Fast, Accurate Detection of 100,000 Object Classes on a Single Machine

Thomas Dean Mark A. Ruzon Mark Segal
Jonathon Shlens Sudheendra Vijayanarasimhan Jay Yagnik†
Google, Mountain View, CA
{tld,ruzon,segal,shlens,svnaras,jyagnik}@google.com

Abstract

Many object detection systems are constrained by the time required to convolve a target image with a bank of filters that code for different aspects of an object's appearance, such as the presence of component parts. We exploit locality-sensitive hashing to replace the dot-product kernel operator in the convolution with a fixed number of hash-table probes that effectively sample all of the filter responses in time independent of the size of the filter bank. To show the effectiveness of the technique, we apply it to evaluate 100,000 deformable-part models requiring over a million (part) filters on multiple scales of a target image in less than 20 seconds using a single multi-core processor with 20GB of RAM. This represents a speed-up of approximately 20,000 times— four orders of magnitude— when compared with performing the convolutions explicitly on the same hardware. While mean average precision over the full set of 100,000 object classes is around 0.16 due in large part to the challenges in gathering training data and collecting ground truth for so many classes, we achieve a mAP of at least 0.20 on a third of the classes and 0.30 or better on about 20% of the classes.

1. Introduction

Many object detection and recognition systems are constrained by the computation time required to perform a large number of convolutions across a dense array of image windows with a bank of filters. This problem can be mitigated somewhat by using cascades and coarse-to-fine architectures to deploy filters conditionally based upon previously computed thresholded responses, or by sampling image windows with a sparse grid or only at locations determined by an interest-point detector. The former method is inherently greedy and can require traversing a tree of filters that is both deep and broad in order to scale to a large number of object classes and achieve a given precision/recall target. The latter method suffers from low-level salience being a poor guide to precise identification of locations to sample and thus requires that filters be invariant to small positional changes, thereby reducing their discrimination.

We exploit locality-sensitive hashing [10] to replace the dot product in the kernel operator of the convolution with a multi-band hash-table lookup. This enables us to determine the filters with highest responses in $O(1)$ time independent of the number of filters. The technique is applicable to a variety of signal processing tasks in image processing as well as other domains. Our approach is not restricted to any particular method or dataset; rather it provides the basis for scaling the number of traditionally compute-intensive signal processing operators from the hundreds or at most thousands applied in current practice to millions. All filters are effectively sampled in one pass over the image with a single probe per location/image window. A descriptor generated from the image window is divided into bands, and each band is hashed in the table associated with that band to retrieve a list of filters potentially responding at that location. The indications from all bands are combined to come up with a set of proposals for the filters with the largest responses, and exact responses are computed for these filters only. We demonstrate the efficacy of the approach by scaling object detection to one hundred thousand object classes employing millions of filters representing objects and their constituent parts across a wide range of poses and scales.

We allow filters to operate on diverse features and achieve additional benefits by embedding patches of the resulting feature map in an ordinal space using a high-dimensional sparse feature descriptor [21]. By operating in this ordinal space, our similarity measure becomes rank correlation instead of the linear correlation implied by the dot product in the kernel of a traditional convolution. Rank correlation largely eliminates both linear and nonlinear scaling effects (including local contrast normalization) and reduces sensitivity to small perturbations in the underlying feature space (see Figure 1). By performing this nonlinear embedding in a high-dimensional space, we are able to simplify

† Corresponding author.

1063-6919/13 $26.00 © 2013 IEEE
DOI 10.1109/CVPR.2013.237

the hashing process and apply large-scale linear solvers in training object models. Furthermore, this hashing scheme can be implemented exactly in the integer domain, a desirable property for achieving high performance.

Are there really 100,000 objects to concern ourselves with? Humans are able to discriminate among around 30,000 visual categories [2], but many more appearances and specific instances of classes, in much the same way as adult vocabularies are estimated to be on the order of 20,000 words for an English speaker but much larger when one considers all the proper nouns, hyphenated words and arcana related to specific areas of expertise and interest. Yet visual media are mostly opaque to machines. Much of the progress in indexing such data depends on metadata, including anchor text, captions, transcripts from audio, and user-supplied tags. Standard visual descriptors relying on features such as SIFT and HOG are semantically too low-level to substantially complement the metadata.

