

Large-Scale Live Active Learning: Training Object Detectors with Crawled Data and Crowds

Sudheendra Vijayanarasimhan · Kristen Grauman

Received: 24 February 2013 / Accepted: 19 March 2014 / Published online: 12 April 2014
© Springer Science+Business Media New York 2014

Abstract Active learning and crowdsourcing are promising ways to efficiently build up training sets for object recognition, but thus far techniques are tested in artificially controlled settings. Typically the vision researcher has already determined the dataset’s scope, the labels “actively” obtained are in fact already known, and/or the crowd-sourced collection process is iteratively fine-tuned. We present an approach for *live learning* of object detectors, in which the system autonomously refines its models by actively requesting crowd-sourced annotations on images crawled from the Web. To address the technical issues such a large-scale system entails, we introduce a novel part-based detector amenable to linear classifiers, and show how to identify its most uncertain instances in sub-linear time with a hashing-based solution. We demonstrate the approach with experiments of unprecedented scale and autonomy, and show it successfully improves the state-of-the-art for the most challenging objects in the PASCAL VOC benchmark. In addition, we show our detector competes well with popular nonlinear classifiers that are much more expensive to train.

Keywords Object detection · Active learning · Large-scale learning · Hashing · Crowdsourcing · Image annotation

1 Introduction

Object detection is a fundamental vision problem: given an image, which object categories are present, and where? Ongoing research is devoted to developing novel representations and classification algorithms in support of this task, and challenge datasets encourage further progress (Dalal and Triggs 2005; Felzenszwalb et al. 2009; Vedaldi et al. 2009; Lampert et al. 2008; Everingham et al. 2010). Today’s best-performing detection methods employ discriminative learning together with window-based search, and assume that a large number of cleanly labeled training examples are available. For example, thousands of bounding box annotations per category is standard.

Given the substantial human effort required to gather good training sets—as well as the expectation that *more* data is almost always advantageous—researchers have begun to explore novel ways to collect labeled data. Both *active learning* and *crowd-sourced labeling* are promising ways to efficiently build up training sets for object recognition. Active learning work shows how to minimize human effort by focusing label requests on those that appear most informative to the classifier (Kapoor et al. 2007; Qi et al. 2008; Vijayanarasimhan and Grauman 2008; Joshi et al. 2009; Siddiquie and Gupta 2010), whereas crowd-sourcing work explores how to package annotation tasks such that they can be dispersed effectively online (von Ahn and Dabbish 2004; Russell et al. 2007; Sorokin and Forsyth 2008; Welinder and Perona 2010; Deng et al. 2009). The interesting questions raised in these areas—such as dealing with noisy labels, measuring reliability, mixing strong and weak annotations—make it clear that data collection is no longer a mundane necessity, but a thriving research area in itself.

However, while ostensibly intended to distance algorithm developers from the data collection process, in practice exist-

Communicated by Martial Hebert.

S. Vijayanarasimhan · K. Grauman (✉)
University of Texas at Austin, Austin, TX, USA
e-mail: grauman@cs.utexas.edu

S. Vijayanarasimhan
e-mail: svnaras@cs.utexas.edu

ing techniques are tested in artificially controlled settings. Specifically, we see four limiting factors. First, previous work uses “sandbox” datasets, where the vision researcher has already determined the dataset’s source and scope, meaning there is a fixed (and possibly biased) set of images that will even be considered for labeling. In fact, active learning methods have only been tested on sandbox data where the true labels are really known, and merely temporarily withheld from the selection algorithm. These common simulations likely inflate the performance of both active and passive learners, since anything chosen for labeling is relevant. Second, nearly all work targets the active *image classification* problem—not detection—and so images in the unlabeled pool are artificially assumed to contain only one prominent object. Third, most crowd-sourced collection processes require iterative fine-tuning by the algorithm designer (e.g., revising task requirements, pruning responses, barring unreliable Mechanical Turkers) before the data is in usable form. Fourth, the computational complexity of the active selection process is generally ignored, yet it is critical when running a live system to avoid keeping the human annotators idle. Thus, it is unknown to what extent current approaches could translate to real settings.

Our goal is to take crowd-sourced active annotation out of the “sandbox”. We present an approach for *live learning* of object detectors, in which the system directly interacts with human annotators online and iteratively poses annotation requests to refine its models. Rather than fill the data pool with some canned dataset, the system itself gathers possibly relevant images via keyword search (we use Flickr). It repeatedly surveys the data to identify unlabeled sub-windows that are most uncertain according to the current model, and generates tasks on an online crowd-sourcing service (Mechanical Turk) to get the corresponding bounding box annotations. After an annotation budget is spent, we evaluate the resulting detectors both on benchmark data, as well as a novel test set from Flickr. Notably, throughout the procedure *we do not intervene with what goes into the system’s data pool, nor the annotation quality from the hundreds of online annotators.*

To make the above a reality requires handling some important technical issues. Active selection for window-based detection is particularly challenging since the object extents (bounding boxes) are unknown in the unlabeled examples; naively one would need to evaluate all possible windows within the image in order to choose the most uncertain one. This very quickly leads to a prohibitively large unlabeled pool to evaluate exhaustively. Thus, we introduce a novel part-based detector amenable to linear classifiers, and show how to identify its most uncertain instances in sub-linear time with a hashing-based solution.

We show that our detector strikes a good balance between speed and accuracy, with results competitive with and even exceeding the state-of-the-art on the PASCAL VOC. Most

importantly, we show successful live learning in an uncontrolled setting. The system learns accurate detectors with much less human effort than strong baselines that rely on human-verified keyword search results.

2 Related Work

Object detection has received various treatments in the literature; see [Everingham et al. \(2010\)](#) and references therein for an overview. Currently window-based approaches based on gradient features and subwindow parts provide state-of-the-art results using discriminative classifiers. A known limitation, however, is their fairly significant computational expense, due both to the need to search exhaustively through all windows in the image, as well as the classifiers’ complexity (e.g., SVMs with nonlinear kernels or latent variables ([Vedaldi et al. 2009](#); [Felzenszwalb et al. 2009](#))).

Various ways to reduce detection time have been explored, including cascades ([Vedaldi et al. 2009](#); [Viola and Jones 2001](#)), branch-and-bound search ([Lampert et al. 2008](#)), or jumping windows ([Chum and Zisserman 2007](#)). To reduce classifier training and testing costs, simpler linear models are appealing. While linear models tend to underperform the state of the art with common representations (e.g., see the experiments reported in ([Vedaldi et al. 2009](#); [Boureau et al. 2010](#))), recent work in image classification shows very good results when also incorporating sparse coding and feature pooling ([Yang et al. 2009](#); [Wang et al. 2010](#); [Boureau et al. 2010](#)). We propose a part-based object model that exploits a related representation, and show it to be competitive with state-of-the-art detection results. To our knowledge, no previous work considers sparse coding and linear models for object detection.

An alternative way to improve detection time for a part-based detector is to reduce the number of filter convolutions it requires to run on a new image. Multi-task learning can encourage a set of object detectors to share features, so that at detection time fewer local part features need to be extracted ([Torralba et al. 2007](#)). With similar motivation, recent work shows how to use a shared dictionary of part filters to estimate filter responses in terms of sparse matrix-vector products ([Song et al. 2012](#); [Pirsiavash and Ramanan 2012](#)); this greatly reduces the computation time compared to standard part filter convolutions.

While these efforts concentrate on limiting feature extraction time, the system of [Dean et al. \(2013\)](#) (introduced subsequent to our work) also pushes on reducing the standard linear run-time dependency on the number of object categories. In that work, a hashing-based solution replaces convolution dot products with hash-table probes, thereby sampling filter responses in time independent of the size of the filter bank. The approach can run a simplified deformable part

model (with no root filter and equally sized parts) for 100K object classes simultaneously on a single machine in just 20 seconds. Our approach also involves hashing, but for the sake of scalable active learning. Using our hyperplane hash functions (Jain et al. 2010; Vijayanarasimhan et al. 2014), we search for uncertain data in sub-linear time, breaking the linear-time dependence on the size of the unlabeled data pool suffered by existing active learning methods.

Active learning has been shown to better focus annotation effort for image recognition tasks (Kapoor et al. 2007; Qi et al. 2008; Joshi et al. 2009) and region labeling (Vijayanarasimhan and Grauman 2008; Siddiquie and Gupta 2010). However, no previous work uses active learning to train a window-based detector. To do so introduces major scalability issues, which we address with a new linear detector combined with a hashing algorithm we developed (Jain et al. 2010; Vijayanarasimhan et al. 2014) for sub-linear time search of the unlabeled pool. Further, all previous work tests active selection only in a sandbox, where the true labels are already known for the data.

Researchers have investigated issues in annotation tools and large-scale database collection for recognition. Keyword-based search is often used for dataset creation, and several recent efforts integrate crowd-sourced labeling (von Ahn and Dabbish 2004; Russell et al. 2007; Deng et al. 2009) or online and incremental learning (Li et al. 2007). Even with a human in the loop, annotation precision varies when using Web interfaces and crowds, and so some research explores ways to automatically provide quality assurance (Sorokin and Forsyth 2008; Welinder and Perona 2010). Other work attempts to directly learn object models from noisy keyword search (Fergus et al. 2005; Li et al. 2007; Vijayanarasimhan and Grauman 2008); however, such methods assume a single prominent object of interest per image, whereas for detection we will have cluttered candidate images that require a bounding box to identify the object of interest.

Overall, previous active learning methods focus on image classification, and/or demonstrate results under controlled settings on prepared datasets of modest scale. Ours is the first complete end-to-end approach for scalable, automatic online learning of object detectors.

This article expands the work we originally published in Vijayanarasimhan and Grauman (2011). The primary additions are two new results sections analyzing the live annotation consistency, variation among annotators, and our automatic consensus procedure (Sect. 4.5, Figs. 13 and 14), and analyzing why live active annotation requests have an advantage over existing keyword search data collection strategies, both quantitatively and qualitatively (Sect. 4.4, Table 4, and Figs. 11, 12a, and b). In addition, we provide examples of the proposed detector's output to portray its strengths and failure modes (Fig. 7), and we refine the overall text for clarity.

