Timely Object Detection

Anonymous CVPR submission

Paper ID 2283

Abstract

In a large, multi-class detection system, the problem of timeliness of results is crucial. We are motivated by situations where running all detectors would take an unacceptably long time, and the best answer must be given by some deadline. Our system for multi-class detection aims to give the best possible results at any single point after a start time; it is terminated at a deadline time. Toward this goal, we formulate a dynamic policy to decide which detector to deploy next. The policy is guided by an estimate of the contents of the image, and is sensitive to the time deadline. We evaluate ways of parametrizing the policy toward best performance in the novel AP vs. Time evaluation on the PASCAL VOC dataset.

1. Introduction

In recent years, the computer vision field has converged on both a general method for and evaluation of object detection. Most current state-of-the-art object detection systems consist of three tasks: proposing regions of the image, evaluating a given region for presence of the given class, and post-processing the results. In the evaluation ground truth, 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 human annotations are used for evaluation of detection algorithms [6].

With few exceptions, current detection systems are not inherently multi-class. Instead, separate detectors are trained per class, and the evaluation of multi-class performance, if at all given, consists of averaging the per-class metrics. Some notable papers do make multi-class detection a priority, and accordingly evaluate in multi-class settings, where false positives can be generated by matching to ground truth of the wrong class.

Although some detection systems make efficiency a goal, this is not always true—and it should not always be in a research area that is still young. That said, for some applications, performance really is time-sensitive. In robotics, a small finite amount of processing power per unit time is all that is available for robust object detection if the robot is to usefully interact with humans. In large-scale detection system deployments, such as for image search, results need to be obtained quickly per image as the number of images to process is large and constantly growing. When processing large photo collection on end-user machines for immediate consumer navigation, the same holds true.

In all these cases, it may be argued that an acceptable answer at a reasonable time is more valuable than the best answer given too late. Furthermore, the value of the answer depends largely on the target application.

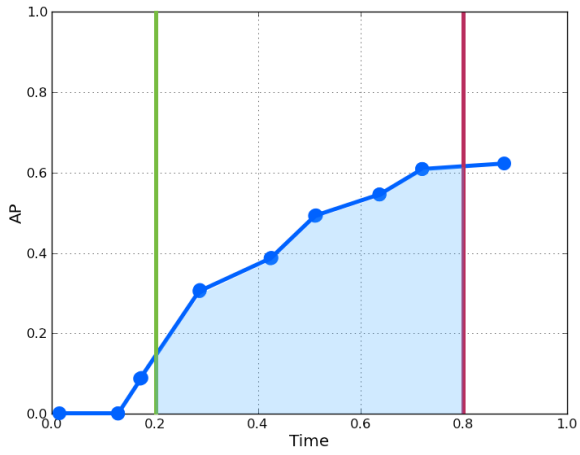


Figure 1: Our objective is to obtain Anytime performance within the bounds of the curve. That is, the policy should give the best possible answer at any given time from start time to deadline.

A hypothetical advertising system deployment presents a great case study. If a detection system gets paid a set amount per each correct detection of objects of class A, and a smaller amount per each correct detection of class B, but the class B detector is simulatenously faster and more accurate than the class A detector, and the backlog of images to be processed is growing in size, the answer to the most rational detection strategy is non-trivial, and depends on all

these variables.

1.1. Proposed Evaluation Metric

We argue that the key to tackling problems of this type is to focus on the evaluation metric of performance vs. time, and answer the question: What is the best performance we can get on a budget?

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.

Figure 1 shows the evaluation strategy and variables for our approach: AP vs. Time. From the green vertical line, signifying the start time, to the red vertical line signifying the deadline, we want to obtain the best possible answer if stopped at any given time. This corresponds to the general notion of Anytime algorithms, and is motivated by desired flexibility in the system.

In the limit of infinite time, we expect optimal performance—the performance of our system should not degrade as it is given more time.

A single-number metric that corresponds to the stated objective is simply the area under the curve between the start and deadline bounds. Just like the metric of Average Precision (area under the Precision vs. Recall) curve was motivated by the inadequacy of any single Precision-Recall operating point to describe the performance of a robust system, our proposed metric is motivated by the inadequacy of any single Performance vs. Time operating point.

