Attentional Object Detection
Report for CS 294-67 Sequential Decisions

Sergey Karayev

13 May 2011

Abstract

Multi-class object detection in cluttered natural scenes is a long-standing open problem in computer vision. Currently best-performing approaches are computationally expensive. We seek to ameliorate this through replacing exhaustive and unprioritized search with a sequential decision process.

1 Introduction

The task of *object detection* as most commonly formulated entails simultaneous recognition and localization of "things" in a static image of a natural scene. Each object is most commonly assumed to belong to one of a fixed set of classes; its localization is approximated by placing a bounding box around pixels belonging to it. Datasets of such annotations by humans are used for evaluation of detection algorithms [1].

Most object detection systems can be subdivided into three parts: a way to propose regions of the image, a way to evaluate a given region, and a way to post-process the results. For example, the deformable parts model of Felzenszwalb *et al.* proposes image regions with an exhaustive sliding window, evaluates them with a window-sized gradient-based feature and a linear SVM (in addition, it fits parts to the model in a Latent-SVM formulation), and uses detection context to prune some detections in post-processing [2, 3]. Another approach, Multiple Kernels of Vedaldi *et al.*, proposes windows in a cascaded manner and evaluates them with bag-of-words non-linear SVM kernels [4]. Both systems have recently shown state-of-the-art performance on the most commonly used detection challenge dataset [1], and have pointed the direction for much subsequent research.

These approaches are quite slow. In the case of the deformable parts model, detection is on the order of a few seconds; recent work to speed it up with coarse-to-fine search and cascaded parts puts it on the order of a second, at slight hit to the accuracy [5, 3]. Multiple Kernels consider increasingly fewer region proposals with increasingly more powerful SVMs; in the end, more than a minute is spent per image [6]. At these rates, real-time detection is problematic.

We argue that the solution to robust but fast object detectors in the notion of attention—meaning a sequential process of looking for something somewhere (as opposed to looking for everything everywhere). We suggest that to work on detection approaches scalable to many classes and to sequential frames of videos, the field needs to consider a new evaluation metric that encourages maximization of correct detections as early as possible in the detection process, given either a fixed or stochastic deadline.

This paper reviews past work toward this goal and formulates a novel method for maximizing performance of a detector with regard to the new evaluation.

2 Related Work

The literature on object detection is vast. Here we briefly summarize work relevant to our contribution.

An early success in efficient object detection used simple Haar features to build up a *cascade* of classifiers, which then considered image regions in a sliding window regime [7]. Plentiful later work improved the behavior of the cascade while maintaining the basic idea [8]. This detection method is fast, but the simple features and classifiers used have not led to the best performing detectors.

The best performance has recently come from detectors that use gradient-based features to represent either local patches or object-sized windows. If local patches are used, it appears important to include additional feature channels such as shape and color. If object-sized windows are used, finer-scale "parts" in object representation boost performance significantly.

The region proposal is usually done exhaustively over the image space, and doing so in an efficient order has not received much attention in the literature. Using "jump windows" (window hypotheses voted on by local features) as region proposals is one common idea [9, 6]. For local features, a bounded search over the space of all possible windows works well (especially for single-object detection) [10]. The method requires derivation of bounds, which have not yet been developed for the best-performing HOG-based detectors [11, 2], limiting the method's usefulness in state-of-the-art systems.

One recent idea is to use a class-independent measure of "objectness" to reject much of the hypothesis space before running any class-specific detectors [12, 13]. However, these approaches have not been shown to be actually faster than exhaustive evaluation, due to an expensive proposal step [13].

Evaluation of regions involves feature computation, which is a time consuming step. Efficient feature computation for HOG-based detectors has been explored in [14]. Another idea is coarse-to-fine model evaluation. A recent work applies coarse-to-fine evaluation and adds a feedback arrow from post-processing to proposals, obtaining an order of magnitude speedup in the deformable part models framework while losing some accuracy [5].

Other systems that add feedback arrows to the conceptual diagram of Proposals -¿ Classifiers -¿ Post-processing are often inspired by biological vision and sequential decision process ideas [15, 16, 17]. These works are very close to our motivation. Another piece of motivation is Anytime performance, which is a largely unexplored idea for vision systems. A pioneering recent paper picks features with maximum value of information in a Hough-voting framework, and explicitly evaluates itself with regard to time [18].

Multi-class detection has its own line of work, focusing largely on detection time sublinear in the number of classes through sharing features [19, 20, 21]. An interesting reinforcement learning approach in a cascade framework is taken in [22].

Context has a long history in vision. One source of context is the scene or non-detector cues; for the PASCAL VOC, these are quantitatively considered in [23]. Another source is inter-object context, used for detection in a random field setting in [24].

Attention modeling is a long-standing research problem in psychophysics, resulting in a few key concepts: saliency, bottom-up vs. top-down effects, and using saccades as a proxy [25, 26, 27]. The concept of saliency has been used for object classification, and has been explored to no great effect for detection [28].

The two papers closest to our contribution are principled multi-class structured prediction [29] and the detection under bounded resources work [18]. We build atop the first work, which assumes that all classes have been evaluated at all regions. We share the motivation of Anytime performance with the second work, but in a significantly more powerful detection regime. Although we rely on the commonly used deformable parts model detector, our method can use any feature extraction and classification method.