3 Approach

Our goal is to enable online active crowd-sourced object detector training. Given the name of a class of interest, our system produces a detector to localize novel instances using automatically obtained images and annotations. To make this feasible, we first propose a part-based linear support vector machine (SVM) detector, and then show how to identify its uncertain examples efficiently using a hashing scheme.

3.1 Object Representation and Linear Classifier

We first introduce our part-based object model. Our goal is to design the representation such that a simple linear classifier will be adequate for robust detection. A linear model has many complexity advantages important to our setting: (i) SVM training requires time linear in the number of training examples, rather than cubic (Joachims 2006), (ii) classification of novel instances requires constant time rather than growing linearly with the number of training examples, (iii) exact incremental classifier updates are possible, which makes an iterative active learning loop practical, and (iv) hash functions enable sub-linear time search to map a *query hyperplane* to its nearest points according to a linear kernel (Jain et al. 2010; Vijayanarasimhan et al. 2014).

Inspired by recent findings in sparse coding for image classification (Yang et al. 2009; Wang et al. 2010; Boureau et al. 2010), we explore a detection model based on sparse coding of local features combined with a max pooling scheme. Previous representations pool coded SIFT features in a single global window or in a fixed class-independent hierarchy of grid cells (i.e., a spatial pyramid (SP) structure). While sufficient for whole-image classification, we instead aim to represent an *object* separately from its context, and to exploit its part-based structure with *class-dependent* subwindow pooling.

To this end, we propose an object model consisting of a root window r , multiple part windows $\{p_1, \dots, p_P\}$ that overlap the root, and context windows $\{c_1, \dots, c_C\}$ surrounding it. See Fig. 1. Let $O = [r, p_1, \dots, p_P, c_1, \dots, c_C]$ denote a candidate object configuration within an image, and let $\phi(W)$ denote the sparse feature encoding for local image descriptors extracted from a given subwindow W (to be defined below). The detector scores a candidate configuration as a simple linear sum:

$$\begin{aligned} f(O) &= \mathbf{w}^T \phi(O) \\ &= \mathbf{w}_r^T \phi(r) + \sum_{i=1}^P \mathbf{w}_{p_i}^T \phi(p_i) + \sum_{i=1}^C \mathbf{w}_{c_i}^T \phi(c_i), \end{aligned} \quad (1)$$

where \mathbf{w} denotes the learned classifier weights, which we obtain with SVM training. As reflected above, this weight vector is comprised of the weights for the root, parts, and

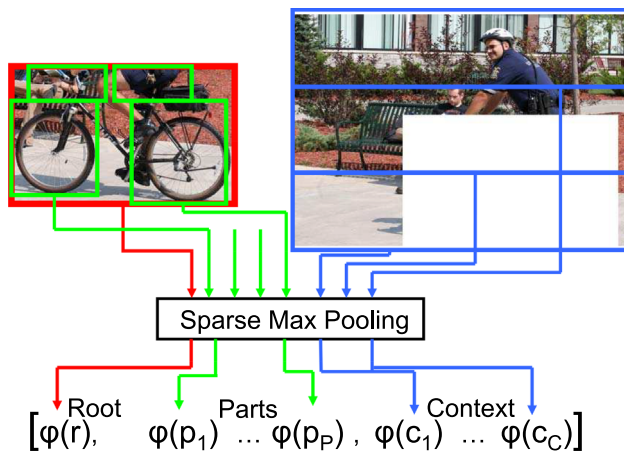


Fig. 1 Our part-based object representation consists of a root window for the entire object, a set of class-specific object part windows located within the root, and a set of context windows surrounding the root. The feature encoding $\phi(\cdot)$ captures the maximum sparse coding coefficient found among all local features within a window for a given visual word

context windows, respectively; that is,

$$\mathbf{w} = [\mathbf{w}_r, \mathbf{w}_{p_1}, \dots, \mathbf{w}_{p_P}, \mathbf{w}_{c_1}, \dots, \mathbf{w}_{c_C}]. \quad (2)$$

We next flesh out the window descriptions and feature encoding details, and then analyze how our model relates to existing detectors. Section 3.2 will explain how we obtain candidate root placements.

3.1.1 Window Descriptions

Given a novel test image, we first extract local image descriptors; we use a dense multi-scale sampling of SIFT in our implementation. Each window type (r , p_i , or c_j) uses these features to create its encoding $\phi(\cdot)$. The *root window* provides a global summary of the object appearance, and is invariant to spatial translations of features within it.

Similarly, each *part window* summarizes the local features within it, discarding their positions; however, the location of each part is defined relative to the current root, and depends on the object class under consideration (i.e., bicycles and cars each have a different configuration of the p_i windows). Thus, they capture the locations and spatial configurations of parts of the object. We train with the part locations and bounds learned by the detector in Felzenszwalb et al. (2009) on an initial labeled set. While our implementation uses a fixed number of parts per object (namely $P = 6$, following Felzenszwalb et al. (2009)), it is possible some objects would perform better with fewer or more parts. The parameter P could be adjusted per class with cross validation in cases where sufficient initial labeled examples are available.

While the parts could alternatively be requested directly from annotators, we prefer to use the automatic learning

approach both for ease and since human annotators need not know which parts will be most discriminative. These parts need not be nameable parts apparent to the human eye; rather, they are those subwindows of the object that appear consistently in the positive exemplars but not in the negative exemplars. For example, for a car detector, a useful part might be where the bottom of the wheel meets the pavement, as opposed to a patch on the entire wheel itself.

The *context windows* incorporate contextual cues surrounding the object, such as the presence of “sky”, “ground”, “road”, etc., and also help discard poorer candidate windows that cover only parts of objects (in which case object features spill into the context window). We create the context windows using a 3×1 partition of the root window r 's complement, as shown in the top right of Fig. 1.

The intuition behind breaking the context windows only vertically in image, as opposed to also dividing features in bins across the horizontal axis, is that other objects appearing relative to the root object window tend to appear with more regularity from top to bottom. In contrast, from left to right they are often interchangeable. For example, sky is above the car and ground is below it, but a person standing beside a car is equally likely on the left or right. In preliminary trials, the 3×1 partition was more effective than variations using fewer cells or horizontal splits, and so we simply fixed this layout for all objects. We find that providing this context strengthens certain categories, which agrees with recent findings (Uijlings et al. 2009; Lee and Grauman 2010).

3.1.2 Feature Encoding

Each window is represented using a nonlinear feature encoding based on sparse coding and max-pooling, which we refer to as sparse max pooling (SMP). The SMP representation is related to the well-known bag-of-features (BoF); however, unlike BoF, each component local descriptor is first encoded as a combination of *multiple* visual words, and the weights are then pooled within a region of interest using the max function.

Offline, we cluster a large corpus of randomly selected features to obtain a dictionary of $|V|$ visual words: $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_{|V|}]$, where each column $\mathbf{v}_i \in \mathfrak{R}^{128}$ is a cluster center in SIFT space. For any window W (whether it is the root, a part, or a context window), let $F = \{\mathbf{f}_i\}_{i=1}^{|F|}$ be the set of local features falling within it, where each $\mathbf{f}_i \in \mathfrak{R}^{128}$ is a SIFT descriptor. We represent this window with a sparse $|V|$ -dimensional vector, as follows.

First, each feature \mathbf{f}_i is quantized into a $|V|$ -dimensional sparse vector \mathbf{s}_i that approximates \mathbf{f}_i using some existing sparse coding algorithm and the dictionary \mathbf{V} , that is, $\mathbf{f}_i \approx \mathbf{s}_i \mathbf{V}$. Taking this encoding for every \mathbf{f}_i as input, the SMP representation of W is given by:

$$\phi(W) = [\phi^1, \dots, \phi^{|V|}], \text{ where} \tag{3}$$

$$\phi^j = \max(s_i(j)), i = 1, \dots, |F|,$$

for $j = 1, \dots, |V|$, and $s_i(j)$ is the j -th dimension of the sparse vector encoding the i -th original feature, f_i . In other words, for a given subwindow W , for each dictionary word $j = 1, \dots, |V|$, we record the maximum coefficient associated with any of the $|F|$ local SIFT descriptors appearing in that window. This yields a $|V|$ -dimensional descriptor for that window. Finally, we normalize $\phi(W)$ by its L_2 norm.¹

The rationale behind the SMP window encoding is twofold: the sparse coding gives a fuller representation of the original features by reflecting nearness to multiple dictionary elements (as opposed to BoF’s usual hard vector quantization), while the max pooling gives better discriminability amidst high-variance clutter (Boureau et al. 2010). See (Boureau et al. 2010; Yang et al. 2009) for useful comparisons between various sparse coding approaches, which shows their clear advantage when combined with a linear kernel as compared to the popular BoF.

3.1.3 Relationship to Existing Detection Models

Our model intentionally strikes a balance between two recent state-of-the-art detection models: (i) a nonlinear SVM with a spatial pyramid (SP) in which each grid cell is a histogram of unordered visual words (Vedaldi et al. 2009), and (ii) a latent SVM (LSVM) with a deformable part model, in which each part is a rigid histogram of ordered oriented gradients (Felzenszwalb et al. 2009). See Fig. 2.

On the one hand, the SP model is robust to spatial translations of local features within each grid cell. On the other hand, its nonlinear kernels (required for good performance, as shown by Vedaldi et al. (2009)) makes the classifier quite expensive to train and test, and rigid class-independent bins may fail to capture the structure of the best parts on an object (see Fig. 2a). In contrast, the LSVM model can robustly capture the parts, since it learns multiple part filters that deform relative to the root. However, its dynamic programming step to compute parts’ alignment makes it expensive to train. Furthermore, its use of the spatially dense gradient histograms for both the root and parts make it less tolerant to *internal* shifts and rotations (see Fig. 2b).

