To be presented at the 3rd Human Computation (HCOMP) Workshop, in conjunction with AAAI, August 2011.

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

Sudheendra Vijayanarasimhan and Kristen Grauman
University of Texas at Austin
{svnaras,grauman}@cs.utexas.edu

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 benchmark. In addition, we show our detector competes well with popular nonlinear classifiers that are much more expensive to train.

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 (1; 2; 3; 4; 5). 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 (6; 7; 8; 9; 10), whereas crowd-sourcing work explores how to package annotation tasks such that they can be dispersed effectively online (11; 12; 13; 14; 15). 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 existing techniques are tested in artificially controlled settings. Specifically, we see four limiting factors:

1. 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, to our knowledge, 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.
2. 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.
3. 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.
4. The computational complexity of the active selection process is generally ignored, yet scalability 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 MTurk 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. 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 recently developed (16).

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, which is "the" challenging benchmark in object detection studied by many vision researchers. 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.[1]

Related Work

We briefly review related work on object detection, active learning for object recognition, and crowd-sourcing efforts with image data.

Object detection has received various treatments in the literature; see (5) 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 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 (3; 2)).

Various ways to reduce detection time have been explored, including cascades (3), branch-and-bound search (4), or jumping windows (18). To reduce classifier training and testing costs, simpler linear models are appealing. While linear models tend to underperform with common representations (e.g., see tests in (3; 19)), recent work in image classification shows very good results when also incorporating sparse coding and feature pooling (20; 21; 19). 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.

Active learning has been shown to better focus annotation effort for image recognition tasks (6; 9; 8) and region labeling (7; 10). 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 (16) for sub-linear time search of the unlabeled pool. Further, all previous work in vision 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 (11; 12; 15) or online and incremental learning (22). 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 (13; 14) or even target tasks according to individuals' reliability (23). Other work attempts to directly learn object models from noisy keyword search (e.g., (24; 22; 25)); 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.

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.

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 SVM detector, and then show how to identify its uncertain examples efficiently using a hashing scheme.

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, 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 (16).

Our object model consists of a root window r, multiple part windows $\{p_1, \ldots, p_P\}$ that overlap the root, and context windows $\{c_1, \ldots, c_C\}$ surrounding it. See Figure 1. Let $O = [r, p_1, \ldots, p_P, c_1, \ldots, c_C]$ denote a candidate object

[1] This work will appear in CVPR 2011 (17), and through HCOMP we hope to share and discuss it with a broader community. We abbreviate the details of the approach in order to include new additional analysis of data annotation results that are relevant to the HCOMP workshop. We refer a reader interested in the technical details of the vision components to our CVPR 2011 paper.

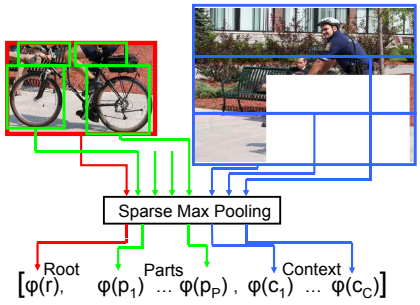


Figure 1: Our part-based object representation.

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:

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

where $\boldsymbol{w}$ denotes the learned classifier weights, which we obtain with SVM training.

Given a novel test image, we first extract local image descriptors; we use a dense multi-scale sampling of SIFT. Each window type (r, p_i, or c_j) uses these features to create its encoding $\phi(\cdot)$. Specifically, each window is represented using a nonlinear feature encoding based on *sparse coding and max-pooling*. This representation is related to the well-known bag-of-features, and is directly inspired by recent sparse coding work in image classification (20; 21; 19). Offline, we cluster a corpus of features to obtain a dictionary of visual words. Then for any window (whether root/part/context), we obtain the sparse codes of its set of local features, and describe the window by the largest code response of each visual word within it. (See (17) for details.)

