An Active Search Strategy for Efficient Object Class Detection

Abel Gonzalez-Garcia
a.gonzalez-garcia@sms.ed.ac.uk

Alexander Vezhnevets
avezhnev@inf.ed.ac.uk

Vittorio Ferrari
vferrari@staffmail.ed.ac.uk

University of Edinburgh

Abstract

Object class detectors typically apply a window classifier to all the windows in a large set, either in a sliding window manner or using object proposals. In this paper, we develop an active search strategy that sequentially chooses the next window to evaluate based on all the information gathered before. This results in a substantial reduction in the number of classifier evaluations and in a more elegant approach in general. Our search strategy is guided by two forces. First, we exploit context as the statistical relation between the appearance of a window and its location relative to the object, as observed in the training set. This enables to jump across distant regions in the image (e.g. observing a sky region suggests that cars might be far below) and is done efficiently in a Random Forest framework. Second, we exploit the score of the classifier to attract the search to promising areas surrounding a highly scored window, and to keep away from areas near low scored ones. Our search strategy can be applied on top of any classifier as it treats it as a black-box. In experiments with R-CNN on the challenging SUN2012 dataset, our method matches the detection accuracy of evaluating all windows independently, while evaluating $9\times$ *fewer windows.*

1. Introduction

Given an image, the goal of object class detection is to place a bounding-box around every instance of a given object class. Modern object detectors [8, 10, 19, 22, 26, 35, 49, 53] partition the image into a set of windows and then score each window with a classifier to determine whether it contains an instance of the object class. The detector finally outputs the windows with the locally highest scores.

In the classical sliding window approach [10, 19, 26, 35], the window set is very large, containing hundred of thousands of windows on a regular grid at multiple scales. This approach is prohibitively expensive for slow, powerful window classifiers which are state-of-the-art nowadays [8, 15, 22, 49, 53] such as Convolutional Neural Networks (CNN) [22]. For this reason, these detectors are

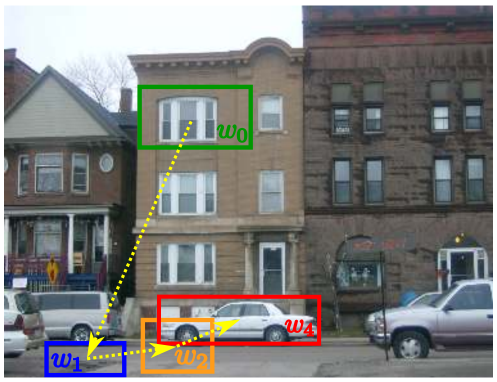


Figure 1: **Example of a car search using our method.**

based instead on object proposal generators [2, 36, 49], which provide a smaller set of a few thousand windows likely to cover all objects. Hence, this reduces the number of window classifier evaluations required. However, in both approaches the window classifier evaluates *all* windows in the set, effectively assuming that they are independent.

In this work, we propose an *active search strategy* that sequentially chooses the next window to evaluate based on previously observed windows, rather than going through the whole window set in an arbitrary order. Observing a window not only provides information about the presence of the object in that particular window, but also about its surroundings and even distant areas of the image. Our search method extracts this information and integrates it into the search, effectively guiding future observations to interesting areas, likely to contain objects. Thereby, our method explores the window space in an intelligent fashion, where future observations depend on all the information gathered so far. This results in a more natural and elegant way of searching, avoiding wasteful computation in uninteresting areas and focusing on the promising ones. As a consequence, our method is able to find the objects while evaluating much fewer windows, typically only a few hundreds (sec. 7.1).

We use two guiding forces in our method: context and window classifier score. Context exploits the statistical relation between the appearance and location of a window and its location relative to the objects, as observed in the training set. For example, the method can learn that cars tend to be on roads below the sky. Therefore, observing a window in the sky in a test image suggests the car is likely to be far below, whereas a window on the road suggests making a smaller horizontal move. We learn the context force, in a Random Forest framework that provides great computational efficiency as well as accurate results. The classifier score of an observed window provides information about the score of nearby windows, due to the smoothness of the classifier function. It guides the search to areas where we have observed a window with high score, while pushing away from windows with low score. Observing a window with part of a car, for example, will attract the search to its surroundings.

Our method effectively combines these two forces. Fig. 1 shows the intuition of our method on detecting cars. It starts at window w_0 and it moves away immediately, since w_0 contains a piece of building, not a car. Context determines the direction of the move, as cars tend to be on streets below buildings. Hence, the next visited location is on the road. After observing w_1, the method continues searching along the road, as indicated by context. For window w_2, however, the score of the classifier is rather high, as it contains a piece of car. Therefore, the search focuses around this area until it finds a tight window on the car.

Experiments on the challenging SUN2012 dataset [55] and PASCAL VOC10 [16] demonstrate that our method explores the image in an intelligent way, effectively detecting objects in only a few hundred iterations. As window classifiers we use the state-of-the-art R-CNN [22] and the popular UvA Bag-of-Words model of [49], both on top of object proposals [49]. For R-CNN on SUN2012, our search strategy matches the detection accuracy of evaluating all proposals independently, while evaluating $9\times$ fewer proposals. As our method adds little overhead, this translates into an actual wall-clock speedup. When computing CNN features on the CPU [29], the processing time for one test image reduces from 320s to 36s ($9\times$ speed-up). When using a GPU, it reduces from 14.4s to 2.5s ($6\times$ speed-up). Hence, our method opens the door to using expensive classifiers by considerably reducing the number of evaluations while adding little overhead. For the UvA window classifier, our search strategy only needs 35 proposals to match the performance of evaluating all of them (a reduction of $85\times$). By letting the search run for longer, we even *improve* accuracy while evaluating $30\times$ fewer proposals, as it avoids evaluating some cluttered image areas that lead to false-positives.