Our model attempts to incorporate positive aspects of the above two models, while maintaining a much lower computational cost. In particular, we have class-specific part configurations, like Felzenszwalb et al. (2009), but they are fixed relative to the root, like Vedaldi et al. (2009). Our SMP-based encoding is robust to shifts within the part and object win-

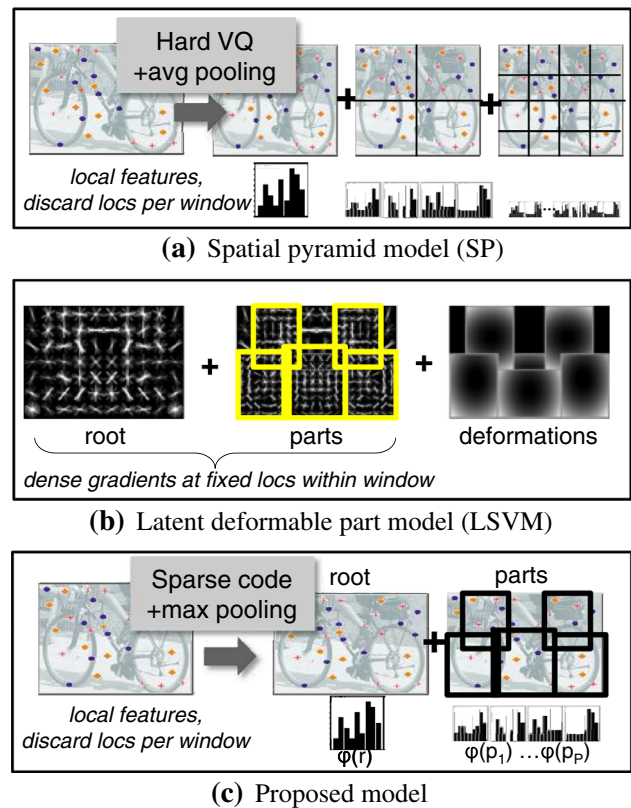


Fig. 2 Sketch to illustrate contrasts with related existing models. See text for details

dows, thereby tolerating some deformation to the exact part placement without needing the additional dynamic programming alignment step during detection. In short, by utilizing a part-based representation and a linear classifier, our approach provides a very good trade-off in terms of model complexity and accuracy.

3.2 Generating Candidate Root Windows

So far we have defined a representation and scoring function for any candidate window. Now we discuss how to generate the candidates, whether in novel test images or unlabeled images the system is considering for annotation. A thorough but prohibitively expensive method would be the standard *sliding window* approach; instead, we use a grid-based variant of the *jumping window* method of Chum and Zisserman (2007), Vijayanarasimhan and Kapoor (2010).

The jumping window approach generates candidate windows with a Hough-like projection using visual word matches, and prioritizes these candidates according to a measure of how discriminative a given word and coarse location is for the object class (see Fig. 3). First, each root window in the training images is divided into an $N \times M$ grid. Let $W_{loc}(r)$ denote a root window’s position and scale. Given a training window r and a visual word v occurring at grid posi-

¹ We use Locality-constrained Linear Coding (LLC) by Wang et al. (2010) to obtain the sparse coding, though other algorithms could also be used for this step.

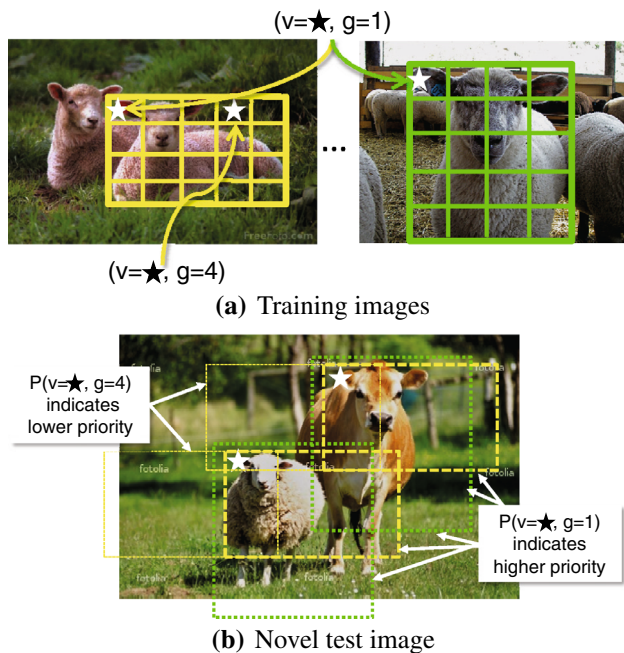


Fig. 3 Illustration of jumping window root candidates. Grid cells serve to refine the priority given to each box (but do not affect its placement). Here, location $g = 1$ has higher priority than $g = 4$ for visual word $v = \star$ since it appears more consistently in the training images (a). Therefore, at test time (b) we prioritize the object search to focus on those windows that are more highly rated as candidate roots

tion $g \in \{1, \dots, NM\}$, we record the triplet $(v, g, W_{loc}(r))$. We build a lookup table indexed by the v entries for all training examples. Then, given a test image, for each occurrence of a visual word v , we use the lookup table to retrieve all possible $W_{loc}(r)$'s, and project a bounding box in the test image relative to that v 's position. Note, candidates can vary in aspect ratio and scale.

The grid cell g in each triple is used to assign a priority score to each candidate, since we may not want to examine all possible candidates mapped from the lookup table. Specifically, we score a given pair (v, g) based on how predictive it is for the true object bounding box across the training set: $P(v, g)$ is the fraction of the occurrences of word v that appear at grid location g . This function gives a higher score to bounding boxes where the visual word occurs consistently across positive training examples at a particular position (see Fig. 3).

Given a test image, we take the top K candidate jumping windows based on their priority scores. The detector is run only on these boxes. In experiments, we obtain 95 % recall on most categories when taking just $K = 3,000$ candidates per test image. The same level of recall would require up to 10^5 bounding boxes if using sliding windows (see Vedaldi et al. (2009)).

As discussed in Sect. 2, there exist other strategies to restrict the search space for an object in an image, includ-

ing cascades that reject many negatives with simple tests (e.g., Viola and Jones (2001)). However, whereas a cascade entails applying a sequence of tests to a given window, the jumping windows approach we take discovers a small set of promising candidates as a single batch. This part of our design is amenable to our hashing-based approach to perform efficient active learning, which we will discuss in the next section. Essentially, having extracted all jumping windows in all unlabeled images offline, we can precompute an efficient data structure to index them for retrieval later in the active learning loop.

3.3 Active Selection of Object Windows

We initialize our online active learning system with a linear SVM trained with a small number of labeled examples for the object. Then, it crawls for a pool of potentially relevant unlabeled data using keyword search with the object name (i.e., it downloads a set of images tagged *dog* when learning to detect dogs). We want to efficiently determine which images among those retrieved should be labeled next by the human annotators. As an active learning criterion, we use the “simple margin” selection method for SVMs (Tong and Koller 2000), a widely used criterion that seeks points that most reduce the version space. Given an SVM with hyperplane normal w and an unlabeled pool of data $\mathcal{U}_O = \{\phi(O_1), \dots, \phi(O_n)\}$, the point that minimizes the distance to the current decision boundary is selected for labeling:

$$O^* = \arg \min_{O_i \in \mathcal{U}_O} |w^T \phi(O_i)|. \quad (4)$$

A naive application of this criterion entails computing the classifier response on all unlabeled data, ranking them by $|w^T \phi(O_i)|$. However, even with a linear classifier, exhaustively evaluating all unlabeled examples at each iteration is prohibitively expensive. Whereas previous active learning work is generally unconcerned about the amount of time it actually takes to compute the next labeling request, it becomes a real issue when working out of the sandbox, since we have live annotators awaiting the next labeling jobs and massive unlabeled data pools. In particular, since we need to apply the active selection function at the level of the *object*, not the entire *image*, we have an enormous number of instances—all bounding boxes within the unlabeled image data. Even using jumping windows, thousands of images yield millions of candidates. Thus, a simple linear scan of all unlabeled data is infeasible.

Therefore, we adopt our *hyperplane-hashing* algorithm (Jain et al. 2010; Vijayanarasimhan et al. 2014) to identify the most promising candidate windows in sub-linear time. The main idea of hyperplane hashing (H-Hash) is to prepare a hash table in such a way that when a “query” hyperplane w

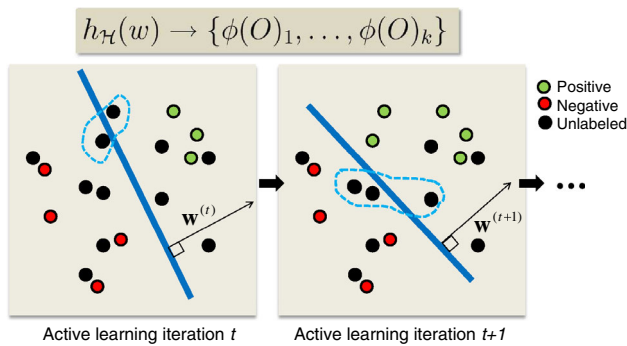


Fig. 4 We use the hyperplane hashing algorithm (Jain et al. 2010; Vijayanarasimhan et al. 2014) to map the linear classifier parameters w directly to a set of unlabeled data points $\{\phi(O)_1, \dots, \phi(O)_k\}$ that are nearest the hyperplane decision boundary. After the actively selected points are labeled by an annotator, the classifier parameters are updated, and the process repeats

is hashed in, it maps directly to hash buckets containing data points that are near its surface. Using such a data structure, we can be assured of finding the unlabeled instances that are most uncertain according to the margin criterion—without exhaustively applying the classifier to every one of them. It is important to distinguish this hashing formulation from widely used hash functions for approximate nearest neighbor search. Whereas work on large-scale similarity search offers hash functions that will map a query *point* to its nearby points, H-Hash provides hash functions that map a *hyperplane* to nearby points anywhere along its surface.

Briefly, the H-Hash algorithm works as follows. Assuming we have normalized data vectors, the instances that are nearest to the surface of a hyperplane parameterized by the normal w are those that are close to perpendicular to w . Therefore, H-Hash generates a randomized hash function that is locality-sensitive for the angle between a database point and the hyperplane normal. The closer their separation is to 90° , the more likely they are to collide in the hash table; that is, they are more likely to be assigned the same hash bit. By concatenating the outputs of a series of such hash functions, we obtain a binary key that indexes a single bucket in the hash table. First, all the unlabeled data is hashed into the table. Then, given a “query hyperplane” w , we can hash directly to the uncertain points, with high probability. See Fig. 4.