3 Time-sensitive evaluation

Average Precision (AP) has become a standard evaluation for detector performance on challenging datasets. AP is the area under the Precision vs. Recall curve, which is obtained by varying the threshold on the confidence of the detector.

Just as we care about the performance of our system at different thresholds of detection, we

should also care about performance as the system is given different amount of time to output detections. Accordingly, we plot the P-R *surface* instead of the P-R curve, with time as one of the axes. The surface can be integrated along the Recall dimension to yield an AP vs. time curve.

The goal of our scheme is to schedule computation optimally, so that the largest part of the detection performance is recovered early on. In the limit of infinite time, we still want optimal performance—the performance of our system should not degrade as it is given more time. Accordingly, our goal is *Anytime* performance starting at some fixed deadline.

As our task is fundamentally in *multi-class* object detection, we rely on a slightly different evaluation than is commonly used: instead of pooling detections across images in the dataset but not classes, we pool detections across classes, but evaluate per-image (and report the average). This has been done before [29].

4 Problem Formulation

Our goal is to output the best detections at some deadline F, and to continue having the best detections at every time step after that. Specifically, we seek to maximize the average multi-class AP of a detector for all classes in all images of a test dataset.

Given enough time, our program will look everywhere for everything, but given a short deadline, it should first look in the most promising places for the most likely objects. To take advantage of semantic and spatial context, where the program looks at time t should depend on the results gathered by $t-1$.

In each of the I images we consider, we look for K classes of objects, and have detectors c_k.

We represent an image as a collection of overlapping windows at different scales: a multi-scale pyramid of N locations $l_i = (x, y, s)$. We assume that each location ℓ can be the center of at most one bounding box, containing an object belonging to one of K classes.

We can consider three variants of object detection as a sequential decision problem:

1. Looking for everything at a location: Selecting a location i, and running a black box program P there that evaluates all detectors as it sees fit.

2. Looking for something at a location: Selecting a location i and a detector c_k. Picking the sequence of partial classifiers for d is then a black box subproblem.

3. Looking for partial evidence of something at a location: Selecting a location i and a partial classifier c_{kp}.

We will be working at the level of the second problem—we assume that we have K powerful classifiers, and want to pick the shortest path through (location, class) pairs to all correct detections.

4.1 POMDP

We face a finite-horizon Partially Observed Markov Decision Process (POMDP), which we define as the tuple $(S, A, O, T, \Omega, R, \gamma, F)$, where:

- S is a set of discrete states, A is a set of discrete actions, and O is a set of continuous observations.

- $T(s, a, s') = P(s_{t+1} = s | a_t = a, s_t = s)$ is the distribution describing the probability of transitioning from state s to state s' upon taking action a.

- $\Omega(o, s, a) = P(o_{t+1} = o | a_t = a, s_{t+1} = s)$ is the distribution describing the probability of observing o from state s after taking action a.

- $R(s, a)$ is the reward signal received when executing action a in state s.

- F is the deadline to first evaluation.

The problem is naturally episodic, where an image presents an episode. Our goal is to learn the optimal policy π, which is a mapping from histories (trajectories) $h_{1:t} = ((a_1, o_1, r_1)..., (a_t, o_t, r_t))$ to actions a.

4.2 State representations

For all approaches described later, the state $s \in S$ is fully defined by the time T, and the class of the bounding box centered at each location. We augment the K classes with a "background" class, which means that there is no object of interest at a location. Note that each object in the image is assigned to exactly one location; this is in contrast to many detection models employing random fields, such as [24]. The number of states is exponential in N.

An interesting aspect of our task is that we are dealing with a static image, meaning that nothing changes the underlying state. Therefore, $T(s, a, s') = \delta_{s,s'}$ where δ is the Kronecker delta function. In the usual POMDP setting, the physical state is non-stationary, and a repeated action a may lead to different observations o. In our case, it does not make sense to run the same detection action at a location more than once, and so there is a finite number of actions $\mathcal{A}$ (not just action types) for us to take in an episode. This means that F can be set to a finite time F_{max}, such that all the possible actions can be taken before termination. This corresponds to an exhaustive search.

Normally in POMDPs, the history h is unbounded (except by the horizon) and quickly becomes intractably large. For this reason, POMDP policies are either *memoryless*, meaning that the next action depends only on the last observation, or depend on some internal state representation that is a compression of the trajectory. A common internal representation is the belief state, $P(s|h_{1:t})$, which can compactly represent everything the program needs to know to make Bayesian-optimal decisions. If the POMDP is known, then we can convert it with this internal representation to an MDP over belief states, and solve it optimally using MDP techniques. However, this optimal solution approach is hard to scale to large POMDP problems [30, 31]. We again note that in our problem, the size of $h_{1:t}$ is always bounded by $\mathcal{A}$.

We model the internal POMDP state as a graphical model, with nodes for the N locations in the image pyramid, and $N*K$ possible observations. A node $y_i \in Y$ is an integer $0 \dots K$, according to the class whose bounding box we believe is centered at location i (background is 0). A node $z_{ik} \in Z$ is the real-valued output of a detector of class k at location i, signifying classifier confidence. Two nodes y_i and z_{ik} are connected with weight w_k, which in a sense represents our confidence in the classifier c_k and also allows us to account for multi-class bias.