If it were possible to identify a large subset of the objects in an image—their semantic categories and relative geometries—we could apply modern document-retrieval methods: term-frequency vectors, inverted indexes, and common word sequences to considerably improve the ability of machines to parse visual data and search it efficiently. We believe the present work will accelerate the state of the art in object detection by increasing the number of visual categories by an order of magnitude or more while simultaneously reducing run times by a comparable factor. We demonstrate our approach in an implementation that achieves respectable performance on a standard benchmark for object detection, and exhibits graceful degradation in performance with larger, automatically curated datasets consisting of tens of thousands of classes.

2. Related Work

Traditionally, object detection is reduced to binary classification and a sliding window classifier is applied at all positions, scales and orientations of an image [20, 4, 8, 19, 22]. This leads to a complexity of $O(LC)$ where L is the number of locations or window proposals that need to be evaluated and C is the number of classifiers (object classes) that are being searched for. Most existing work on improving object detection complexity focuses on reducing L.

Following the success of Viola and Jones [20], cascades have been successfully applied to the star-structured models of [8] in [7] and [12] for this purpose. Cascades work by applying simpler tests to each hypothesized object location in order to eliminate most of them very quickly. Felzenszwalb *et al*. [7] utilize a variation on *probably approximate learning* (PAC) to learn a cascade of detectors that provides a 20-fold speedup of their earlier model, and [12] introduce a new coarse-to-fine method that complements the approach in [7], and can be combined with it to achieve up to two orders of magnitude speedup in some cases.

Alternative approaches to reducing L advocate the use of interest points in the form of jumping windows [19], salience operators like objectness [1, 14], and segmentations [18, 9] as cues for generating object-like window proposals, thus pruning out most of the background regions that a naive approach like sliding window cannot avoid. While these approaches produce similar accuracies to sliding-window-based methods by evaluating only a fraction of the windows in an image, they are primarily used for bag-of-words-based models that require large vocabularies and costly sparse coding or kernel operations in order to obtain state-of-the-art accuracies.

Very recently, [13, 17] consider scaling category-level detection by learning a set of shared basis parts in the form of steerable filters or *sparselets*. Using these learned basis parts, approximate part responses can be computed for a large number of classes using a sparse matrix-vector product. They perform convolutions only for the basis part filters. While sparse products reduce the dependence of filter convolutions on the number of classes, the overall complexity is still linear in C and it remains to be seen if these approaches can scale to hundreds of thousands of categories. It also remains to be seen just how large the set of basis filters has to grow in order to support a large number of classes. We suspect it will be significant, and that our approach will complement the sparselet work. Moreover, once we largely eliminate the overhead of performing convolutions using a method such as ours, we expect researchers will find ample use for new classes of filters.

Overall, in contrast to most previous work, our approach reduces object detection complexity by targeting its dependence on the number of object classes C. Our hashing-based framework is the first approach to reduce object detection complexity from $O(LC)$ to $O(L)$. Furthermore, since most previous methods concentrate on reducing L, they complement our technique and can therefore be combined with ours to further reduce computational costs.

3. Technical Details

The architecture described in this paper applies to a wide range of feature types, e.g., *histogram of oriented gradients* (HOG) [4] and *locally adaptive regression kernels* (LARK) [16]. The application and experiments presented here make use of the *deformable part model* (DPM) of Felzenszwalb *et al*. [8]. Many other object detection models can be adapted to use our approach, including multi-class cascades [14], recursive compositional models [22], and variants of convolutional neural networks [11].

3.1. Winner-Take-All Hashing

Following [8], we compute a HOG feature pyramid by converting each level of a standard image pyramid into a

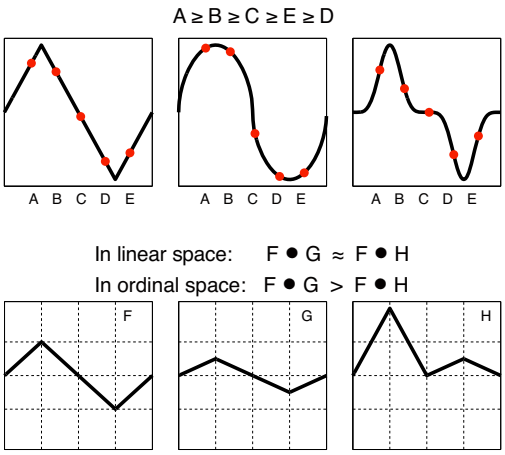


Figure 1. An ordinal measurement $A \geq B \geq C \geq E \geq D$ is robust to variations in individual filter values (top row). Ordinal measures of similarity capture filter response differences not reflected in linear measures (dot product) of similarity (bottom row).