In considering "time" performance of detectors, it is always hard to be precise about the meaning. Between the possibilities of using wall clock, "flop" counts, or theoretical runtime analysis, there is no good single answer, as all have benefits and shortcomings. Purely theoretical evaluation can hide performance inefficiencies due to computer architecture; purely empirical depends on the low-level details of the implementation and the specifications of the hardware. Wall clock is our choice, because our multi-class detection system treats underlying detectors as a black box, and so a more involved analysis is not in order.

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 [4].

2. Related Work

The literature on object detection is vast. Here we briefly summarize detection work that contextualizes 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 [22]. 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 [2, 20]. 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 [3, 8], limiting the method's usefulness in state-of-the-art systems.

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 [16].

Other systems that add "feedback arrows" from region evaluation to region proposals are often inspired by biological vision and sequential decision process ideas [1, 23, 15].

Anytime performance in vision systems is a surprisingly little-explored idea. A pioneering recent paper picks features with maximum value of information in a Hough-voting framework, and explicitly evaluates itself with regard to time [21]. Another detection paper motivated by time constraints concerns real-time person tracking [11].

Multi-class detection has its own line of work, focusing largely on detection time sublinear in the number of classes through sharing features [19, 7, 17].

A recent post-processing extension to detection systems uses structured prediction to incorporate multi-class context as a principled replacement for the common step of non-maximum suppression [4]. This work aims to tie up a couple of loose threads of object detection: 1) joint multi-class detection, with the mutual exclusion and contextual effects it entails; 2) the hack of non-maximum suppression of single-class detections.

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 [5]. Another source is inter-object context, used for detection in a random field setting in [18].

The two papers closest to our contribution are principled multi-class structured prediction [4] and the detection under bounded resources work [21]. 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 system can use any detection method.

3. Multi-class detection policy

We present a system that is deploys multiple detectors to find all object instances in the image. The system receives a time deadline, at which point the performance of all detections to this point is evaluated.

The performance of the system is evaluated in the multi-class setting, meaning that detected boxes of a given class are matched to ground truth boxes of all classes, with the corresponding additional capacity for false positives. The underlying performance metric is the standard Average Precision (area under the Precision vs. Recall curve) that is commonly used in the community.

Because the system must deliver results on a time budget, the policy must dynamically decide which of the detectors in its arsenal to deploy next. To leverage the signal present in inter-object context, the decision of which detector to run next should be based on the observed results of detector actions that have already been run. Of course, the observed results of detector actions do not give full confidence to the presence of a class in the image, necessitating an intermediate step from detection statistics to class presence in our system.

First, we look at how to construct such a system with only one detector per class. We explain the general partially-observable decision process that we face and our choices for the different modular components of it. We present evaluations comparing different ways of obtaining probabilities of object presence, and of evaluating the value of running actions.

Next, we generalize the system to be able to handle multiple detector types per class, and show results on this task.

4. Policy with a single detector per class

The structure of the problem may be formalized as follows. There are $|A|$ actions a. For now, we assume that each action consists in running a detector, and so generates a (possibly empty) list of detections when taken. As the same action will always generate the same list of detections, there is no reason to ever take the same action more than once.

Furthermore, we assume that $|A| = K$, the number of target classes in the dataset, so that there is exactly one detector that we can run per class. The operation of the system, as shown in Figure 2, consists in

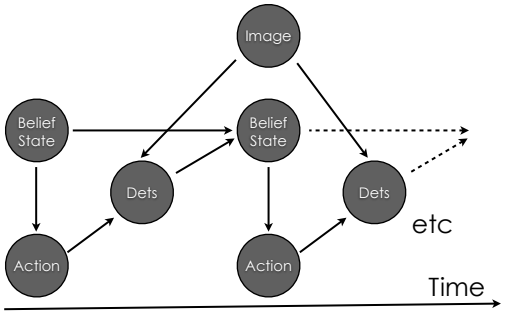
1. selecting which action to take, based on the current estimate of the image contents

2. updating the estimate based on the observed detections upon taking the action.