2. Related Work

Object proposals. Recent, highly accurate window classifiers like high-dimensional Bag-of-Words [49] or CNN [15, 22, 30] are too expensive to evaluate in a sliding window fashion. For this reason, recent detectors [8, 22, 49, 53] evaluate only a few thousands windows produced by object proposals generators [2, 36, 49]. The state-of-the-art detector [22] follows this approach, using CNN features [30] with Selective Search proposals [49]. Although proposals already reduce the number of window classifier evaluations, our work brings even further reductions.

Improving sliding window. Some works reduce the number of window classifier evaluations. Lampert et al. [31] use a branch-and-bound scheme to efficiently find the maximum of the classifier over all windows. However, it is limited to classifiers for which tight bounds on the highest score in a subset of windows can be derived. Lehman et al. [33] extend [31] to some more classifiers. Sznitman et al. [46] avoid exhaustive evaluation for face detection by using a hierarchical model and pruning heuristics.

An alternative approach is to reduce the cost of evaluating the classifier on a window. For example, [26, 51] first run a linear classifier over all windows and then evaluate a complex non-linear kernel only on a few highly scored windows. Several techniques are specific to certain types of window classifiers and achieve a speedup by exploiting their internal structure (e.g. DPM [18, 43, 57], CNN-based [27], additive scoring functions [54], cascaded boosting on Haar features [45, 52]. Our work instead can be used with any window classifier as it treats it as a black-box.

A few works develop techniques that make sequential fixations inspired by human perception for tracking in video [4], image classification [12, 32, 37] and face detection [6, 47]. However, they only use the score of a (foveated) window classifier, not exploiting the valuable information given by context. Moreover, they experiment on simple datasets, far less challenging than SUN2012 [55] (MNIST digits, faces).

Context. Many works use context as an additional cue on top of object detectors, complementing the information provided by the window classifier, but without altering the search process. Several works [19, 26, 41, 48, 49] predict the presence of object classes based on global image descriptors, and use it to remove out-of-context false-positive detections. The response of detectors for multiple object classes also provides context, as it enables to reason about co-occurrence [44] and spatial relations between classes [7, 13, 21, 28]. Other works incorporate regions outside the object into the window classifier [11, 34, 40, 49] Divvala et al. [14] analyze several context sources and their impact on object detection.

The most related work to ours is [3], which proposes a search strategy driven by context. Here we go beyond in several ways: (1) They used context in an inefficient way, involving a nearest-neighbour search over all windows in all training images. This caused a large overhead that compromised the actual wall-clock speedup they made over evalu-

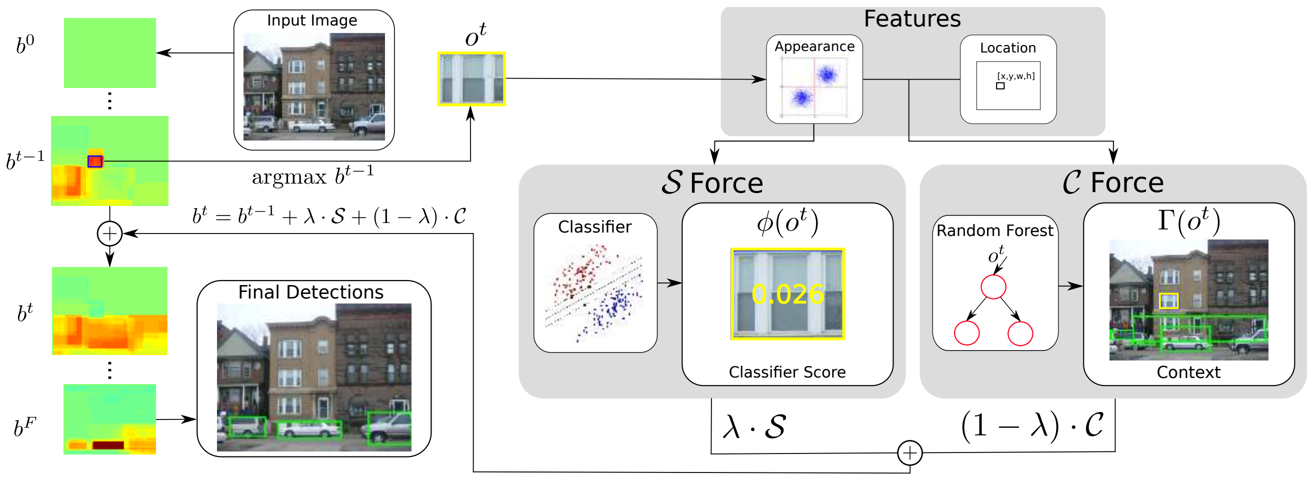


Figure 2: **Search model.** *The next observed window o^t is the maximum of the current belief map b^{t-1}. The method extracts appearance and location features for o^t, and uses them to compute its context $\mathcal{C}$ and window classifier $\mathcal{S}$ outputs. Then, it combines these outputs with the current belief map b^{t-1} into the next iteration's belief map b^t. The final belief map b^F combines all the performed observations. The output detections are the observed windows with highest scores.*

ating all windows in the test image. In contrast, we present a very efficient technique based on Random Forests, which has little overhead (sec. 7.2). (2) While [3] uses only context, we guide the search also by the classifier score, and learn an optimal combination of the two forces (sec. 3). (3) They perform single-view and single-instance detection, whereas we detect multiple views and multiple instances in the same image. (4) We adopt the state-of-the-art R-CNN [22] as the reference detector and compare to it, as opposed to the weaker DPM detector [19]. (5) While [3] performs experiments only on PASCAL VOC10, we also use SUN2012 [55], which has more cluttered images with smaller objects.