Formally, let \mathcal{U}_I denote the set of unlabeled images, and \mathcal{U}_O denote the pool of candidate object windows obtained using the jumping window predictor on \mathcal{U}_I . Note that $|\mathcal{U}_O| = K \times |\mathcal{U}_I|$. The locality-sensitive hash family \mathcal{H} generates randomized functions with two-bit outputs:

$$h_{\mathcal{H}}(z) = \begin{cases} h_{u,v}(\phi(O_i), \phi(O_i)), & \text{if } z \text{ is a database vector,} \\ h_{u,v}(w, -w), & \text{if } z \text{ is a query hyperplane,} \end{cases}$$

where the component function is defined as

$$h_{u,v}(a, b) = [\text{sign}(u^T a), \text{sign}(v^T b)]. \tag{5}$$

The function $\text{sign}(u^T a)$ returns 1 if $u^T a \geq 0$, and 0 otherwise, and u and v are sampled from a standard multivariate Gaussian, $u, v \sim \mathcal{N}(0, I)$. As the notation indicates, the functions accommodate both hyperplane parameter vectors and unlabeled data vectors as input. These functions guarantee high probability of collision for a query hyperplane and the points nearest to its boundary. The two-bit hash limits the retrieved points’ deviation from the perpendicular by constraining the angle with respect to both w and $-w$. See Jain et al. (2010), Vijayanarasimhan et al. (2014) for more details.

We use these functions to hash the crawled data into the table. Then, at each iteration of the active learning loop, we hash the current classifier as a query, and directly retrieve examples closest to its decision boundary.² We search only those examples, i.e., we compute $|w^T \phi(O_i)| = |f(O_i)|$ for each one, and rank them in order of increasing value. At this point, each unlabeled image is associated with the score of its jumping window with the smallest margin criterion value. Finally, the system issues a label request for the top T images under this ranking. Since we only need to evaluate the classifier for examples that fall into a particular hash bucket—typically less than 0.1 % of the total number of unlabeled examples—this strategy combined with our new detector makes online selection from large datasets feasible.

We stress that the unlabeled data is only hashed into the table once, before any active learning iterations take place. Then, during active learning, it is only the current classifier hyperplane that is hashed in, and it is the only thing that changes as learning proceeds. This property is essential to the design of H-Hash, since it means the primary hashing overhead is all paid up front. Only minimal overhead is incurred during online live learning, namely, two inner products for each hash bit to compute the hyperplane’s single hash key.

3.4 Online Annotation Requests

To automatically obtain annotations on the actively selected examples, our system posts jobs on Mechanical Turk, where it can pay workers to provide labels. The system gathers the images containing the most uncertain bounding boxes, and the annotators are instructed to use a rectangle-drawing tool to outline the object of interest with a bounding box (or else to report that none is present). We ask annotators to further subdivide instances into “normal”, “truncated”, or “unusual”, consistent with PASCAL annotations, and to flag images containing more than three instances. Figure 5 shows the annotation interface.

² Hyperplane hashes can be used with existing approximate nearest neighbor search algorithms; we use the formulation by Charikar (2002), which guarantees the probability with which the nearest neighbor will be returned.



Fig. 5 Our Mechanical Turk interface used to obtain bounding boxes on the actively selected examples. The worker is asked whether the object of interest is present in the image (here, the object of interest is *bicycle*), and if there is one, he/she must outline it with a bounding box. Furthermore, we request the PASCAL-style qualifiers as to whether the object instance is truncated or unusual

While MTurk provides easy access to a large number of annotators, the quality of their labels varies. Thus, we design a simple but effective approach to account for the variability. We issue each request to 10 unique annotators, and then cluster their bounding boxes using mean shift to obtain a consensus. We keep only those clusters with boxes from more than half of the annotators. Finally, we obtain a single representative box from each cluster by selecting the one with the largest mean overlap with the rest.

Note how each image consists of thousands of unlabeled window instances, each of which serves as a candidate active learning query. Once a single image annotation is obtained, however, it tells us the labels for all windows within it.

3.5 Training the Detector

Training our detector entails learning the linear SVM weights in Eq. 2 to distinguish windows that contain the object of interest from all others. To limit the number of negative windows used to train, we mine for “hard” negatives: at each iteration, we apply the updated classifier to the newly labeled images, and add the 10 top-scoring windows as negatives if they overlap the target class by less than 20 %.

We can now actively train an object detector automatically using minimal crowd-sourced human effort. To recap, the main loop consists of using the current classifier to generate candidate jumping windows, storing all candidates in a hash table, querying the hash table using the hyperplane classifier, giving the actively selected examples to online annotators, taking their responses as new ground truth labeled data, and updating the classifier. See Fig. 6 for a summary of the complete system.

4 Results

There are six main components in our experiments. First, we compare the proposed detector to the most closely related

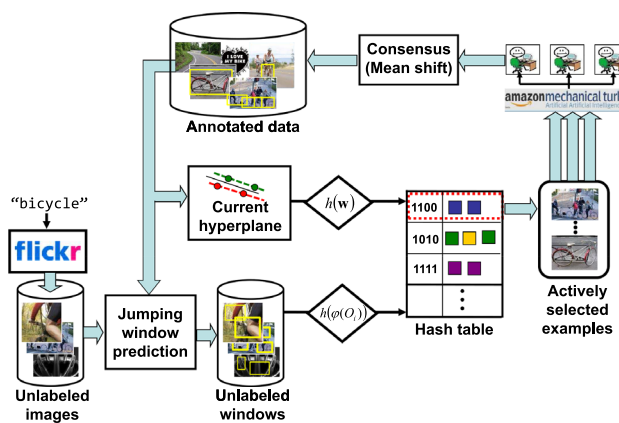


Fig. 6 Summary of our system for live learning of object detectors

state-of-the-art techniques (Sect. 4.1). Second, we validate our large-scale active selection approach with benchmark data (Sect. 4.2). Third, we deploy our complete live learning system with crawled images, and compare to strong baselines that request labels for the keyword search images in a random sequence (Sect. 4.3). Fourth, we analyze the types of requests made by the live active learning system, and try to understand their advantage over the passive keyword-search protocol (Sect. 4.4). Fifth, we analyze the live annotation collection results (Sect. 4.5). Finally, we report comparative run-times for our method and prior detectors in order to emphasize how our contributions make live learning scalable (Sect. 4.6).

We use two datasets in our experiments: the PASCAL VOC 2007, and a new Flickr dataset. More details about the datasets are given below.

Implementation Details We use dense SIFT at three scales (16, 24, 32 pixels) with grid spacing of 4 pixels, for 30K features per image. We obtain $|V| = 56,894$ visual words with two levels of hierarchical k -means on a sample of training images. We use the fast linear SVM code `svm_perf` (Joachims 2006), $C = 100$. We use the LLC code (Wang et al. 2010), and set k , the number of non-zero values in the sparse vector s_i to 5, following Wang et al. (2010). We use $P = 6$ parts per object from each of a 2-mixture detector from Felzenszwalb et al. (2009) trained on PASCAL data, take $T = 100$ instances per active cycle, set $K = 3000$, and $N, M = 4$. We fix $N^p = 500$ and $\epsilon' = 0.01$ for the hash table (Jain et al. 2010). During detection we run non-max suppression on top ranked boxes and select 10 per image. We score all results with standard PASCAL metrics and train/test splits.

4.1 Comparison to State-of-the-Art Detectors

First we compare our detector to the algorithms with the current best performance on VOC 2007 benchmark of 20

Table 1 Average precision compared to a spatial pyramid BoF baseline (BoF SP), a sparse coding max pooling spatial pyramid baseline modeled after Wang et al. (2010) (LLC SP), and two state-of-the-art approaches (Felzenszwalb et al. 2009; Vedaldi et al. 2009) on the PASCAL VOC, where all methods are trained and tested on the standard benchmark splits

Ours	Classif	Parts		Feats		Cands	Aero.	Bicyc.	Bird	Boat	Bottl	Bus	Car	Cat	Chair	Cow	Dimin.	Dog	Horse	Motor.	Person	Potte.	Sheep	Sofa	Train	Tvmom.	Mean
		Yes	Single	Jump	48.4																						
BoF SP	Linear	No	Single	Jump	30.4	43.1	6.9	3.5	10.8	35.8	45.0	17.7	11.5	24.6	3.5	18.0	43.5	44.0	15.3	1.5	19.1	14.7	35.7	34.9	23.0		
LLC SP	Linear	No	Single	Jump	35.9	46.7	6.4	6.3	16.5	45.6	49.8	26.7	12.5	27.3	6.8	18.2	44.9	45.0	18.2	4.6	23.2	22.6	41.3	42.0	27.0		
LSVM+HOG Felzenszwalb et al. (2009)	Nonlinear	Yes	Single	Slide	32.8	56.8	2.5	16.8	28.5	39.7	51.6	21.3	17.9	18.5	25.9	8.8	49.2	41.2	36.8	14.6	16.2	24.4	39.2	39.1	29.1		
SP+MKL Vedaldi et al. (2009)	Nonlinear	No	Multiple	Jump	37.6	47.8	15.3	15.3	21.9	50.7	50.6	30.0	17.3	33.0	22.5	21.5	51.2	45.5	23.3	12.4	23.9	28.5	45.3	48.5	32.1		

Bold values indicate the best performing method per class

objects (as found at the time of these experiments), as well as our own implementation of two other relevant baselines. All methods are trained and tested with the same PASCAL-defined splits.

Table 1 shows the results. The first three rows all use the same original SIFT features, a linear SVM classifier, and the same jumping windows in the test images. They differ, however, in the feature coding and pooling. The BoF SP baseline maps the local features to a standard 3-level spatial pyramid bag-of-words descriptor with L_2 -normalization. The LLC SP baseline applies sparse coding and max pooling within the SP cells. LLC SP is the method of Wang et al. (2010); note, however, we are applying it for detection, whereas the authors propose their approach for image classification.

The linear classifier with standard BoF coding is the weakest. The LLC SP baseline performs quite well in comparison, but its restriction to a global SP structure appears to hinder accuracy. In contrast, our detector improves over LLC SP noticeably for most objects (compare rows 1 and 3), likely due to its part windows.