HOG image using a filter roughly equivalent to the Dalal and Triggs model [4]. Rather than operate in the space of HOG features, we encode each filter-sized window of the HOG pyramid as a high-dimensional sparse binary descriptor called a *winner-take-all* (WTA) hash [21].

We restrict our attention to a subfamily of the hash functions introduced in [21] such that each WTA hash is defined by a sequence of N permutations of the elements in the fixed-sized window that we use in computing filter responses in the HOG pyramid. Each permutation consists of a list of indices into the vector obtained by flattening such a window of HOG coefficients; we need only retain the first K indices for each of the N permutations to implement a WTA hash function.

The term "winner take all" refers to each $(N * K)$-length descriptor comprising N spans of length K consisting of all zeros except the k^{th} entry, which is set to one to encode the index of the maximum value in the first K entries of the permutation. Each descriptor is compactly represented in $N * \lceil \log_2 K \rceil$ bits. For example, given $N = 2$ and $K = 3$, the flattened-window vector $[0.3, 0.7, 0.2, 0.5]$, and the permutations $[1, 4, 3, 2]$ and $[2, 1, 4, 3]$, the resulting WTA descriptor is $[0110]$: the first K indices of each permutation, $[1, 4, 3]$ and $[2, 1, 4]$ select $[0.3, 0.5, 0.2]$ and $[0.7, 0.3, 0.5]$ whose maximum values have the indices 2 and 1 encoded—least significant bit leftmost—in the binary vector. Note that since the only operation involved in the entire process is comparison, the hashing scheme can (a) be implemented completely with integer arithmetic and (b) can be efficiently coded without branching, and thus without branch prediction penalties.

Each WTA hash function defines an ordinal embedding and an associated rank-correlation similarity measure, which offers a degree of invariance with respect to perturbations in numeric values [21] and is well suited as a basis for locality-sensitive hashing. These deterministic functions are nonlinear and produce sparse descriptors that have been shown to yield significant improvements on VOC 2010 using simple linear classifiers that can be trained quickly [21].

Intuitively, the information stored in a WTA hash allows one to reconstruct a partial ordering of the coefficients in the hashed vector. The top row of Figure 1 highlights how partial-order statistics can be invariant to deformations and perturbations of individual filter values. In an image processing domain, all three filters in the top row of Figure 1 are qualitatively similar edge filters — this qualitative property follows from all three filters obeying the same ordinal statistic. Likewise, linear measures of similarity can be insensitive to qualitative differences easily captured by an ordinal measure of similarity. For instance, in the bottom row of Figure 1, filter F is qualitatively more similar to filter G than it is to filter H. A linear measure of similarity (dot product), however, regards F to be equally similar to G and H. An ordinal measure of similarity based on partial-order statistics would correctly identify F to be more similar to G than H.

3.2. Deformable Parts Models

A DPM consists of a set of part filters, a set of deformations that provide geometric information regarding the expected placement of parts in a patch, a scoring function that provides the basis for combining the deformations and part-filter responses, and a threshold tuned to meet a given precision-recall target. A filter is a matrix specifying weights for subwindows of a HOG pyramid. The score of a filter with respect to a subwindow of a HOG pyramid is the dot product of the weight vector and the features comprising the subwindow. A deformation is a symmetric, two-dimensional Gaussian mask superimposed on the target subwindow, with mean location specified relative to the subwindow.

The model for a particular object class is implemented as a mixture of DPMs, in which each component DPM is designed to capture one aspect of the class. In our implementation, each model comprises three mixture components representing three common aspect ratios for the bounding box of an object instance. This implies that a single model with, say, ten parts per component could have as many as thirty filters, requiring the same number of convolutions.