In this way, the *root window* provides a global summary of the object appearance, and is invariant to translations of features. Similarly, each *part window* summarizes the local features within it, discarding their mutual positions but capturing their spatial layout relative to the root. The *context windows* incorporate contextual cues surrounding the object, such as the presence of "sky", "ground", or "road", as shown in Figure 1.

As we discuss in detail in (17), our object model intentionally captures positive aspects of recent state-of-the-art detection models (3; 2), but does so while maintaining a much lower computational cost. In particular, training the proposed model is significantly more scalable, as we will see in the results.

Generating Candidate Root Windows

Rather than index an exhaustive list of all windows in all unlabeled images coming from keyword search, we generate a set of candidate root windows per image using a variant of the *jumping window* search method (18; 26). The approach generates a Hough-like projection of the proposed

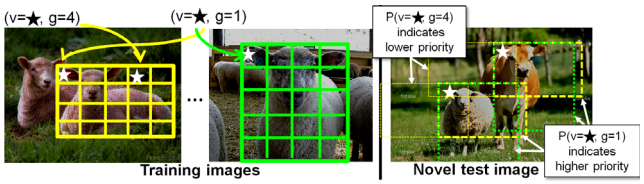


Figure 2: Illustration of jumping window root candidates. Grid cells serve to refine the priority given to each box (but do not affect its placement). Here, location $g = 1$ has higher priority than $g = 4$ for visual word $v = \star$ since it appears more consistently in the training images (left two images).

object bounding box using individual visual word matches between the new example and all training instances, and it prioritizes these candidates according to a measure of how discriminative a given word and coarse location is for the object class (see Figure 2). We take the top $K = 3,000$ jumping windows per image based on their priority scores.

This allows us to focus on only the most viable candidates for possible selection by the active learning procedure. Similarly, at test time, we use jumping windows rather than sliding windows to limit the search time.

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 (27), a widely used criterion that seeks points that most reduce the version space. Given an SVM with hyperplane normal $\boldsymbol{w}$ and an unlabeled pool of data $\mathcal{U}_O = \{\phi(O_1), \ldots, \phi(O_n\})$, the point that minimizes the distance to the current decision boundary is selected for labeling: $O^* = \arg\min_{O_i \in \mathcal{U}_O} |\boldsymbol{w}^T \phi(O_i)|$.

A naive application of this criterion would entail computing the classifier response on all unlabeled data, and ranking them by $|\boldsymbol{w}^T \phi(O_i)|$. However, this is prohibitively expensive, especially since we have live annotators awaiting the next labeling jobs and massive unlabeled data pools.

Therefore, we adopt our *hyperplane-hashing* algorithm (16) to identify the most promising candidate windows in sub-linear time. The algorithm maps inputs to binary keys using a randomized hash function that is locality-sensitive for the angle between the hyperplane normal and a database point. Given a "query hyperplane", with our algorithm one can hash directly to those points that are nearest to the hyperplane $\boldsymbol{w}$, with high probability.

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

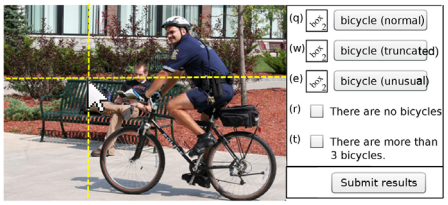


Figure 3: Mechanical Turk interface to obtain bounding boxes on actively selected examples.

randomized functions with two-bit outputs:

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

where the component function is defined as

$$h_{\boldsymbol{u},\boldsymbol{v}}(\boldsymbol{a}, \boldsymbol{b}) = [\text{sign}(\boldsymbol{u}^T\boldsymbol{a}), \text{sign}(\boldsymbol{v}^T\boldsymbol{b})], \quad (2)$$

$\text{sign}(\boldsymbol{u}^T\boldsymbol{a})$ returns 1 if $\boldsymbol{u}^T\boldsymbol{a} \geq 0$, and 0 otherwise, and $\boldsymbol{u}$ and $\boldsymbol{v}$ are sampled from a standard multivariate Gaussian, $\boldsymbol{u}, \boldsymbol{v} \sim \mathcal{N}(0, I)$. 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 $\boldsymbol{w}$ and $-\boldsymbol{w}$. See (16) for 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. We search only those examples, i.e., we compute $|\boldsymbol{w}^T\phi(O_i)| = |f(O_i)|$ for each one, and rank them in order of increasing 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.

