances of class C besides those in boxes \mathcal{B} (**verify-cover** and **draw-box** tasks)
4. $\text{morecls}(\mathcal{C})$ for whether there are more object classes in the image \mathcal{C} (for **name-image** task)
5. $\text{obj}(B)$ for whether box B is a tight box around *some* object (for **verify-object** and **name-box** tasks)

Computing the *true* answer to a question. In order to make a decision about whether the user made a mistake or not in u_t , we first need to determine the true answer to the question a_t given that event E_k^T happened. This is shown in Table 8.2. An event E_k^T sometimes determines the correct answers to more than one question.

For example, consider the event E_k^T of class C not present in the image (this is event “ $\text{cls}(C) = 0$ ” in Table 8.2). The true answer is then determined to be “no” to three questions: (1) when asked if C is contained in the image (verify-image), (2) when asked if C is contained in any box B (verify-box), and (3) when asked if there are more instances of class C in the image (verify-cover). For other questions we have no information about the true answer given the event E_k^T .

Event E_k^T	True answer to each question type a_t given E_k^T				
	Verify-box: is box B for class C ?	Verify-image: is class C in image?	Verify-cover: are there more instances of class C besides in B' ? (same for draw-box)	Name-image: name another object in image besides C' ?	Verify-object: is there an object in box B ? (similarly for name-object)
$\det(B, C) = 1$	✓	✓	✓ if $B \notin B'$	-	✓
$\det(B, C) = 0$	✗	-	-	-	-
$\text{cls}(C) = 1$	-	✓	-	-	-
$\text{cls}(C) = 0$	✗	✗	✗	-	-
$\text{more}(\mathcal{B}, C) = 1$	-	✓	✓ if $B' \subseteq \mathcal{B}$	-	-
$\text{more}(\mathcal{B}, C) = 0$	✗ if $B \notin \mathcal{B}$	-	✗ if $\mathcal{B} \subseteq B'$	-	-
$\text{morecls}(\mathcal{C}) = 1$	-	-	-	✓ if $C' \subseteq \mathcal{C}$	-
$\text{morecls}(\mathcal{C}) = 0$	-	✗ if $C \notin \mathcal{C}$	-	✗ if $\mathcal{C} \subseteq C'$	-
$\text{obj}(B) = 1$	-	-	-	-	✓
$\text{obj}(B) = 0$	✗	-	-	-	✗

Table 8.2: For every event E (row) and question (column), the table reports what the *true* answer to the question if the event E happened. ✓ means “yes” is the true answer, ✗ means “no” is the true answer, and - means that event E provides no information about the true answer. - is the default when not specified. Here every question is treated as a binary question: for example, for **draw-box** question, the answer of drawing a box is simply “yes” and the answer of refusing to draw a box is “no”.

Judging user input as correct, wrong or undecided Having computed the true answers, we can now judge the user input u_t in response to a question a_t . An answer is judged as *wrong* if it doesn’t match the correct answer in Table 8.2. An answer is judged as *correct* if it matches the correct answer in Table 8.2 *and* the event corresponds precisely to the question, as shown by the shaded boxes in Table 8.2. All other answers are judged as *undecided*.

To understand the extra layer of complication with judging correct answers, consider the influence of the question “does box B contain an instance of class C ” (verify-box) on the probability of the object class C being in the image. If the answer to the question is “yes,” then this can only happen if class C is in the image and is certain to be the wrong answer otherwise. There is a direct influence. If the answer is “no,” then it might be correct whether or not class C is in the image. To simplify the computation, we then judge the answer as undecided.

User input accuracy probabilities. Finally, after the answers are all judged as correct, wrong or undecided, we incorporate them back into the probability computation.

Every user input u_t is obtained in response to a question. Each input u_t is then associated with a probability β_t which depends on the average user accuracy rate for this type of question. Let E_0^t correspond to the answer “no” and E_1^t correspond to “yes”. Similarly, let $u_t = 1$ if user says “yes” and $u_t = 0$ if user says “no”. Then $\beta_t = P(u_t = j|E_j^t)$ for $j \in \{0, 1\}$. The empirically obtained error rates $1 - \beta_t$ are reported later in Table 8.3.

We can then compute $P(u_t|E_j^t) = \beta_t$ if answer u_t is *correct*, $P(u_t|E_j^t) = 1 - \beta_t$ if answer u_t is *wrong*, and use the prior from Eqn. 8.5 otherwise.

Special case 1: More instances computation. We briefly note some exceptions to the computation described above. Recall that computing complicated events such as $P(\text{more}(\mathcal{B}, C)|I)$ in Eqn. 8.8 relies on additionally on detection probabilities $P(\text{det}(B, C)|I)$ as well as image classification probabilities $P(\text{cls}(C)|I)$. To effectively utilize all user input \mathcal{U} in this computation, we break up the set \mathcal{U} into \mathcal{U}_1 , which is the set of user input directly relevant to $\text{more}(\mathcal{B}, C)$ and \mathcal{U}_2 , which is all other input. Then have

$$P(\text{more}(\mathcal{B}, C)|I, \mathcal{U}) \propto P(\text{more}(\mathcal{B}, C)|I, \mathcal{U}_2)P(\mathcal{U}_1|\text{more}(\mathcal{B}, C)) \quad (8.10)$$

\mathcal{U}_1 consists of all user input obtained from verify-cover or draw-box tasks that we can judge based on the event $\text{more}(\mathcal{B}, C)$ (Table 8.2). \mathcal{U}_2 is all other user input. We use $P(\text{det}(B, C) = 1|I, \mathcal{U}_2)$ and $P(\text{cls}(B, C) = 1|I, \mathcal{U}_2)$ in this computation, which allows us to successfully incorporate the entire set of user input $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

The computation for $P(\text{morecls}(\mathcal{C}))$ is similar.

Special case 2: New object computation. The computation for $P(\text{new}(B)|I, \mathcal{U})$ also requires breaking up \mathcal{U} into two parts, but in a slightly different way. Generalizing Eqn. 8.9 we have

$$P(\text{new}(B)|I, \mathcal{U}) = P(\text{obj}(B)|I, \mathcal{U}_1) \prod_C (1 - P(\text{det}(B, C)|I, \mathcal{U}_2)) \quad (8.11)$$

with \mathcal{U}_1 is all input from verify-object or name-object tasks related to box $\text{obj}(B)$ (Table 8.2) and $\mathcal{U}_2 = \mathcal{U} - \mathcal{U}_1$. This is to ensure that the independence assumption is not violated.

Special case 3: Open-ended questions For open-ended question such as draw-box we have to consider both the probability that the user draws a box when there is one and that the drawn box is indeed a good box around an object instance. The latter is noted in Eqn. 8.7 as $P(\hat{E}|u_{\hat{T}})$, where $u_{\hat{T}}$ is the fact that user drew the box at time \hat{T} and \hat{E} is fact that the box is correct. We stated that this is computed from user error rates which is true but somewhat subtle: computing this error

rate directly would be influenced by the distribution of positive and negative images shown to the user (positive images being the ones that indeed contain an unannotated instance). To avoid this problem we add an extra variable pos corresponding to the fact that the image is positive, so an unannotated box indeed existed in the image. Then $P(\hat{E}|u_{\hat{T}}) = P(\hat{E}, pos|u_{\hat{T}})$. So

$$P(\hat{E}|u_{\hat{T}}) \propto P(\hat{E}|u_{\hat{T}}, pos)P(u_{\hat{T}}|pos)P(pos) \quad (8.12)$$

Now $P(\hat{E}|u_{\hat{T}}, pos)$ can be estimated as the probability that a bounding box that the user drew on a positive image is indeed correct (reported later in caption of Table 8.3), $P(u_{\hat{T}}|pos)$ is the true positive accuracy for the draw-box task (reported later in Table 8.3), and $P(pos)$ is the current estimate of the image being positive.

8.6 Experiments

We evaluate both the accuracy and cost of our proposed object annotation system that combines multiple computer vision models with multiple types of human input in a principled framework. We begin by describing the experimental setup (Section 8.6.1), then discuss the challenges of designing the variety of human tasks and collecting accurate error rates (Section 8.6.2), showcase the quality of annotation obtained by our system (Section 8.6.3) and conclude by proving that our system is capable of self-monitoring (Section 8.6.4).

8.6.1 Setup

We perform experiments on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) detection dataset [152]. The dataset consists of 400K training images, 20K validation images and 40K test images. The validation and test images are fully annotated with all instances of 200 object classes ranging from accordion to zebra. Since test set annotations are kept hidden by the challenge organizers, we split the validation set into two sets (val1 and val2) and evaluate on val2 following [66]. We use 2216 images of val2 that contain at least 4 ground truth object instances. The average number of instances per image is 7.0 compared to 7.7 of COCO [118], and the average object size is 9.8% of image area compared to 10.5% in SUN [213].

Computer vision input. We use publicly available code and models as computer vision input. The object detectors are pre-trained RCNN models released by [66]. Image classifiers are convolutional neural network (CNN) classifiers trained with Caffe [92] on ILSVRC2013 detection training set (full images, no bounding box labels) [81].

In order to use computer vision models in our framework, we need to obtain accurate probability estimates from the models. The output of object detectors and image classifiers x is commonly

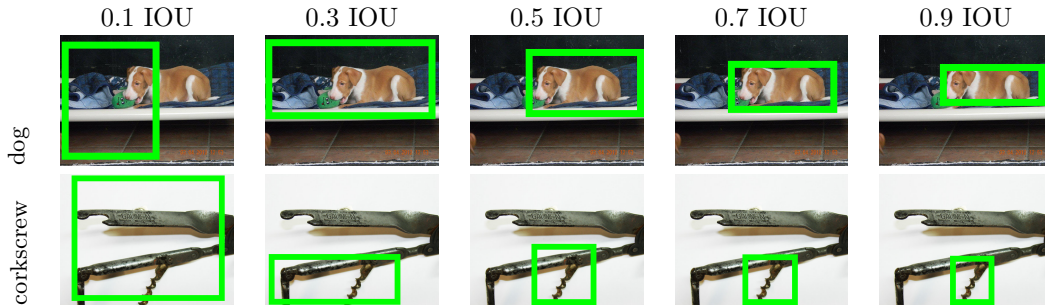


Figure 8.4: Bounding boxes with increasing intersection over union (IOU) with the optimal tight box. Training human annotators to make binary decision on whether or not a bounding box is a good detection is quite difficult; this is the primary contributor to human error rates. Guidance such as “the object occupies more than half the bounding box” is confusing since objects like corkscrews (bottom row) occupy a small area even at perfect IOU.

converted to a probability by fitting a 2-parameter sigmoid

$$p = \frac{1}{1 + \exp(a_1x + a_2)} \quad (8.13)$$

to the CNN output on the ILSVRC val1 set [146]. We bin the detections from all classes into 20 bins in increments of 5% confidence, and compute the error between the expected probability produced by the model (2.5%, 7.5%, 12.5%, etc.) and the actual fraction of positive examples in that bin. The average absolute probability error is 8.7% on ILSVRC val2 set due to the model being overconfident on the high-scoring examples. To compensate, we learn the 3rd parameter $a_3 \in [0, 1]$ with

$$p = \frac{a_3}{1 + \exp(a_1x + a_2)} \quad (8.14)$$

to allow the model to automatically estimate its level of confidence. This reduced the error down to 5.2%. More accurate models such as [164] can also be used. After the conversion, detections and classifications with probability less than 0.1 are discarded.

We use non-maximum suppression on the output of detectors to (1) avoid multiple detections around the same instance, and (2) reduce the computational burden. The probability distribution for $P(\text{more}|n \text{ inst})$ is computed empirically for all classes jointly on the val1 set. The probability $P(\text{morecls}|n \text{ classes})$ is from [71].

Human-computer interaction. Setting up a system that integrates computer vision knowledge with human input requires finding common ground between the two. One necessary decision is what bounding box is considered a correct detection. In object detection, a bounding box is commonly considered correct if its intersection over union (IOU) with a ground truth box is greater than 0.5. [152, 48] However, training humans to visually inspect a bounding box with IOU of 0.3 and

Human task	FP	FN	Cost
Verify-image: class C in image?	0.13	0.02	5.34s
Verify-box: class C in box B ?	0.23	0.07	5.89s
Verify-cover: more boxes of C ?	0.25	0.26	7.57s
Draw-box: draw new box for C	0.28	0.16	10.21s
Verify-object: B some object?	0.29	0.04	5.71s
Name-object: name object in B .	0.25	0.08	9.67s
Name-image: name object in image.	0.02	0.12	9.46s

Table 8.3: Human annotations tasks with the corresponding accuracy rates and costs. Detailed explanations of each task are in Table 8.1. FP column is the false positive probability of user answering “no” (or refusing to draw a box, or write a name) when the answer should in fact be “yes.” For the open-ended tasks, if the answer was given, we also need to estimate the probabilities of the given answer being *wrong*: these probability are draw-box 0.29, name-object 0.06, name-image 0.05. FN column is the true negative probability of the user answering “yes” when the answer should be “no.” Cost is median human time in seconds.

distinguish it from one with IOU 0.5 is surprisingly difficult (Figure 8.4). In our experiments we choose 0.7 as the target IOU as the halfway point between the targets of object detection and human annotation.⁵

The higher IOU further reduces the accuracy of automated object detection, from 34.1% mAP with IOU of 0.5 and non-maximum suppression (nms) of 0.3 as in [66] to 18.7% mAP with IOU of 0.7 and nms of 0.5.

