Timely Object Recognition

Anonymous Author(s)
Affiliation
Address
email

Abstract

In a large visual multi-class detection framework, the timeliness of results can be crucial. Our method for *timely* multi-class detection aims to give the best possible performance at any single point after a start time; it is terminated at a deadline time. Toward this goal, we formulate a dynamic, *closed-loop* policy that infers the contents of the image in order to decide which detector to deploy next. In contrast to previous work, our method significantly diverges from the predominant greedy strategies, and is able to learn to take actions with deferred values. We evaluate our method with a novel *timeliness* measure, computed as the area under an Average Precision vs. Time curve. Experiments are conducted on the eminent PASCAL VOC object detection dataset. If execution is stopped when only half the detectors have been run, our method obtains 66% better AP than a random ordering, and 14% better performance than an intelligent baseline. On the timeliness measure, our method obtains at least 11% better performance. Our code, to be made available upon publication, is easily extensible as it treats detectors and classifiers as black boxes and learns from execution traces using reinforcement learning.

1 Introduction

In most real-world applications of object recognition, performance is time-sensitive and inherently tied to the many-class nature of the world. 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 systems (e.g. image search), results need to be obtained quickly per image as the number of items to process is constantly growing. In these cases, an acceptable answer at a reasonable time may be more valuable than the best answer given too late.

The conventional approach to evaluation of visual category recognition does not consider efficiency and evaluates performance independently across classes. We argue that the key to tackling problems of dynamic recognition resource allocation is to start asking a new question: *What is the best performance we can get on a budget?*

We propose a new *timeliness* measure of performance vs. time (shown in Figure 1), and present a method based on reinforcement learning that takes various detectors and classifiers as black boxes, and learns a dynamic policy for selecting detector and other actions to achieve the highest performance under this evaluation. We are able to obtain better performance than all baselines when there is less time available than is needed to exhaustively run all detectors.

A hypothetical system for vision-based advertising presents a case study. The system will have different values and accuracies for objects of different classes, and the queue of unprocessed images will vary in size. The detection strategy to maximize profit in such an environment has to be dynamic. We explore this scenario using the PASCAL VOC datasets and evaluation regimes, and show that our learned dynamic policy offers a better strategy than a random or optimal static ordering of detectors.

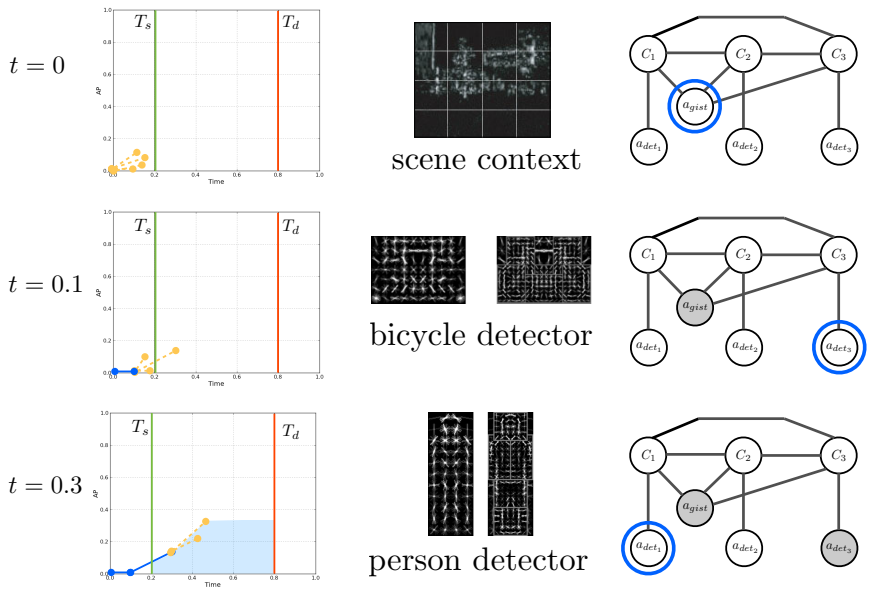


Figure 1: A sample trace of our method. At each time step beginning at $t = 0$, potential actions are considered according to their *value*, and the one with the highest value is picked. The selected action is performed and returns observations, which influence the selection of the next action. The final evaluation of a detection episode is the area of the AP vs. Time curve between the start and end times. The value of an action is the expected value of the final evaluation if the action is taken and the policy continues to be followed, which allows actions without an immediate benefit to be scheduled.