The connections between two nodes y_i and y_j are defined by the spatial context feature d_{ij} (represented in Figure 1) and the weights w_{y_i,y_j}. These weights encode valid geometric configurations of object classes y_i and y_j.

In this way, we can think of the internal state as a sort of CRF, where y_i are conditioned on the image X through z_{ik}. Crucially, the extra nodes z_{ik} can be unobserved. This internal state therefore both encodes the full trajectory h (albeit in an orderless way), and provides a model-based probability distribution over s. We note that we do not encode rewards received in the internal representation, and discuss this further in section 4.6.

This way of modeling the underlying state of the image is similar to the one used in [29]. In fact, we rely on their method of greedy forward search to do inference in the CRF and learn the parameters; inference of the physical state from our belief state is described in section 4.4.

4.3 Actions and Observations

There are two distinct ways to formulate the action space.

The first defines actions by a choice of location i and detector c_k, making for NK possibilities. This action space allows any possible action to be taken at any time, and with the right policy could capture structure of the problem, such as non-maximum suppression and spatial cueing. The space is considerably large, however, and policies have to be highly complex to be effective.

The second restricts the action space with our domain knowledge of the problem. For instance, we know there is no point in running the same action twice. Therefore, we can offer just one

Figure 1: Figure from [29].

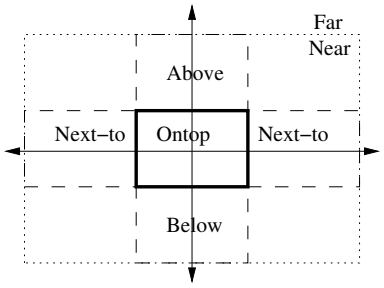


Fig. 3 A visualization of our spatial histogram feature d_{ij}. We consider the location of the center of window j with respect to a coordinate frame defined by window i, denoted by the thickly outlined box. The dashed and dotted rectangles represent regions over which the center of window j are binned. The relative location of j must either be `far` or `near`. For near windows, we consider `above`, `ontop`, `below`, and symmetric `next-to` bins as shown. To allow our model to reproduce the behavior of baseline modules that perform NMS with a criteria of 50% relative overlap, we also include a binary overlap feature. This makes d_{ij} a 7 dimensional sparse binary vector.

action to the agent: "pick a random unexamined (location,class) pair." Of course, the only possible policy then is a random exploration of the space. To structure the exploration, we can offer action such as "pick a random class for an unexamined location with the highest entropy of z_i" (entropy is discussed in section 4.6). We can separate picking the class from picking the location, to for example, obtain a policy of picking an unexplored location of maximum saliency, as determined by some non-state featurization of the image, and picking the class most likely to be in the image, as determined by some other classification output of the image. Some example policies following this approach are described in section 4.5.

In either case, upon executing an action, the observation o we get back is the confidence of the classifier. The observation updates the corresponding node z_{ik}. We can propagate this update through the graphical model with an iteration of belief propagation, or alternatively, we can find the MAP assignment to Y, given this new information, by running a greedy forward search (details in section 4.4). We can do this at every action, every fixed number of actions, or upon taking a special action.

4.4 Outputting Detections, Receiving Rewards, and the Deadline

Y in our belief state represents the detections, and is conditioned on Z.

Recall that we cannot have detections of two different classes at the same location, and that the PASCAL evaluation will penalize all but one detections of the same class at overlapping locations. For this reason, most detection systems that "assign content to locations" like this run a post-processing non-maximum suppression step [2]. In a multi-class setting, it is better to resolve such ambiguities with a principled system to model both inter- and intra-class, and short- and long-range interactions.

Our belief state is modeled after such a system, and we simply run a greedy forward search to maximize the energy of our graphical model to pick our final detections [29]. The energy of the model is given as

$$S(Y,Z) = \sum_{i,j} w^T_{y_i,y_j} d_{ij} + \sum_{i,k} w_k z_{ik} \tag{1}$$

In brief, the greedy search starts out with an empty set of detections, and proceeds to pick those y_i that maximize $\Delta S(Y, Z)$. This method has been shown to be optimal for nearly all images ($\approx 98\%$) in the PASCAL set (and compared against Loopy BP and Tree-ReWeighted BP) [29].

The theoretical explanation for this comes from the work on *submodular functions*. The rough idea is that to maximize functions where adding an element to a large set has less effect than adding an element to a small set, greedy algorithms often have theoretical guarantees of near-optimal performance. The more pairwise connections are non-positive in the CRF, the more submodular it is. Since most connections in the image CRF are inhibitory (the CRF mostly performs non-maximum suppression, after all), the theory may apply here.

After every action starting at deadline F, $R(s, a)$ is assigned according to the AP of these detections. The program is not terminated at F, but runs to F_{max}, which is determined by the time it would take to run every possible detector action once. Therefore, the reward structure is such that there is a large first reward at F, and following rewards at every time step. It is not clear whether a cost-of-living negative reward should be assigned at each time step; in our toy experiments reported in section 5, this did not seem to make a difference. We intend to try both ways on the actual problem.