8.6.2 Human annotation design and observations

To compute the expected output of an action (Section 8.4.2) we need to collect user accuracy rates for each human task of Table 8.1. We assume that user error is dependent only on the type of task (for example, on the clarity of instructions or the effectiveness of filtering spam workers) and not on the exact question: i.e., a user is equally likely to misclassify a cat as she is to misclassify a hammer. In these section we describe how to obtain these error rates; we then use them in simulation to evaluate the human-in-the-loop labeling in Section 8.6.3.

Generating positive and negative sets. In order to estimate human error rates, we perturb the annotations from ILSVRC detection val1 set to obtain a representative positive and negative examples. For most tasks, this is straight-forward. For example, for verify-box task (“is B a good box around an instance of class C ?”) we generate the positive set by sampling ground truth boxes for this class and perturbing them to between 0.7 and 1 IOU. We generate the negative set by sampling boxes between 0 and 0.5 IOU, as well as boxes corresponding to other object classes.

⁵When human annotators are used to collect object detection datasets, the average difference in bounding boxes for the same instance between two annotators is about 5 pixels on each side. [152] For an 200x200 pixel object, this corresponds to approximately 0.90 intersection over union.

However, the negative sets of verify-object (“is there an object in box B ?”), name-object (“name the object in box B ”) and name-image (“name another object in this image”) tasks can’t be generated automatically: there are only 200 classes labeled in ILSVRC, and so any randomly generated region or image from ILSVRC might accidentally contain some other object class beyond the annotated 200. Thus, we sampled some likely negative candidates from ILSVRC and asked 4 AMT workers to determine if there is an additional object in there. If 3 or more workers said that the box does not contain an object, we manually verified it to confirm and then included it in the negative set.

The likely negative candidates for verify-object and name-object were simply random regions on the image that had overlap less than 0.3 with any annotated box. The likely negative candidates for name-image were generated by selecting images where the annotated bounding boxes cover more than 90% of the image area.

Quality control. The questions are presented to users in batches of 20-25 questions. Each batch contains 4-5 “gold standard” questions. These are questions which were verified by trusted subjects (previously unfamiliar with our UI) who deemed that the correct answer should be “obvious” to a careful annotator. When deployed, the annotation UI prevents users from submitting their work if they incorrectly answer more than 1 gold standard question.

The UI also had additional sanity checks built in. First, it prevents users from drawing a bounding box around an object instance if it is too close to a known bounding box. Second, it requires that users spend at least 1 second on each question. In our in-house experiments with trusted workers, this was the minimum amount of time necessary to answer a question correctly. This control was implemented since the interface has keyboard shortcuts (1 and 2 to answer “yes” or “no”, and left and right arrows to move between questions) which makes it possible for spammers to potentially blindly answer all 20 questions in just a few seconds (if all questions are binary).

Accuracy and time per question. The accuracy rates and costs (in median human time [38]) are reported in Table 8.3. By far the biggest source of error is getting users to make binary decisions on tasks with a bounding box: the average accuracy is 0.92 for image-level tasks (verify-image and name-image) and 0.81 for the box-level tasks. For the open-ended tasks (draw-box, name-object, name-image) we needed to compute both the probability that the user *answers* the question affirmatively (i.e., attempts to draw a box) as well as as the probability that the user is *correct*. For name-object and name-image we manually verified the responses on 100 images each. Some common mistakes were misclassifications (calling a sheep “goat” or a cello “violin”) and annotations that were too general (e.g., “food”) despite instructions.

Data collection cost. On Amazon Mechanical Turk we pay workers 10 cents to answer 20 questions. Since on average it takes 7.69 seconds (Table 8.3) to answer a question this comes out to \$2.34 per hour.

Task	Cost (seconds)	
	Positive response	Negative response
Verify-image	5.64	4.88
Verify-box	6.22	5.45
Verify-cover	7.47	7.63
Draw-box	12.34	8.67
Verify-object	6.08	5.21
Name-object	12.19	7.09
Name-image	12.53	7.67

Table 8.4: Cost of each task (in median human time) broken down by positive versus negative responses. On average, positive responses take longer than negative ones. One interesting potential extension to our human-machine object annotation framework would be to incorporate this fact.

To simulate the real use case (where different types of questions are automatically generated by our system out of order) we assigned a *random* selection of tasks to each batch. This slowed down the worker responses since it required reading multiple sets of instructions. Some workers even complained to us via email about this.⁶One interesting extension to our current human-machine object annotation framework would be to consider the reduced human cost if asking multiple questions of the same type consecutively.

Finally, we observed that for many types of questions answering positively took longer than answering negatively (especially for the open-ended questions). Table 8.4 documents this. Another potential extension in our framework would be to incorporate this fact when making decisions about the optimal next question to ask the annotators.

8.6.3 Evaluating labeling quality

We evaluate our proposed annotation system in simulation using the human accuracy rates collected in Section 8.6.2 to simulate the real-world labeling scenario. Figure 8.5 shows the average number of objects labeled as a function of budget (human labeling time). For the purposes of simulation, since only the 200 object category names in the image are known, we omit the verify-object and name-object tasks. We reach several conclusions based on these results:

Computer vision and human input are mutually beneficial. The object detectors (*CV only* in Figure 8.5) are able to label on average 0.95 objects per image at zero cost. Human-only annotation (*H only*) starts at zero labeled objects but improves over time. After 30 seconds of annotation, our joint method (*CV+H*) labels 1.5x more objects than the computer vision-only method and 2.8x

⁶Given that the random selection of questions was necessary in our setting, we attempted to at least simplify the process for the labeler as much as possible by arranging the question types in logical order, roughly from “easiest” to “hardest”: verify-image, name-image, verify-box, draw-box, verify-cover, verify-object, name-object. This is to help the annotators to slowly familiarize themselves with the questions; this was proven effective in our in-house annotation experiments.

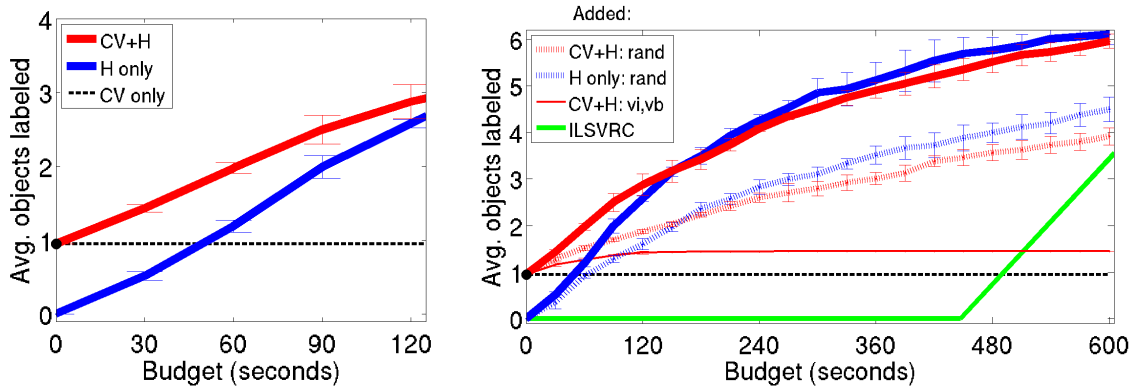


Figure 8.5: Our computer vision+human model ($CV+H$) compares favorably with others. The number of labeled objects (y-axis) is computed by averaging across multiple levels of precision on each image and then across all test images. Error bars correspond to one standard deviation across simulation runs. *Left*. The joint model handily outperforms the human-only (H only) and vision-only (CV only) baselines at low budget. *Right*. Our principled MDP is significantly better than choosing questions at random (*rand*). Variety of human interventions is critical; using only *verify-image* and *verify-box* human tasks is insufficient ($CV+H:vi,vb$). Our model also outperforms the ILSVRC-DET annotation baseline of [152].

more objects than the human-only method (Figure 8.5 left). This means that given 30 seconds of human time per image, adding in computer vision input can almost triple the accuracy of the human labeling.⁷

An MDP is an effective model for selecting human tasks. Figure 8.5 (right) shows that selecting questions at random is a surprisingly effective strategy that can label 3.9 ± 0.4 objects on average after 600 seconds of labeling ($CV+H: rand$). Our MDP approach significantly outperforms this baseline, labeling 6.0 ± 0.3 objects after 600 seconds.

Complex human tasks are necessary for effective annotation. We demonstrate that simple binary tasks are ineffective in our setting, by considering an MDP with just the *verify-image* and *verify-box* tasks ($CV+H: vi,vb$ in Figure 8.5 (right)). It labels 1.5x more objects than the CV-only baseline after 4.5 minutes of labeling and then plateaus. Our full system with all human tasks ($CV+H: all\ tasks$) achieves this improvement after just 1 minute, and then further improve to label 6.3x more objects than CV-only.

⁷Given a large labeling budget (Figure 8.5 right), human feedback contributes more than computer vision to the labeling. In fact, the human-only method even slightly outperforms the joint method in this case, partially due to the fact that it’s difficult to perfectly calibrate object detectors to estimate their level of uncertainty (this is where e.g., [222] may be useful).

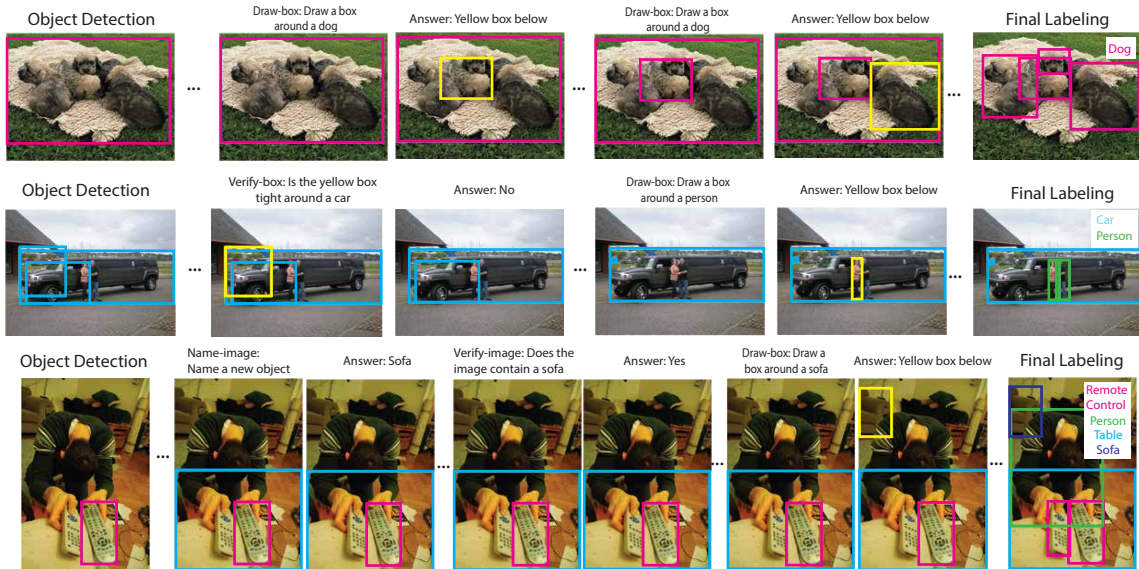


Figure 8.6: Some example results from our system integrating computer vision with multiple types of user feedback to annotate objects.

Our annotation strategy is more effective than the ILSVRC-DET system [152]. The ILSVRC detection system consists of two steps: (1) determining what object classes are present in the images, followed by (2) asking users to draw bounding boxes. (1) is described in [38, 152]. Using their annotation times, hierarchical annotation method and optimistically assuming just 2.5 workers per label, determining the presence/absence of all 200 object categories with 95% accuracy would take 446.9 seconds per image. [175] describes Step 2 and reports time per bounding box (including quality control) as 42.4 seconds. This baseline is shown in green in Figure 8.5 (right), and labels 3.6 objects after 600 seconds, on par with our random baseline (*CV+H: rand*) and significantly below our joint model.⁸

Figure 8.6 shows qualitative results of our labeling system.

Reducing error rates

We also consider how reducing human error rates affects labeling accuracy. Figure 8.7 demonstrates the labeling quality in simulation as the human error rates on all tasks are reduced. We conclude that reducing the human errors (through providing more clear labeling instructions, designing better user interfaces, or formulating simpler labeling questions) can significantly improve the speed and quality of the human-in-the-loop labeling.

⁸One important difference to note is that the ILSVRC-DET system was optimized for annotating the desired 200 object classes while our system allows users to freely name new object classes.