2 Recognition Problems and Related Work

We deal with a dataset of images $\mathcal{D}$, where each image $\mathcal{I}$ contains zero or more objects. Each object is labeled with exactly one category label $k \in \{1, \ldots, K\}$.

The multi-class, multi-label **classification** problem asks whether $\mathcal{I}$ contains at least one object of class k. We write the ground truth for an image as $\mathbf{C} = \{C_1, \ldots, C_K\}$, where $C_k \in \mathbb{B} = \{0, 1\}$ is set to 1 if an object of class k is present.

The **detection** problem is to output a list of bounding boxes (sub-images defined by four coordinates), each with a real-valued confidence that it encloses a single instance of an object of class k, for each k. The answer for a single class is given by an algorithm *detect*$(\mathcal{I}, k)$, which outputs a list of sub-image bounding boxes B and their associated confidences.

The answer is evaluated by plotting precision vs. recall across dataset $\mathcal{D}$ (by progressively lowering the confidence threshold for a positive detection). The area under the curve yields the Average Precision (AP) metric, which has become the standard evaluation for recognition performance on challenging datasets in vision [1]. A common measure of a correct detection is the PASCAL overlap: two bounding boxes are considered to match if they have the same label and the ratio of their intersection to their union is at least $\frac{1}{2}$.

To highlight the hierarchical structure of these problems, we note that (1) the confidences for each sub-image $b \in B$ may be given by *classify*(b, k); (2) correct answer to the detection problem also answers the classification problem.

Multi-class performance is evaluated by averaging the individual per-class AP values. We generalize this metric to a weighted average, with the weights set by the *values* of the classes.

2.1 Related Work

The literature on object recognition is vast. Here we briefly summarize work that sufficiently contextualizes our contribution.

Single-class detection The best recent performance has come from detectors that use gradient-based features to represent objects as either a collection of local patches or as object-sized windows [2, 3]. Classifiers are then used to distinguish between featurizations of a given class and all other possible contents of an image window. Window proposal is most often done exhaustively over the image space, as a "sliding window". Some approaches use "jump windows" (hypotheses voted on by local features) [4, 5], or a bounded search over the space of all possible windows [6].

None of the best-performing systems treat window proposal and evaluation as a closed-loop system, with feedback from evaluation to proposal. Some work has been done on this topic, mostly inspired by ideas from biological vision and attention research [7, 8].

Context Context has a foundational role in vision. One source of context is the scene or other non-detector cues; the most common scene-level feature is the GIST [9] of the image. For the commonly used PASCAL VOC dataset [1], GIST and other sources of context are quantitatively explored in [10]. Inter-object context has also been shown to be useful for improving detection [11]. A critical summary of the main approaches to using context for object and scene recognition is given in [12].

Multi-Class Detection Work on inherently multi-class detection focuses largely on making detection time sublinear in the number of classes through sharing features [13, 14]. A post-processing extension to detection systems uses structured prediction to incorporate multi-class context as a principled replacement for non-maximum suppression [15].

Cascades An early success in efficient object detection used simple, fast features to build up a *cascade* of classifiers, which then considered image regions in a sliding window regime [16]. Most recently, cyclic optimization has been applied to optimize cascades with respect to feature computation cost as well as classifier performance [17]. However, cascades are not dynamic policies—they cannot change the order of execution based on observations obtained during execution, which is our goal.