3.3. DPM with WTA

Figure 2 illustrates the basic detection and training architecture. In the results reported in this paper, we use the same basic training method as [8], and so to economize on space we focus the discussion on detection. The first step is to compute an image pyramid and the associated HOG pyramid for the target image. Felzenszwalb *et al.* [8] employ a

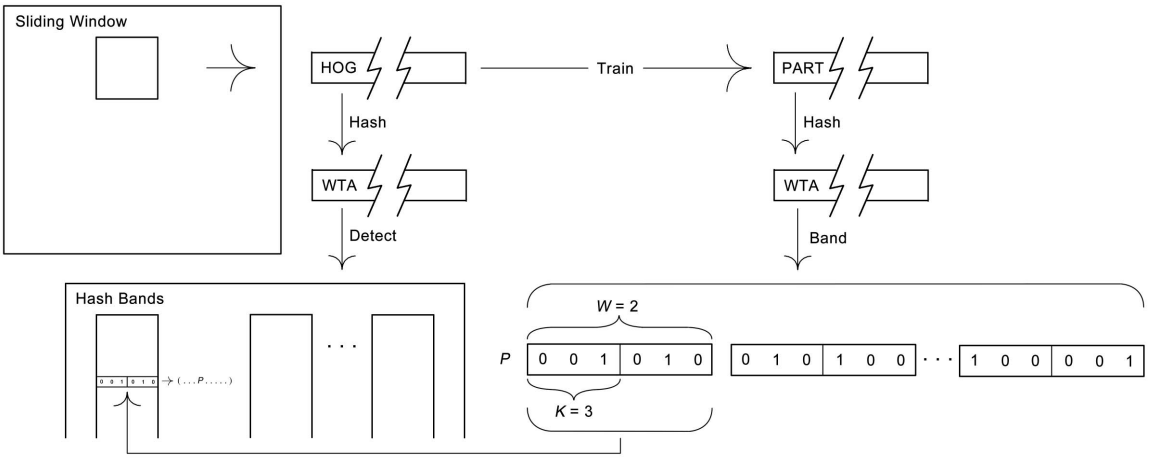


Figure 2. An illustration of the key steps in detecting objects and training the models: Training takes as input examples comprising filter-sized subwindows of a HOG pyramid as in [8] and produces as output a model that includes a set of filter-sized weight vectors called part filters. We compute the WTA hash for each weight vector, decompose the resulting descriptor into M bands consisting of W spans of length K, construct M hash tables and store each band in its respective table along with the index P of the corresponding part filter. During detection we compute the WTA hash of each filter-sized window of the target HOG pyramid, break the hash into bands, look up each band in its corresponding hash table, and count of how many times each part turns up in a table entry. We use these counts to identify candidate filters for which we compute the exact dot products.

root filter in addition to the part filters. In scoring a window at a specified level in the HOG pyramid, they combine the responses of the root filter within the selected window at that level and the responses of the part filters in an appropriately adjusted window at a level one octave higher than the specified level.

The contribution of a given part filter to the score of a subwindow is obtained by convolving the filter with the selected subwindow of the HOG pyramid, multiplying it by the deformation Gaussian mask and taking the maximum value. Instead of a response corresponding to the dot product of the filter weights and a filter-sized block of the HOG pyramid, we compute the WTA hash of that same filter-sized block of the feature map. We employ locality sensitive hashing [10] to recover the R largest responses corresponding to what the dot products would have been had we actually computed them.

During detection we employ hashing as a proxy for explicitly computing the dot product of the convolution subwindow with each of the weight vectors for all the part filters in all object models without actually enumerating the filters. To do so, we employ a multi-band LSH-style hash table to reconstruct the dot products that have a significant number of hash matches [15].

Consider the $(N * K)$-dimensional binary vectors produced by the WTA hash function. Divide the vector into equal-sized bands of size $W * K$. For $N = 2400$ and $K = 16$, $W = 4$ produces $N/W = M = 600$ bands, each taking $W * \lceil \log_2 K \rceil = 16$ bits. Create one hash table for each band to accommodate the hashed subvectors of length $W * K$. Let $\mathbf{w}$ be the vector of weight coefficients for a part filter and $\mathbf{u}$ be the binary vector corresponding to the WTA hash of $\mathbf{w}$.

Let B_m denote the m^{th} hash table and $\mathbf{u}_m$ the m^{th} $(W * K)$ span of weight coefficients in $\mathbf{u}$. The hash bin returned from hashing $\mathbf{u}_m$ in B_m is a list of filter indices, which we use to update a histogram of *filter match scores*. After we select the filters that are above threshold, we continue with Felzenszwalb *et al.*'s method, spreading now-sparse filter activations into a more dense map using the part deformation scores. Maxima in this map are our object detections.