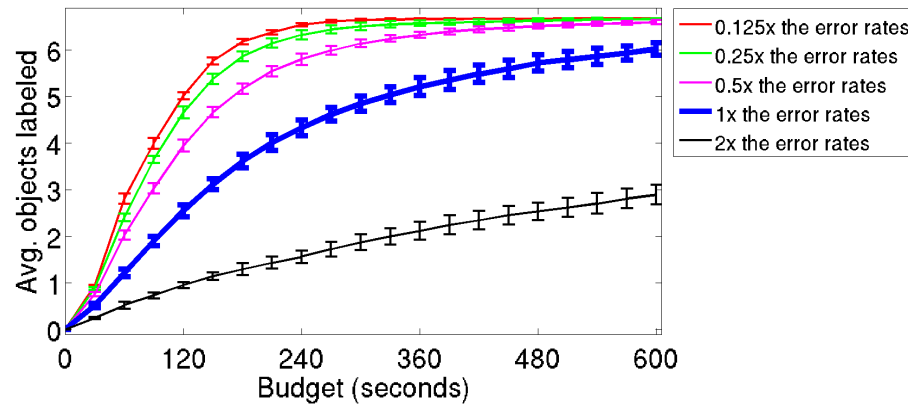


Figure 8.7: The average number of objects labeled as a function of human error rates. Reducing human error rates can significantly increase the quality of labeling.

8.6.4 Satisfying requester constraints

One of the key aspects of our system is the ability to allow the requester to provide constraints on the desired annotation (Section 8.3). After the annotation process (600 seconds), we queried the system for image annotations at 0.5 precision; 0.519 of the returned objects were indeed correct detections. We repeated the process at 0.1 intervals up to 0.9 precision; the model returned detections with an average of 0.041 higher precision than queried. Thus, the system is well-calibrated and we can do requester queries.

Figure 8.8(a) plots requested budget (x-axis) and requested precision (colored lines) versus the utility of returned labeling. We observe that, given the same budget, requesting a higher level of precision causes the system to be more cautious about returned detections and thus results in lower-utility labelings. After incorporating 5 minutes of human input and requesting a labeling at 0.9 precision the system will return on average 4 correctly labeled objects.

Instead of specifying the desired budget and precision, the requester can also specify the desired budget and utility. However, this may not be feasible in all cases, as shown in Figure 8.8(b). For example, obtaining a utility of 3 objects labeled after 60 seconds of labeling is feasible in only 50% of the images. For the images where it is feasible, however, we can refer to Figure 8.8(c) to get the expected precision of the labeling. In this case, the expected precision is 21.2%.

Providing this detailed analysis of the tradeoff between precision, utility and budget can help requesters interested in obtaining a dense labeling of objects in an image a-priori estimate the quality of the labeling given the scope of their problem and their constraints.

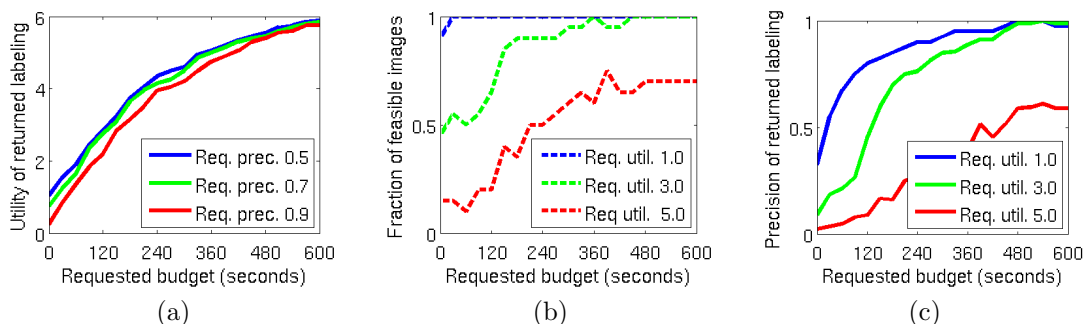


Figure 8.8: Quality of returned labeling as a function of requester constraints; details in Section 8.6.4.

8.7 Conclusions

We presented a principled approach to unifying multiple inputs from both computer vision and humans to label objects in images. We conclude with several take-home messages. First, from the **computer vision perspective**, current object detectors are far from perfect and can only detect a couple of objects in an average image. Further, accuracy drops rapidly when a tighter bounding box (with IOU higher than 0.5) is required. Our work can be used for collecting large-scale datasets with minimal supervision to improve the current state-of-the-art object detectors; in turn, the improved models will make our system more effective.

From a **crowd engineering perspective**, we demonstrated that it is worthwhile to combine multiple tasks in a principled framework. One interesting observation is that the verify-cover task (asking if all instances of an object class are already labeled in the image) inspired by ILSVRC data collection process [152] turned out in practice to have almost no impact on the labeling accuracy as it was selected by the model less than 0.1% of the time. This confirmed more of the intuitions of the later COCO [118] dataset that asking slightly more complex human tasks (such as immediately asking users to put a dot on the object rather than merely asking if the object appears) may be more efficient.

Finally, from an **application developer perspective**, we show that even though computer vision is not yet ready to detect all objects, we have a principled way of labeling all objects in a scene, trading off precision, utility and budget.

Chapter 9

Conclusions

This thesis presents my work on scaling up object detection from tens to hundreds of object categories and from tens to hundreds of thousands of images. The work presented here has focused on core detection algorithms (Chapters 2 and 4), on new ways of describing objects (Chapter 3), on constructing and evaluating large-scale datasets (Chapters 5-7), and on human-in-the-loop detection approaches (Chapter 8).

The running thread of this thesis has been **effective utilization of human effort for building computer vision systems**. Works such as learning generic attributes in Chapter 3 or weakly supervised localization models in Chapter 4 are motivated by the need to minimizing human annotation efforts while scaling up the capabilities of object recognition systems. Effective crowd engineering strategies of Chapters 6 can be used to minimize human effort while collecting large-scale training annotations as needed. Chapter 8 introduces effective strategies for minimizing the amount of human effort required for correcting errors made by the imperfect computer vision algorithm at test time.

Effectively utilizing human effort for improving computer vision systems will become even more important as the domains of interest grow. Future areas of exploration include analyzing even larger scale datasets (such as the Yahoo's Flickr Creative Commons 100M¹), focusing on more detailed image understanding with pixel-level image segmentation [150], or analyzing video rather than image data [217]. As the domains scale, the human cost associated with designing effective computer vision systems for these domains cannot scale accordingly. We will need to become more and more effective at targeting our human efforts of algorithmic design, data annotation and test-time intervention to attaining the optimal computer vision accuracy.

¹<http://webscope.sandbox.yahoo.com/catalog.php?datatype=i&did=67>

9.1 Lessons Learned

We learned several lessons about the effective use of human input for scaling up object detection.

Unsupervised region proposals Using unsupervised techniques to effectively focus attention to the most important visual regions becomes progressively more important as the domains of interest grow and human input become more sparse. Chapter 2 focused on automatically producing regions likely to contain the target object as to avoid exhaustive search. This theme has continued throughout the thesis, with Chapter 4 using a similar region proposal algorithm to regularize the search space for weakly supervised localization, Chapter 5 describing state-of-the-art large-scale algorithms most of which rely on unsupervised region proposals, and Chapter 8 using this method to make the human-in-the-loop computation tractable. This idea was relatively novel five years ago and has now emerged as the default approach for object detection.

The importance of datasets. Much of the thesis has focused on collecting, analyzing and utilizing large-scale datasets. In Chapter 4 we studied how object detection can be performed in a weakly supervised setting without the need for large-scale annotations; however, the detection accuracy was inferior to that of strongly supervised techniques. In order to scale up object detection from twenty to hundreds of object categories, diverse annotated large-scale training data is necessary. Multiple decisions had to be made along the way for collecting such data: from defining the feasible scale of the data and the appropriate evaluation metrics in Chapter 5 to formulating the most accurate and cost-efficient crowd sourcing techniques for data collection in Chapters 6 and 8. Progress in object recognition would have been significantly hindered without the availability of such large-scale datasets [104, 48, 152].

As we continue to expand the capabilities of our computer vision systems, the need for datasets will remain. Standardized test sets with detailed ground truth annotations will still be necessary for tracking progress in the field and for evaluating the relative effectiveness of different algorithms. The need to scale up and diversify these benchmarks will create new interesting challenges. For example, finding a sufficient number of photographs of screwdrivers in cluttered images on the web to test the accuracy of a screwdriver detector is already difficult. Designing benchmarks for testing our algorithms in rare visual situations may require more creative approaches than simply turning to crowdsourced annotation of web images.

Formulating data annotation as an optimization problem. Another underlying theme of this thesis has been formulating the data annotation problem into a coherent optimization framework. As we continue along the path of scaling up computer vision systems while minimizing human effort this is an important lesson to keep in mind. Both Chapter 6 and Chapter 8 demonstrated that both the cost and accuracy can be significantly improved by a principled mathematical annotation

framework compared to an ad hoc data annotation approach. This will become especially critical as the size of datasets continues to increase in future years.

The importance of designing effective user interfaces. The other critical component besides carefully formulating data annotation is to design effective user interfaces for both data annotation and human-in-the-loop approaches. Chapters 3, 6, 8 all discuss the successes and failures of effective interfaces. Some common threads are that multiple rounds of iterations are often necessary in the design and that simpler questions are often more cost-efficient than more complex questions even when considering the fact that a larger number of simpler questions are often necessary. However, the concept of a simpler question is often not immediately obvious without conducting user studies with human annotators unfamiliar with the experiment.

9.2 Future directions

We tackled the problem of scaling up object detection systems in both object classes and number of images that can be processed. One natural next question is: what would it take to build a system capable of meaningfully understanding all objects in any scene? Based on this thesis, three key research directions emerge. First, current algorithms are notoriously bad at detecting certain types of objects as shown in Chapter 7: thin, man-made, untextured objects tend to be particularly challenging. Scaling up object detections to hundreds of object categories brought about this insight, but it may be time to pay close attention to these challenging object categories instead of just focusing on average accuracy across many generic categories. Second, as we need to revisit the idea of large-scale generic attribute descriptions of Chapter 3. Simply naming the object categories is not enough to disambiguate between all objects and to describe everything in the scene. In this case there is a need to study generic object attributes at large scale instead of focusing our research efforts on just specific domains. Finally, in attempting to understand all objects in a scene, automatic object detection systems will inevitably encounter unknown objects. We need to continue exploring effective human-machine collaboration systems both at training time for efficiently collecting targeted training data as well as at test time for unexpected circumstances such as previously unseen objects.

More generally, as we explore richer visual domains with visual data and richer output spaces, I believe that the question of more effective utilization of human effort will come to the forefront. Efficiently collecting optimal training datasets, obtaining sufficiently diverse test benchmarks, seamlessly correcting computer vision errors, effectively soliciting occasional real-time human feedback – all of these research directions become progressively more important in the near future.

Appendix A

Detailed analysis of the ILSVRC localization dataset

In addition to the size of the dataset, we also analyze the level of difficulty of object localization in these images compared to the PASCAL VOC benchmark. This serves three purposes:

1. It illustrates the difficulty of both the image classification and single-object localization tasks in ILSVRC,
2. It justifies that the novel single-object localization task we introduces is sufficiently challenging to be of interest to the community, and
3. It provides a set of metrics for quantifying localization difficulty in Chapter 7 to better understand the performance of object recognition algorithms

In Section 7.3 we introduced four metrics of localization difficulty: number of object instances per image, object scale, chance performance of localization and the level of clutter. We compute statistics on the ILSVRC2012 single-object localization validation set images compared to PASCAL VOC 2012 validation images. Table A.1 and Figure A.1 summarize the results.

Number of instances. The average object category in ILSVRC has 1.61 target object instances on average per positive image, with each instance having on average 0.47 neighbors (adjacent instances of the same object category). This is comparable to 1.69 instances per positive image and 0.52 neighbors per instance for an average object class in PASCAL.

Object scale. In the average object category in PASCAL the object occupies 24.1% of the image area, and in ILSVRC 35.8%. However, PASCAL has only 20 object categories while ILSVRC has 1000. The 537 object categories of ILSVRC with the smallest objects on average occupy the same

Property	PASCAL (num classes)	ILSVRC subset (num classes)	ILSVRC (num classes)
Instances	1.69 (20)	1.69 (843)	1.61 (1000)
Scale	24.08% (20)	24.06% (537)	35.83% (1000)
CPL	8.76% (20)	8.74% (562)	20.83% (1000)
Clutter	5.90 (20)	5.90 (250)	3.59 (1000)

Table A.1: Comparison of the PASCAL VOC 2012 object detection dataset and the ILSVRC 2012 single-object localization dataset on several key properties of object localization difficulty (please see Section 5.4.1 for definitions). For each property, the ILSVRC subset is chosen by selecting the largest set of ILSVRC classes with the same average difficulty as that of the PASCAL classes. The number of object classes is indicated in parentheses. Note that according to any of these measures of difficulty there is a subset of ILSVRC which is as challenging as PASCAL but more than an order of magnitude greater in size.