The system proceeds as such until deadline time, when its operation is terminated, the current detections are evaluated in the Precision vs. Recall regime, and the resulting AP score is collected.

The objective of the system, as motivated in Section 1.1 is to provide the best possible AP score not only at deadline time, but at every time between the start time and the deadline. We first cover the second modular part of the system: updating the latent belief representation having taken an action.

Figure 2: Summary of our approach to the problem. Our model consists of two modular parts: predicting the value of taking an action, and updating the latent belief representation having taken an action.



4.1. Belief state model

The belief state maintains inference of the presence of all the classes in the image, as well as the actions taken, the current time into the episode, and the estimated level of AP gained.

The purpose of the class prior model is to be able to model co-occurrences between objects. Depending on the current belief state, which is a set of classifier results for detectors that have already been run, we choose the next action, i.e. a detector to run. We assume that there is some significance in the context of images: Moreover, we try to learn and exploit co-occurences in the image set. What we basically would want to do is to examine the dependencies of all the possible object classes and with that deduce from

our current belief state, which object classes are most possible to observe next and hence which choice of detector yields the highest improve to our value function. This is best modeled with a CRF for which there are well known algorithm for exact inference. As this would consume a fairly long time at each step, we want to utilize and compare fast approximations to the dependencies.

We use statistics of our training set to formulate initial joint probabilities over classes by empirical counts. These are updated each round using the result of the action run. Now we can express the likelihood for a class occurence C_1 in an image given observations $C_2 \dots C_k$ as:

$$P(C_1|C_2 \dots C_k) = \frac{P(C_1, C_2 \dots C_k)}{P(C_2 \dots C_k)}$$

This model will not satisfy our requirements too good, since we want to get statements for the belief state given any preceding observations. If during test time we observe a new constellation of objects, this will have never been covered in the training set and therefore the probability for all following detections is going to be zero. To prevent these effects caused by our empirical approach, we need to do some kind of smoothing. The complete and correct way to do this would be to incorporate all 2^{k-1} conditionals of C_1 given any subset of the $k-1$ observed variables. A faster approximation of the conditional is to write it out as follows:

$$P_{back}(C_1|C_2 \dots C_k) = \frac{P(C_1, C_2 \dots C_k)}{P(C_2 \dots C_k)} +$$

$$\lambda_1 \cdot \left(\sum_{i=2}^{k} P(C_1|C_i) \right) + \lambda_2 \cdot P(C_1)$$

This model implicitly assume independence of C_i and C_1 given any C_j, s.t. $i \neq j$ in the smoothing part. In the following we will refer to this model as **backoff** and confirm its suitability. The model *fixed policy* ignores any conditionals and just chooses the next action according to the priors of the single class:

$$P_{fix}(P_1|P_2 \dots P_k)$$

With a set of action $\mathcal{A}$, iterating over the belief states will work as follows:

1. We start with belief state $\mathcal{B}_0 = \emptyset$.
2. Choose an action $\alpha_i \in \mathcal{A}_i$ that maximizes the value function for the current belief state.
3. Execute α_i and receive detections β_i.
4. Update: $\mathcal{B}_{i+1} = \mathcal{B}_i \cup \beta_i$, $\mathcal{A}_{i+1} = \mathcal{A}_i \backslash \alpha_i$
5. If $\mathcal{A}_{i+1} \neq \emptyset$ and there is time left, go to 2.

Class	Detector	Detector-fast
aeroplane	0.7427	0.5669
bicycle	0.8249	0.7342
bird	0.5747	0.5565
boat	0.6935	0.6565
bottle	0.7027	0.6276
bus	0.5	0.7533
car	0.8565	0.7448
cat	0.6678	0.6213
chair	0.7029	0.5
cow	0.7160	0.6928
diningtable	0.6639	0.6393
dog	0.5520	0.5281
horse	0.8492	0.7077
motorbike	0.7588	0.7117
person	0.8063	0.7263
pottedplant	0.6213	0.6115
sheep	0.6978	0.6336
sofa	0.6803	0.5901
train	0.8163	0.7149
tvmonitor	0.7622	0.7066