Anytime Algorithms and Active Classification This surprisingly little-explored line of work in vision is closest to our approach. A recent application to the problem of visual detection picks features with maximum value of information in a Hough-voting framework [18]. There has also been work on active classification [19] and active sensing [20], in which intermediate results are considered in order to decide on the next classification step. Most commonly, the scheduling in these approaches is greedy with respect to some manual quantity such as expected information gain. In contrast, we learn policies that take actions without any immediate reward.

3 Multi-class Recognition Policy

Our goal is a multi-class recognition policy π that takes an image $\mathcal{I}$ and outputs a list of multi-class detection results by running detector and global scene *actions* sequentially.

The policy repeatedly selects an action $a_i \in \mathcal{A}$, executes it, receiving observations o_i, and then selects the next action. The set of actions $\mathcal{A}$ can include both classifiers and detectors: anything that would be useful for inferring the contents of the image.

Each action a_i has an expected cost $c(a_i)$ of execution. Depending on the setting, the cost can be defined in terms of algorithmic runtime analysis, an idealized property such as number of *flops*, or simply the empirical runtime on specific hardware. We take the empirical approach: every executed action advances t, the *time into episode*, by its empirical runtime.

As shown in Figure 1, the system is given two times: the setup time T_s and deadline T_d. From the setup time to the deadline, we want to obtain the best possible answer if stopped at any given time. A single-number metric that corresponds to this objective is the area captured under the curve between the start and deadline bounds, normalized by the total area. We evaluate policies by this more robust metric and not simply by the final performance at deadline time for the same reason that Average Precision is used instead of a fixed Precision vs. Recall point in the conventional evaluations.

3.1 Sequential Execution

An *open-loop* policy takes actions in a sequence that does not depend on observations received from previous actions. The common classifier cascade is an example [16]. In contrast, our goal is to learn a dynamic, or *closed-loop*, policy, which would exploit the signal in scene and inter-object context for a maximally efficient path through the actions.

We refer to the information available to the decision process as the *state* s. The state includes the currently believed distribution over class presence variables $P(\mathbf{C}) = \{P(C_0), \ldots, P(C_K)\}$, where we write $P(C_k)$ to mean $P(C_k = 1)$.

Additionally, the state records that an action a_i has been taken by adding it to the initially empty set $\mathcal{O}$ and recording the resulting observations o_i. We refer to the current set of observations as $\mathbf{o} = \{o_i | a_i \in \mathcal{O}\}$. The state also keeps track of the time into episode t, and the setup and deadline times T_s, T_d.

A recognition *episode* takes an image $\mathcal{I}$ and proceeds from the initial state s^0 and action a^0 to the next pair (s^1, a^1), and so on until (s^J, a^J), where J is the last step of the process with $t \leq T_d$. At that point, the policy is terminated, and a new episode can begin on a new image.

The specific actions we consider in the following exposition are detector actions a_{det_i}, where det_i is a detector class C_i, and a scene-level context action a_{gist}, which updates the probabilities of all classes.

3.2 Selecting actions

As our goal is to pick actions dynamically, we want a function $Q(s, a) : S \times \mathcal{A} \mapsto \mathbb{R}$, where S is the space of all possible states, to assign a value to a potential action $a \in \mathcal{A}$ given the current state s of the decision process. We can then define the desired policy π as simply taking the (untaken) action with the maximum value:

$$\pi(s) = \underset{a_i \in \mathcal{A} \setminus \mathcal{O}}{\operatorname{argmax}} Q(s, a_i) \tag{1}$$

Although the action space $\mathcal{A}$ is quite manageable, consisting of the detectors and global classifier we would like to run on the image, the space of possible states S is intractable. Therefore we cannot learn a tabular representation of $Q(s, a)$, and must use function approximation to represent it [21]. We featurize the state-action pair and assume linear structure: $Q^\pi(s, a_i) = \theta_\pi^\top \phi(s, a_i)$.