We express F in units of expected time per action, such that taking action a with time cost τ advances T by τ. We determine τ by averaging the wall clock time of executing a; this is done for all a prior to running the POMDP learner.

4.5 Some Example Policies

We can briefly describe how different detection strategies fit into our framework. Instead of committing to one strategy, we can define our action space to allow multiple strategies. We can then find a policy that optimally combines the strategies, perhaps as a function of T and F, to maximize the reward under our definitions.

Sliding window, Random, and Saliency-driven As described, earlier in the text, our definition of the action space can drive any of these policies. We first compute a simple, fast self-similarity saliency map for the image (for example, as in [12]). The internal state is featurized by the location of the current unexamined (picked less than K times) max in the saliency map and the class k of the last action. The policy could then be to simply sample locations in order of saliency, running all detectors at a location before proceeding to the next.

Class prior or posterior-driven On the training set, we compute K canonical object likelihood maps. We featurize the belief state with the current unexamined (picked less than 1 time) max of each likelihood map. Our policy is to run the detector for the class k at the location i with the largest likelihood among these maps.

Of course, with our random field model of the environment, we can compute class posteriors for all locations. As the policy proceeds, it could start taking belief propagation actions to update the class posteriors and improve the quality of the actions.

4.6 Unobserved rewards and Augmented MDPs

Our problem formulation briefly noted an important detail: at test time, the program does not receive rewards. Therefore, we should not learn a policy that depends on the rewards during execution. For this reason, there is nothing in our internal state representation that encodes rewards received: the only information is what we currently believe each node y_i to be, and what observations z_{ik} we have made.

We can view the observations $z_i = \{z_{i1} : z_{iK}\}$ as an unnormalized distribution over y_i. Accordingly, we can derive the entropy term $H(z_i)$, which represents our uncertainty in the value of y_i. The idea of adding uncertainty variables to the state space of a decision process is known as Augmented MDPs [32, 33]. Augmented MDPs have the advantage of increased tractability over

corresponding POMDPs, while still retaining the modeling power of allowing uncertainty. For example, policies can account for the uncertainty in the state through the entropy variables, and direct actions accordingly. This model has been successfully applied to self-localizing robots [33] and to the problem of active sensing in soccer-playing Aibos [32].

If we model our problem as an Augmented MDP, we will lack one thing: knowledge of the reward function. Accordingly, we can apply the well-developed array of efficient techniques for model-free reinforcement learning [34]. Additionally, we can shape the reward function with *a priori* domain knowledge, by, for example, encouraging minimal entropy.

4.7 Solving the RL problem

We consider several ways to solve the posed problem.

First, we can attempt to solve the POMDP as a belief-state MDP, computing the state value function for it. This approach is hard to scale to a problem as large as ours, but efficient methods have been developed for medium-sized problems [35].

Second, we can solve directly for the $Q(s, a)$ function using reinforcement learning techniques. For general POMDPs, this seldom works, as the model is likely non-Markovian in s. We consider this approach seriously, because for reasons outlined in section 4.6, direct model-free learning could work well for us. A compromise between this and the first approach was proposed in the Q_{MDP} framework [36], which computes Q as if the environment was Markovian at every future time step past the current one, and so defines $Q(b, a) = \sum_x b(x)Q_{MDP}(x, a)$. In our toy experiments, we use the technique of Least-squares Policy Iteration [37], an extension of Least-Squares Temporal Difference learning [38] to model-free learning. This is an efficient linear method allowing a large number of parameters, and recovering the global maximum of the value function.

Third, we can remember that the ultimate goal is to find the best policy, and search for it directly, forgetting about standard indirect approaches of policy evaluation. Direct policy search is motivated by indirect approaches' several problems when applied to POMDPs:

- Q may be unstable due to non-Markovian s;
- function approximation may make it hard for reinforcement learning to converge to a stable policy;
- stochastic policies may be better suited than deterministic ones to POMDPs.

Direct policy search does not suffer from these problems, but of course has the additional problem of how to parametrize and learn the policy effectively [30, 39]. An effective way to represent a policy is as a Finite State Machine [40], which can lead to efficient search solutions. The PEGASUS framework samples trajectories most efficiently, allowing fast search through a huge space [31]. A series of reports on the GPOMDP algorithm extend the ideas began in Williams's REINFORCE algorithm [41] for policy gradient ascent and review much of the relevant literature [42, 43]. Lastly, the problem of searching for the best policy can be treated at its most basic, as a coordinate ascent problem [44].

In our opinion, direct policy search is most suitable for general POMDP problems, as it minimizes the assumptions made about the belief state, and at least directly ensures that the policy is improved at each iteration of learning (value function learning does not guarantee this). However, the special structure of our POMDP, and the possibility of a reasonable representation as an Augmented MDP, leads us to at least try a reinforcement learning approach to our task. We describe initial experiments to this end in **??**.

5 Experiments

Toward the goal of evaluation on a current challenge dataset, we spent time to setup the baselines, before proceeding with learning approaches. Here we report the baseline, and describe the next step in that direction.

To explore appropriate learning methods for the problem, we developed a toy problem that represents the salient details of the full image detection tasks. Here we report our initial experiments, using a model-free batch reinforcement learning method.