3. Search model

Let I be a test image represented by a set of object proposals [49], $I = \{o_i\}_{i=1}^N$. The goal of our method is to efficiently detect objects in I, by evaluating the window classifier on only a subset of the proposals. Our method is a class-specific iterative procedure that evaluates one window at a time. At every iteration t, it selects the next window o^{t+1} according to all the observations $\{o^k\}_{k=1}^t$ performed so far.[1] We assign a belief value $b^t(o_i, \{o^k\}_{k=1}^t; \Theta)$ to each object proposal o_i and update it after every iteration. This belief indicates how likely it is that o_i contains the object, given all previously observed windows $\{o^k\}_{k=1}^t$. Here $\Theta = \{\lambda, \sigma_\mathcal{S}, \sigma_\mathcal{C}\}$ are hyperparameters and t indexes the iteration.

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

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

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

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

$$\begin{aligned} b^t(o_i, \{o^k\}_{k=1}^t; \Theta) &= b^{t-1}(o_i, \{o^k\}_{k=1}^{t-1}; \Theta) \\ &+ \lambda \cdot \mathcal{S}(o_i, o^t; \sigma_\mathcal{S}) + (1-\lambda) \cdot \mathcal{C}(o_i, o^t; \sigma_\mathcal{C}) \end{aligned} \tag{2}$$

The observation o^t provides two kind of information: the context $\mathcal{C}$ and the classifier score $\mathcal{S}$ (explained below). These are linearly combined with a mixing parameter $\lambda \in [0, 1]$. Fig. 2 illustrates our pipeline.

Context force $\mathcal{C}$ points to areas of the image likely to contain the object, relative to the observation o^t. It uses the statistical relation between the appearance and location of training windows and their position relative to the objects. The context force may point to any area of the image, even those distant from o^t. For the car detection example, if o^t contains a patch of building, $\mathcal{C}$ will point to windows far below it, as cars are below buildings (fig. 1, 6). If o^t contains a patch of road, $\mathcal{C}$ will propose instead windows next to o^t, as cars tend to be on roads (fig. 1).

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

[1] o_i indexes through the input set of proposals I, whereas o^t is the proposal actively chosen by our strategy in the t-th iteration.

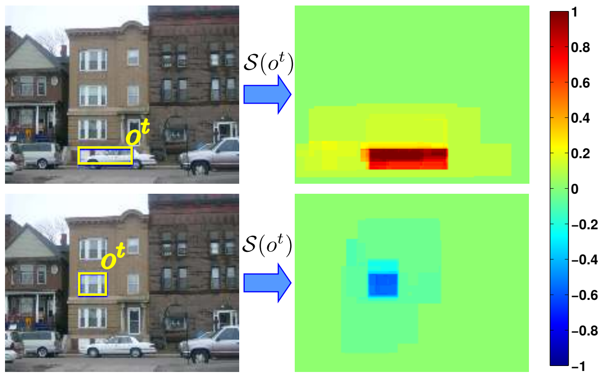


Figure 3: **Classifier score force $\mathcal{S}$.** *(Left) Image and observation o^t. (Right) Belief map produced by $\mathcal{S}$. Colours correspond to belief values.*

relative position to the objects in their own images. We explain how our context extractor works in sec. 4.

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

$$\mathcal{C}(o_i, o^t; \sigma_{\mathcal{C}}) = \sum_{w_j \in \Gamma(o^t)} K(w_j, o_i; \sigma_{\mathcal{C}}) \tag{3}$$

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

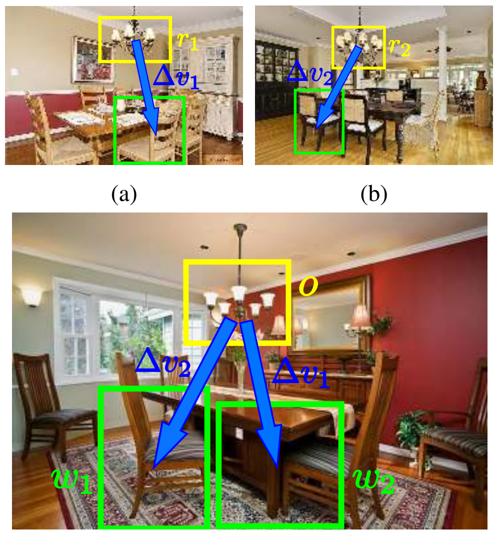
$$K(w, o; \sigma) = e^{-(1-\mathrm{IoU}(w,o))^2/(2\sigma^2)} \tag{4}$$

This choice of kernel assumes smoothness in the presence of an object for nearby windows. Indeed, adjacent windows to a window containing an object will also contain part of the object. The further apart the windows are, the lower the probability of containing the object is. We use the inverse overlap 1 - intersection-over-union [16] (IoU) as distance between two windows.

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

$$\mathcal{S}(o_i, o^t; \sigma_{\mathcal{S}}) = K(o_i, o^t; \sigma_{\mathcal{S}}) \cdot (\phi(o^t) - 0.5) \tag{5}$$

where $\phi(o^t) \in [0, 1]$ is the window classifier score of o^t (sec.4 details our choice of window classifier). We translate $\phi(o^t)$ into the $[-0.5, 0.5]$ range and weight it using the smoothing kernel (4). Therefore, $\mathcal{S}$ operates in the surroundings of o^t, spreading the classifier score to windows near o^t. When $\mathcal{S}$ is positive, it attracts the search to the region surrounding o^t. For example, if o^t contains part of a car, it will probably have a high classifier score (fig. 3). Then $\mathcal{S}$ will guide the search to stay in this area, as some nearby window is likely to contain the whole car. On the other hand, when $\mathcal{S}$ values are negative, it has a repulsive effect. It pushes the search away from uninteresting regions with low classifier score, such as background (sky, buildings, etc).