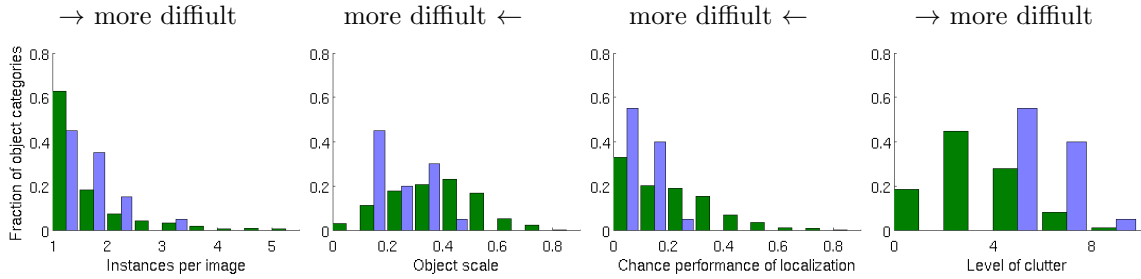
fraction of the image as PASCAL objects: 24.1%. Thus even though on average the object instances tend to be bigger in ILSVRC images, there are more than 25 times more object categories than in PASCAL VOC with the same average object scale.

Chance performance of localization (CPL). Some of the most difficult ILSVRC categories to localize according to this metric are basketball, swimming trunks, ping pong ball and rubber eraser, all with less than 0.2% CPL. The average CPL across the 1000 ILSVRC categories is 20.8%. The 20 PASCAL categories have an average CPL of 8.7%, which is the same as the CPL of the 562 most difficult categories of ILSVRC.

Clutter. The measure of clutter is defined in Section 7.3 as a logarithm of the number of objectness windows [3] it takes on average to localize an instance of the target class. If an object can't be localized with the first 1000 windows (as is the case for 1% of images on average per category in ILSVRC and 5% in PASCAL), we set $\text{OBJ}(m) = 1001$. The fact that more than 95% of objects can be localized with these windows imply that the objectness cue is already quite strong, so objects that require many windows on average will be extremely difficult to detect: e.g., ping pong ball (clutter of 9.57, or 758 windows on average), basketball (clutter of 9.21), puck (clutter of 9.17) in ILSVRC. The most difficult object in PASCAL is bottle with clutter score of 8.47. On average, ILSVRC has clutter score of 3.59. The most difficult subset of ILSVRC with 250 object categories has an order of magnitude more categories and the same average amount of clutter (of 5.90) as the PASCAL dataset.

Summary. We compared the statistics of PASCAL2012 validation dataset to ILSVRC2012 validation dataset according to four measures of localization difficulty of Section 7.3: number of object instances per image, average object scale, chance performance of localization, and level of clutter. According to all the metrics there is a subset of ILSVRC which is as challenging as PASCAL but more than an order of magnitude greater in size. As a result of its scale, the ILSVRC allows

**1000 classes of ILSVRC2012-2014 single-object localization (dark green)
versus 20 classes of PASCAL 2012 (light blue)**



**200 hardest classes of ILSVRC2012-2014 single-object localization (dark green)
versus 20 classes of PASCAL 2012 (light blue)**

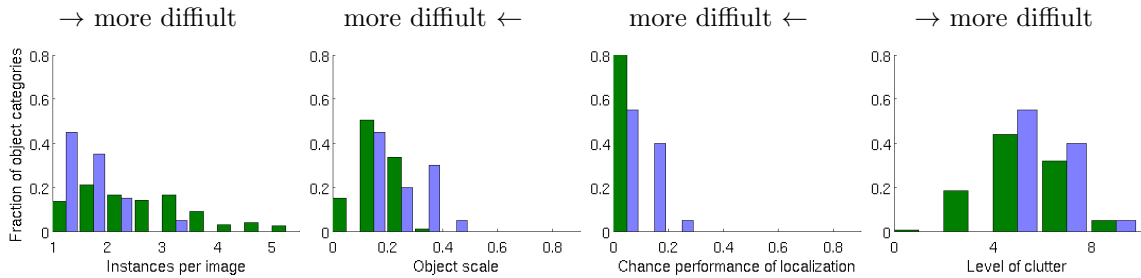


Figure A.1: Distribution of various measures of localization difficulty on the ILSVRC2012-2014 single-object localization (dark green) and PASCAL VOC 2012 (light blue) validation sets. The plots on top contain the full ILSVRC validation set with 1000 classes; the plots on the bottom contain 200 ILSVRC classes with the lowest chance performance of localization. All plots contain all 20 classes of PASCAL VOC.

for evaluation of performance of detectors under a variety of image-level statistics, documented in Chapter 7.

Appendix B

Analysis of the top-5 ILSVRC evaluation criteria

In Chapter 5 we introduced the ImageNet Large Scale Visual Recognition Challenge and in Section 5.5 described the top-5 evaluation criteria used on the ILSVRC classification and localization tasks: an algorithm is allowed to make up to 5 guesses per image without penalty. We briefly justify this choice.

Top-5 evaluation of algorithms. Chapter 7 we analyze the performance of leading object detection algorithms as a function of object class properties. In Section 7.4 we specifically focus on the breakthrough year 2012. Here we consider just the state-of-the-art algorithms from year 2012, SV and VGG as described in Section 7.4.2.

Figure B.1 plots the accuracy of the methods SV and VGG as a function of the number of guesses (ignoring the black curves). As the algorithms are allowed to make between 1 and 5 guesses, the relative performance remains reasonably consistent: the difference in classification accuracy between the two methods ranges from 0.108 and 0.117, and the difference in localization accuracy ranges from 0.140 to 0.160. Since these patterns are consistent, we follow the intuition of the ImageNet challenge evaluation (the images are not exhaustively labeled, so unannotated objects may be present and thus the algorithms should not be penalized for potentially predicting an unlabeled object as the top scoring detection) and use top-5 evaluation in our analysis. Similar conclusions can be drawn from the data when using just top-1 evaluation.

Top-10 evaluation of upper bound. Section 7.4.2 presented an upper bound of the two methods, which combines the outputs of the VGG and SV on every image and considers the object to be correctly detected if any of the 10 proposed (class, location) pairs is correct. Here we provide some analysis and insight.

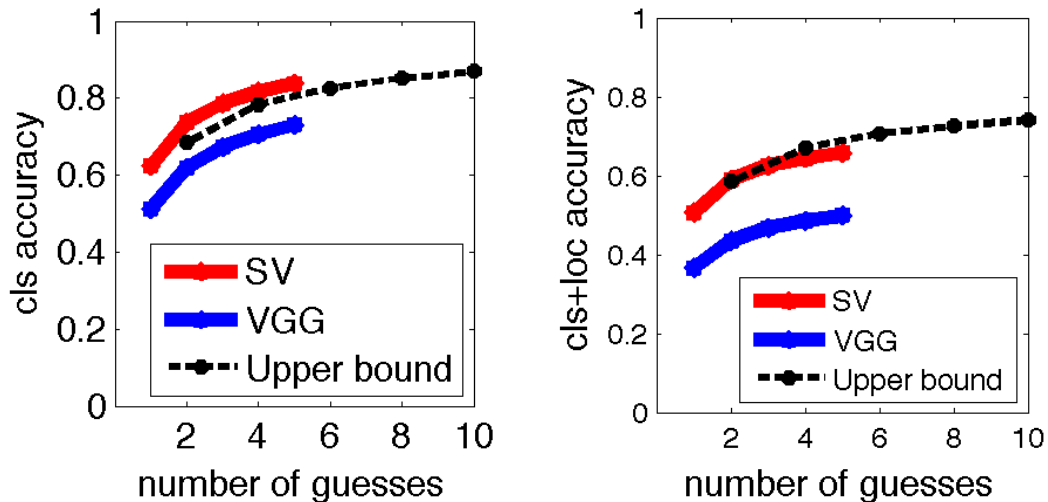


Figure B.1: (a) Classification and (b) classification with localization accuracies of the three methods as a function of the number of guesses allowed during evaluation on ILSVRC2012.

The idea is to put an upper bound on how well a combination of the two systems might work – for example, given an oracle which would select the “best” box from the 10 boxes proposed by SV and VGG (in this setting, top-10 evaluation would actually be the same as top-1).

Figure B.1 shows the number of guesses versus accuracy for the upper bound (in black) by taking the top 1, 2, 3, 4, 5 guesses from each method, so 2, 4, 6, 8, 10 guesses. It is important to keep in mind that the scores between the two algorithms are uncalibrated, so this upper bound is in fact suboptimal. However, a few interesting trends are still worth noting:

- For classification, SV is an impressively strong algorithm: given a budget of only 5 guesses it’s better to use the top-5 guesses from SV (accuracy of 0.838) rather than combining it with VGG (at least in the current setting of uncalibrated scores; upper bound top-6 accuracy is only 0.827).
- For localization, taking the top 2 detections from SV (accuracy of 0.590) is also slightly better than taking the top detection from each algorithm (upper bound top-2 accuracy is only 0.586)
- For localization, when considering top-5 detections the combination algorithm is in fact stronger than SV alone: SV top-5 accuracy is 0.658, and upper bound top-4 accuracy is 0.671 (top-6 is 0.707)

Further investigation is outside the scope of this work but may yield more interesting insights and stronger combined detection algorithms.

Appendix C

Manually curated queries for ILSVRC detection images

In Section 6.3 we discussed three types of queries we used for collecting the object detection images: (1) single object category name or a synonym; (2) a pair of object category names; (3) a manual query, typically targetting one or more object categories with insufficient data. Here we provide a list of the 129 manually curated queries:

afternoon tea, ant bridge building, armadillo race, armadillo yard, artist studio, auscultation, baby room, banjo orchestra, banjo rehearsal, banjo show, califone headphones & media player sets, camel dessert, camel tourist, carpenter drilling, carpentry, centipede wild, coffee shop, continental breakfast toaster, continental breakfast waffles, crutch walking, desert scorpion, diner, dining room, dining table, dinner, dragonfly friendly, dragonfly kid, dragonfly pond, dragonfly wild, drying hair, dumbbell curl, fan blow wind, fast food, fast food restaurant, firewood chopping, flu shot, goldfish aquarium, goldfish tank, golf cart on golf course, gym dumbbell, hamster drinking water, harmonica orchestra, harmonica rehearsal, harmonica show, harp ensemble, harp orchestra, harp rehearsal, harp show, hedgehog cute, hedgehog floor, hedgehog hidden, hippo bird, hippo friendly, home improvement diy drill, horseback riding, hotel coffee machine, hotel coffee maker, hotel waffle maker, jellyfish scuba, jellyfish snorkling, kitchen, kitchen counter coffee maker, kitchen counter toaster, kitchenette, koala feed, koala tree, ladybug flower, ladybug yard, laundromat, lion zebra friendly, lunch, mailman, making breakfast, making waffles, mexican food, motorcycle racing, office, office fan, opossum on tree branch, orchestra, panda play, panda tree, pizzeria, pomegranate tree, porcupine climbing trees, power drill carpenter, purse shop, red panda tree, riding competition, riding motor scooters, school supplies, scuba starfish, sea lion beach, sea otter, sea urchin habitat, shopping for school supplies, sitting in front of a fan, skunk and cat, skunk park, skunk wild, skunk yard, snail flower, snorkling starfish, snowplow cleanup, snowplow pile, snowplow winter, soccer game, south

american zoo, starfish sea world, starts shopping, steamed artichoke, stethoscope doctor, strainer pasta, strainer tea, syringe doctor, table with food, tape player, tiger circus, tiger pet, using a can opener, using power drill, waffle iron breakfast, wild lion savana, wildlife preserve animals, wiping dishes, wombat petting zoo, zebra savana, zoo feeding, zoo in australia

Appendix D

Hierarchy of questions for ILSVRC detection annotation

The following is a hierarchy of questions manually constructed for crowdsourcing full annotation of images with the presence or absence of 200 object detection categories in ILSVRC2013 and ILSVRC2014. All questions are of the form “is there a ... in the image?” Questions marked with • are asked on every image. If the answer to a question is determined to be “no” then the answer to all descendant questions is assumed to be “no”. The 200 numbered leaf nodes correspond to the 200 object detection categories.

The goal in the hierarchy construction is to save cost (by asking as few questions as possible on every image) while avoiding any ambiguity in questions which would lead to false negatives during annotation. This hierarchy is not tree-structured; some questions have multiple parents.