5.1 PASCAL Baseline

Among published detection frameworks, the Cascaded DPM detector [3] has probably the highest performance to speed ratio. To use it as a baseline, we had to first evaluate it in the AP vs. Time regime as described in section 3.

Toward this end, we write out detections as each stage of the cascade is passed, at each location, for each class. This of course results in far too many detection hypotheses. To reduce them to a reasonable set that would lead to the best performance according to the PASCAL criterion, we employ the standard technique of Non-Maximum Suppression (NMS), where two overlapping detections are reduced to just one: the detection with the higher score. Specifically: at discrete time intervals, we pool all surviving detection hypotheses from previous intervals, add in new ones from the current interval, and run NMS. The resulting set of detections is evaluated and measured by the Average Precision.

Figure 2 shows a typical AP vs. T plot for a single image, and the same data averaged across all images in the test dataset. Note the slope of rise to the final AP score; our goal is to make this steep leading up to some deadline, and either plateau (if final AP is reached) or increase to final AP after that. We can formulate the desired performance as a scalar measure: the discounted AP starting at deadline. We note that to construct these evaluations, we had to pick an ordering over classes. To remove the effect of a particular ordering being better suited to the dataset, we randomized the ordering for each image. Figure 3 shows that this method of evaluation results in final detections that achieve the same level of performance as if the cascade had run un-interrupted to completion.

The baseline represents the output of a particular policy that samples locations and classes in a fixed order. We can find the optimal order for the validation dataset by exhaustive search; that would produce an optimal baseline for the policy class. This is current work, along with a simple novel policy of sampling in order from a saliency map computed on the image, as described in section 4.5.

5.2 Toy problem

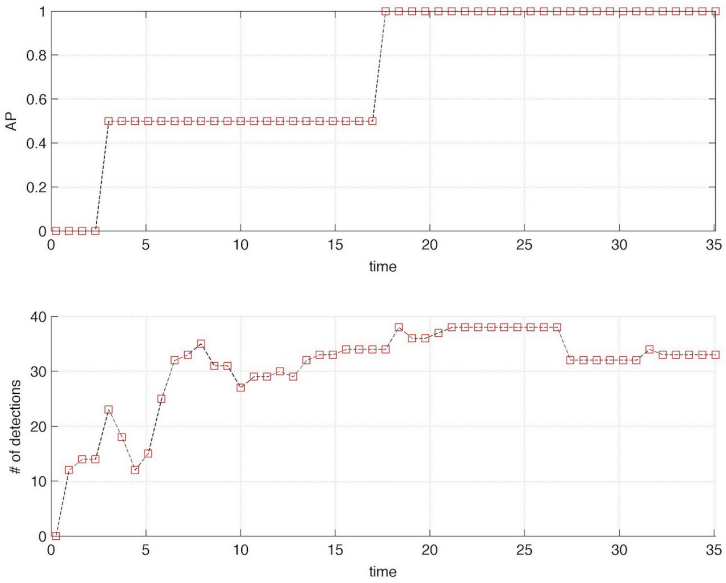
To experiment with different reinforcement learning problem formulations, we created a toy problem. In it, we consider an image with N locations, with some number of locations containing objects of some class $\{1 : K\}$. For initial experiments, we assume that we have perfect classifiers, so if a detect action is run, we become fully certain about the presence of the examined class at the examined location. This is a simplifying assumption about the detectors, that moves the problem away from the domain of POMDPs. Still, as described in section 4.6, an Augmented MDP representation of the original problem is roughly the same as this formulation, except with an unknown, stochastic reward function.

To be able to generalize to large state and action spaces, and to the Augmented MDP formulation, we use the model-free Least Squares Policy Iteration approach [37]. The approach consists of two parts.

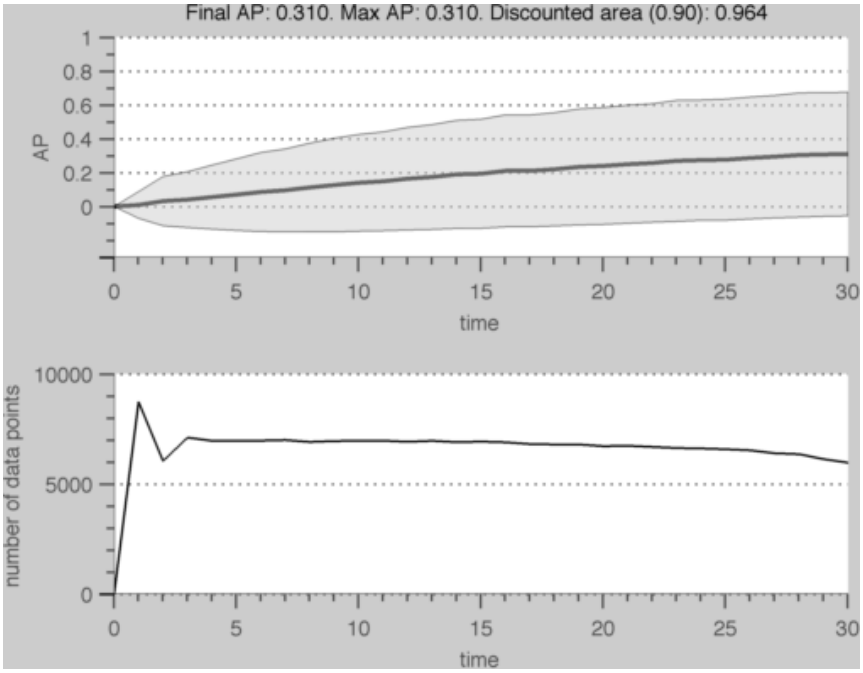
First, linear function approximation is used for the Q function, such that:

$$Q^{\Pi}(s,a) \approx \hat{Q}^{\Pi}(s,a) = \sum_{i=1}^{k} \phi_i(s,a) w_i = \phi(s,a)^T w \tag{2}$$