(c)

Figure 4: **Context extractor.** *(a,b) The displacement vectors Δv_1 and Δv_2 of training samples r_1 and r_2 point to their respective ground-truth objects. (c) By applying Δv_1 and Δv_2 to test observation o, we obtain displaced windows w_1 and w_2, covering likely locations for the object with respect to o.*

4. Context extractor

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

The context extractor is trained from the same data as the window classifier, i.e. images with annotated object bounding-boxes. Hence our approach requires the same annotation as standard object detectors [8, 10, 19, 22, 26, 35, 49, 53]. The training set consists of pairs $\{(r_n, \Delta v_n)\}_{n=1}^N$; r_n is a proposal from a training image, and Δv_n is a 4D displacement vector, which transforms r_n into the closest object bounding-box in its training image (fig. 4a-b). Here index n runs over all object proposals in all the training images. For 500 images and 3200 proposals per image, $N = 500 \cdot 3200 = 1'600'000$.

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

Random Forests. We use Random Forests (RF) [5] as our context extractor. A RF is an ensemble of J binary decision trees, each tree inputs the window o and outputs a displacement vector Δv_j. The final output of RF are all displacement vectors $\{\Delta v_j\}_{j=1}^J$ produced by each tree.

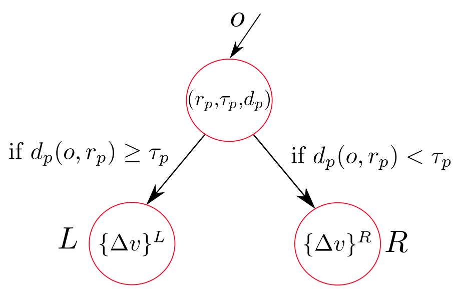


Figure 5: **Internal node at test time.** *The test compares distance $d_p(o, r_p)$ between test window o and training sample r_p with the threshold τ_p.*

RF have been successfully applied in several learning problems such as classification, regression or density estimation [9]. RF in computer vision [9, 17, 20, 23, 38] typically use axis-aligned separators (thresholding on one feature dimension) as tests in the internal nodes. However, we found that tests on distances to training samples perform better in our case, as they are more informative. Hence, we build our RF based on distance tests. This is related to Proximity Forests [42], although [42] uses RF for clustering, not geometric regression.

When the test window o goes down a tree, it traverses it from the root to a leaf guided by tests in the internal nodes. At each node p the decision whether o goes left or right is taken by comparing the distance $d_p(o, r_p)$ between o and a pivot training point r_p to the threshold τ_p. The test window o proceeds to a left child if $d_p(o, r_p) \geq \tau_p$ or to the right child otherwise (fig. 5). This process is repeated until a leaf is reached. Each leaf stores a displacement vector, which the tree returns. The triplet (r_p,τ_p,d_p) at each internal node is chosen during training (the process can choose between two different distance functions, sec. 6).

RF training. For each class, we train one RF with $J = 10$ trees. To keep the trees diverse, we train each one on windows coming from a random sub-sample of 40 training images. As shown in [9], this procedure improves generalization. We construct each tree by recursively splitting the training set at each node. We want to learn tests in the internal nodes such that leaves contain samples with a compact set of displacement vectors. This way a tree learns to group windows using features that are predictive of their relative location to the object. To create an internal node p, we need to select a triplet (r_p,τ_p,d_p), which defines our test function. Following the extremely randomized forest framework [39] we generate a random set of possible triplets. We then pick the triplet that achieves maximum information gain:

$$IG = H(S) - \frac{|S^L|}{|S|} \cdot H(S^L) - \frac{|S^R|}{|S|} \cdot H(S^R), \qquad (6)$$

where S is the set of training samples at the node and L, R denote the left and right children with samples S^L and S^R, respectively. H is the approximated Shannon entropy of

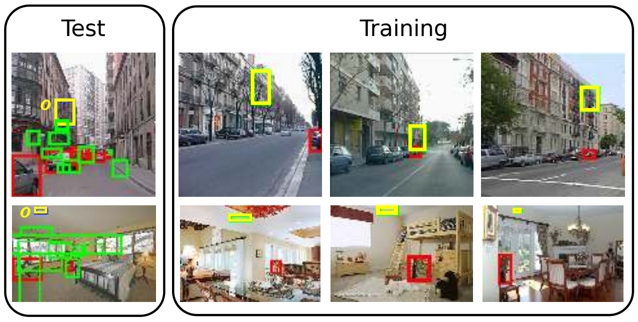


Figure 6: **RF examples.** *(Test) Example image and observation o for classes car (top) and chair (bottom). Windows displaced by the displacement vectors regressed by the RF are in green, whereas ground-truth objects are in red. (Training) Training samples (yellow) in leaves reached by o when input into our RF and associated ground-truth objects (red).*

the 4D displacement vectors in S: we first compute a separate histogram per dimension, and then sum their individual entropies [25]. Finally, we keep in each leaf the mediod displacement vector of the training samples that fell into it.

Fig. 6 shows examples test windows passed through the RF, along with example training windows in leaves they reach. Note how these training windows are similar in appearance to the test window, and produce displaced windows covering areas likely to contain objects of that class in the test image. This demonstrates that RF is capable of learning the relation between the appearance of a window and its position relative to the object.