Table 1: Accuracies for Classifiers: Input is a set of detections on an image, decide for class occurence in a better way than thresholding

After executing an action, we get a set of detections on the image. Instead of just tresholding the scores of these detection, we combine all of them to a feature vector that can then be classified by an SVM. This vector consists of a count of detections on the image, as well as a histogram over the values of scores for the detector on that image. For the histogram we examine different numbers of bins as well as lower and upper bounds on the considered range. The intuition behind this approach is that on the one hand positive detections will yield a strong signal in the higher bins of this histogram, which corresponds to just thresholding the scores. On the other hand we also want to correctly detect cases in which we mostly do not have high signals, but still a certain distribution of scores and a high enough number of detections. Table 2 shows the performance of different classifiers. We measure accuracy as number of correct decisions versus overall number of decisions. In each image the classifiers have to decide for each class whether or not an object of this class is present. *detector* shows the results of the histogrammed CSC-classifier and *detector-fast* is CSC with parameters settings to speed it up to about half the time.

Both object class presence probability models are evaluated in Section 1.1.

4.2. Value Function and the Definition of Rewards

Having updated the belief state, the system must again pick the next action. We now cover our approach to this problem.

Remember that toward the goal of *Anytime* performance, a good single-number metric to optimize is the area under the performance vs. time curve. Accordingly, we set the reward accrued by a policy to be precisely the area under the curve between start and deadline times.

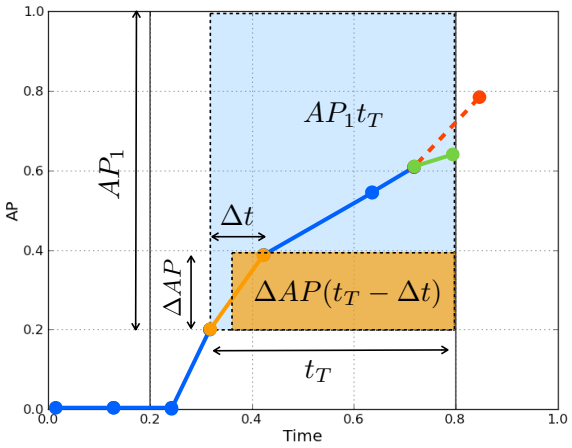


Figure 3: A per-action greedy value function that corresponds to the maximization of our objective function is the ratio of the area of the horizontal slice under the curve due to the action to the maximum possible area the action could have captured. The figure shows this analysis for the action highlighted in orange.

It is desirable to be able to construct this final reward as a sum of individual rewards accrued by each action. Figure 3 demonstrates our choice of such a reward function. The intuitive explanation is that each action's reward is defined by the ratio of the area of AP vs. Time plot that it captured to the area of the plot that it could have captured if it instantaneously obtained the maximum possible performance.

Specifically, we define the reward of an action as

$$\frac{\Delta AP(t_T - \Delta t)}{AP_1 t_T} \tag{1}$$

where AP_1 is the difference between 1 and the multi-class AP of the current detection set, t_T is the time left until deadline, and Δt and ΔAP are the time and AP change taken by the action.

Examining (1), we see that the normalization factor can go away, as we are only interested in comparing values at the same level of AP accrued and time passed. The equation then breaks down into a term to maximize, $\Delta AP t_T$, and a term to minimize, $\Delta AP \Delta t$. This agrees with the intuition that to capture the most area under the curve, the slope needs to be maximized at each point. Additionally, the equation shows that if Δt exceeds t_T, the value of taking the action is negative.

We can formulate a simpler value function that has the same properties:

$$\frac{\Delta AP}{\Delta t}(t_T - \Delta t) \tag{2}$$

This function always maximizes the slope, except for when doing so would lead to taking more time than alloted by the deadline. (Consider the branching of the performance curve in 3 into green and red branches when it approaches the deadline. The branches represent two actions of the case where maximizing slope would not be the correct behavior.)