LSPI learns the weights w, which fully determine the policy π using batch temporal difference learning. First, the agent interacts with the environment following some (at first random) policy to generate samples $< s, a, r, s' >$, with the obvious meanings. The tuples are used to update a $k \times k$



(a) Example of performance on a single image.



(b) Average performance across the dataset. Gray area is bounded by one standard deviation away from the mean.

Figure 2: Plot of baseline performance of the Cascaded DPM Detector [3], in the AP vs. Time evaluation regime.

matrix A and $k \times 1$ vector b:

$$A \leftarrow A + \phi(s,a)(\phi(s,a)^T - \gamma\phi(s',\pi(s'))^T) \tag{3}$$

$$b \leftarrow b + \phi(s,a)r \tag{4}$$

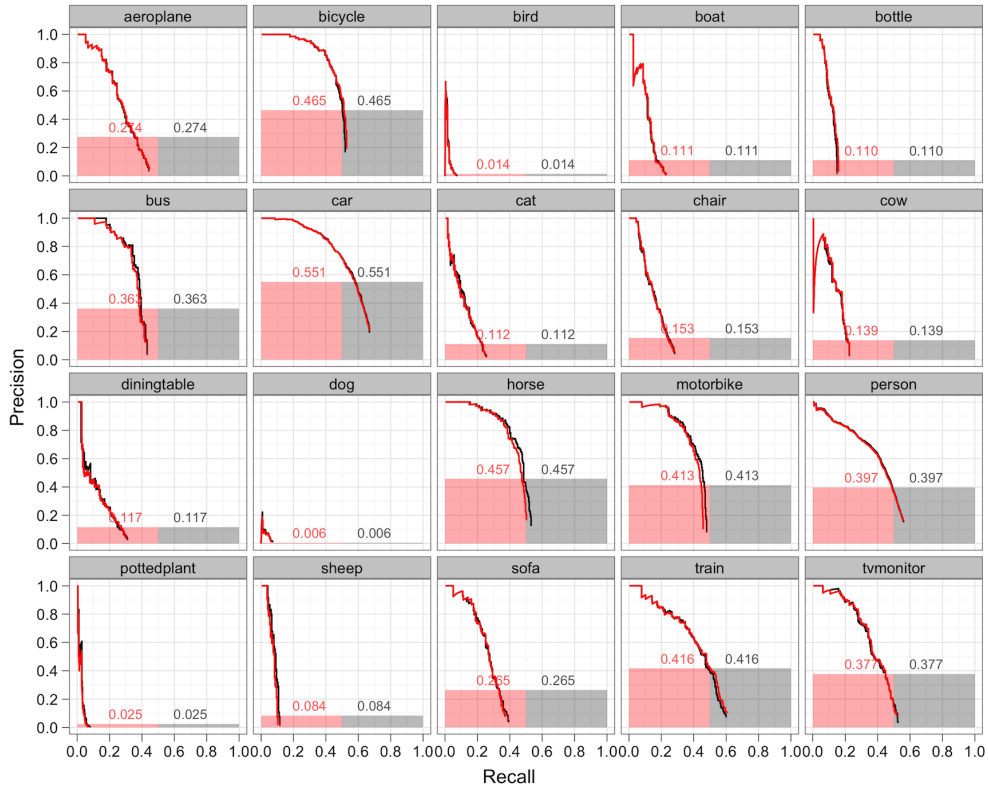


Figure 3: Our method of computing Anytime detections results in final detections that perform as well as the final output of the cascade.

The weights vector w can be extracted by solving $A^{-1}b$. The learning procedure (1) generates S samples by following the current policy, beginning with random; (2) updates the current policy by first setting A, b to 0, updating them with the samples as above, an deriving the new policy; (3) repeating step (2) until the error between two sequential policy vectors is $< \delta$; (4) generating new samples by following an ϵ-greedy policy. This procedure was followed for an active learning problem [32].

LSPI is an off-policy algorithm that is able to use the same set of samples to improve the policy at each iteration of step (2). It converges faster and with less samples than TD(λ) or SARSA(λ), due to its least squares formulation.

As mentioned in section 4.3, we face two distinct ways to define the action space. For initial experiments, we chose the fully general action space, such that $|A| = NK$, meaning that any detector can be run at any location.

The k features are designed manually. For each action, we learn a separate Q-function (given a, $\phi(s, a)$ is only non-zero at locations corresponding to a). The feature vector is simple:

$$\phi(s, a) =< E_i, 1 > \tag{5}$$

E_i is a binary feature that is 1 if the (location,class) pair given by a has been examined, according to our internal state s, and -1 otherwise. 1 is a constant allowing the policy to select actions. w therefore has $2NK$ parameters.