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 the PASCAL challenge's annotation protocol, and to flag images containing more than 3 instances. Figure 3 shows the annotation interface.

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.

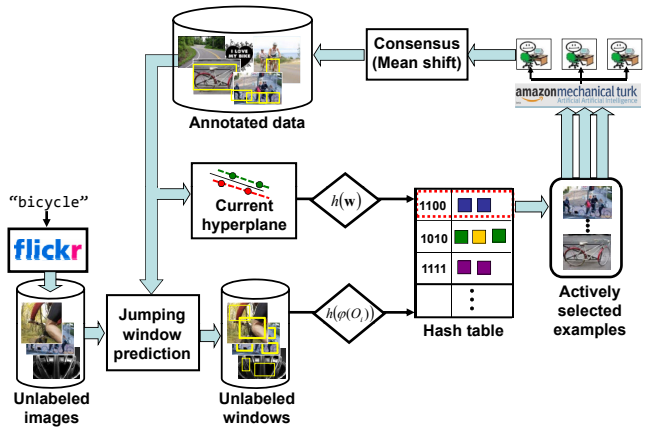


Figure 4: Summary of our live learning system.

	bird	boat	chair	dog	pottedplant	sheep
Flickr-crawled	2936	3138	2764	1831	1566	1570
Flickr-test	655	628	419	780	364	820

Table 1: Number of images in the Flickr data.

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

Training the Detector

Training our detector entails learning the linear SVM weights in Eqn. 1 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 $< 20\%$.

Summary of the Approach

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 Figure 4.

Results

The goal of our experiments is to 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. We use two datasets: the PASCAL VOC 2007, and a new Flickr dataset.

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

	bird	boat	dog	potted plant	sheep	chair
Ours (live learning)	**15.8***	**18.9***	**25.3***	11.6*	28.4*	9.1*
Proposed detector	14.1	13.6	21.8	11.1	28.8	11.6
Previous best	15.3	16.8	21.5	**14.6**	**23.9**	**17.9**

Table 2: Live learning results on PASCAL test set, vs. the best results we found in the literature. (*means extra Flickr data automatically obtained by our system used to train.)

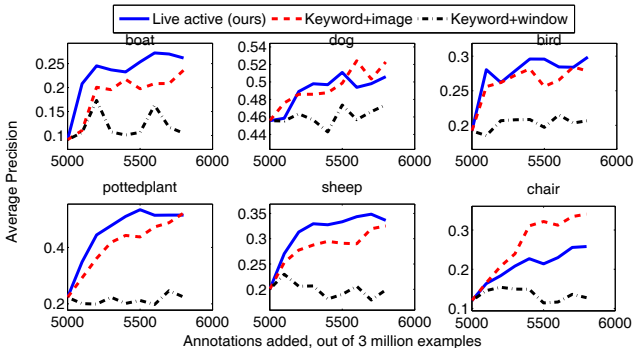


Figure 5: Live learning results on Flickr test set.

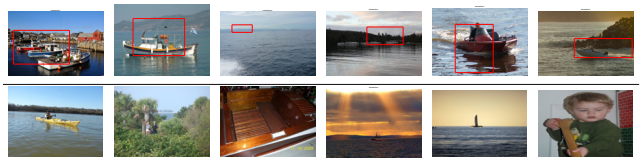


Figure 6: Selections by our live approach (top) and Keyword+image (bottom) when learning "boat".