A limitation of our hashing approach is that all part filters must be the same size. Because the object's bounding box has arbitrary aspect ratio, we cannot use a corresponding root filter. Since hashing is inexpensive, however, we mitigate the lack of a root filter by tiling the bounding box with part filters and obtaining activations for all of the tiles as we slide the bounding box over the scaled array of features.

Additionally, the filter match score generated through hashing is a lower bound on the filter response at that patch (due to hash grouping in bands). Once a candidate part is found, therefore, we compute the actual dot product of the underlying image features with the associated filter and use this in evaluating the location-based deformation. This result is combined with those of other parts for the candidate object to obtain the object's score, and we use this combined score to suppress all but the best detection in a region.

It is possible to train DPM models not on HOG data as explained in this section but rather on a hashed WTA version of the HOG data. This approach has several technical advantages but represents a more significant departure from the original DPM approach than we have space to present here. In the technical supplement associated with this paper, however, we sketch the basic algorithms for training and applying such WTA-based models.

4. Experimental Results

In the following, we use the term *baseline algorithm* or just *baseline* to refer to the detection algorithm described in [6] utilizing models generated by the training method from the same paper. By performing several experiments, we compare the performance of the baseline algorithm with the hashing-based detection algorithm described in the previous section, utilizing models generated by the training method from [6] but *sans* root filters as mentioned earlier. First, we demonstrate that the hashing-based algorithm compares favorably with the baseline algorithm on the PASCAL VOC 2007 dataset. Second, we show that the hashing-based algorithm can scale object detection to hundreds of thousands of object classes and provide insight into the trade-offs involving accuracy, memory and computation time. Finally, we show the results of training 100,000 object classes using our system and running the hashing-based algorithm on the resulting models on a single machine.

4.1. Datasets and Implementation Details

We employed the standard benchmark detection dataset, PASCAL VOC 2007, to test our detector. The PASCAL dataset contains images from 20 different categories with 5000 images for training and validation and a test set of size ∼5000. Further, to test our system at scale, we created a new dataset called ImageSearch-100k which contains training and validation images for 100,000 object concepts.

We compiled a list of 100,000 Freebase entities [3] from notable types deemed to contain visual objects and queried Google Image Search with these entities. For each query, we downloaded at most 500 images resulting in a dataset of 32 million images with an average of 300 images per query. We divided this dataset into a training set containing 80% of the images and a validation set containing the other 20%. Note that, since Image Search does not provide bounding box annotations, this is a weakly supervised dataset, and furthermore the dataset is noisy since the labels are based on text accompanying the images. Hand labeling this data even at the image level is infeasable.

For all experiments we used the extended HOG features with three mixture components, as proposed in [6]. All models were trained using parts consisting of a 6×6 grid of HOG cells. Training was carried out training in two stages initializing the models using warped examples in the first iteration and computing latent variables in subsequent iterations. We fixed the C parameter to 0.003 as recommended in [6] based on cross-validation on the PASCAL dataset.

4.2. PASCAL VOC 2007

In this section we compare the hashing-based algorithm with the baseline algorithm. Table 1 shows the average precision scores of each category for the two algorithms. Note that we report the results for the base system of [6], since we do not use bounding box prediction or context rescoring which were reported to provide further improvements. For the hashing parameters in this experiment, we use $K = 4$ and $W = 4$ for 8-bit hash keys and $M = 3000$ hash tables.

Our hashing-based algorithm compares favorably with the baseline with a mAP of 0.24 compared to their 0.26. We perform better than the baseline on three categories, similar to the baseline on eight categories and are worse on nine categories. The lack of a root filter in our implementation is responsible for much of the deficit, especially the person category which includes many examples too small to be captured by the high-resolution part filters. We are exploring options for restoring the root filter that would reduce this deficit with a small added computational cost.

4.3. Accuracy Versus Speed and Memory

In the previous section we showed that our hashing-based detector performs comparably to the baseline exhaustive detector using the best set of hash parameters. In this section we illustrate how the performance, along with the speed and memory requirements of the detector, change with the hash parameters.