- first aid/ medical items
 - (1) stethoscope
 - (2) syringe
 - (3) neck brace
 - (4) crutch
 - (5) stretcher
 - (6) band aid: an adhesive bandage to cover small cuts or blisters
- musical instruments
 - (7) accordion (a portable box-shaped free-reed instrument; the reeds are made to vibrate by air from the bellows controlled by the player)
 - (8) piano, pianoforte, forte-piano
 - percussion instruments: chimes, maraccas, drums, etc
 - (9) chime: a percussion instrument consisting of a set of tuned bells that are struck with a

- hammer; used as an orchestral instrument
- (10) maraca
- (11) drum
- stringed instrument
 - (12) banjo, the body of a banjo is round. please do not confuse with guitar
 - (13) cello: a large stringed instrument; seated player holds it upright while playing
 - (14) violin: bowed stringed instrument that has four strings, a hollow body, an unfretted fingerboard and is played with a bow. please do not confuse with cello, which is held upright while playing
 - (15) harp
 - (16) guitar, please do not confuse with banjo. the body of a banjo is round
- wind instrument: a musical instrument in which the sound is produced by an enclosed column of air that is moved by the breath (such as trumpet, french horn, harmonica, flute, etc)
 - (17) trumpet: a brass musical instrument with a narrow tube and a flared bell, which is played by means of valves. often has 3 keys on top
 - (18) french horn: a brass musical instrument consisting of a conical tube that is coiled into a spiral, with a flared bell at the end
 - (19) trombone: a brass instrument consisting of a long tube whose length can be varied by a u-shaped slide
 - (20) harmonica
 - (21) flute: a high-pitched musical instrument that looks like a straight tube and is usually played sideways (please do not confuse with oboes, which have a distinctive straw-like mouth piece and a slightly flared end)
 - (22) oboe: a slender musical instrument roughly 65cm long with metal keys, a distinctive straw-like mouthpiece and often a slightly flared end (please do not confuse with flutes)
 - (23) saxophone: a musical instrument consisting of a brass conical tube, often with a u-bend at the end
- food: something you can eat or drink (includes growing fruit, vegetables and mushrooms, but does not include living animals)
 - food with bread or crust: pretzel, bagel, pizza, hotdog, hamburgers, etc
 - (24) pretzel
 - (25) bagel, beigel
 - (26) pizza, pizza pie
 - (27) hotdog, hot dog, red hot
 - (28) hamburger, beefburger, burger
 - (29) guacamole
 - (30) burrito

- (31) popsicle (ice cream or water ice on a small wooden stick)
- fruit
 - (32) fig
 - (33) pineapple, ananas
 - (34) banana
 - (35) pomegranate
 - (36) apple
 - (37) strawberry
 - (38) orange
 - (39) lemon
- vegetables
 - (40) cucumber, cuke
 - (41) artichoke, globe artichoke
 - (42) bell pepper
 - (43) head cabbage
- (44) mushroom
- items that run on electricity (plugged in or using batteries); including clocks, microphones, traffic lights, computers, etc
 - (45) remote control, remote
 - electronics that blow air
 - (46) hair dryer, blow dryer
 - (47) electric fan: a device for creating a current of air by movement of a surface or surfaces (please do not consider hair dryers)
 - electronics that can play music or amplify sound
 - (48) tape player
 - (49) iPod
 - (50) microphone, mike
 - computer and computer peripherals: mouse, laptop, printer, keyboard, etc
 - (51) computer mouse
 - (52) laptop, laptop computer
 - (53) printer (please do not consider typewriters to be printers)
 - (54) computer keyboard
 - (55) lamp
 - electric cooking appliance (an appliance which generates heat to cook food or boil water)
 - (56) microwave, microwave oven
 - (57) toaster
 - (58) waffle iron

- (59) coffee maker: a kitchen appliance used for brewing coffee automatically
- (60) vacuum, vacuum cleaner
- (61) dishwasher, dish washer, dishwashing machine
- (62) washer, washing machine: an electric appliance for washing clothes
- (63) traffic light, traffic signal, stoplight
- (64) tv or monitor: an electronic device that represents information in visual form
- (65) digital clock: a clock that displays the time of day digitally
- kitchen items: tools, utensils and appliances usually found in the kitchen
 - electric cooking appliance (an appliance which generates heat to cook food or boil water)
 - (56) microwave, microwave oven
 - (57) toaster
 - (58) waffle iron
 - (59) coffee maker: a kitchen appliance used for brewing coffee automatically
 - (61) dishwasher, dish washer, dishwashing machine
 - (66) stove
 - things used to open cans/bottles: can opener or corkscrew
 - (67) can opener (tin opener)
 - (68) corkscrew
 - (69) cocktail shaker
 - non-electric item commonly found in the kitchen: pot, pan, utensil, bowl, etc
 - (70) strainer
 - (71) frying pan (skillet)
 - (72) bowl: a dish for serving food that is round, open at the top, and has no handles (please do not confuse with a cup, which usually has a handle and is used for serving drinks)
 - (73) salt or pepper shaker: a shaker with a perforated top for sprinkling salt or pepper
 - (74) plate rack
 - (75) spatula: a turner with a narrow flexible blade
 - (76) ladle: a spoon-shaped vessel with a long handle; frequently used to transfer liquids from one container to another
 - (77) refrigerator, icebox
 - furniture (including benches)
 - (78) bookshelf: a shelf on which to keep books
 - (79) baby bed: small bed for babies, enclosed by sides to prevent baby from falling
 - (80) filing cabinet: office furniture consisting of a container for keeping papers in order
 - (81) bench (a long seat for several people, typically made of wood or stone)
 - (82) chair: a raised piece of furniture for one person to sit on; please do not confuse with benches or sofas, which are made for more people

- (83) sofa, couch: upholstered seat for more than one person; please do not confuse with benches (which are made of wood or stone) or with chairs (which are for just one person)
- (84) table
- clothing, article of clothing: a covering designed to be worn on a person's body
 - (85) diaper: Garment consisting of a folded cloth drawn up between the legs and fastened at the waist; worn by infants to catch excrement
 - swimming attire: clothes used for swimming or bathing (swim suits, swim trunks, bathing caps)
 - (86) swimming trunks: swimsuit worn by men while swimming
 - (87) bathing cap, swimming cap: a cap worn to keep hair dry while swimming or showering
 - (88) maillot: a woman's one-piece bathing suit
 - necktie: a man's formal article of clothing worn around the neck (including bow ties)
 - (89) bow tie: a man's tie that ties in a bow
 - (90) tie: a long piece of cloth worn for decorative purposes around the neck or shoulders, resting under the shirt collar and knotted at the throat (NOT a bow tie)
 - headdress, headgear: clothing for the head (hats, helmets, bathing caps, etc)
 - (87) bathing cap, swimming cap: a cap worn to keep hair dry while swimming or showering
 - (91) hat with a wide brim
 - (92) helmet: protective headgear made of hard material to resist blows
 - (93) miniskirt, mini: a very short skirt
 - (94) brassiere, bra: an undergarment worn by women to support their breasts
 - (95) sunglasses
- living organism (other than people): dogs, snakes, fish, insects, sea urchins, starfish, etc.
 - living organism which can fly
 - (96) bee
 - (97) dragonfly
 - (98) ladybug
 - (99) butterfly
 - (100) bird
 - living organism which cannot fly (please don't include humans)
 - living organism with 2 or 4 legs (please don't include humans):
 - mammals (but please do not include humans)
 - feline (cat-like) animal: cat, tiger or lion
 - (101) domestic cat
 - (102) tiger
 - (103) lion
 - canine (dog-like animal): dog, hyena, fox or wolf
 - (104) dog, domestic dog, canis familiaris

- (105) fox: wild carnivorous mammal with pointed muzzle and ears and a bushy tail (please do not confuse with dogs)
- animals with hooves: camels, elephants, hippos, pigs, sheep, etc
 - (106) elephant
 - (107) hippopotamus, hippo
 - (108) camel
 - (109) swine: pig or boar
 - (110) sheep: woolly animal, males have large spiraling horns (please do not confuse with antelope which have long legs)
 - (111) cattle: cows or oxen (domestic bovine animals)
 - (112) zebra
 - (113) horse
 - (114) antelope: a graceful animal with long legs and horns directed upward and backward
- (115) squirrel
- (116) hamster: short-tailed burrowing rodent with large cheek pouches
- (117) otter
- (118) monkey
- (119) koala bear
- (120) bear (other than pandas)
- (121) skunk (mammal known for its ability to spray a liquid with a strong odor; they may have a single thick stripe across back and tail, two thinner stripes, or a series of white spots and broken stripes)
- (122) rabbit
- (123) giant panda: an animal characterized by its distinct black and white markings
- (124) red panda: Reddish-brown Old World raccoon-like carnivore
- (125) frog, toad
- (126) lizard: please do not confuse with snake (lizards have legs)
- (127) turtle
- (128) armadillo
- (129) porcupine, hedgehog
- living organism with 6 or more legs: lobster, scorpion, insects, etc.
 - (130) lobster: large marine crustaceans with long bodies and muscular tails; three of their five pairs of legs have claws
 - (131) scorpion
 - (132) centipede: an arthropod having a flattened body of 15 to 173 segments each with a pair of legs, the foremost pair being modified as prehensors

- (133) tick (a small creature with 4 pairs of legs which lives on the blood of mammals and birds)
- (134) isopod: a small crustacean with seven pairs of legs adapted for crawling
- (135) ant
- living organism without legs: fish, snake, seal, etc. (please don't include plants)
- living organism that lives in water: seal, whale, fish, sea cucumber, etc.
 - (136) jellyfish
 - (137) starfish, sea star
 - (138) seal
 - (139) whale
 - (140) ray: a marine animal with a horizontally flattened body and enlarged winglike pectoral fins with gills on the underside
 - (141) goldfish: small golden or orange-red fishes
- living organism that slides on land: worm, snail, snake
 - (142) snail
 - (143) snake: please do not confuse with lizard (snakes do not have legs)
- vehicle: any object used to move people or objects from place to place
 - a vehicle with wheels
 - (144) golfcart, golf cart
 - (145) snowplow: a vehicle used to push snow from roads
 - (146) motorcycle (or moped)
 - (147) car, automobile (not a golf cart or a bus)
 - (148) bus: a vehicle carrying many passengers; used for public transport
 - (149) train
 - (150) cart: a heavy open wagon usually having two wheels and drawn by an animal
 - (151) bicycle, bike: a two wheeled vehicle moved by foot pedals
 - (152) unicycle, monocycle
 - a vehicle without wheels (snowmobile, sleighs)
 - (153) snowmobile: tracked vehicle for travel on snow
 - (154) watercraft (such as ship or boat): a craft designed for water transportation
 - (155) airplane: an aircraft powered by propellers or jets
- cosmetics: toiletry designed to beautify the body
 - (156) face powder
 - (157) perfume, essence (usually comes in a smaller bottle than hair spray)
 - (158) hair spray
 - (159) cream, ointment, lotion
 - (160) lipstick, lip rouge

- carpentry items: items used in carpentry, including nails, hammers, axes, screwdrivers, drills, chain saws, etc
 - (161) chain saw, chainsaw
 - (162) nail: pin-shaped with a head on one end and a point on the other
 - (163) axe: a sharp tool often used to cut trees/ logs
 - (164) hammer: a blunt hand tool used to drive nails in or break things apart (please do not confuse with axe, which is sharp)
 - (165) screwdriver
 - (166) power drill: a power tool for drilling holes into hard materials
- school supplies: rulers, erasers, pencil sharpeners, pencil boxes, binders
 - (167) ruler, rule: measuring stick consisting of a strip of wood or metal or plastic with a straight edge that is used for drawing straight lines and measuring lengths
 - (168) rubber eraser, rubber, pencil eraser
 - (169) pencil sharpener
 - (170) pencil box, pencil case
 - (171) binder, ring-binder
- sports items: items used to play sports or in the gym (such as skis, raquets, gymnastics bars, bows, punching bags, balls)
 - (172) bow: weapon for shooting arrows, composed of a curved piece of resilient wood with a taut cord to propel the arrow
 - (173) puck, hockey puck: vulcanized rubber disk 3 inches in diameter that is used instead of a ball in ice hockey
 - (174) ski
 - (175) racket, racquet
 - gymnastic equipment: parallel bars, high beam, etc
 - (176) balance beam: a horizontal bar used for gymnastics which is raised from the floor and wide enough to walk on
 - (177) horizontal bar, high bar: used for gymnastics; gymnasts grip it with their hands (please do not confuse with balance beam, which is wide enough to walk on)
 - ball
 - (178) golf ball
 - (179) baseball
 - (180) basketball
 - (181) croquet ball
 - (182) soccer ball
 - (183) ping-pong ball
 - (184) rugby ball

- (185) volleyball
- (186) tennis ball
- (187) punching bag, punch bag, punching ball, punchball
- (188) dumbbell: An exercising weight; two spheres connected by a short bar that serves as a handle
- liquid container: vessels which commonly contain liquids such as bottles, cans, etc.
 - (189) pitcher: a vessel with a handle and a spout for pouring
 - (190) beaker: a flatbottomed jar made of glass or plastic; used for chemistry
 - (191) milk can
 - (192) soap dispenser
 - (193) wine bottle
 - (194) water bottle
 - (195) cup or mug (usually with a handle and usually cylindrical)
- bag
 - (196) backpack: a bag carried by a strap on your back or shoulder
 - (197) purse: a small bag for carrying money
 - (198) plastic bag
- (199) person
- (200) flower pot: a container in which plants are cultivated

Appendix E

User interfaces for human-machine object annotation

In this appendix we present the user interfaces for the human-machine collaboration for object annotation system described in Chapter 8. Figure E.1 shows a zoomed-out view of the annotation interface. The general instructions at the bottom of the page (shown in Figure E.2) discuss what is considered a good bounding box (with examples) and what types of objects should be annotated. These instructions are shown to workers at beginning of the work and are always available for reference at the bottom of the page. Each type of task has a separate interface with brief specific instructions shown in Figures E.3-E.9.