Efficiency. A key benefit of RF over a simple nearest-neighbour regressor [3] is computational efficiency. Since the observation o is only compared to at most as many pivot points as the depth of the tree, the runtime of RF grows only logarithmically with the size of the training set. In contrast, the runtime of nearest-neighbour grows linearly, since it compares o to all training samples. This makes a substantial different in runtime in practice (sec. 7.2).

5. Classifier scores

As our method supports any window classifier ϕ, we demonstrate it on two very different ones: R-CNN [22] and UvA [49].

R-CNN. This detector is based on the CNN model of [30], which achieved winning results on the ILSVRC-2012 image classification competition [1]. The CNN is then fine-tuned from a image classifier to a window classifier on ground-truth bounding-boxes. Finally, a linear SVM classifier is trained on normalized 4096-D features, obtained from the 7th layer of the CNN. We use the R-CNN implementation provided by the authors [22], based on *Caffe* [29].

UvA. The Bag-of-Words technique of [49] was among the best detectors before CNNs. A window is described by a 3x3 spatial pyramid of bag-of-words. The codebook has 4096 words and is created using Random Forest on PCA-reduced dense RGB-SIFT [50] descriptors. Overall, the window descriptor has 36864 dimensions. The window classifier is an SVM with a histogram intersection kernel on these features. We use the implementation provided to us kindly by the authors [49].

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

6. Technical details

Object proposals. We use the fast mode of Selective Search [49], giving 3200 object proposals per image on average. These form the set of windows o_i visible to our method (both to the context extractor and window classifier). Note how both the R-CNN and UvA detectors as originally proposed [22, 49] also evaluate their window classifier on these proposals.

Features and distances for context extractor. We represent a proposal by two features: location and appearance. A proposal location $[x/W, y/H, w/W, h/H]$ is normalized by image width W and height H. Here x, y, w, h are the top-left coordinates, width and height of the proposal. The distance function is the inverse overlap $1 -$ IoU.

The appearance features used by the context extractor match those in the window classifier. We embed the 4096-dimensional CNN appearance features [29] in a Hamming space with 512 bits using [24]. This reduces the memory footprint by 256$\times$ (from 131072 to just 512 bits per window). It also speeds up distance computations, as the Hamming distance between these binary strings is 170$\times$ faster than L2-distance in the original space. We do the same for the Bag-of-Words features of [49]. These Hamming embeddings are used only for the context extractor (sec. 4). The window classifiers work on the original features (sec. 4).

Training hyperparameters Θ. For each object class, we find optimal hyperparameters $\sigma_{\mathcal{S}}$, $\sigma_{\mathcal{C}}$, and λ by maximizing object detection performance by cross-validation on the training set (by grid search in ranges $\sigma_{\mathcal{S}}, \sigma_{\mathcal{S}} \in [0.01, 1]$ and $\lambda \in [0, 1]$). Performance is quantified by the area under the Average Precision (AP) curve, which reports AP as a function of the number of proposals evaluated by the strategy (fig. 7). Interestingly, the learned σ values correspond to intuition. For $\sigma_{\mathcal{S}}$ we obtain small values, as the classifier score informs only about the immediate neighborhood of the observed window. Values for $\sigma_{\mathcal{C}}$ are larger, as the context force informs about a broader region of the image. Furthermore, $\mathcal{C}$ uses arbitrary windows, hence the distance to proposals is generally larger.

7. Experiments

We perform experiments on two datasets: SUN2012 [55] and PASCAL VOC10 [16].

SUN2012. We use all available images for the 5 most frequent classes in the highly challenging SUN2012 dataset [55]: Chair, Person, Car, Door and Table. This amounts to 2920 training images and 6794 test images, using the official train/test split provided with the dataset [56]. Each image is annotated with bounding-boxes on instances of these 5 classes. This dataset contains large cluttered scenes with small objects, as it was originally created for scene recognition [55]. This makes it very challenging for object detection, and also well suited to show the benefits of using context in the search strategy.

PASCAL VOC10. We use all 20 classes of PASCAL VOC10 [16]. While also challenging, on average this dataset has larger and more centered objects than SUN2012. We use the official splits, train on the `train` set (4998 images) and test on the `val` set (5105 images).

Protocol. We train the window classifier (including fine-tuning for R-CNN), the Random Forest regressor and the hyperparameters Θ on the training set. We measure performance on the test set by Average Precision (AP), following the PASCAL protocol [16] (i.e. a detection is correct if it overlaps a ground-truth object > 0.5). Previous to the AP computation, we use Non-Maxima Suppression [19] to remove duplicate detections.