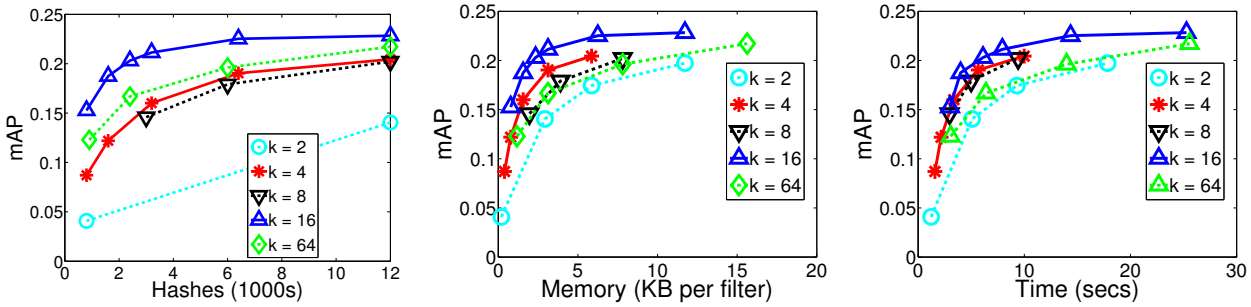
Figure 3(a) shows the mAP on the PASCAL VOC 2007 dataset for $K = \{2, 4, 8, 16, 64\}$ and the different numbers of hashes. For each case we choose W such that we obtain comparable hash keys of 16 or 18 bits. For all values of K we see that the accuracy increases with more hashes (as expected) and saturates beyond a certain number of hashes. $K = 16$ performs best for all values of K, which we believe is a property of the feature dimension and the percentage of useful dimensions.

Figure 3(b) shows the same set of results mapped onto the amount of memory used by the system. The memory required is a function of M, the number of hash tables. For $K = 16$, performance saturates at 5 KB per filter, which translates to 5 GB for storing 100,000 classes with 10 filters each. This demonstrates we can store filters for up to 100,000 classes on a single modestly-equipped workstation.

Figure 3(c) describes the trade-off between the accuracy and the total computation time for detecting all 20 object classes. At 5 seconds per image, we obtain a speed-up of approximately $20\times$ over the base exhaustive system, which compares well with cascade-based approaches for speeding up detection [7]. Note, however, that our approach scales

	arp	bike	bird	boat	bttl	bus	car	cat	chr	cow	tbl	dog	hrs	mbke	prsn	plnt	shp	sofa	trn	tv	Mean
Ours	0.19	0.48	**0.03**	0.10	0.16	**0.41**	0.44	0.09	0.15	**0.19**	0.23	**0.10**	**0.52**	0.34	0.20	**0.10**	0.16	**0.28**	**0.34**	0.34	0.24
[6] (base)	**0.29**	**0.55**	0.01	**0.13**	**0.26**	0.39	**0.46**	**0.16**	**0.16**	0.17	**0.25**	0.05	0.44	**0.38**	**0.35**	0.09	**0.17**	0.22	0.34	**0.39**	**0.26**

Table 1. Comparison of the hashing-based and baseline algorithms on the PASAL VOC 2007 dataset



(a) (b) (c)

Figure 3. Effect of hashing parameters on the accuracy, speed and memory required by the system.

constantly over the number of object classes after a fixed overhead, and, therefore, our speed-up increases with an increasing number of classes.

4.4. Training 100,000 Object Detectors

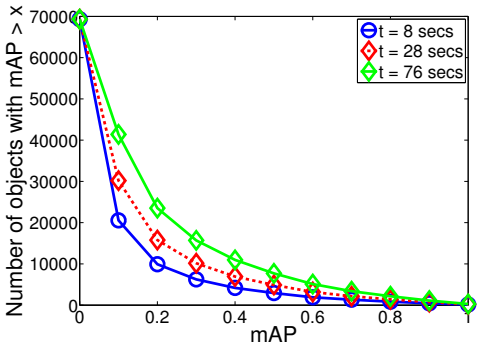
In this section we show that our hashing-based algorithm can be used to scale object detection to 100,000 classes. For this experiment, we train our object models using the ImageSearch-100K training set. Since this dataset does not contain bounding boxes we pursue a simple bootstrapping approach.

Bootstrapping. In the first iteration of training we initialize the model by using either the whole image as the bounding box, or, in simple images with plain backgrounds, by automatically cropping out any smoothly varying background regions. For most images, this is highly inaccurate. In subsequent iterations, we first compute a bounding box by running the detectors on each training image. If the detection score is higher than the score for the bounding box from the previous iteration we use the newly computed bounding box. This strategy works especially well on Image Search images, since most of the images contain the object of the interest and there is significantly less clutter than in the PASCAL VOC dataset. Training was performed using a cluster of 5000 machines for 10 iterations in under 24 hours. For detection, we use $K = 16$, $W = 4$ and $N = 2400$ in order to obtain a model of size 20 GB which can be run on a single machine.