Our detector is competitive with both of the state-of-the-art approaches discussed in Sect. 3.1: SP+MKL (Vedaldi et al. 2009), which uses a cascade of classifiers that culminates with a learned combination of nonlinear SVM kernels over multiple feature types, and LSVM+HOG (Felzenszwalb et al. 2009), which uses the LSVM and deformation models for parts. In fact, our detector outperforms all existing results for six of the 20 objects, improving the state-of-the-art. At the same time, it is significantly faster to train (about 50 to 600 times faster; see Table 5).

The classes where we see most improvements seem to make sense, too: our approach outperforms the rigid SP representation used in Vedaldi et al. (2009) for cases with more class-specific part structure (*aeroplane, bicycle, train*), while it outperforms the dense gradient parts used in Felzenszwalb et al. (2009) for the more deformable objects (*dog, cat, cow*).

Figure 7 shows some example detections (high-scoring true and false positives) by our detector for five representative categories.

4.2 Active Detector Training on PASCAL

We next compare our active selection scheme to a passive learning baseline that randomly selects images for bounding box annotation. We select six representative categories from PASCAL: we take two each from those that are “easier” (>40 AP), “medium” (25–40 AP) and “hard” (0–25 AP) according to the state-of-the-art result (max of rows 4 and 5 in Table 1). We initialize each object’s classifier with 20 examples, and then let the remainder of the training data serve as the unlabeled pool, a total of 4.5 million examples. At each iteration, both methods select 100 examples, add their true bounding boxes (if any) to the labeled data, and retrain. This qualifies

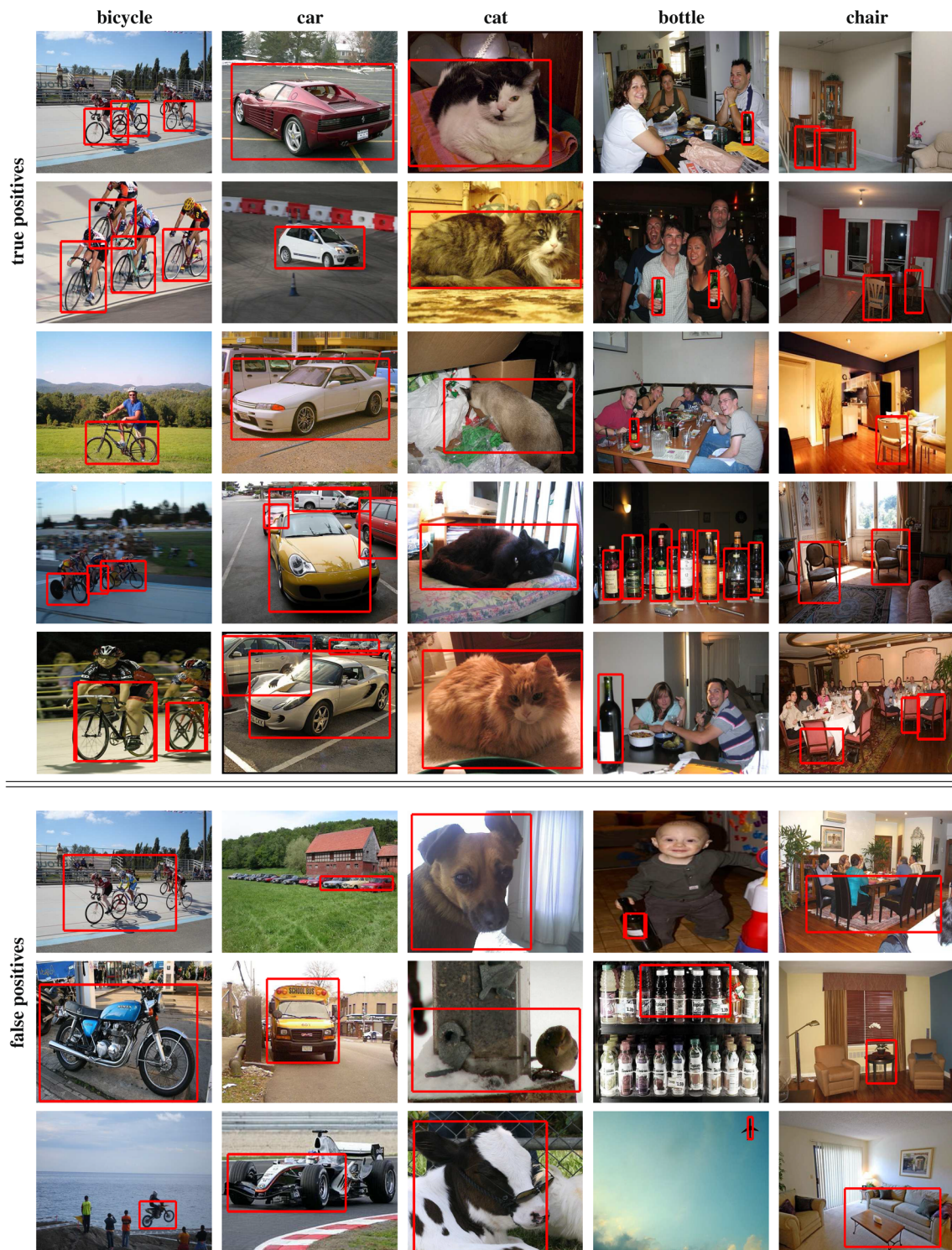


Fig. 7 Example detections on the PASCAL dataset obtained by our detector for five representative categories (*bicycle*, *car*, *cat*, *bottle*, *chair*). Our detector provides accurate localization despite large variations in appearance, pose, and the number of objects. Top scoring false

positives are mainly from similar categories (e.g. *cat* vs. *dog*, *bicycle* vs. *motorbike*) or due to the presence of a large number of similar objects (row 6 column 1, row 7 column 4) or inaccurate localization

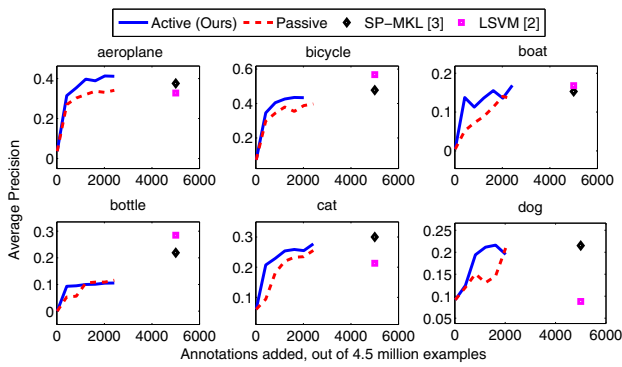


Fig. 8 Active detector training on PASCAL. Our large-scale active selection yields steeper learning curves than passive selection, and reaches peak state-of-the-art performance using only ~30 % of the training data

as learning in the “sandbox”, but is useful to test our jumping window and hashing-based approach. Furthermore, the natural cluttered images are significantly more challenging than data considered by prior active object learning approaches, and our unlabeled pool is orders of magnitude larger.

Figure 8 shows the results. We see our method’s clear advantage; the steeper learning curves indicate it improves accuracy on the test set using fewer labels. In fact, in most cases our approach reaches state-of-the-art performance (see markers above 5,000 labels) using only one-third of the available training data.

It is worth noting that in this sandbox testing scenario, the passive learning baseline is much stronger than a method that simply “randomly” labels images. Rather, the pool of unlabeled examples is guaranteed to contain many relevant images of the object of interest, and the passive learner will uncover them in an arbitrary order.

4.3 Online Live Learning on Flickr

Finally, we deploy our complete live learning system, where new training data is crawled on Flickr. We consider all object

Table 2 Number of images per category in the crawled dataset and the new Flickr test set

	Bird	Boat	Chair	Dog	Pottedplant	Sheep
Flickr-crawled	2936	3138	2764	1831	1566	1570
Flickr-test	655	628	419	780	364	820

classes for which state-of-the-art AP is less than 25.0 (*boat*, *dog*, *bird*, *pottedplant*, *sheep*, *chair*) in order to provide the most challenging case study, and to seek improvement through live learning where other methods have struggled most. To form the Flickr test set, we download images tagged with the class names dated in 2010; when running live training, our system is restricted to images dated in 2009. Table 2 shows the data statistics, in terms of the number of images per category in the training and testing sets.

We compare to two baselines: (1) a **Keyword+image baseline** that uses the same crawled image pool, but randomly selects images to get annotated on MTurk, and (2) a **Keyword+window baseline** that randomly picks jumping windows to get labeled. These are strong baselines since most of the images will contain the relevant object. In fact, they exactly represent the status quo approach, where one creates a dataset by manually pruning keyword search results. We initialize all methods with the PASCAL-trained models (5000 training images), and run for 10 iterations.

4.3.1 Live Learning Applied to PASCAL Test Set

Figure 9 shows the live learning results on the PASCAL test set. For four of the six categories, our system improves test accuracy, and outperforms the keyword approaches. The final AP also exceeds the current state-of-the-art for three categories (see Table 3, where we compare our method to the best previous result between Felzenszwalb et al. (2009) or Vedaldi et al. (2009)). This is an exciting result, given the size of the

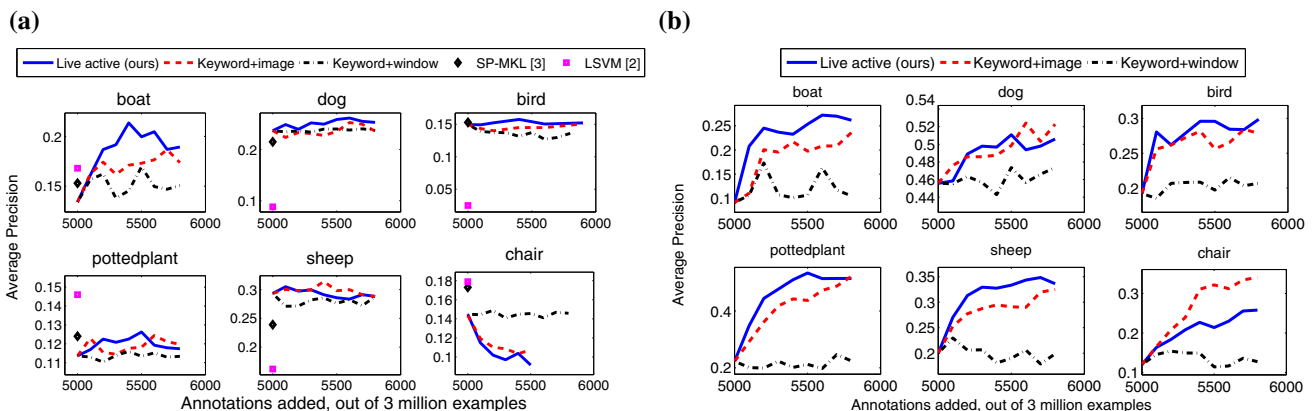


Fig. 9 Live learning results on (a) PASCAL test set and (b) Flickr test set

Table 3 Categories for which our method yields the best AP on PASCAL VOC 2007, compared to any result we found in the literature

	Aeroplane	Bird	Boat	Cat	Dog	Sheep	Sofa	Train
Ours	48.4	15.8*	18.9*	30.7	25.3*	28.8	33.0	47.7
Previous best	37.6	15.3	16.8	30.0	21.5	23.9	28.5	45.3

*Means extra Flickr data automatically obtained by our system was used to train

unlabeled pools (~3 million examples), and the fact that the system refined its models completely automatically.