The policy's performance at time t is determined by the detections that are part of the set of observations $\mathbf{o}^j$ at the last state s^j before t. Therefore, the final AP vs. Time evaluation of an episode is a function $eval(h, T_s, T_d)$ of the history of execution $h = s^0, s^1, \ldots, s^J$. It is precisely the normalized area under the AP vs. Time curve between T_s and T_d, as determined by the detections in $\mathbf{o}^j$ for all steps j in the episode.

As shown in 3b, this evaluation function is additive per action, as each action can generate detections that either raise or lower the mean AP of the results so far (Δap) and takes a certain time (Δt). From these and T_s and T_d, we can find the area under the curve that was contributed by the action. So, we can then represent the final evaluation $eval(h, T_s, T_d)$ in terms of individual rewards: $\sum_{j=0}^{J} R(s^j, a^j)$.

Specifically, as shown in Figure 3b, we define the *reward* of an action as

$$R(s^j, a_i) = \Delta \text{ap}_i(t_T^j - \frac{1}{2}\Delta t_i) \tag{2}$$

where t_T^j and ap^j are the time left until the deadline and the AP at state s^j, and Δt_i and Δap_i are the time taken and AP change produced by the action a_i. (For clarity of exposition, we do not account for T_s here.)

3.3 Learning the policy

The expected value of the final evaluation can be written recursively in terms of the value function:

$$Q^\pi(s^j, a_i) = \mathbb{E}_{s^{j+1}}[R(s^j, a_i) + \gamma Q^\pi(s^{j+1}, \pi(s^{j+1}))] \tag{3}$$

where $\gamma \in [0, 1]$ is a *discount* value that can mitigate the effects of increasing state-transition uncertainty over long episodes.

While we can't directly compute the expectation in (3), we can sample it by running actual episodes to gather $< s, a, r, s' >$ samples, where r is the reward obtained by taking action a in state s, and s' is the following state.

Learning the policy is then a problem of repeatedly gathering samples with the current policy, minimizing the error between the discounted reward to the end of the episode as predicted by our current $Q(s^j, a_i)$ and the actual values gathered, and updating the policy with the resulting weights. This is fitted Q-iteration, a variant of generalized policy iteration [22, 21].

To ensure sufficient exploration of the state space, we implement ϵ-greedy action selection during training: with a probability that decreases with each training iteration, a random action is selected instead of following the policy. During test time, ϵ is set to 0.05.

We use L_2-regularized regression to minimize the error. We run 15 iterations of accumulating samples by running 350 episodes, starting with a baseline policy which will be described in section 4, and cross-validating the regularization parameter at each iteration; samples are not thrown away between iterations.

A meta-parameter of the approach is the discount γ. With $\gamma = 0$, the value function is determined entirely by the immediate reward. Learning in this case can only result in completely greedy policies.

With $\gamma = 1$, the value function is determined by the expected rewards to the end of the episode, and so should be a close approximation to the final evaluation metric. However, the highest value for γ is not necessarily best if the action-value function is not expressive enough to represent the actual state transition behavior of the world. We experiment with several values of γ, and find a mid-level value (0.4) to work best.

3.4 Feature representation

Our policy is at its base determined by a linear function of the features of the state: $\pi(s) = \text{argmax}_{a_i \in \mathcal{A} \setminus \mathcal{O}} \theta_\pi^\top \phi(s, a_i)$. Since we want to be able to learn a dynamic policy, the observations $\mathbf{o}$ that are part of the state s should play a role in determining the value of a potential action.

We include the following quantities as features $\phi(s, a)$:

$P(C_a)$ The prior probability of the class that corresponds to the detector of action a

$P(C_0|\mathbf{o}) \dots P(C_K|\mathbf{o})$ The probabilities of all classes, conditioned on the current set of observations.

$H(C_0|\mathbf{o}) \dots H(C_K|\mathbf{o})$ The entropies of all classes, conditioned on the current set of observations.