Of course, we cannot be aware of the true value of Δt until having taken the action, and the policy can never be fully certain of the value of ΔAP. When estimating the value function, we therefore use their expected values. Δt_{avg} is the average running time of the detector on an image, estimated on a validation set of the data at training time. ΔAP is estimated as $P(C)\Delta AP_{avg}$, where $P(C)$ is the current belief in the presence of class C in the image, and ΔAP_{avg} is the "naive" AP contribution averaged over images containing class C[1].

Both value functions are evaluated in Section 1.1, where we also discuss the meaning of "naive" AP contribution.

5. Evaluation

Our system treats the detector as a black box that takes an image and class and returns a list of detections for that class. In our implementation, we use the Cascaded Deformable Part Model detector, which is among both the fastest and most robust detectors currently available [8]. The detectors take on average around one second to process a standard dataset image.

We evaluate on the standard PASCAL VOC 2007 detection task dataset [6]. We note that the dataset is not known for having high levels of signal in the inter-object and scene-object context [5], so the parts of our model that seek to exploit inter-object context may be at a disadvantage. We follow the approach set forth in Section 1.1, but set the start time at 0 and ignore the feature computation time.

During training time, we learn the behavior of different detectors: the average time taken per image, and the average AP contribution on images that do and do not contain any objects of the desired class. For the latter statistic, we collect two variants. The "naive" AP contribution is the performance of the detector in the multi-class regime if its detections were added to an empty set of detections. That is, true detections cannot become false positives due to being

[1]We additionally experimented with estimating ΔAP as $P(C)\Delta AP_{avg|present} + (1 - P(C))\Delta AP_{avg|absent}$, but the results were worse than with the given equation.

overshadows by a more confident but wrong detection of a different class in this evaluation.

The "actual" AP contribution is the real performance of the detector in the multi-class regime: its detections are added to an existing set of detections, unless it was the result of the first action to run. This case, although more realistic, is subject to signficant noise due to other detectors' performance that may obscure the evaluated detector's characteristics.

In Figure 4, the "Oracle" curve is obtained by running detectors in the order of greatest "naive" AP contributions on each image. The performance is formidable, which suggests that the "naive" statistics is worth using in our policies.

Due to the large disparity in performance between detectors for different classes, it is not surprising that the curve of the "Oracle" policy turns downward: after obtaining the initial true positives, the policy has no choice but start running worse-performing detectors which generate false positives, bringing the overall AP down.

6. Policy with multiple detectors per class

We now expand our model to deal with being able to run multiple detectors per class. We simulate a faster but less robust detector, as would be obtained by modifying the stride or classifier strength, by randomly taking half of the detections output by our base detector, described below in Section 1.1.

Simply adding such a detector to the existing system as described would lead to pathological performance. For example, let's assume that the image contains an object of class 'dog;' we run our most powerful detector for that class and thereafter update our belief state with the strong posterior of this class presence. Our value function estimates for running the other 'dog' detectors are now increased from before, despite the fact that actually running them is unlikely to produce any new true positives and will in fact probably hurt performance due to false positives. (Remember that in our evaluation, a ground truth object may only be matched to one detection proposal—all other detections matching that ground truth are considered false positives.)

For this reason, we augment the value function with this consideration. The only change that needs to be made is to the expected score increase:

$$\mathbb{E}[\Delta AP^i] = P(C)\Delta AP^i_{avg} - P(C)\sum_{j \neq i} \delta_j \Delta AP^j_{avg} \quad (3)$$

where ΔAP^i is the score increase of the detector belonging to the action under consideration, the sum is over all other detector actions of the same class as i, and δ_j is an indicator function for whether action j has been taken.

Figure 5 shows the performance comparison of the usual random, oracle, and the two evaluated policies.

Policy	Fixed-order	Backoff	Random	Time
Advanced	2.17 (0.34)	1.65 (0.3)	0.74	10
	5.51 (0.38)	4.97 (0.38)	3.28	20
	9.27 (0.39)	8.7 (0.39)	6.93	30
Slope	2.39 (0.35)	2.21 (0.32)	0.74	10
	6.10 (0.38)	5.76 (0.38)	3.28	20
	9.97 (0.39)	9.63 (0.39)	6.93	30
Slope (Double)	2.16 (0.3)	1.78 (0.24)	-	10
	5.4 (0.34)	4.24 (0.27)	-	20
	8.85 (0.36)	7.11 (0.32)	-	30
	12.36 (0.36)	10.43 (0.35)	-	40
Slope (Pair)	2.16 (0.3)	1.78 (0.24)	-	10
	5.39 (0.34)	4.27 (0.27)	-	20
	8.83 (0.36)	7.22 (0.33)	-	30
	12.34 (0.36)	10.60 (0.35)	-	40

Table 2: Detailed results of our policies.