However, for two classes (*chair*, *sheep*), live learning decreases accuracy. Of course, more data cannot guarantee improved performance on a fixed test set. We suspect the decline is due to stark differences in the distribution of PASCAL and Flickr images, since the PASCAL dataset creators do some manual preparation and pruning of all PASCAL data. Our next result seems to confirm this.

4.3.2 Live Learning Applied to Flickr Test Set

Figure 9b shows the live learning results on the new Flickr test set, where we apply the same live-learned models from above. While this test set appears more challenging than PASCAL, the improvements made by our approach are dramatic—both in terms of its absolute climb, as well as its margin over the baselines. The results indicate that our large-scale live learning approach can autonomously build models appropriate for detection tasks with realistic and unbiased data.

The gains over the passive keyword-based methods are noticeable and fairly consistent. Again, it is important to note that the passive baselines are much stronger than truly “random” selection of images to label. The baselines do not select images at random from a generic pool of images. Rather, they select among those that have the keyword tag for the object

of interest. This means there is great chance of at least the object being present, if not somewhat useful to the classifier. Furthermore, this procedure—keyword search followed by manual pruning in some arbitrary order—is exactly the status quo approach taken by researchers. The fact that live learning can outperform this strong baseline is very encouraging.

4.4 Analysis of Active Selections

Next we analyze the selections made by the live active learning system, compared to those made by the traditional passive (Keyword+image) baseline. Since the active learner typically outperforms the passive learner, we would like to understand what it is about the images it selects that makes them more informative. To this end, we first examine some qualitative examples, and then attempt to quantify the differences between what each method chooses to have labeled.

Figure 10 shows selections made by either method when learning the object category *boat*. The red boxes shown on the examples chosen by our method (see part (a)) illustrate the typical quality of its jumping windows. While not perfectly aligned with the tight ground truth bounding boxes, they have sufficient overlap to appear as useful candidate positives to the system.

Comparing the two sets of images, the examples in Fig. 10 also illustrate how our approach effectively focuses human attention among the crawled tagged images. We see that the first images our method selects (Fig. 10a) contain instances of a boat with variable appearance, which are intuitively useful to expand the learned model. In contrast, while the baseline’s selections (Fig. 10b) often will contain a boat of some kind—since the source images are all from keyword search on Flickr—the instances it chooses to get labeled first are unlikely to refine the boat detector well. For example, note

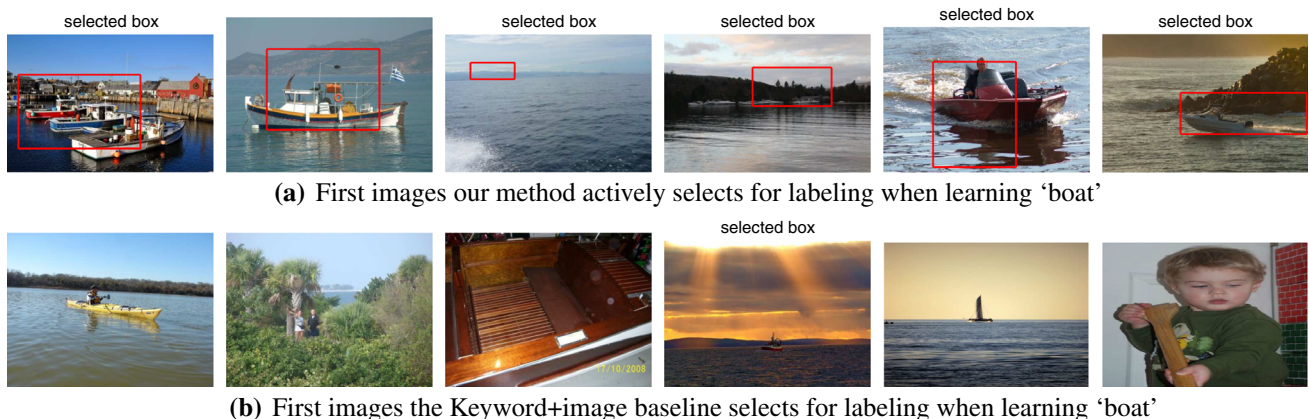


Fig. 10 Selections by (a) our live active approach and (b) the Keyword+image passive baseline. The red bounding boxes in (a) show the root window among all candidate jumping windows in that image that led to the image’s selection

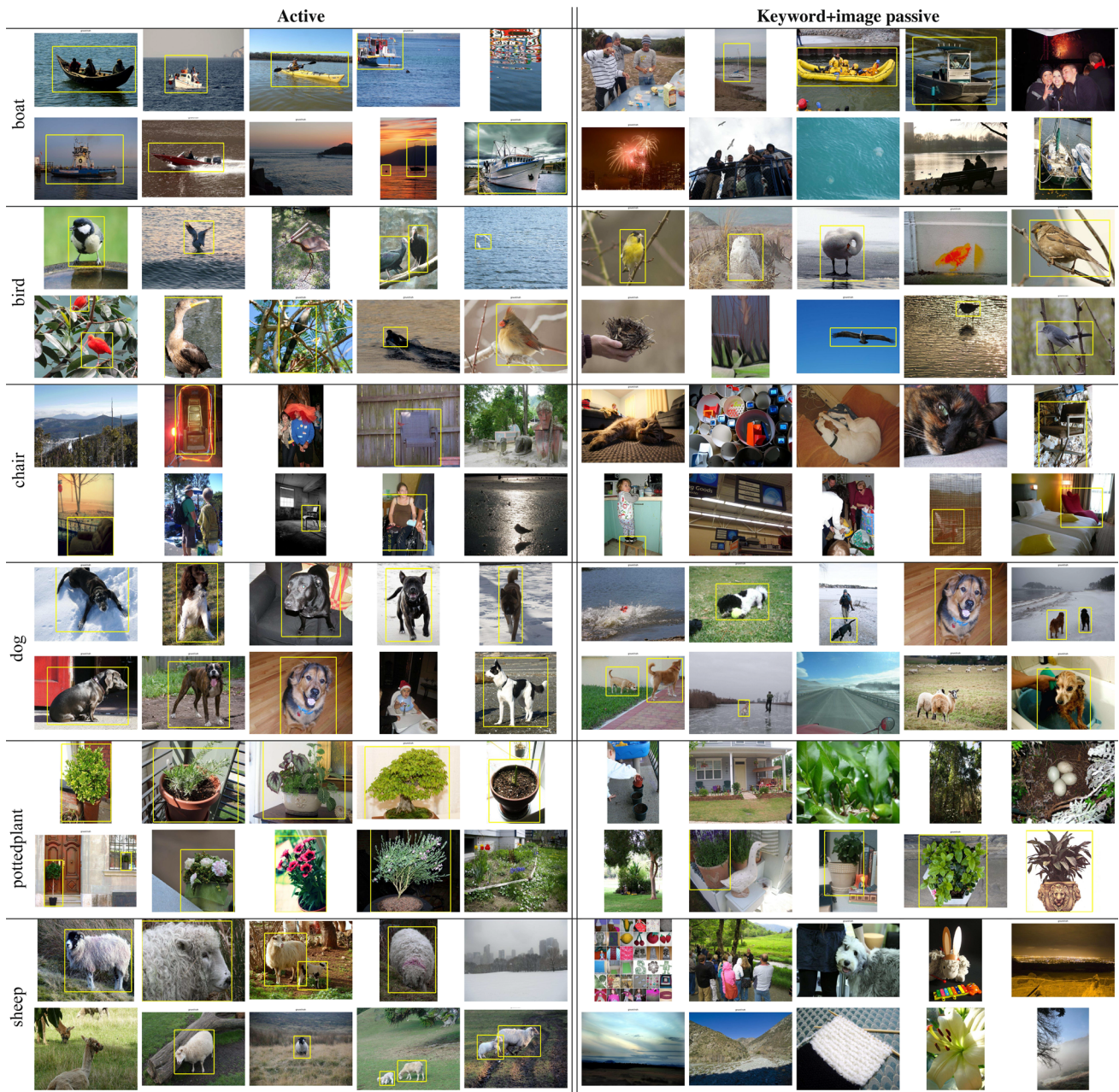


Fig. 11 Representative set of images selected for labeling by our approach (left) and the Keyword+image passive baseline (right), for six different object categories. The yellow boxes denote the ground truth position of the object. See text for details

the closeup of a side of the boat, or the picture of a boy holding a toy boat. This accentuates an important aspect of live learning. Its benefit is not solely in finding *more* examples of the object category; even basic keyword search can accomplish that. Rather, it helps the system identify *useful* examples of the object category in the context of what the detector already knows.

To examine this difference more closely, Fig. 11 shows the first 10 unlabeled images selected for labeling by either method, for each of six object categories. The bounding boxes denote the ground truth object positions (which are unknown

to our algorithm during the selection process). Here we notice that the live active learner finds more positive examples in its first label requests. While more positive examples that confirm what the detector already knows would be redundant, by definition these are positives that are most uncertain to the current model, since they fall nearest to the decision boundary. In contrast, the passive method yields fewer positives, and so it is more susceptible to the noisy tags on the Flickr images.