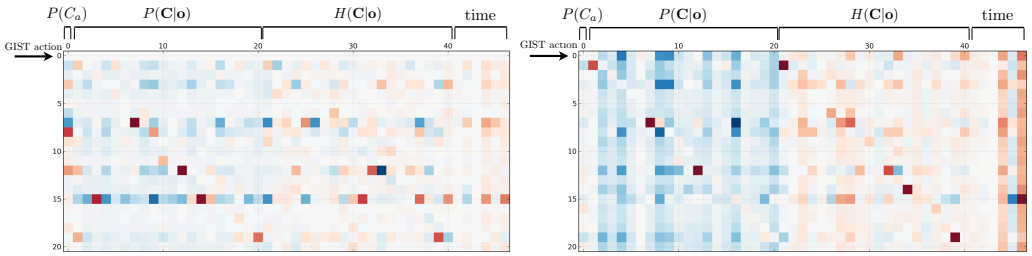
Additionally, we include the mean and maximum entropies of all classes, the expected time of the action $c(a)$, and time features that represent the times until start and deadline, for a total of F features.

We note that this setup is commonly used to solve Markov Decision Processes [21]. There are two related limitations of MDPs when it comes to most systems of interesting complexity, however: the state has to be functionally approximated instead of exhaustively enumerated; and some aspects of the state are not observed, making the problem a Partially Observed MDP (POMDP), for which exact solution methods are intractable for all but rather small problems [23].

There isn't much to do about the necessity of approximating the state but design good features. Our initial solution to the partial observability problem is to include uncertainty features into the feature representation to *augment* the MDP [24].

To formulate learning the policy as a single regression problem, we represent the features in block form, where $\phi(s, a_i)$ is a vector of size $F|\mathcal{A}|$, with all values set to 0 except for the block corresponding to a_i.

As an illustration, we visualize the learned weights on these features in Figure 2, reshaped such that each row shows the weights learned for an action, in order. The featurization for the a_{gist} scene-context action, which concerns all classes, omits the first $P(C_a)$ feature.



(a) Greedy (b) RL

Figure 2: Learned policy weights θ_π (best viewed in color: red corresponds to positive, blue to negative values). The first row corresponds to the scene-level action, which does not generate detections itself but only helps reduce uncertainty about the contents of the image. Note that in the greedy learning case, this action is learned to never be taken—whereas it is shown to be useful in the reinforcement learning case.

3.5 Updating with observations

The bulk of our feature representation is formed by probability of individual class occurrence, conditioned on the observations so far: $P(C_0|\mathbf{o}) \dots P(C_K|\mathbf{o})$. This allows the action-value function to learn correlations between presence of different classes, and so the policy can look for the most probable classes given the observations.

However, higher-order co-occurrences are not well represented in this form. Additionally, updating $P(C_i|\mathbf{o})$ presents choices regarding independence assumptions between the classes.

We evaluate two approaches for updating probabilities: *direct* and *MRF*. In the *direct* method, $P(C_i|\mathbf{o}) = score(C_i)$ if $\mathbf{o}$ includes the observations for class C_i and $P(C_i|\mathbf{o}) = P(C_i)$ otherwise. This means that an observation of class i does not influence the estimated probability of any class but C_i. $score(C_i)$ for a_{det_i} is obtained by training a probabilistic classifier on the detections output. $score(C_i)$ for a_{gist} is obtained by training probabilistic classifiers on the GIST feature, for all classes.

The *MRF* approach employs a pairwise fully-connected Markov Random Field (MRF), as shown in Figure 1, with the observation nodes set to $score(C_i)$ appropriately, or considered unobserved.

The graphical model structure is set as fully-connected, but some classes are overwhelmingly unlikely to co-occurr in our dataset. Accordingly, the graph edge weights are learned with L_1 regularization, which obtains the desired sparse structure [25]. All parameters of the model are trained on fully-observed data, and Loopy Belief Propagation inference is implemented with an open-source graphical model package [26].