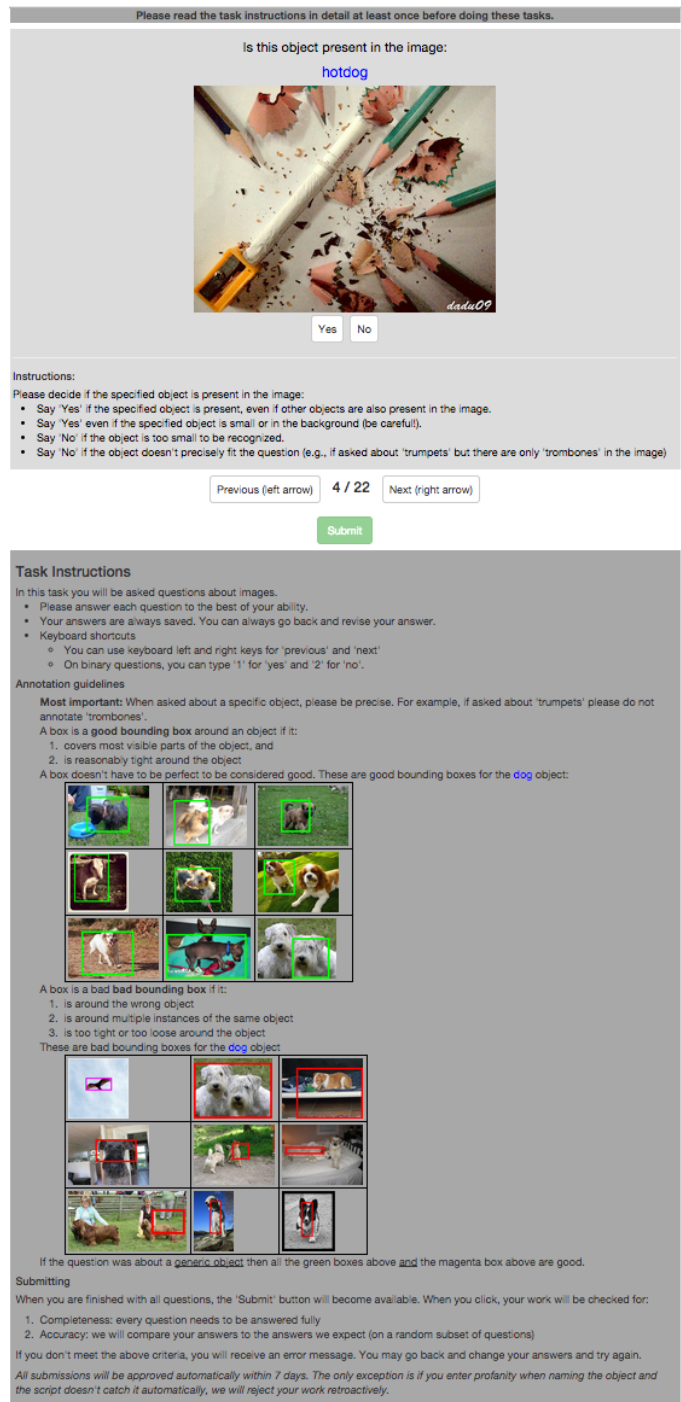


Figure E.1: Overview of our annotation layout. A close-up of the instructions (bottom) is in Figure E.2. Only one type of task is shown here (top); closeups of each of the seven types are in Figures E.3-E.9).


Annotation guidelines

Most important: When asked about a specific object, please be precise. For example, if asked about 'trumpets' please do not annotate 'trombones'.

A box is a **good bounding box** around an object if it:

1. covers most visible parts of the object, and
2. is reasonably tight around the object


A box doesn't have to be perfect to be considered good. These are good bounding boxes for the **dog** object:



A box is a **bad bounding box** if it:

1. is around the wrong object
2. is around multiple instances of the same object
3. is too tight or too loose around the object

These are bad bounding boxes for the **dog** object




If the question was about a generic object then all the green boxes above and the magenta box above are good.

Figure E.2: General instructions for our human annotation tasks.

Please read the task instructions in detail at least once before doing these tasks.

Is this object present in the image:

hotdog



Yes No

Instructions:


Please decide if the specified object is present in the image:

- Say 'Yes' if the specified object is present, even if other objects are also present in the image.
- Say 'Yes' even if the specified object is small or in the background (be careful!).
- Say 'No' if the object is too small to be recognized.
- Say 'No' if the object doesn't precisely fit the question (e.g., if asked about 'trumpets' but there are only 'trombones' in the image)

Figure E.3: User interface for **verify-image** task. The correct answer is “no.”

Please read the task instructions in detail at least once before doing these tasks.

Is this a good box around a
guitar



Yes, it is a good box No, it is not a good box


Instructions:
Please decide if the provided box is a good bounding box around a single instance of **the specified object**. See the examples below to learn what is considered a good bounding box.

Figure E.4: User interface for **verify-box** task. The correct answer is “yes.”

Please read the task instructions in detail at least once before doing these tasks.

Is there a good box around EVERY instance of:

dumbbell



Yes, every instance is covered No, not every instance is covered

Instructions:


Please check if there is a good blue bounding box around every instance of **the specified object**.

- See the examples below to learn what is considered a good bounding box.
- Ignore any bad blue boxes, if any.
- Say 'Yes' if every instance of the object has a good box around it.
- Say 'Yes' if there are no instances of this object in the image.
- Say 'No' if at least one instance of the object does not have a good box.

Figure E.5: User interface for **verify-cover** task. The correct answer is “no.”

Please read the task instructions in detail at least once before doing these tasks.

Draw a good box around another
chair



Click here if no other box can be drawn

Instructions:

Please draw a tight bounding box around one new instance of the specified object class. See the examples below to learn what is considered a good bounding box.

If there are yellow boxes on the image:

- Your box should not be around an object instance that already has a good yellow bounding box around it.
- The system will not let you draw a box too close to an existing box by giving a 'Too close to existing box' error. You will not be able to submit unless you fix this error.
- Some yellow boxes may be incorrect; simply ignore them.

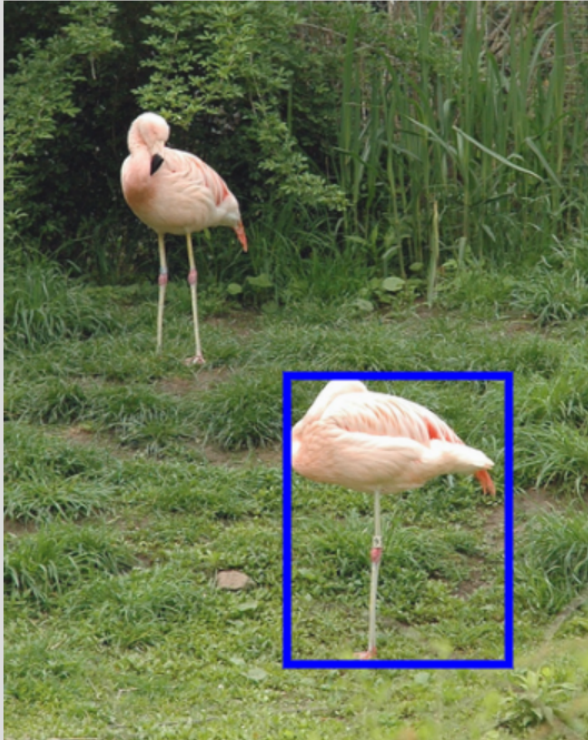
Click the 'Click here if no box can be drawn' button if:

1. The object is not present in the image, or
2. The object is present but all instances are already correctly annotated by yellow boxes, or
3. The only remaining unannotated instances are so close to existing yellow boxes that the system will not let you draw a box around them by giving a 'Too close to existing box' error.

Figure E.6: User interface for **draw-box** task. The correct answer is “no other box can be drawn.”

Please read the task instructions in detail at least once before doing these tasks.

Is this a good box around AN object?



Yes, it is a good box

No, it is not a good box

Instructions:
Please decide if the provided box is a good bounding box around a single instance of **some** object (for example, it is a good bounding box around one cat, or one person, or one screwdriver, or one instance of any other object). See the examples below to learn what is considered a good bounding box.

Figure E.7: User interface for **verify-object** task. The correct answer is “yes.”

Please read the task instructions in detail at least once before doing these tasks.

Name the object in the blue box (only if the box is good!):

Click here if not a good box



Instructions:

First, please see the examples below to learn what is considered a good bounding box around an object.

- If you believe the blue box is not a good box around **any** object, you can click the 'Click here if not a good box' button.

If the blue box is good, please type in the name of the object within it:

- In general, the most 'basic' word that comes to mind is good: for example, 'chair', 'car', 'dog' are all good annotations
- As you type, the box may autocomplete to offer some suggestions.
- Do not be too vague: 'object' or 'thing' are not good annotations.
- Do not be too specific (unless you're sure): better to say 'cat' than to incorrectly label a 'Persian cat' as a 'Burmese cat'
- Do not enter profanity -- your work will be rejected.

Figure E.8: User interface for **name-object** task. The correct answer is “not a good box.”

Please read the task instructions in detail at least once before doing these tasks.

Name another object in the image:

Current objects: chair; person

Click here if no other objects



Instructions:

Please write the name of any object that is in the image but is not listed.

Please avoid synonyms of the listed objects

- In general, the most 'basic' word that comes to mind is good: for example, 'chair', 'car', 'dog' are all good annotations
- Do not be too vague: 'object' or 'thing' are not good annotations.
- Do not be too specific (unless you're sure): better to say 'cat' than to incorrectly mention a 'Persian cat' instead of a 'Burmese cat'
- Do not enter profanity -- your work will be rejected.
- You will not be able to submit if you repeat one of the existing words

Figure E.9: User interface for **name-image** task. The correct answer is, for example, “umbrella.”

Bibliography

- [1] The sun attribute database: Beyond categories for deeper scene understanding. *International Journal on Computer Vision (IJCV)*, 2014.
- [2] T. Ahonen, A. Hadid, and M. Pietikinen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28, 2006.
- [3] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2012.
- [4] R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [5] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33, 2011.
- [6] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [7] S. Bell, P. Upchurch, N. Snavely, and K. Bala. OpenSurfaces: A richly annotated catalog of surface appearance. In *ACM Transactions on Graphics (SIGGRAPH)*, 2013.
- [8] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2002.
- [9] A. Berg, R. Farrell, A. Khosla, J. Krause, L. Fei-Fei, J. Li, and S. Maji. Fine-Grained Competition. <https://sites.google.com/site/fgcomp2013/>, 2013.
- [10] A. C. Berg, T. L. Berg, H. Daume, J. Dodge, A. Goyal, X. Han, A. Mensch, M. Mitchell, A. Sood, K. Stratos, et al. Understanding and predicting importance in images. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

- [11] M. S. Bernstein, J. Brandt, R. C. Miller, and D. R. Karger. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *ACM User Interface Software and Technology Symposium (UIST)*, 2011.
- [12] J. P. Bigham, C. Jayant, H. Ji, G. Little, A. Miller, R. C. Miller, R. Miller, A. Tatarowicz, B. White, S. White, and T. Yeh. VizWiz: Nearly real-time answers to visual questions. In *ACM User Interface Software and Technology Symposium (UIST)*, 2010.
- [13] H. Bilen, V. P. Namboodiri, and L. V. Gool. Object and action classification with latent variables. In *British Machine Vision Conference (BMVC)*, 2010.
- [14] S. Bileschi. *StreetScenes: Towards Scene Understanding in Still Images*. PhD thesis, Massachusetts Institute of Technology (MIT), 2006.
- [15] A. Biswas and D. Parikh. Simultaneous active learning of classifiers & attributes via relative feedback. In *Computer Vision and Pattern Recognition (CVPR)*, 2013.
- [16] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771, 2004.
- [17] J. Bragg, Mausam, and D. S. Weld. Crowdsourcing multi-label classification for taxonomy creation. In *Conference on Human Computation and Crowdsourcing (HCOMP)*, 2013.
- [18] S. Branson, K. E. Hjorleifsson, and P. Perona. Active annotation translation. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [19] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie. Visual recognition with humans in the loop. In *European Conference on Computer Vision (ECCV)*, 2010.
- [20] R. Caruana. Multitask learning: A knowledge-based source of inductive bias. *Machine Learning*, 1997.
- [21] B. Catanzaro, B.-Y. Su, N. Sundaram, Y. Lee, M. Murphy, and K. Keutzer. Efficient, high-quality image contour detection. In *International Conference on Computer Vision (ICCV)*, 2009.
- [22] Y. Chai, V. Lempitsky, and A. Zisserman. BiCoS: A bi-level co-segmentation method for image classification. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.
- [23] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines. <http://www.csie.ntu.edu.tw/~cjlin/libsvm>, 2001.
- [24] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, and S. Guha. Approximation algorithms for directed Steiner problems. In *Symposium on Discrete Algorithms*, 1998.