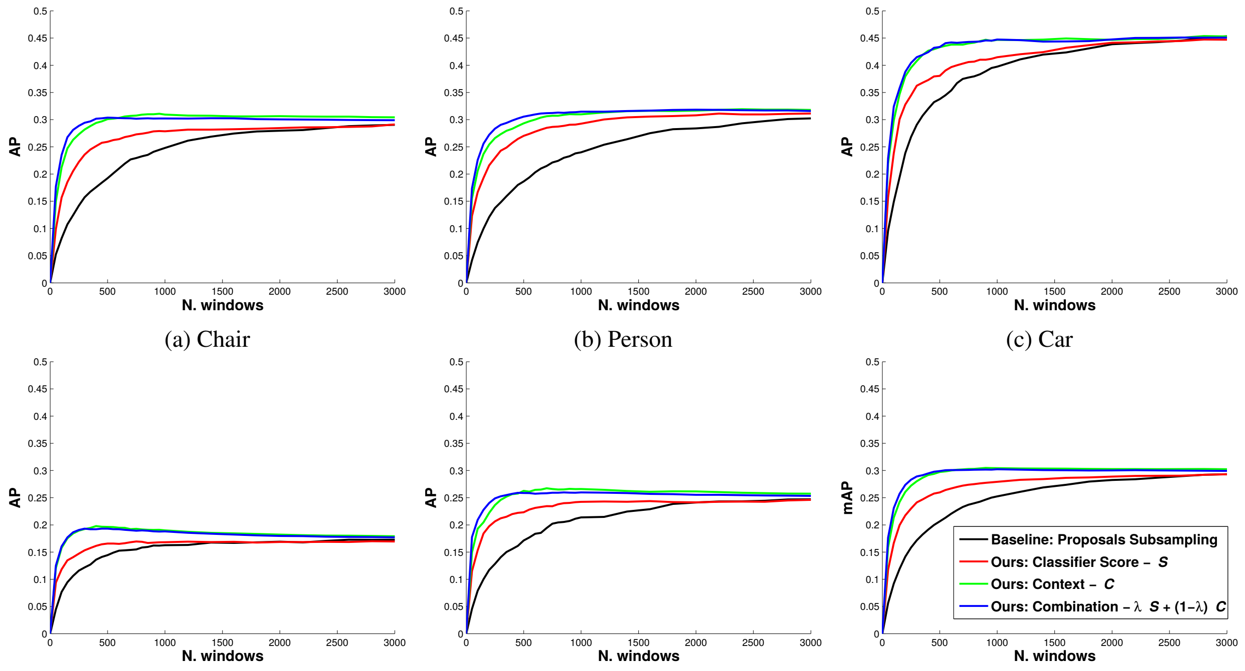
7.1. Results

R-CNN on SUN2012. Fig. 7 presents results for our full system ('Combination') and when using each force $\mathcal{S}$ or $\mathcal{C}$ alone. The figure shows the evolution of AP as a function of the number of proposals evaluated. As a baseline, we compare to evaluating proposals in a random sequence ('Proposals Subsampling'). This represents a naive way of reducing the number of evaluations. The rightmost point on the curves represent the performance of evaluating all proposals, i.e. the original R-CNN method.

Our full method clearly outperforms Proposals Subsampling, by selecting a better sequence of proposals to evaluate. On average over all classes, by evaluating about 350 proposals we match the performance of evaluating all proposals (fig. 7f). This corresponds to a 9$\times$ reduction in the number of window classifier evaluations.

In general, we achieve our best results by combining both forces $\mathcal{S}$ and $\mathcal{C}$. When using one force alone, $\mathcal{C}$ performs better, in some cases even reaching the accuracy of our combined strategy. Nevertheless, force $\mathcal{S}$ achieves surprisingly good results by itself, providing a rather simple method to speed-up state-of-the-art object detectors while maintaining high accuracy.

Fig. 9 shows our search strategy in action. After just a few iterations the belief maps are already highlighting areas containing the objects. Uninteresting areas such as the sky or ceiling are barely ever visited, hence we waste little computation. Our method detects multiple object instances and



(a) Chair (b) Person (c) Car

(d) Door (e) Table (f)

Figure 7: **Results for SUN2012**. *Results for the baseline Proposals Subsampling and our method on SUN2012, using each force $\mathcal{S}$, $\mathcal{C}$ alone and in combination. The x-axis shows the number of evaluated windows. The y-axis in (a-e) shows the AP of each class, while in (f) it shows the mean AP over all classes (mAP).*

viewpoints, even in challenging images with small objects. In these examples, it finds the first instance in fewer than 50 iterations, showing its efficiency in guiding the search. After finding the first instance, it continues exploring other areas of the image looking for more instances.

As a last experiment, we compare our RF context extractor with one based on nearest-neighbour search, as in [3]. We run our method only using the context force $\mathcal{C}$, but substituting RF with nearest-neighbours, on the same training set and input features. The results show that both ways of extracting context lead to the same performance. The AP at 500 windows, averaged over all classes, differs by only 0.006. Importantly, however, RF is $60\times$ faster (sec. 7.2).

UvA on SUN2012. We re-trained all the elements of our method on the Bag-of-Words features of [49]: window classifier, RF regressor, and hyperparameters. Our method matches the performance of evaluating all proposals with 35 windows on average, a reduction of $85\times$ (fig. 8-left) . Interestingly, the curve for our method reaches an even *higher* point when evaluating just 100 windows (+0.02 mAP). This is due to avoiding some cluttered areas where the window classifier would produce false-positives. This effect is less noticeable for the R-CNN window classifier, as UvA is more prone to false-positives.

R-CNN on PASCAL VOC10. As fig. 8-right shows, our method outperforms the Proposal Subsampling baseline again, having a very rapid growth in the first 100 windows.

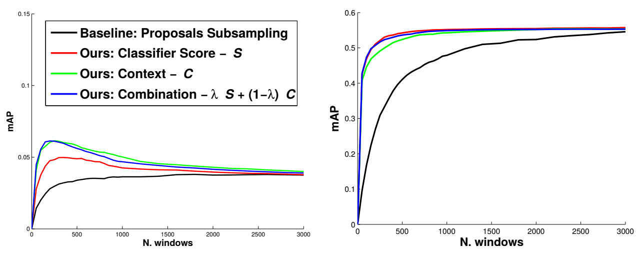


Figure 8: *(Left) Results on SUN2012 using the UvA detector [49]. (Right) Results on PASCAL VOC10 using R-CNN.*

7.2. Runtime

We measure runtimes on an Intel Xeon E5-1620v2 CPU and a GeForce GTX 770 GPU, for the R-CNN detector. The most expensive component is computing CNN features, which takes 4.5 ms per window on the GPU. Evaluating the 3200 proposals in an average image takes 14.4 seconds. The total overhead added by our method is 2.6 ms per iteration. Therefore, processing one image while maintaining the same AP performance (350 iterations on SUN2012) takes $350 \cdot (4.5 + 2.6)$ ms $= 2.5$ seconds, i.e. $6\times$ faster than evaluating all proposals [2].