Although we do not evaluate this, our system allows for setting priority weights on different detector classes; this results in different policy behaviors.

7. Future Work

Extra-detection actions In our implementation of the system, we leave open the possibility of extra-detection actions—those actions that may modify the belief state but not generate any detections directly. In fact, we implemented one such action: a regression from the GIST [14] feature of the image to object class presence. In our experiments, this action was always taken first in every policy evaluation; we did not observe an improvement over not using this special action, but leave the question open for further investigation.

Policy Iteration While our current implementation shows significant gains over baseline, the gap between our policies' performance and the oracle performance shows that there is yet room to improve. In particular, we are looking at using the general method of policy iteration to derive approximations of the value function that are able to capture not just greedy but expected rewards to the end of the episode. Solving POMDPs, of which our stated problem is an example, has not in general been robustly done for problems of this size [12, 13], but there are promising relaxes approaches [9].

References

[1] N. Butko and J. Movellan. Optimal scanning for faster object detection. *CVPR*, (1):2751–2758, June 2009. 2

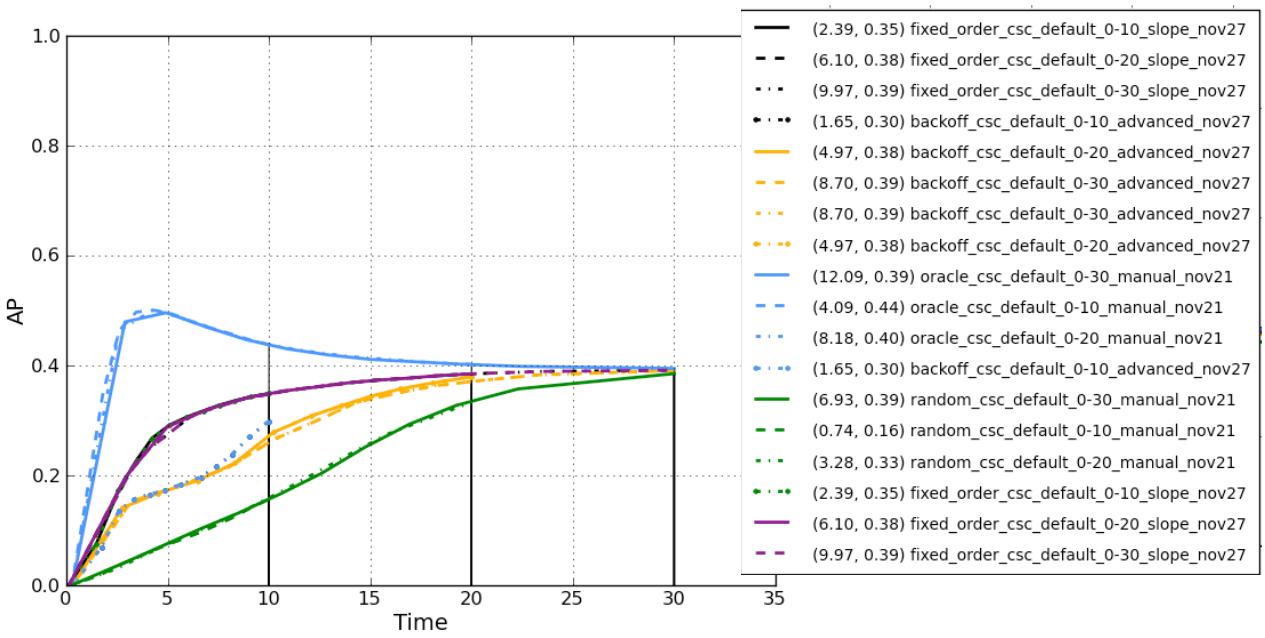


Figure 4: The random policy shows the typical path of an uninformed multi-class detector through the image, while the oracle policy shows the room to improve. Our policy results in performance between the two, showing clear improvement over the random baseline.