- [25] Q. Chen, Z. Song, Z. Huang, Y. Hua, and S. Yan. Contextualizing object detection and classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2014.
- [26] X. Chen, A. Shrivastava, and A. Gupta. NEIL: Extracting Visual Knowledge from Web Data. In *International Conference on Computer Vision (ICCV)*, 2013.
- [27] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *Computer Vision and Pattern Recognition (CVPR)*, 2007.
- [28] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, and Z. Popovic. Predicting protein structures with a multiplayer online game. *Nature*, 466:756–760, 2010.
- [29] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.
- [30] D. Crandall and D. Huttenlocher. Weakly supervised learning of part-based spatial models for visual object recognition. In *European Conference on Computer Vision (ECCV)*, 2006.
- [31] A. Criminisi. Microsoft Research Cambridge (MSRC) object recognition image database (version 2.0). <http://research.microsoft.com/vision/cambridge/recognition>, 2004.
- [32] P. Dai, Mausam, and D. S. Weld. Decision-theoretic control of crowd-sourced workflows. In *AAAI Conference on Artificial Intelligence*, 2010.
- [33] P. Dai, D. S. Weld, et al. Decision-theoretic control of crowd-sourced workflows. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.
- [34] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2005.
- [35] T. Dean, M. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *Computer Vision and Pattern Recognition (CVPR)*, 2013.
- [36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [37] J. Deng, J. Krause, and L. Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2013.
- [38] J. Deng, O. Russakovsky, J. Krause, M. Bernstein, A. C. Berg, and L. Fei-Fei. Scalable multi-label annotation. In *ACM Conference on Human Factors in Computing Systems (CHI)*, 2014.

- [39] C. Desai, D. Ramanan, and C. Fowlkes. Discriminative models for multi-class object layout. *International Journal on Computer Vision*, 2011.
- [40] T. Deselaers, B. Alexe, and V. Ferrari. Localizing objects while learning their appearance. In *European Conference on Computer Vision (ECCV)*, 2010.
- [41] S. Divvala, A. Farhadi, and C. Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [42] S. Divvala, D. Hoiem, J. Hays, A. Efros, and M. Hebert. An empirical study of context in object detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [43] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [44] D. E. Drake and S. Hougardy. On approximation algorithms for the terminal Steiner tree problem. *Information Processing Letters*, 89(1), 2004.
- [45] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In *Neural Information Processing Systems (NIPS)*, 2001.
- [46] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) challenge - a Retrospective. *International Journal on Computer Vision (IJCV)*, 2014.
- [47] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman. PASCAL Visual Object Classes Challenge (VOC). <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>, 2005-2012.
- [48] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) challenge. *International Journal on Computer Vision (IJCV)*, 88(2):303–338, June 2010.
- [49] A. Farhadi, I. Endres, and D. Hoiem. Attribute-Centric Recognition for Cross-category Generalization. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.
- [50] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [51] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few examples: an incremental bayesian approach tested on 101 object categories. In *Computer Vision and Pattern Recognition (CVPR)*, 2004.

- [52] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition (CVPR)*, 2005.
- [53] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
- [54] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32, 2010.
- [55] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32, 2010.
- [56] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal on Computer Vision (IJCV)*, 59(2), 2004.
- [57] J. Feng, B. Ni, Q. Tian, and S. Yan. Geometric ℓ_p -norm feature pooling for image classification. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.
- [58] V. Ferrari and A. Zisserman. Learning Visual Attributes. In *Neural Information Processing Systems (NIPS)*, 2007.
- [59] A. Frome, G. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov. Devise: A deep visual-semantic embedding model. In *Neural Information Processing Systems (NIPS)*, 2013.
- [60] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *International Conference on Computer Vision (ICCV)*, 2009.
- [61] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [62] H. Gao, G. Barbier, and R. Goolsby. Harnessing the crowdsourcing power of social media for disaster relief. *IEEE Intelligent Systems*, 26(3):10–14, 2011.
- [63] S. Gao, L.-T. Chia, and I. W. Tsang. Multi-layer group sparse coding – for concurrent image classification and annotation. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.
- [64] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. Freeman, 1978.
- [65] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *IJRR*, 2013.

- [66] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [67] B. Goodrich and I. Arel. Reinforcement learning based visual attention with application to face detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [68] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *International Conference on Computer Vision (ICCV)*, 2009.
- [69] S. Gould, T. Gao, and D. Koller. Region-based segmentation and object detection. In *Neural Information Processing Systems (NIPS)*, 2009.
- [70] S. Gould, O. Russakovsky, I. Goodfellow, P. Baumstarck, A. Y. Ng, and D. Koller. The STAIR vision library. <http://ai.stanford.edu/~sgould/svl>, 2009.
- [71] M. R. Greene. Statistics of high-level scene context. *Frontiers in Psychology*, 4, 2013.
- [72] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, Caltech, 2007.
- [73] C. Gu, J. J. Lim, P. Arbelaez, and J. Malik. Recognition using regions. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [74] M. Guillaumin, J. Verbeek, and C. Schmid. Multimodal semi-supervised learning for image classification. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.
- [75] T. Harada and Y. Kuniyoshi. Graphical gaussian vector for image categorization. In *Neural Information Processing Systems (NIPS)*, 2012.
- [76] G. Hartmann, M. Grundmann, J. Hoffman, D. Tsai, V. Kwatra, O. Madani, S. Vijayanarasimhan, I. Essa, J. Rehg, and R. Sukthankar. Weakly supervised learning of object segmentations from web-scale video. In *Workshop on Web-scale Vision and Social Media, European Conference on Computer Vision (ECCV)*, 2012.
- [77] K. He, X. Zhang, S. Ren, , and J. Su. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision (ECCV)*, 2014.
- [78] H. Hedi, J. Frederic, and S. Cordelia. Combining efficient object localization and image classification. In *International Conference on Computer Vision (ICCV)*, 2009.
- [79] G. Heitz, S. Gould, A. Saxena, and D. Koller. Cascaded classification models: Combining models for holistic scene understanding. In *Neural Information Processing Systems (NIPS)*, 2008.

- [80] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.
- [81] J. Hoffman, S. Guadarrama, E. Tzeng, J. Donahue, R. Girshick, T. Darrell, and K. Saenko. LSDA: Large scale detection through adaptation. *CoRR*, abs/1407.5035, 2014.
- [82] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. In *European Conference on Computer Vision (ECCV)*, 2012.
- [83] D. Hoiem, C. Rother, and J. Winn. 3D layout CRF for multi-view object class recognition and segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2007.
- [84] A. Howard. Some improvements on deep convolutional neural network based image classification. In *International Conference on Learning Representations (ICLR)*, 2014.
- [85] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.
- [86] Y. Huang, K. Huang, and T. Tan. Salient coding for image classification. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.
- [87] R. Hwang, D. Richards, and P. Winter. The Steiner Tree problem. *Annals of Discrete Mathematics*, 53, 1992.
- [88] S. J. Hwang and K. Grauman. Reading between the lines: Object localization using implicit cues from image tags. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(6):1145–1158, 2012.
- [89] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *Conference on Human Computation and Crowdsourcing (HCOMP)*, 2010.
- [90] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 1998.
- [91] S. D. Jain and K. Grauman. Predicting sufficient annotation strength for interactive foreground segmentation. *International Conference on Computer Vision (ICCV)*, December 2013.
- [92] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. <http://caffe.berkeleyvision.org/>, 2013.
- [93] T. Kadir and M. Brady. Saliency, scale and image description. *International Journal on Computer Vision (IJCV)*, 45(2), 2001.

- [94] E. Kamar, S. Hacker, and E. Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012.
- [95] E. Kamar, S. Hacker, and E. Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012.
- [96] A. Kanezaki, S. Inaba, Y. Ushiku, Y. Yamashita, H. Muraoka, Y. Kuniyoshi, and T. Harada. Hard negative classes for multiple object detection. In *ICRA*, 2014.
- [97] S. Karayev, M. Fritz, and T. Darrell. Anytime recognition of objects and scenes. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [98] S. Kazemzadeh, V. Ordonez, M. Matten, and T. L. Berg. Referitgame: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014.
- [99] M. Khodabandeh, A. Vahdat, G. Zhou, H. Hajimirsadeghi, M. J. Roshtkhari, G. Mori, and S. Se. Discovering human interactions in videos with limited data labeling. *CoRR*, abs/1502.03851, 2015.
- [100] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, Computer Vision and Pattern Recognition (CVPR)*, 2011.
- [101] G. Kim and A. Torralba. Unsupervised detection of regions of interest using iterative link analysis. In *Neural Information Processing Systems (NIPS)*, 2009.
- [102] K. Murphy, A. Torralba, D. Eaton, and W. Freeman. Object detection and localization using local and global features. *Lecture Notes in Computer Science*, 2006.
- [103] A. Kovashka, S. Vijayanarasimhan, and K. Grauman. Actively selecting annotations among objects and attributes. *International Conference on Computer Vision (ICCV)*, 2011.
- [104] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Neural Information Processing Systems (NIPS)*, 2012.
- [105] D. Kuettel, M. Guillaumin, and V. Ferrari. Segmentation Propagation in ImageNet. In *European Conference on Computer Vision (ECCV)*, 2012.
- [106] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and Simile Classifiers for Face Verification. In *International Conference on Computer Vision (ICCV)*, 2009.
- [107] S. Lad and D. Parikh. Interactively guiding semi-supervised clustering via attribute-based explanations. In *European Conference on Computer Vision (ECCV)*, 2014.

- [108] C. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. *Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [109] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31:2129–2142, Dec 2009.
- [110] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.
- [111] D. Larlus and F. Jurie. Combining appearance models and markov random fields for category level object segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.
- [112] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial Pyramid Matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition (CVPR)*, 2006.
- [113] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal on Computer Vision (IJCV)*, 2008.
- [114] C. Li, D. Parikh, and T. Chen. Automatic discovery of groups of objects for scene understanding. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [115] L.-J. Li, R. Socher, and L. Fei-Fei. Towards total scene understanding: classification, annotation and segmentation in an automatic framework. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [116] T. Li and M. Ogihara. Detecting emotion in music. In *International Society for Music Information Retrieval (ISMIR)*, 2003.
- [117] M. Lin, Q. Chen, and S. Yan. Network in network. In *International Conference on Learning Representations (ICLR)*, 2014.
- [118] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)*, 2014.
- [119] Y. Lin, F. Lv, L. Cao, S. Zhu, M. Yang, T. Cour, K. Yu, and T. Huang. Large-scale image classification: Fast feature extraction and SVM training. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.
- [120] C. J. Lintott, K. Schawinski, A. Slosar, K. Land, S. Bamford, D. Thomas, M. J. Raddick, R. C. Nichol, A. Szalay, D. Andreescu, P. Murray, and J. Vandenberg. Galaxy zoo: morphologies

- derived from visual inspection of galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society (MNRAS)*, 389(3):1179–1189, 2008.
- [121] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller. Turkit: tools for iterative tasks on mechanical turk. In *SIGKDD workshop on human computation*, 2009.
- [122] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing via label transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(12), 2011.
- [123] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal on Computer Vision (IJCV)*, 60(2):91–110, 2004.
- [124] S. Maji and J. Malik. Object detection using a max-margin hough transform. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [125] T. Malisiewicz and A. A. Efros. Improving spatial support for objects via multiple segmentations. In *British Machine Vision Conference (BMVC)*, 2007.
- [126] D. Martin, C. Fowlkes, , and J. Malik. Learning to detect natural image boundaries using brightness and texture. *Neural Information Processing Systems (NIPS)*, 2002.
- [127] V. Melkonian. New primal-dual algorithms for Steiner Tree problems. *Computers and Operations Research*, 34(7), 2007.
- [128] G. A. Miller. Wordnet: A lexical database for english. *Communications ACM*, 38(11), Nov. 1995.
- [129] M. H. Nguyen, L. Torresani, F. de la Torre, and C. Rother. Weakly supervised discriminative localization and classification: a joint learning process. In *International Conference on Computer Vision (ICCV)*, 2009.
- [130] J. Noronha, E. Hysen, H. Zhang, and K. Z. Gajos. PlateMate: crowdsourcing nutritional analysis from food photographs. In *ACM User Interface Software and Technology Symposium (UIST)*, 2011.
- [131] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal on Computer Vision (IJCV)*, 2001.
- [132] A. Oliva and A. Torralba. The role of context in object recognition. *Trends in Cognitive Sciences*, 11(12), 2007.
- [133] V. Ordonez, J. Deng, Y. Choi, A. C. Berg, and T. L. Berg. From large scale image categorization to entry-level categories. In *International Conference on Computer Vision (ICCV)*, 2013.