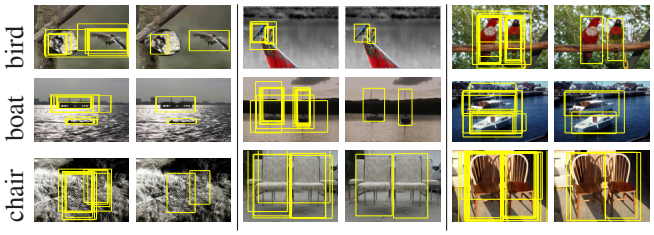


Figure 7: Representative examples of our annotation collection. In each pair, the left image shows bounding boxes from 10 annotators, and the right shows the consensus computed by our method. Best viewed in pdf with zoom.

gled 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. See Table 1 for the data stats.

Live Learning Applied to PASCAL Test Set Table 2 compares the final AP obtained by our live learning process, our initial detector trained on PASCAL only, and the current state-of-the-art (best of (2; 3)). Our results exceed the state-of-the-art on three categories. This is an exciting result, given the size of the 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 our detector's accuracy (see first two rows). 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.

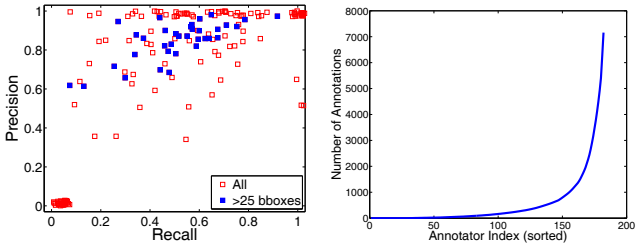
Live Learning Applied to Flickr Test Set Figure 5 shows the learning curves on the new Flickr test set, where we apply the same live-learned models from above. We compare to (1) a **Keyword+image baseline** that uses the same crawled image pool, but randomly selects images to get annotated on MTurk, 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* in computer vision, 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.

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. Figure 6 shows selections made by our method and the keyword baseline when learning "boat", illustrating how ours better focuses human attention among the crawled tagged images. In all, the results indicate that our large-scale live learning approach can autonomously build models appropriate for detection tasks with realistic and unbiased data.

Annotation Collection Figure 7 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. Yet, 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. We find that our consensus approach provides such 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 provide 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 8(a) 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



(a) Annotator performance (b) Annotation statistics

Figure 8: 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.

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 8(b) summarizes the number of jobs completed per annotator who contributed to the live learning process. There were in total 7182 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 rest worked on very few (< 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.

Computation Time An important part of our contribution is to make live learning for detectors feasible computationally. Table 3 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 (2; 3). Our jumping window+hashing scheme requires on average 2-3 seconds to retrieve 2,000 examples nearest the current hyperplane, and an additional 250 seconds to rank and select 100 images to query. In contrast, a linear scan over the entire unlabeled pool would require about 60 hours.

The entire online learning process requires 45-75 minutes per iteration: 5-10 min. to retrain, 5 min. for selection, and $\sim$1 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 a direct application of existing detectors (2; 3) would require about 8 hours to 1 week, respectively.

Conclusions

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.

This work stresses the importance of scalability and live gathering of data, in order to study active learning's impact in a realistic setting. While other researchers have certainly examined active learning with challenging and realistic *datasets*, we are not aware of any comparable quantitative anaylsis of a live deployed system in the vision or learning literature. However, there is an array of other interesting challenges we do *not* address in this work that may also hinder active learning in real problems—such as how to choose the best learner or selection criteria, the cold start problem, and skewed data—as summarized very well in (28), and tackled for biomedical citation screening applications in (29).

In future work we are interested in expanding the problem domain for live learning beyond object detection, and exploring effective ways to target the requests online to the appropriate annotator.

	Active selection	Training	Detection per image
Ours + active	10 mins	5 mins	150 secs
Ours + passive	0 mins	5 mins	150 secs
LSVM (2)	3 hours	4 hours	2 secs
SP+MKL(3)	93 hours	$>$ 2 days	67 secs

Table 3: Run-time comparisons. Our detection time is mostly spent pooling the sparse codes. Active times are estimated for (2; 3) models based on linear scan. Our approach's efficiency makes live learning practical.