In Figure 4, we show the results of our initial experiments with learning a policy. We are able to learn policies for deterministically or probabilistically-placed object classes. The learning appears to be heavily influenced by having enough samples from a random policy; we keep ϵ high such that even stable policies generate enough random samples. In general, a lot of parameter tuning is

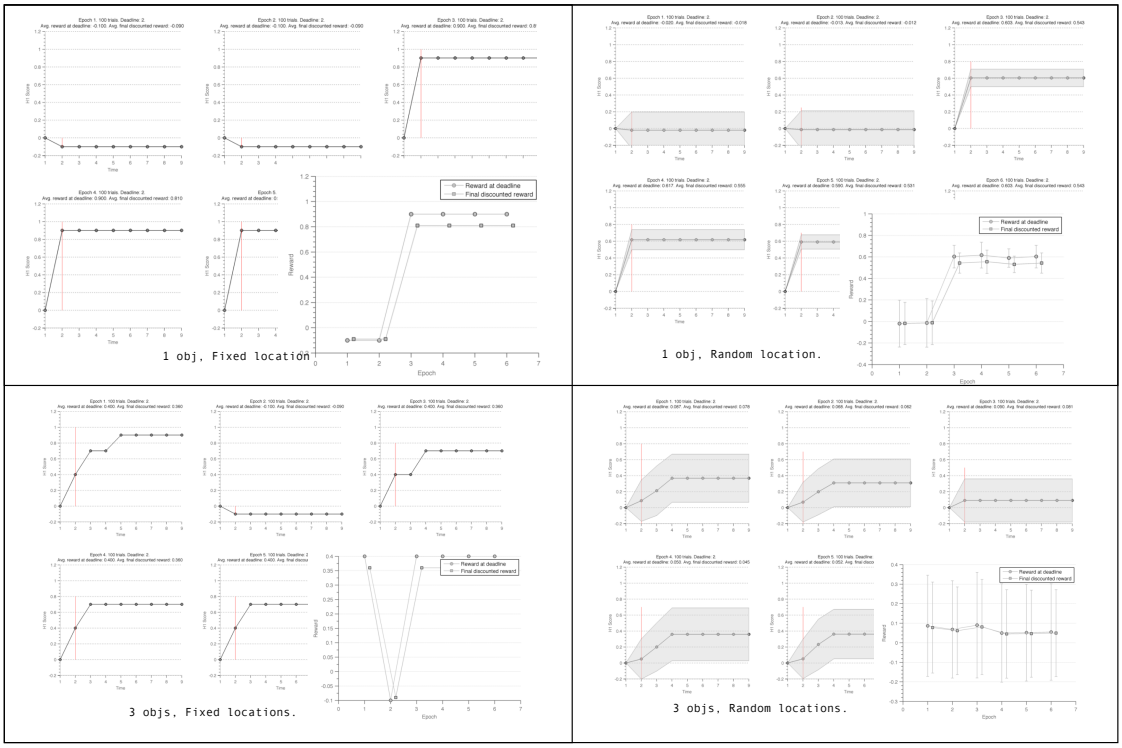


Figure 4: Some initial experiments using LSPI to solve the toy problem described here. The approach easily learns to find one object at a fixed location, and performs well with random single-object placement. It does not recover multiple objects at fixed locations successfully, and sometimes struggles when they are randomly placed. These are initial experiments, in ongoing work.

needed for successful learning, and this is something we are still actively exploring. We have not yet tried to learn policies corresponding to more complicated object placement rules.

We attribute some of the difficulty of learning a successful policy partly to the highly complex action space. In current experiments, we are considering actions of a more restricted class, such as "pick a random unexamined (location,class) pair."

References

[1] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL VOC Challenge 2010 Results." http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html, 2010. 1

[2] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models.," *PAMI*, vol. 32, pp. 1627–45, Sept. 2010. 1, 2, 5

[3] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *CVPR*, pp. 2241–2248, IEEE, June 2010. 1, 8, 9

[4] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," *ICCV*, pp. 606–613, Sept. 2009. 1

[5] M. Pedersoli, A. Vedaldi, and J. Gonzalez, "A Coarse-to-fine approach for fast deformable object detection," in *CVPR*, 2011. 1, 2

[6] S. Vijayanarasimhan and K. Grauman, "Large-Scale Live Active Learning: Training Object Detectors with Crawled Data and Crowds," in *CVPR*, 2011. 1, 2

[7] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR*, 2001. 2

[8] L. Bourdev and J. Brandt, "Robust Object Detection via Soft Cascade," in *CVPR*, pp. 236–243, Ieee, 2005. 2

[9] O. Chum and A. Zisserman, "An Exemplar Model for Learning Object Classes," in *CVPR*, pp. 1–8, Ieee, June 2007. 2

[10] C. Lampert and M. Blaschko, "A Multiple Kernel Learning Approach to Joint Multi-class Object Detection," *Lecture Notes in Computer Science: Pattern Recognition*, vol. 5096, 2008. 2

[11] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *CVPR*, pp. 886–893, Ieee, 2005. 2