- [134] W. Ouyang, P. Luo, X. Zeng, S. Qiu, Y. Tian, H. Li, S. Yang, Z. Wang, Y. Xiong, C. Qian, Z. Zhu, R. Wang, C. C. Loy, X. Wang, and X. Tang. Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection. *CoRR*, abs/1409.3505, 2014.
- [135] W. Ouyang and X. Wang. Joint deep learning for pedestrian detection. In *International Conference on Computer Vision (ICCV)*, 2013.
- [136] M. Pandey and S. Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *International Conference on Computer Vision (ICCV)*, 2011.
- [137] C. Pantofaru, C. Schmid, and M. Hebert. Object recognition by integrating multiple image segmentations. In *European Conference on Computer Vision (ECCV)*, 2008.
- [138] D. Parikh and K. Grauman. Relative attributes. In *International Conference on Computer Vision (ICCV)*, 2011.
- [139] D. Parikh, C. L. Zitnick, and T. Chen. Unsupervised learning of hierarchical spatial structures in images. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [140] A. Parkash and D. Parikh. Attributes for classifier feedback. In *European Conference on Computer Vision (ECCV)*, 2012.
- [141] J. Peng, Q. Liu, A. Ihler, and B. Berger. Crowdsourcing for structured labeling with applications to protein folding. In *Proc. ICML Machine Learning Meets Crowdsourcing Workshop*, 2013.
- [142] F. Perronnin, Z. Akata, Z. Harchaoui, and C. Schmid. Towards good practice in large-scale learning for image classification. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [143] F. Perronnin and C. R. Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition (CVPR)*, 2007.
- [144] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *European Conference on Computer Vision (ECCV)*, 2010.
- [145] N. Pinto, D. Cox, and J. DiCarlo. Why is real-world visual object recognition hard? *PLoS Computational Biology*, 4, 2008.
- [146] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*, 1999.
- [147] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

- [148] M. Rohrbach, M. Stark, G. Szarvas, I. Gurevych, and B. Schiele. What Helps Where – And Why? Semantic Relatedness for Knowledge Transfer. *Computer Vision and Pattern Recognition (CVPR)*, 2010.
- [149] H. A. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scenes. In *Neural Information Processing Systems (NIPS)*, 1996.
- [150] O. Russakovsky, A. L. Bearman, V. Ferrari, and L. Fei-Fei. What’s the point: Semantic segmentation with point supervision. *ArXiv e-prints*, 2015.
- [151] O. Russakovsky, J. Deng, Z. Huang, A. Berg, and L. Fei-Fei. Detecting avocados to zucchinis: what have we done, and where are we going? In *International Conference on Computer Vision (ICCV)*, 2013.
- [152] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal on Computer Vision (IJCV)*, 2015.
- [153] O. Russakovsky and L. Fei-Fei. Attribute learning in large-scale datasets. In *European Conference of Computer Vision (ECCV), International Workshop on Parts and Attributes*, 2010.
- [154] O. Russakovsky, L.-J. Li, and L. Fei-Fei. Best of both worlds: human-machine collaboration for object annotation. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [155] O. Russakovsky, Y. Lin, K. Yu, and L. Fei-Fei. Object-centric spatial pooling for image classification. In *European Conference on Computer Vision (ECCV)*, 2012.
- [156] O. Russakovsky and A. Y. Ng. A Steiner tree approach to efficient object detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.
- [157] B. C. Russell, W. T. Freeman, A. A. Effros, J. Sivic, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *Computer Vision and Pattern Recognition (CVPR)*, 2006.
- [158] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. *International Journal on Computer Vision (IJCV)*, 77(1-3), 2008.
- [159] A. L. S. Fidler, M. Boben. Similarity-based cross-layered hierarchical representation for object categorization. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.
- [160] M. A. Sadeghi and A. Farhadi. Recognition using visual phrases. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.

- [161] J. Sanchez and F. Perronnin. High-dim. signature compression for large-scale image classification. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.
- [162] J. Sanchez, F. Perronnin, and T. de Campos. Modeling spatial layout of images beyond spatial pyramids. In *Pattern Recognition Letters (PRL)*, 2012.
- [163] R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2-3):135–168, 2000.
- [164] W. Scheirer, N. Kumar, P. N. Belhumeur, and T. E. Boult. Multi-attribute spaces: Calibration for attribute fusion and similarity search. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [165] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.
- [166] E. Sharon, A. Brandt, and R. Basri. Fast multiscale image segmentation. In *International Conference on Computer Vision (ICCV)*, 1999.
- [167] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *SIGKDD*, 2008.
- [168] B. Siddiquie and A. Gupta. Beyond active noun tagging: Modeling contextual interactions for multi-class active learning. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.
- [169] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep fisher networks for large-scale image classification. In *Neural Information Processing Systems (NIPS)*, 2013.
- [170] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
- [171] H. O. Song, R. Girshick, S. Jegelka, J. Mairal, Z. Harchaoui, and T. Darrell. On learning to localize objects with minimal supervision. In *ICML*, 2014.
- [172] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan. Contextualizing object detection and classification. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.
- [173] A. Sorokin and D. Forsyth. Utility data annotation with Amazon Mechanical Turk. In *Inter-Net08*, 2008.
- [174] M. Spain and P. Perona. Some objects are more equal than others: Measuring and predicting importance. In *European Conference on Computer Vision (ECCV)*, 2008.

- [175] H. Su, J. Deng, and L. Fei-Fei. Crowdsourcing annotations for visual object detection. In *AAAI Human Computation Workshop*, 2012.
- [176] M. J. Swain and D. H. Ballard. Color indexing. *International Journal on Computer Vision (IJCV)*, 1991.
- [177] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.
- [178] K. Tang, A. Joulin, L.-J. Li, and L. Fei-Fei. Co-localization in real-world images. In *CVPR*, 2014.
- [179] K. Tang, R. Sukthankar, J. Yagnik, and L. Fei-Fei. Discriminative segment annotation in weakly labeled video. In *Computer Vision and Pattern Recognition (CVPR)*, 2013.
- [180] S. Thorpe, D. Fize, C. Marlot, et al. Speed of processing in the human visual system. *Nature*, 381(6582):520–522, 1996.
- [181] S. Todorovic and N. Ahuja. Learning subcategory relevances for category recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.
- [182] A. Torralba and A. Efros. An unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.
- [183] A. Torralba, R. Fergus, and W. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2008.
- [184] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27, 2007.
- [185] N. Ueda and K. Saito. Parametric mixture models for multi-labeled text. In *Neural Information Processing Systems (NIPS)*, 2002.
- [186] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *International Journal on Computer Vision (IJCV)*, 2013.
- [187] R. Urtasun, R. Fergus, D. Hoiem, A. Torralba, A. Geiger, P. Lenz, N. Silberman, J. Xiao, and S. Fidler. Reconstruction meets recognition challenge. <http://ttic.uchicago.edu/~rurtasun/rmrc/>, 2013-2014.
- [188] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(9):1582–1596, 2010.

- [189] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Empowering visual categorization with the gpu. *IEEE Transactions on Multimedia*, 13(1):60–70, 2011.
- [190] K. E. A. van de Sande, C. G. M. Snoek, and A. W. M. Smeulders. Fisher and vlad with flair. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [191] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders. Segmentation as selective search for object recognition. In *International Conference on Computer Vision (ICCV)*, 2011.
- [192] S. Vijayanarasimhan and K. Grauman. Multi-level active prediction of useful image annotations for recognition. In *Neural Information Processing Systems (NIPS)*, 2009.
- [193] S. Vijayanarasimhan and K. Grauman. Predicting effort vs. informativeness for multi-label image annotations. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [194] S. Vijayanarasimhan and K. Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. *International Journal on Computer Vision (IJCV)*, 108(1-2), 2014.
- [195] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal on Computer Vision (IJCV)*, 57(2), 2004.
- [196] S. Vittayakorn and J. Hays. Quality assessment for crowdsourced object annotations. In *British Machine Vision Conference (BMVC)*, 2011.
- [197] L. von Ahn and L. Dabbish. Esp: Labeling images with a computer game. In *AAAI Spring Symposium: Knowledge Collection from Volunteer Contributors*, 2005.
- [198] C. Vondrick, D. Patterson, and D. Ramanan. Efficiently scaling up crowdsourced video annotation. *International Journal on Computer Vision (IJCV)*, 2013.
- [199] C. Wah, S. Branson, P. Perona, and S. Belongie. Multiclass recognition and part localization with humans in the loop. In *International Conference on Computer Vision (ICCV)*, 2011.
- [200] C. Wah, G. V. Horn, S. Branson, S. Maji, P. Perona, and S. Belongie. Similarity comparisons for interactive fine-grained categorization. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [201] D. Walther, U. Rutishauser, C. Koch, and P. Perona. Selective visual attention enables learning and recognition of multiple objects in cluttered scenes. *Computer Vision and Image Understanding (CVIU)*, 2005.
- [202] C. Wang, W. Ren, K. Huang, and T. Tan. Weakly supervised object localization with latent category learning. In *ECCV*, 2014.

- [203] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained Linear Coding for image classification. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.
- [204] X. Wang, T. Han, and S. Yan. An HOG-LBP human detector with partial occlusion handling. In *International Conference on Computer Vision (ICCV)*, 2009.
- [205] X. Wang, M. Yang, S. Zhu, and Y. Lin. Regionlets for generic object detection. In *International Conference on Computer Vision (ICCV)*, 2013.
- [206] D. S. Weld, P. Dai, et al. Human intelligence needs artificial intelligence. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.
- [207] P. Welinder, S. Branson, S. Belongie, and P. Perona. The multidimensional wisdom of crowds. In *Neural Information Processing Systems (NIPS)*, 2010.
- [208] J. Whitehill, T.-f. Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Neural Information Processing Systems (NIPS)*, 2009.
- [209] C. Wojek, S. Walk, S. Roth, K. Schindler, and B. Schiele. Monocular visual scene understanding: Understanding multi-object traffic scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2013.
- [210] B. Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *International Conference on Computer Vision (ICCV)*, 2005.
- [211] B. Wu and R. Nevatia. Simultaneous object detection and segmentation by boosting local shape feature based classifier. In *Computer Vision and Pattern Recognition (CVPR)*, 2007.
- [212] J. Wu, J. M. Rehg, and M. D. Mullin. Learning a rare event detection cascade by direct feature selection. In *Neural Information Processing Systems (NIPS)*, 2004.
- [213] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from Abbey to Zoo. *Computer Vision and Pattern Recognition (CVPR)*, 2010.
- [214] K. Yanai and K. Barnard. Image Region Entropy: A Measure of Visualness of Web Images Associated with One Concept. In *ACM Multimedia*, 2005.
- [215] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [216] B. Yao, X. Yang, and S.-C. Zhu. Introduction to a large scale general purpose ground truth dataset: methodology, annotation tool, and benchmarks. In *Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, 2007.

- [217] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *ArXiv e-prints*, 2015.
- [218] X. C. Z Tu, A. L. Yuille, and S. C. Zhu. Image parsing: Unifying segmentation, detection, and recognition. *International Journal on Computer Vision (IJCV)*, 63(2), 2005.
- [219] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.
- [220] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *International Conference on Computer Vision (ICCV)*, 2011.
- [221] A. Zelikovsky. A series of approximation algorithms for the acyclic directed Steiner tree problem. *Algorithmica*, 18, 1997.
- [222] P. Zhang, J. Wang, A. Farhadi, M. Hebert, and D. Parikh. Predicting failures of vision systems. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [223] Y. Zhang, S. Burer, and W. N. Street. Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research (JMLR)*, 7:1315–1338, 2006.
- [224] Y. Zhang and T. Chen. Weakly supervised object recognition and localization with invariant high order features. In *British Machine Vision Conference (BMVC)*, 2010.
- [225] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. *Neural Information Processing Systems (NIPS)*, 2014.
- [226] D. Zhou, J. Platt, S. Basu, and Y. Mao. Learning from the wisdom of crowds by minimax entropy. In *Neural Information Processing Systems (NIPS)*, 2012.
- [227] X. Zhou, K. Yu, T. Zhang, and T. Huang. Image classification using super-vector coding of local image descriptors. In *European Conference on Computer Vision (ECCV)*, 2010.
- [228] X. Zhu, C. Vondrick, D. Ramanan, and C. C. Fowlkes. Do we need more training data or better models for object detection. In *British Machine Vision Conference (BMVC)*, 2012.
- [229] L. Zosin and S. Khuller. On directed Steiner trees. In *Symposium on Discrete Algorithms*, 2002.