Figure 4(a) shows a summary of the mean average precisions of 100,000 objects on the validation set for three different parameter settings that trade off speed and accuracy for the same memory. For a speed-up of $17,857\times$ (5 hours to 1 second), we obtain a mAP of 0.11 over 100,000 object classes, which is a significant result when considering the fact that the current state-of-the-art for object detection

time	t = 8	t = 28	t = 76
mAP	0.07	0.11	0.16
speed-up	$62,500\times$	$17,857\times$	$6,580\times$

(a)



(b)

Figure 4. Summary of mean average precision scores over the 100,000 objects for three different parameter settings.

is at 0.40 mAP over just 20 classes [5]. Furthermore, the Image Search dataset frequently contains images from multiple concepts for ambiguous entities such as apple (fruit, company), jaguar (animal, vehicle), etc.

Figure 4(b) shows the number of object classes with mAP above different levels. As expected, this follows an exponential trend; however, close to one-third of the objects have a mAP of 0.20, and one-fifth of the objects have a mAP of 0.30 for comparison with results on PASCAL VOC. Finally, Figure 5 shows the top six detection results (from both positive and negative images) for a representative set of categories for $t = 8$ seconds. Our detector is able to correctly rank images containing each object concept with the highest scores despite the large size of the negative set (images

Figure 5. Top few detections on the validation set for a representative set of categories ordered by detection score.

from all other categories).

Hand-labeled Ground Truth. Because our training sets come from Google Image Search, there is considerable noise in the data. We set a threshold at 80% precision on the validation set and sampled a set of 545 common objects whose detectors fired on a statistically significant number of images out of a set of 12 million random thumbnails from YouTube for which we had no ground truth. A number of detections for each object were labeled by hand using Mechanical Turk operators, and the full set was used to calculate the mean average precision.

Figure 6 shows the results. Some objects converged on generic patterns, resulting in precision near zero, while more than 10% have precision of at least 90%. 21 objects, including "rubber duck," "eyeglasses," "malamute," and "person," have 100% precision. Of course, since we cannot calculate the false negative rate of these detectors on our test set, 100% precision means only that all true positives in the labeled set appeared before the first false positive when sorted by decreasing score. It does show that there are acceptable high-precision operating points for many objects even using weakly-labeled, noisy training data.

4.5. Revisiting VOC 2007 with 10,000 Detectors

In this section we examine the prediction accuracy of the hashing-based detector as the number of unique objects in the system systematically increases. Section 4.3 demonstrated the inherent trade-off between the prediction accuracy and computational resources for our object detector. While it is theoretically possible to maintain the prediction accuracy of the object detector by increasing the number

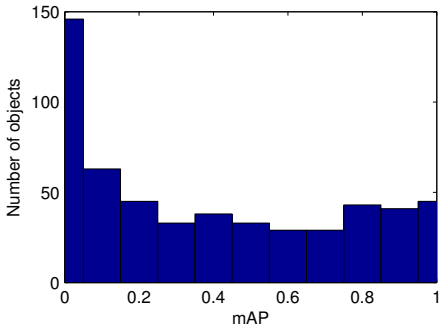


Figure 6. Histogram of mean average precisions on ground truth data for 545 objects with significant detection rate.

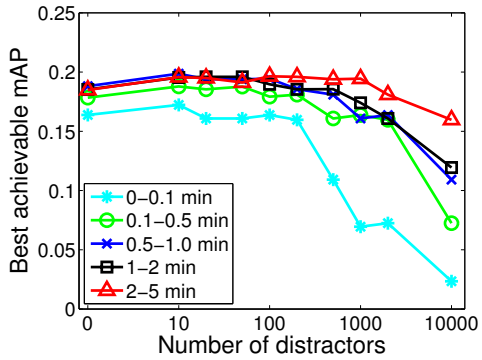


Figure 7. Increasing the number of objects gracefully degrades prediction accuracy on PASCAL VOC 2007 for a fixed computational budget.

hashes, as we vary the number of objects, this approach is problematic in that arbitrarily increasing the number of objects will inevitably lead to impractical computational demands and unacceptably slow computation times.