[12] B. Alexe, T. Deselaers, and V. Ferrari, "What is an object?," in *CVPR*, IEEE, 2010. 2, 6

[13] I. Endres and D. Hoiem, "Category Independent Object Proposals," in *ECCV*, pp. 575–588, 2010. 2

[14] P. Dollár, S. Belongie, and P. Perona, "The fastest pedestrian detector in the west," in *BMVC*, Citeseer, 2010. 2

[15] N. Butko and J. Movellan, "Optimal scanning for faster object detection," *CVPR*, pp. 2751–2758, June 2009. 2

[16] J. Vogel and N. de Freitas, "Target-directed attention: Sequential decision-making for gaze planning," *ICRA*, pp. 2372–2379, May 2008. 2

[17] L. Paletta, G. Fritz, and C. Seifert, "Q-learning of sequential attention for visual object recognition from informative local descriptors," in *ICML*, (New York, New York, USA), pp. 649–656, ACM Press, 2005. 2

[18] S. Vijayanarasimhan and A. Kapoor, "Visual Recognition and Detection Under Bounded Computational Resources," in *CVPR*, pp. 1006–1013, 2010. 2

[19] A. Torralba, K. P. Murphy, and W. T. Freeman, "Sharing visual features for multiclass and multiview object detection.," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, pp. 854–69, May 2007. 2

[20] X. Fan, "Efficient Multiclass Object Detection by a Hierarchy of Classifiers," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pp. 716–723, Ieee, 2005. 2

[21] N. Razavi, J. Gall, and L. V. Gool, "Scalable Multi-class Object Detection," in *CVPR*, 2011. 2

[22] R. Isukapalli and A. Elgammal, "Learning Policies for Efficiently Identifying Objects of Many Classes," *18th International Conference on Pattern Recognition (ICPR'06)*, no. 1, pp. 356–361, 2006. 2

[23] S. Divvala, D. Hoiem, J. Hays, A. Efros, and M. Hebert, "An empirical study of context in object detection," in *CVPR*, pp. 1271–1278, Ieee, June 2009. 2

[24] A. Torralba, K. P. Murphy, and W. T. Freeman, "Contextual Models for Object Detection Using Boosted Random Fields," *MIT Computer Science and Artificial Intelligence Laboratory Technical Report*, 2004. 2, 4

[25] L. Itti and C. Koch, "Computational modelling of visual attention.," *Nature reviews. Neuroscience*, vol. 2, pp. 194–203, Mar. 2001. 2

[26] S. S. Chikkerur, T. Serre, C. Tan, and T. Poggio, "What and where: A Bayesian inference theory of attention," *Vision Research*, May 2010. 2

[27] T. Judd, K. Ehinger, F. Durand, and A. Torralba, "Learning to predict where humans look," *2009 IEEE 12th International Conference on Computer Vision*, pp. 2106–2113, Sept. 2009. 2

[28] C. Kanan and G. Cottrell, "Robust Classification of Objects , Faces , and Flowers Using Natural Image Statistics," in *CVPR*, 2010. 2

[29] C. Desai, D. Ramanan, and C. Fowlkes, "Discriminative models for multi-class object layout," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 229–236, Ieee, Sept. 2009. 2, 3, 4, 5, 6

[30] K. P. Murphy, "A Survey of POMDP Solution Techniques," Tech. Rep. September, 2000. 4, 7

[31] A. Y. Ng and M. Jordan, "PEGASUS: A policy search method for large MDPs and POMDPs," *UAI*, 2000. 4, 7

[32] C. Kwok and D. Fox, "Reinforcement Learning for Sensing Strategies," in *IROS*, 2004. 6, 7, 10

[33] N. Roy and S. Thrun, "Coastal Navigation with Mobile Robots," in *NIPS*, no. figure 1, 1999. 6, 7

[34] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998. 7

[35] J. Pineau, "Anytime Point-Based Approximations for Large POMDPs," *Machine Learning*, vol. 27, pp. 335–380, 2006. 7

[36] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," tech. rep., 1995. 7

[37] M. G. Lagoudakis, "Least-Squares Policy Iteration," *Journal of Machine Learning Research*, vol. 49, pp. 161–1149, Jan. 2003. 7, 8

[38] J. A. Boyan, "Technical Update: Least-Squares Temporal Difference Learning," *Machine Learning*, vol. 49, pp. 233–246, 2002. 7

[39] M. Hauskrecht, "Value-Function Approximations for Partially Observable Markov Decision Processes," *Journal of Artificial Intelligence Research*, vol. 13, pp. 33–94, 2000. 7

[40] E. A. Hansen, "Solving POMDPs by Searching in Policy Space," in *UAI*, 1998. 7

[41] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, May 1992. 7

[42] J. Baxter and P. Bartlett, "Direct Gradient-Based Reinforcement Learning: II . Gradient Ascent Algorithms and Experiments," tech. rep., 1999. 7

[43] J. Baxter and P. L. Bartlett, "Direct Gradient-Based Reinforcement Learning: I . Gradient Estimation Algorithms," tech. rep., 1999. 7

[44] N. Kohl and P. Stone, "Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion," in *ICRA*, no. May, pp. 2619–2624, 2004. 7