References

[1] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, 2005.

[2] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part Based Models. *TPAMI*, 99(1), 2009.

[3] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple Kernels for Object Detection. In *ICCV*, 2009.

[4] C. Lampert, M. Blaschko, and T. Hofmann. Beyond Sliding Windows: Object Localization by Efficient Subwindow Search. In *CVPR*, 2008.

[5] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes Challenge. *IJCV*, 88(2):303–338, June 2010.

[6] G. Qi, X. Hua, Y. Rui, J. Tang, and H. Zhang. Two-Dimensional Active Learning for Image Classification. In *CVPR*, 2008.

[7] S. Vijayanarasimhan and K. Grauman. Multi-Level Active Prediction of Useful Image Annotations for Recognition. In *NIPS*, 2008.

[8] A. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-Class Active Learning for Image Classification. In *CVPR*, 2009.

[9] P. Jain and A. Kapoor. Active Learning for Large Multi-class Problems. In *CVPR*, 2009.

[10] B. Siddiquie and A. Gupta. Beyond Active Noun Tagging: Modeling Context for Multi-Class Active Learning. In *CVPR*, 2010.

[11] L. von Ahn and L. Dabbish. Labeling Images with a Computer Game. In *CHI*, 2004.

[12] B. Russell, A. Torralba, K. Murphy, and W. Freeman. Labelme: a Database and Web-Based Tool for Image Annotation. *IJCV*, 2007.

[13] A. Sorokin and D. Forsyth. Utility Data Annotation with Amazon Mechanical Turk. In *Internet Vision*, 2008.

[14] P. Welinder and P. Perona. Online Crowdsourcing: Rating Annotators and Obtaining Cost-Effective Labels. In *ACVHL*, 2010.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[16] P. Jain, S. Vijayanarasimhan, and K. Grauman. Hashing Hyperplane Queries to Near Points with Applications to Large-Scale Active Learning. In *NIPS*, 2010.

[17] S. Vijayanarasimhan and K. Grauman. Large-Scale Live Active Learning: Training Object Detectors with Crawled Data and Crowds. In *CVPR*, 2011.

[18] O. Chum and A. Zisserman. An Exemplar Model for Learning Object Classes. In *CVPR*, 2007.

[19] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning Mid-level Features for Recognition. In *CVPR*, 2010.

[20] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear Spatial Pyramid Matching Sparse Coding for Image Classification. In *CVPR*, 2009.

[21] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-Constrained Linear Coding for Image Classification. In *CVPR*, 2010.

[22] L. Li, G. Wang, and L. Fei-Fei. Optimol: Automatic Online Picture Collection via Incremental Model Learning. In *CVPR*, 2007.

[23] P. Donmez and J. Carbonell. Proactive Learning: Cost-Sensitive Active Learning with Multiple Imperfect Oracles. In *CIKM*, 2008.

[24] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning Object Categories from Google's Image Search. In *ICCV*, 2005.

[25] S. Vijayanarasimhan and K. Grauman. Keywords to Visual Categories: Multiple-Instance Learning for Weakly Supervised Object Categorization. In *CVPR*, 2008.

[26] S. Vijayanarasimhan and A. Kapoor. Visual Recognition and Detection Under Bounded Computational Resources. In *CVPR*, 2010.

[27] S. Tong and D. Koller. Support Vector Machine Active Learning with Applications to Text Classification. In *ICML*, 2000.

[28] J. Attenberg and F. Provost. Inactive Learning? Difficulties Employing Active Learning in Practice. *SIGKDD Explorations Newsletter*, 12(2), December 2010.

[29] B. Wallace, K. Small, C. Brodley, and T. Trikalinos. Active Learning for Biomedical Citation Screening. In *KDD*, 2010.