Figure 12 quantitatively confirms this observation. It shows the percentage of positive examples in the selections

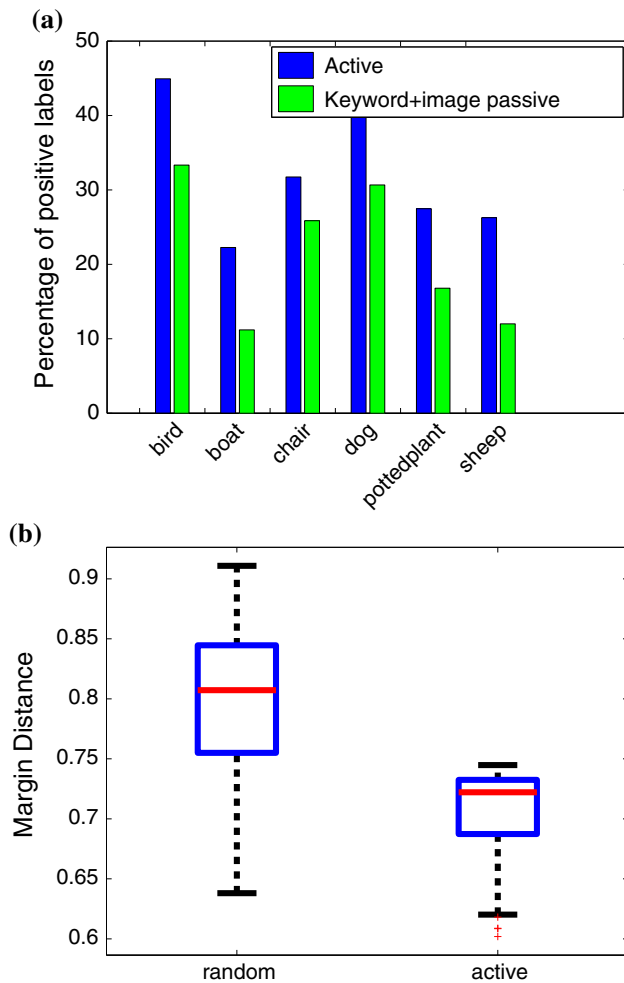


Fig. 12 (a) The percentage of positive examples selected by each method for each object category in the first five rounds of learning. (b) The distributions of the distances of selected examples to the classifier hyperplane, for the Keyword+passive baseline (left) and our approach (right). See text for details

made by each method, per object category. These percentages correspond to all label queries made within five rounds of learning, where each round accumulates $T = 100$ new examples. Notably, for those categories where our active approach is significantly better than the passive approach (e.g., *boat*, *sheep*, *pottedplant*), our method selects twice as many positive examples as the baseline. Furthermore, for the one class where we underperform the passive baseline (*chair*), our method's positive ratio, while still a bit higher, is much closer to the baseline's. Therefore, we see that focusing the annotations on probable positive instances is a benefit of live learning.

Aside from finding more positives among the unlabeled image pool, our method tends to select “cleaner” and more varied instances of an object category. For example, in Fig. 11, the images we select for *dog* have higher resolution views and dogs with some variety in textures (spots,

Table 4 Recall rate for the jumping windows as a function of the number of initial training examples. See text for details

#Examples	Recall	
	Sheep (%)	Chair (%)
20	46	28
40	64	48
60	63	51
80	63	51
100	69	49

solid). For *sheep*, we request labels for instances that vary in size (baby and adult) as well as pose (facing left or right). However, in the extreme, annotating highly diverse examples can also be problematic, especially when the total amount of labeled data is too low. For example, in the *chair* class, the variable appearance and high occlusion among the selected instances are not sufficiently representative and/or tend to have different contexts than those found in PASCAL. In fact, they are more in the vein of the chair examples the PASCAL Challenge deems as “difficult” and thus does not include in standard evaluation.

To quantify the informativeness of the examples selected by either method, we can look at how uncertain they are under the simple margin criterion. Figure 12b plots the distance to the margin for the selected examples in the first iteration. Compared to the passive Keyword+image baseline, our active approach identifies instances significantly closer to the decision boundary, which (as the learning curves above indicate) tend to be more informative overall. While this is to be expected—it is exactly the criterion our method aims to optimize—it is nonetheless good evidence that the approximation from hashing is strong enough to substantially outperform an arbitrary draw from the pool of images already pruned by keyword search.

Finally, we analyze the sensitivity of active selection with respect to the initial training data. The “cold start” problem is well known in active learning: in order to start exploiting the current model to choose informative things, the system has to already know *something* about the classes! Our method had a natural “warm” starting point in the live learning results presented above, since we could seed it with the labeled training examples from PASCAL. We found that the initial detectors trained with that data were good enough to achieve about 95 % recall in the jumping windows, which in turn means that our hash table of unlabeled instances will have (at least some) good positive candidate bounding boxes.

However, what if we have much less data to initialize? How sensitive will the quality of candidates for active selection be? To test this, we analyze the *sheep* and *chair* classes, as they represent a success and failure case for our method, respectively. Using randomly selected initial training exam-

ples from the Flickr training data, we train models using a varying number of examples ($\{20, 40, 60, 80, 100\}$). Note that because these are “cold start” instances that are chosen at random, they are likely to have a disproportionate number of negatives among them. Then, we apply each resulting model to the PASCAL test set, and record the maximum recall obtained by their jumping windows. The higher this recall rate is, the better chance the active learner will have to discover good candidates.

Table 4 shows the results. We observe two main outcomes. First of all, the recall rates climb with more initial training data, as expected. If the models are initialized with too few examples, recall may be so low as to produce a weak set of candidates in the hash table. Still, with only 40 instances, recall is already at 64 % for the *sheep* category. Second of all, we see another hint at why our method fails to outperform standard passive learning on the *chair* category. Its recall even with 100 labeled examples remains at about 50 %. Therefore, its candidates in the hash table could be poorly focused, leading the method to favor less informative examples in subsequent iterations. This echoes the analysis above about the *chair* class and the high intra-class appearance variation in selected examples.

4.5 Annotation Collection

Next we analyze the live online annotation collection process itself. Figure 13 shows some example annotations obtained from multiple annotators on Mechanical Turk and the consensus automatically obtained by our mean shift approach. The bounding boxes obtained from the annotators have a fairly large variability in their location and their tightness with respect to the object of interest. In order to train accurate detectors it is critical to obtain bounding boxes that

fit tightly around all the objects of interest in the image. Our consensus approach is able to provide such accurate tightly fitting bounding boxes in the majority of the cases. The last columns for ‘boat’ and ‘chair’ show the most common failure cases, where a majority of annotators provided a single bounding box surrounding all objects in the image.

Having obtained consensus on all images, we can go back and evaluate every annotator’s performance based on how much they agree. We score a detection as correct if the bounding box provided by an annotator has an intersection score of at least 80 % with the consensus. Figure 14a shows the precision and recall of all the annotators computed on all categories for “normal” instances of the object.

We see that most annotators have a precision of at least 50 %, which suggests there were very few spammers and most are competent for this task. However, the recall values are fairly low, even among frequent annotators (those providing at least 25 bounding boxes). This could be because the PASCAL division of the object of interest into {normal, truncated, unusual} categories is subjective. However, despite the low recall of most annotators, by using 10 annotators per instance we are able to detect all the objects in most images. Interestingly, there are clusters of annotators near the precision values of both 0 and 1. Perhaps these correspond to annotators who were tasting the task and found it too easy or hard to try more.

Figure 14b summarizes the number of jobs completed per annotator who contributed to the live learning process. There were in total 7,182 annotations (bounding boxes) collected from all six categories, and 182 annotators provided them. We see the graph follows an exponential shape, where a few annotators provided large portions of annotations while the

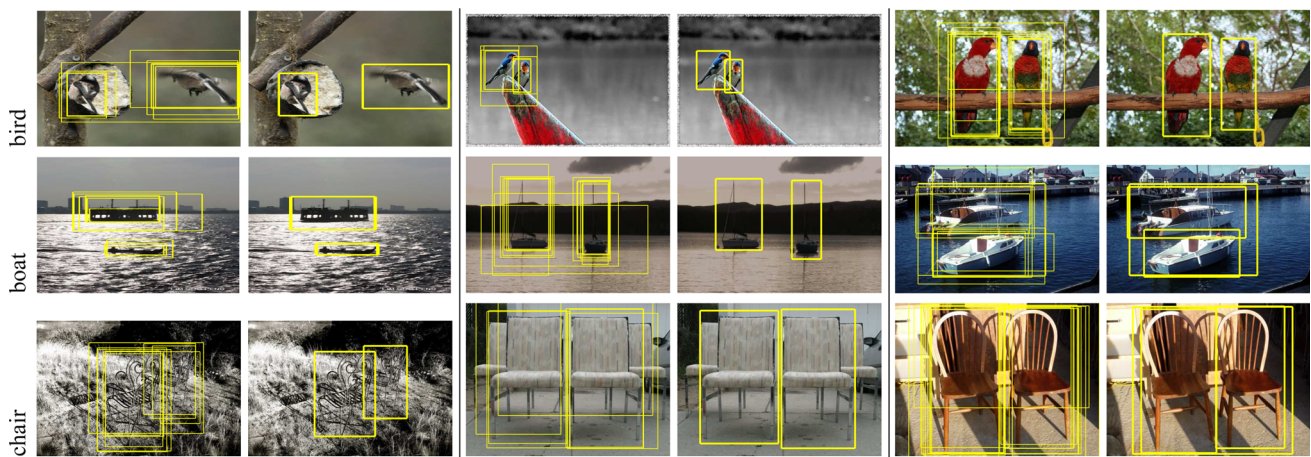


Fig. 13 Representative examples of our annotation collection. In each pair of images, the left image shows bounding boxes from 10 annotators, and the right image shows the consensus computed automatically

by our method. In spite of the variability in precision between the annotators, our consensus approach yields fairly accurate and tight bounding boxes. Best viewed in color