To explore how a hashing-based object detector scales to a large number of objects, we instead measure the prediction accuracy as we increase the number of unique objects for a *fixed* computational budget. We systematically perform an exhaustive search of ∼1000 sets of hashing model parameters [1]. For each set of model parameters we calculate (1) the average computational time for processing an image and (2) the mean average precision (mAP) on the subset of 20 objects labels from the PASCAL VOC 2007 validation data set. For a given number of object labels, we record the highest mAP from all of the models that process images within a specified time window.

Figure 7 plots the best achievable mAP for the subset of 20 PASCAL objects for a given time window across a range of object labels. Several trends are worth noting. First,

[1] We explored all combinations of the following parameters: K = 4; number of hash tables = 100, 200, 500, 1000; number of permutations per hash band = 2, 3, 4, 5, 6, 7; number of additional objects = 0, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000; number of matches per hash = 0, 1, 2, 3, 4.

models that process images faster exhibit worse prediction accuracy (e.g. 0-0.1 min, cyan curve). This observation reinforces the result from Figure 3. Second, as the number of objects increases, the prediction accuracy on the subset of 20 PASCAL VOC 2007 objects gracefully decreases. This is to be expected as the frequency of errant hash collisions increases, and the entropy of the hash table reaches capacity. Nonetheless, the far right point on the red curve indicates that for a particular set of model parameters, we achieve a minimal degradation in the prediction accuracy while still achieving a throughput of 2-5 minutes per image. This set of model parameters demonstrates that a hashing-based object detector can be scaled up to a large number of objects while achieving reasonable balance between prediction accuracy and evaluation throughput. We note that the absolute times reported here are based on unoptimized code.

5. Conclusions

Our key contribution is a scalable approach to object detection that replaces linear convolution with ordinal convolution by using an efficient LSH scheme. This approach is applicable to a variety of object detection methods. Through extensive empirical tests on DPM detectors, we have shown that (a) the system performs comparably to the original DPM detectors, (b) performance degrades gracefully as the number of object classes is increased, and (c) up to 100,000 object classes can be simultaneously detected on a single machine in under 20 seconds.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 73–80, 2010. 2

[2] I. Biederman. Aspects and extensions of a theory of human image understanding. In Z. W. Pylyshyn, editor, *Computational processes in human vision: An interdisciplinary perspective*, pages 370–428. Ablex, Norwood, NJ, 1988. 2

[3] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, New York, NY, USA, 2008. 5

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005. 2, 3

[5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/. 6

[6] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1627–1645, 2010. 5, 6

[7] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2241–2248, 2010. 2, 5

[8] P. F. Felzenszwalb, D. A. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. 2, 3, 4

[9] C. Gu, P. A. Arbeláez, Y. Lin, K. Yu, and J. Malik. Multi-component models for object detection. In *European Conference on Computer Vision*, pages 445–458, 2012. 2

[10] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *ACM Symposium on Theory of Computing*, pages 604–613, 1998. 1, 4

[11] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012. 2

[12] M. Pedersoli, A. Vedaldi, and J. Gonzàlez. A coarse-to-fine approach for fast deformable object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1353–1360, 2011. 2

[13] H. Pirsiavash and D. Ramanan. Steerable part models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3226–3233, 2012. 2

[14] E. Rahtu, J. Kannala, and M. B. Blaschko. Learning a category independent object detection cascade. In *IEEE International Conference on Computer Vision*, pages 1052–1059, 2011. 2

[15] A. Rajaraman and J. D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2011. 4

[16] H. J. Seo and P. Milanfar. Training-free, generic object detection using locally adaptive regression kernels. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 32(9):1688–1704, 2010. 2

[17] H. Song, S. Zickler, T. Althoff, R. Girshick, M. Fritz, C. Geyer, P. Felzenszwalb, and T. Darrell. Sparselet models for efficient multiclass object detection. In *European Conference on Computer Vision*, 2012. 2

[18] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders. Segmentation as selective search for object recognition. In *IEEE International Conference on Computer Vision*, pages 1879–1886, 2011. 2

[19] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *IEEE International Conference on Computer Vision*, pages 606–613, 2009. 2

[20] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–574, 2002. 2

[21] J. Yagnik, D. Strelow, D. A. Ross, and R.-s. Lin. The power of comparative reasoning. In *IEEE International Conference on Computer Vision*, 2011. 1, 3

[22] L. Zhu, Y. Chen, A. L. Yuille, and W. T. Freeman. Latent hierarchical structural learning for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1062–1069, 2010. 2