4 Evaluation

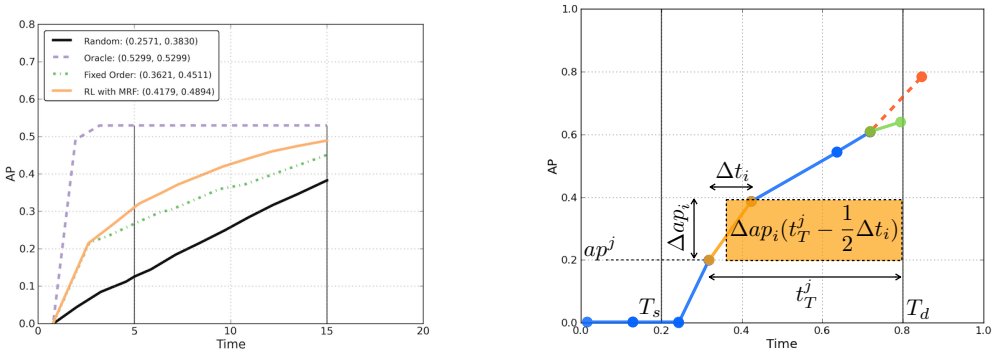
We evaluate our system on the multi-class, multi-label detection task as previously described. Each detection episode takes an image and outputs detections with associated times, based on the order of actions taken. We evaluate on a popular detection challenge task: the PASCAL VOC 2007 dataset [1], which exhibits only a rather modest amount of class co-occurrence: the "person" class is highly likely to occur, and less than 10% of the images have more than two classes.

We learn weights on the training and validation sets, and run our policy on all images in the testing set. The final evaluation pools all detections up to a certain time, and computes their multi-class AP per image, averaging over all images. This is done for different times to plot the AP vs. Time curve over the whole dataset. Our method of averaging per-image performance follows [15].

For the detector actions, we use one-vs-all cascaded deformable part-model detectors on a HOG featurization of the image [27], with associated linear classification of the detections. There are 20

classes in the PASCAL challenge task, so there are 20 detector actions. Running a detector on a PASCAL image takes about 1 second.

We test three different settings of the start and deadline times. In the first one, the start time is immediate and execution is cut off at 20 seconds, which is enough time to run basically all actions. In the second one, execution is cut off after only 10 seconds. Lastly, we measure performance between 5 seconds and 15 seconds. These operating points show how our method behaves when deployed in different conditions. The results are given in rows of Table 1.



(a) (b)

Figure 3: (a) AP vs. Time curves for Random, Oracle, the Fixed Order baseline, and our best-performing policy, with start time of 5 seconds and deadline of 15 seconds. (b) Reward function explanation.

We establish the first baseline for our system by selecting actions randomly at each step. As shown in Figure 3a, this **Random** policy results in a roughly linear gain of AP vs. time. This is expected: the detectors are capable of obtaining a certain level of performance; if half the detectors are run, the expected performance level is half of the maximum level.

To establish an upper bound on performance, we plot the **Oracle** policy, obtained by re-ordering the actions with the hindsight of the results they produced.

We consider another baseline: selecting actions in a fixed order based on the value they bring to the AP vs. Time evaluation, which is roughly proportional to their occurrence probability. We refer to this as **Fixed Order**.

Then there are our proposed systems, described in the previous section : **RL w/ Direct** and **RL w/ MRF**. In experiments, the two were not exceedingly different in performance, with the MRF model consistently performing slightly better.

In Figure 3a, we can see that due to the dataset bias, the fixed-order policy performs well at first, as the person class is disproportionately likely to be in the image, but is significantly overtaken by our model as execution goes on and more rare classes have to be detected.

Lastly, we include an additional scene-level GIST feature that updates the posterior probabilities of all classes. This is considered one action, and takes about 0.3 seconds. This setting always uses the MRF model to properly update the class probabilities with GIST observations. This brings another boost in performance. All results are shown in Table 1.