The small overhead added by our method is mainly due to the RF query performed at each iteration for the context

[2]Extracting CNN descriptors on a GPU is more efficient in batches than one at a time, and is done in R-CNN [22] by batching many proposals in a single image. In our sequential search we can form batches by processing one window each from many different images.

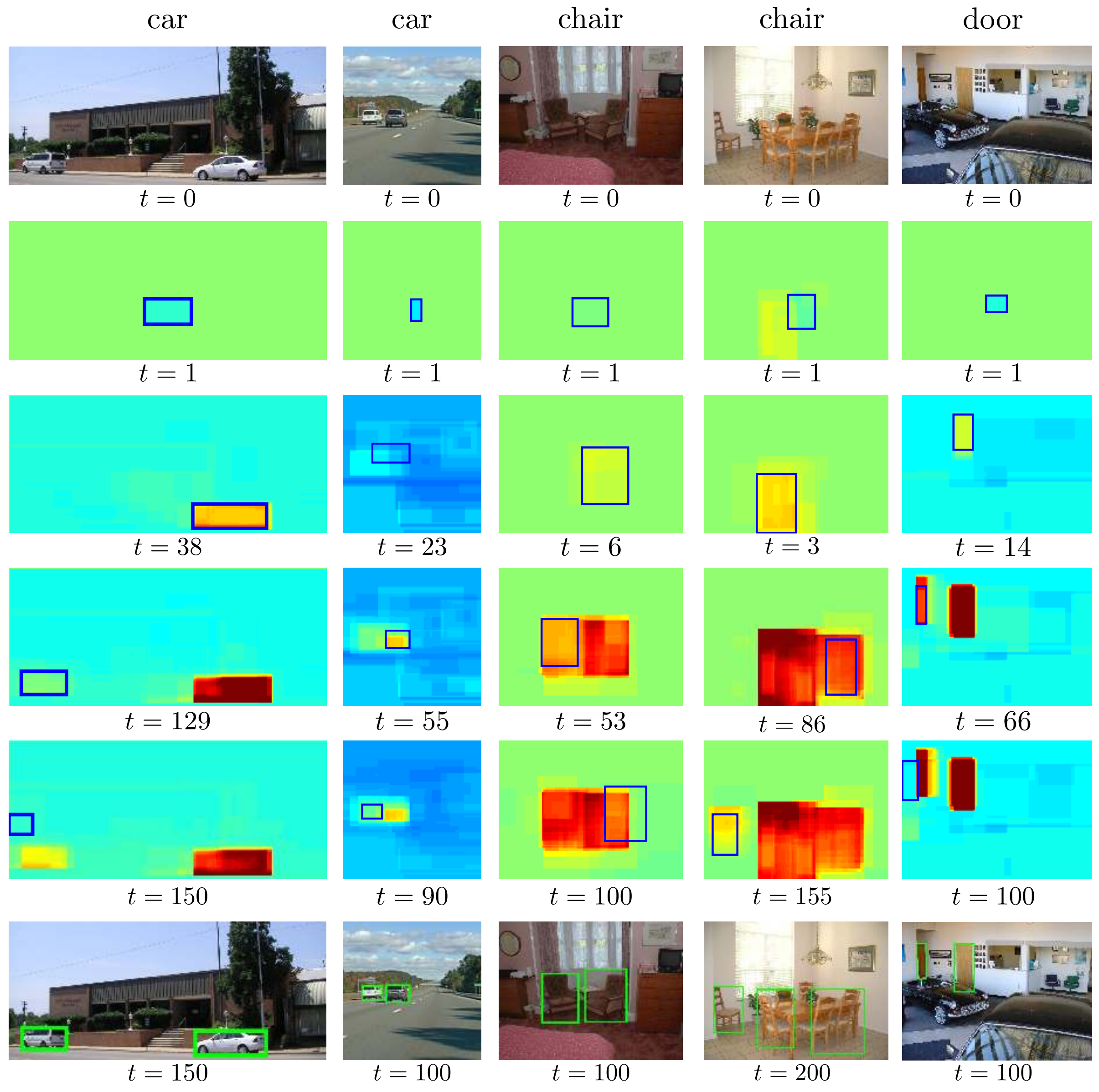


Figure 9: **Qualitative results.** *(Top) Original image. (Middle) Belief maps for some iterations. The blue window indicates the observation at that iteration. (Bottom) The top scored detections.*

force, which amounts to 1.9 ms including the Hamming embedding of the appearance features. In comparison, a context extractor implemented using nearest-neighbours as in [3] takes 57 ms per iteration, which would lead to no actual total runtime gain over evaluating all proposals.

As the runtime of the window classifier grows, the speedup made by of our method becomes more important. Extracting CNN features on the CPU takes 100 ms per window [29]. In this regime, the overhead added by our method becomes negligible, only 3% of running the window classifier. Evaluating all 3200 proposals in an image would require 320 seconds, in contrast to just 36 seconds for evaluating 350 proposals with our method, a $9\times$ speed-up.

7.3. Conclusion

Most object detectors independently evaluate a classifier on all windows in a large set. Instead, we presented an active search strategy that sequentially chooses the next window to evaluate based on all the information gathered before. Our search effectively combines two complementing driving forces: context and window classifier score.

In experiments on SUN2012 and PASCAL VOC10, our method substantially reduces the number of window classifier evaluations. Due to the efficiency of our proposed context extractor based on Random Forests, we add little overhead to the detection pipeline, obtaining significant speedups in actual runtime.