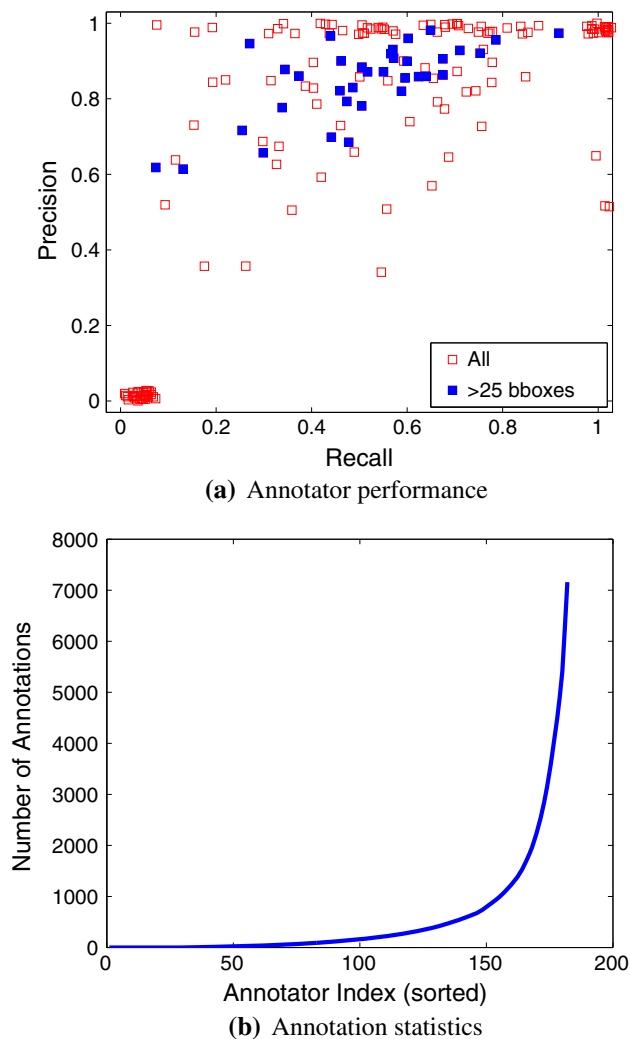


Fig. 14 Analysis of annotations collected. (a) Precision-recall of all annotators computed using the obtained consensus. Points in blue highlight results of frequent annotators. (b) Number of annotations per annotator, in sorted order

rest worked on very few (less than 25) annotations. In fact, about one-sixth of the annotators provided more than 90 % of the annotations. This suggests that one could target a small group of annotators and further optimize the collection process based on their expertise.

4.6 Computation Time

Table 5 shows the time complexity of each stage, and illustrates our major advantages for selection and retraining compared to existing strong detectors. Our times are based on a dual-core 2.8 GHz CPU, comparable to Felzenszwalb et al. (2009) and Vedaldi et al. (2009). Our jumping window+hashing scheme requires on average 2–3 s to retrieve 2,000 examples nearest the current hyperplane, and an additional 250 s to rank and select 100 images to query. In con-

Table 5 Run-time comparisons

	Active selection	Training	Detection per image
Ours + active	10 min	5 min	150 s
Ours + passive	0 min	5 min	150 s
LSVM	3 h	4 h	2 s
Felzenszwalb et al. (2009)			
SP+MKL	93 h	> 2 days	67 s
Vedaldi et al. (2009)			

Our detection time is mostly spent pooling the sparse codes. Active times are estimated for Felzenszwalb et al. (2009), Vedaldi et al. (2009) models based on linear scan. Our approach’s efficiency makes live learning practical

trast, a linear scan over the entire unlabeled pool would require about 60 h.

The entire online learning process requires 45–75 min per iteration: 5–10 min to retrain, 5 min for selection, and about one hour to wait for the MTurk annotations to come back (typically 50 unique workers gave labels per task). Thus, waiting on MTurk responses takes the majority of the time, and could likely be reduced with better payment. In comparison, the same selection with the method of Felzenszwalb et al. (2009) or Vedaldi et al. (2009) would require about 8 h to 1 week, respectively.

5 Conclusions and Future Work

Our contributions are (i) a novel efficient part-based linear detector that provides excellent performance, (ii) a jumping window and hashing scheme suitable for the proposed detector that retrieves relevant instances among millions of candidates, and (iii) the first active learning results for which both data and annotations are automatically obtained, with minimal involvement from vision experts. Tying it all together, we demonstrated an effective end-to-end system on two challenging datasets.

Looking forward, there are a number of challenging issues in active learning that will influence how successful future live learners can be. As in any pool-based active learning approach, the system’s model of what looks useful is inherently tied to the particular classifier it employs. An interesting challenge for future work is to identify images for labeling that could benefit an array of competing approaches. Furthermore, while our system does well working in a “myopic” mode, where at each cycle of learning the top T most uncertain images are chosen for labeling, a scalable solution for far-sighted selection of jointly informative sets of images would be interesting to consider. Similarly, while our pipeline does some standard hard negative mining as new data is added, sub-linear time strategies to find negatives far from the hyperplane across all data might better steepen learning curves.

Finally, while our system focuses on the “exploitation” aspect by leveraging the current classifier to find useful examples, it could be useful to also incorporate an “exploration” aspect that attempts to cover diverse areas of the feature space.

Future work could also investigate applying language models and word sense disambiguation methods to enhance the way the images are crawled using keywords. For example, rather than simply gather up candidate images that have a keyword match, the system could automatically broaden the queries to other nouns in the same synset, or use descriptive phrases to isolate image content most likely aligned with the target visual concept. In terms of the crowdsourcing aspect of this work, we would like to investigate automated methods for adjusting payments offered to workers according to the marketplace dynamics and relative difficulty of the actively selected annotation tasks.

Acknowledgments The authors thank the anonymous reviewers for their helpful comments. This research is supported in part by NSF CAREER IIS-0747356 and DARPA Mind’s Eye.

References

- Boureau, Y.-L., Bach, F., LeCun, Y., Ponce, J. (2010). Learning mid-level features for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Charikar, M. (2002). Similarity estimation techniques from rounding algorithms. In *Symposium on Theory of Computing*.
- Chum, O., Zisserman, A. (2007). An exemplar model for learning object classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Dalal, N., Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Dean, T., Ruzon, M., Segal, M., Shlens, J., Vijayanarasimhan, S., Yagnik, J. (2013). Fast, accurate detection of 100,000 object classes on a single machine. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). A large-scale hierarchical image database: Imagenet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Everingham, M., Van Gool, L., Williams, C., Winn, J., & Zisserman, A. (2010). The pascal visual object classes challenge. *International Journal of Computer Vision*, 88(2), 303–338.
- Felzenszwalb, P., Girshick, R., McAllester, D., & Ramanan, D. (2009). Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(1), 5555.
- Fergus, R., Fei-Fei, L., Perona, P., Zisserman, A. (2005). Learning object categories from Google’s image search. In *Proceedings of the International Conference on Computer Vision (ICCV)*.
- Jain, P., Vijayanarasimhan, S., Grauman, K. (2010). Hashing hyperplane queries to near points with applications to large-scale active learning. In *Advances in Neural Information Processing Systems (NIPS)*.
- Joachims, T. (2006). Training linear SVMs in linear time. In *International Conference on Knowledge Discovery and Data Mining (KDD)*.
- Joshi, A., Porikli, F., Papanikolopoulos, N. (2009). Multi-class active learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Kapoor, A., Grauman, K., Urtasun, R., Darrell, T. (2007). Active learning with Gaussian processes for object categorization. In *International Conference on Computer Vision (ICCV)*.
- Lampert, C., Blaschko, M., & Hofmann, T. (2008). Object localization by efficient subwindow search: Beyond sliding windows. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (ICCV)*.
- Lee, Y. J., Grauman, K. (2010). Object-graphs for context-aware category discovery. In *Proceedings of IEEE International Conference on Computer Vision (CVPR)*.
- Li, L., Wang, G., & Fei-Fei, G. (2007). Automatic online picture collection via incremental model learning: Optimol. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Pirsiavash, H., Ramanan, D. (2012). Steerable part models. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Qi, G., Hua, X., Rui, Y., Tang, J., Zhang, H. (2008). Two-dimensional active learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Russell, B., Torralba, A., Murphy, K., & Freeman, W. (2007). Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77, 157–173.
- Siddiquie, B., & Gupta, A. (2010). Modeling context for multi-class active learning: Beyond active noun tagging. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Song, H., Zickler, S., Althoff, T., Girshick, R., Fritz, M., Geyer, C., Felzenszwalb, P., Darrell, T. (2012). Sparselet models for efficient multiclass object detection. In *Proceedings of the European Conference on Computer Vision*.
- Sorokin, A., Forsyth, D. (2008). Utility data annotation with Amazon mechanical turk. In *Workshop on Internet Vision*.
- Tong, S., Koller, D. (2000). Support vector machine active learning with applications to text classification. In *Proceedings of the International Conference on Machine Learning (ICML)*.
- Torralba, A., Murphy, K., & Freeman, W. (2007). Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5), 854–869.
- Uijlings, J., Smeulders, A., Scha, R. (2009). What is the spatial extent of an object? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Vedaldi, A., Gulshan, V., Varma, M., Zisserman, A. (2009). Multiple kernels for object detection. In *International Conference on Computer Vision (ICCV)*.
- Vijayanarasimhan, S., & Grauman, K. (2008). Multiple-instance learning for weakly supervised object categorization: Keywords to visual categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Vijayanarasimhan, S., & Grauman, K. (2011). Training object detectors with crawled data and crowds: Large-scale live active learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Vijayanarasimhan, S., Grauman, K. (2008). Multi-level active prediction of useful image annotations for recognition. In *Advances in Neural Information Processing Systems (NIPS)*.
- Vijayanarasimhan, S., Kapoor, A. (2010). Visual recognition and detection under bounded computational resources. In *Proceedings of IEEE International Conference on Computer Vision (CVPR)*.
- Vijayanarasimhan, S., Jain, P., & Grauman, K. (2014). Hashing hyperplane queries to near points with applications to large-scale active learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(2), 276–288.

- Viola, P., Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- von Ahn, L., Dabbish, L. (2004). Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*.
- Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., Gong, Y. (2010). Locality-constrained linear coding for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Welinder, P., & Perona, P. (2010). Rating annotators and obtaining cost-effective labels: Online crowdsourcing. In *Workshop on Advancing Computer Vision with Humans in the Loop (ACVHL)*.
- Yang, J., Yu, K., Gong, Y., Huang, T. (2009). Linear spatial pyramid matching sparse coding for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.