Additionally, we evaluated different settings of the discount parameter γ, and found that a mid-range value works best. We only report results for that value on the test set. Figure 2 shows how changing γ affects the learned weights: the GIST action is learned to never be taken in the greedy setting, because it does not bring any immediate detections results. In the reinforcement learning setting, however (higher γ), GIST is learned to be taken for its value in quickly refining the class presence probabilities.

Figure 4 illustrates the action trajectories of different policies.

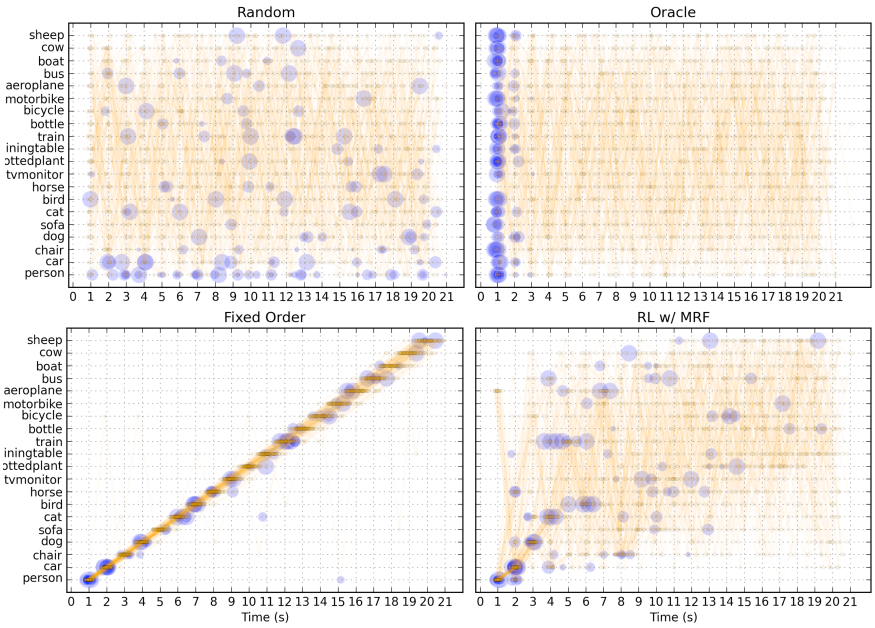


Figure 4: Visualizing the action trajectories of different policies. Action selection traces are plotted in orange over many episodes; the size of the blue circles correspond to the Δap obtained by the action. We see that the **Random** policy selects actions and obtains rewards randomly, while the **Oracle** policy obtains all rewards in the first few actions. The **Fixed Order** policy selects actions in a static optimal order. Our policy does not stick a static order but selects actions dynamically to maximize the rewards obtained early on.

Table 1: The areas under the AP vs. Time curve for different experimental conditions. Our method reliably beats the intelligent static baseline; including the GIST action further boosts performance.

Bounds	Random	Fixed Order	RL w/ MRF	w/ GIST	Oracle
(0,20)	0.250	0.342	0.378	**0.382**	0.488
(0,10)	0.119	0.240	0.266	**0.267**	0.464
(5,15)	0.257	0.362	0.418	**0.420**	0.530

5 Conclusion

We presented a method for learning "closed-loop" policies for multi-class object recognition, given existing object detectors and classifiers and a metric to optimize. The method learns the optimal policy using reinforcement learning, by observing execution traces in training. If detection on an image is cut off after only half the detectors have been run, our method does 66% better than random, and 14% better than an intelligent baseline. In particular, our method learns to take action with no intermediate reward in order to improve the overall performance of the system.

As always with reinforcement learning problems, defining the reward function requires some manual work. Here, we derive it for the novel detection AP vs. Time evaluation that we suggest is most useful for evaluating efficiency in recognition. Although computation devoted to scheduling actions is much less significant than the computation due to running the actions, an interesting future research direction is to explicitly consider this decision-making cost. At time of publication, we will release the code of our modular framework, that allows for easy incorporation of other classifiers and detectors on different tasks and rewards.