Acknowledgements This work was supported by the ERC Starting Grant VisCul. A. Vezhnevets is also supported by SNSF fellowship PBEZP-2142889.

References

[1] Imagenet large scale visual recognition challenge (ILSVRC). http://www.image-net.org/challenges/LSVRC/2012/index, 2012.

[2] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Trans. on PAMI*, 2012.

[3] B. Alexe, N. Heess, Y. Teh, and V. Ferrari. Searching for objects driven by context. In *NIPS*, 2012.

[4] L. Bazzani, N. de Freitas, H. Larochelle, V. Murino, and J. Ting. Learning attentional policies for tracking and recognition in video with deep networks. In *ICML*, 2011.

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

[6] N. J. Butko and J. R. Movellan. Optimal scanning for faster object detection. In *CVPR*, 2009.

[7] M. Choi, J. Lim, A. Torralba, and A. Willsky. Exploiting hierarchical context on a large database of object categories. In *CVPR*, 2010.

[8] R. Cinbis, J. Verbeek, and C. Schmid. Segmentation driven object detection with fisher vectors. In *ICCV*, 2013.

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

[10] N. Dalal and B. Triggs. Histogram of Oriented Gradients for human detection. In *CVPR*, 2005.

[11] N. Dalal and B. Triggs. Histogram of Oriented Gradients for Human Detection. In *CVPR*, volume 2, pages 886–893, 2005.

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

[13] C. Desai, D. Ramanan, and C. Folkess. Discriminative models for multi-class object layout. In *ICCV*, 2009.

[14] S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert. An empirical study of context in object detection. In *CVPR*, pages 1271–1278, 2009.

[15] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[16] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[17] G. Fanelli, J. Gall, and L. Van Gool. Real time head pose estimation with random regression forests. In *CVPR*, pages 617–624. IEEE, 2011.

[18] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade Object Detection with Deformable Part Models. In *CVPR*, 2010.

[19] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. on PAMI*, 32(9), 2010.

[20] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. In *CVPR*, 2009.

[21] C. Galleguillos, A. Rabinovich, and S. Belongie. Object categorization using co-occurrence, location and appearance. In *CVPR*, 2008.

[22] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[23] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *ICCV*, pages 415–422. IEEE, 2011.

[24] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011.

[25] P. Hall and S. C. Morton. On the estimation of entropy. *AISM*, 45(1):69–88, 1993.

[26] H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In *ICCV*, 2009.

[27] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

[28] G. Heitz and D. Koller. Learning spatial context: Using stuff to find things. In *ECCV*, 2008.

[29] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. `http://caffe.berkeleyvision.org/`, 2013.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[31] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*, 2008.

[32] H. Larochelle and G. E. Hinton. Learning to combine foveal glimpses with a third-order Boltzmann machine. In *NIPS*, 2010.

[33] A. Lehmann, P. V. Gehler, and L. J. Van Gool. Branch&rank: Non-linear object detection. In *BMVC*, volume 2, page 1, 2011.

[34] C. Li, D. Parikh, and T. Chen. Extracting adaptive contextual cues from unlabeled regions. In *ICCV*, pages 511–518. IEEE, 2011.

[35] T. Malisiewicz, A. Gupta, and A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[36] S. Manen, M. Guillaumin, and L. Van Gool. Prime object proposals with randomized prim's algorithm. In *ICCV*, 2013.

[37] V. Mnih, N. Heess, and A. Graves. Recurrent models of visual attention. In *NIPS*, pages 2204–2212, 2014.

[38] A. Montillo and H. Ling. Age regression from faces using random forests. In *Proceedings of the IEEE International Conference on Image Processing*, pages 2465–2468. IEEE, 2009.

[39] F. Moosman, B. Triggs, and F. Jurie. Fast discriminative visual codebook using randomized clustering forests. In *NIPS*, 2006.

[40] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object

detection and semantic segmentation in the wild. In *CVPR*, pages 891–898. IEEE, 2014.

[41] K. Murphy, A. Torralba, and W. T. Freeman. Using the forest to see the trees: A graphical model relating features, objects, and scenes. In *NIPS*, 2003.

[42] S. O'Hara and B. A. Draper. Are you using the right approximate nearest neighbor algorithm? In *wacv*, 2013.

[43] M. Pedersoli, A. Vedaldi, and J. Gonzales. A coarse-to-fine approach for fast deformable object detection. In *CVPR*, 2011.

[44] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *ICCV*, 2007.

[45] M. Saberian and N. Vasconcelos. Boosting algorithms for detector cascade learning. *JMLR*, 15(1):2569–2605, 2014.

[46] R. Sznitman and B. Jedynak. Active testing for face detection and localization. *IEEE Trans. on PAMI*, 2010.

[47] Y. Tang, N. Srivastava, and R. Salakhutdinov. Learning generative models with visual attention. In *NIPS*, pages 1808–1816, 2014.

[48] A. Torralba. Contextual priming for object detection. *IJCV*, 53(2):153–167, 2003.

[49] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013.

[50] K. E. A. Van De Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Trans. on PAMI*, 32(9):1582–1596, 2010.

[51] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009.

[52] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, pages 511–518, 2001.

[53] X. Wang, M. Yang, S. Zhu, and Y. Lin. Regionlets for generic object detection. In *ICCV*, pages 17–24. IEEE, 2013.

[54] T. Wu and S. Zhu. Learning near-optimal cost-sensitive decision policy for object detection. In *ICCV*, pages 753–760, 2013.

[55] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from Abbey to Zoo. In *CVPR*, 2010.

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

[57] M. Zhu, N. Atanasov, G. Pappas, and K. Daniilidis. Active Deformable Part Models Inference. In *ECCV*, 2014.