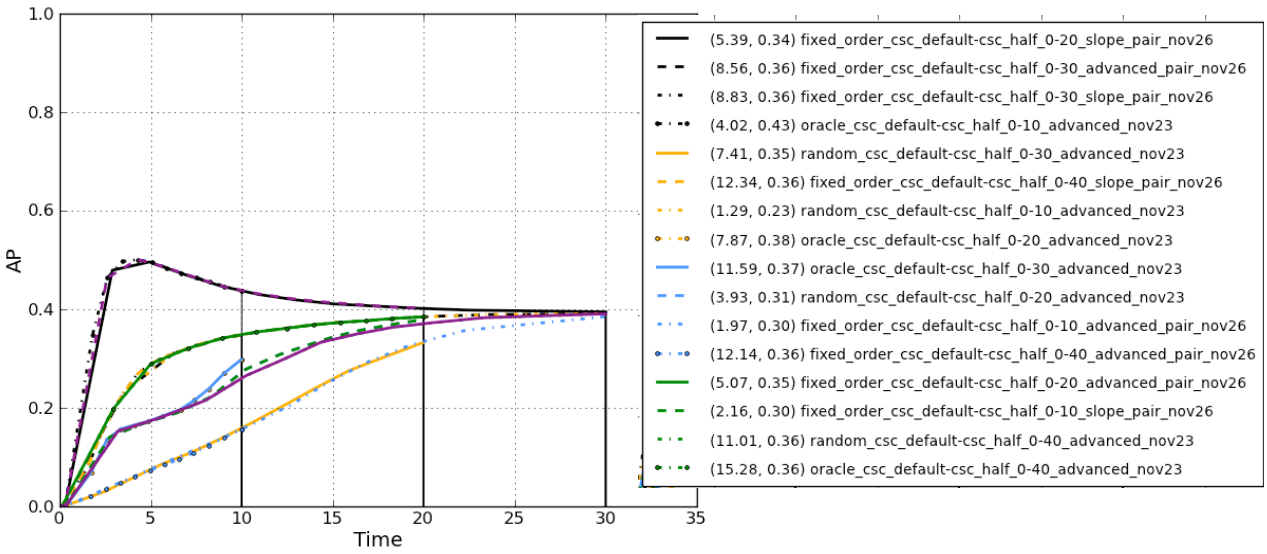


Figure 5: When using multiple detectors per class, there is a clear benefit to using a policy that is sensitive to the finite amount of reward that can be obtained from a single class.

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

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

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

[5] S. Divvala, D. Hoiem, J. Hays, A. Efros, and M. Hebert. An empirical study of context in object detection. In *CVPR*, pages 1271–1278. Ieee, June 2009. 2, 5

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html. 1, 5

[7] X. Fan. Efficient Multiclass Object Detection by a Hierarchy of Classifiers. In *CVPR*, pages 716–723. Ieee, 2005. 2

[8] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–45, Sept. 2010. 2, 5

[9] C. Kwok and D. Fox. Reinforcement Learning for Sensing Strategies. In *IROS*, 2004. 6

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

[11] D. Mitzel, P. Sudowe, and B. Leibe. Real-Time Multi-Person Tracking with Time-Constrained Detection. In *BMVC*, 2011. 2

[12] K. P. Murphy. A Survey of POMDP Solution Techniques. Technical Report September, 2000. 6

[13] A. Y. Ng and M. Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. *UAI*, 2000. 6

[14] A. Oliva and A. Torralba. Modeling the Shape of the Scene : A Holistic Representation of the Spatial Envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001. 6

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

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

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

[18] 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

[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*, 29(5):854–69, May 2007. 2

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

[21] S. Vijayanarasimhan and A. Kapoor. Visual Recognition and Detection Under Bounded Computational Resources. In *CVPR*, pages 1006–1013, 2010. 2, 3

[22] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, pages I–511–I–518, 2001. 2

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