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. 2, 3, 6

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

[3] David G Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. 3

[4] Andrea Vedaldi, Varun Gulshan, Manik Varma, and Andrew Zisserman. Multiple kernels for object detection. *ICCV*, pages 606–613, September 2009. 3

[5] Sudheendra Vijayanarasimhan and Kristen Grauman. Large-Scale Live Active Learning: Training Object Detectors with Crawled Data and Crowds. In *CVPR*, 2011. 3

[6] Christoph H Lampert, Matthew B Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. *CVPR*, pages 1–8, June 2008. 3

[7] N J Butko and J R Movellan. Optimal scanning for faster object detection. *CVPR*, (1):2751–2758, June 2009. 3

[8] Julia Vogel and Nando de Freitas. Target-directed attention: Sequential decision-making for gaze planning. *ICRA*, pages 2372–2379, May 2008. 3

[9] Aude Oliva and Antonio Torralba. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *IJCV*, 42(3):145–175, 2001. 3

[10] 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. Ieee, June 2009. 3

[11] Antonio Torralba, Kevin P Murphy, and William T Freeman. Contextual Models for Object Detection Using Boosted Random Fields. *MIT Computer Science and Artificial Intelligence Laboratory Technical Report*, 2004. 3

[12] Carolina Galleguillos and Serge Belongie. Context based object categorization: A critical survey. *Computer Vision and Image Understanding*, 114(6):712–722, June 2010. 3

[13] Antonio Torralba, Kevin P Murphy, and William T Freeman. Sharing visual features for multiclass and multiview object detection. *PAMI*, 29(5):854–869, May 2007. 3

[14] Xiaodong Fan. Efficient Multiclass Object Detection by a Hierarchy of Classifiers. In *CVPR*, pages 716–723. Ieee, 2005. 3

[15] Chaitanya Desai, Deva Ramanan, and Charless Fowlkes. Discriminative models for multi-class object layout. In *2009 IEEE 12th International Conference on Computer Vision*, pages 229–236. Ieee, September 2009. 3, 6

[16] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 3, 4

[17] Minmin Chen, Zhixiang Eddie, Xu Kilian, and Saint Louis. Classifier Cascade for Minimizing Feature Evaluation Cost. In *AISTATS*, 2012. 3

[18] Sudheendra Vijayanarasimhan and Ashish Kapoor. Visual Recognition and Detection Under Bounded Computational Resources. In *CVPR*, pages 1006–1013, 2010. 3

[19] Tianshi Gao and Daphne Koller. Active Classification based on Value of Classifier. In *NIPS*, pages 1–9, 2011. 3

[20] Shipeng Yu, Balaji Krishnapuram, Romer Rosales, and R Bharat Rao. Active Sensing. In *AISTATS*, pages 639–646, 2009. 3

[21] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998. 4, 5

[22] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research*, 6:503–556, 2005. 5

[23] Nicholas Roy and Geoffrey Gordon. Exponential Family PCA for Belief Compression in POMDPs. In *NIPS*, 2002. 5

[24] Cody Kwok and Dieter Fox. Reinforcement Learning for Sensing Strategies. In *IROS*, 2004. 5

[25] Su-In Lee, Varun Ganapathi, and Daphne Koller. Efficient Structure Learning of Markov Networks using L 1 -Regularization. In *NIPS*, 2006. 6

[26] Ariel Jaimovich and Ian Mcgraw. FastInf : An Efficient Approximate Inference Library. *Journal of Machine Learning Research*, 11:1733–1736, 2010. 6

[27] Pedro F Felzenszwalb, Ross B Girshick, and David McAllester. Cascade object detection with deformable part models. In *CVPR*, pages 2241–2248. IEEE, June 2010. 6