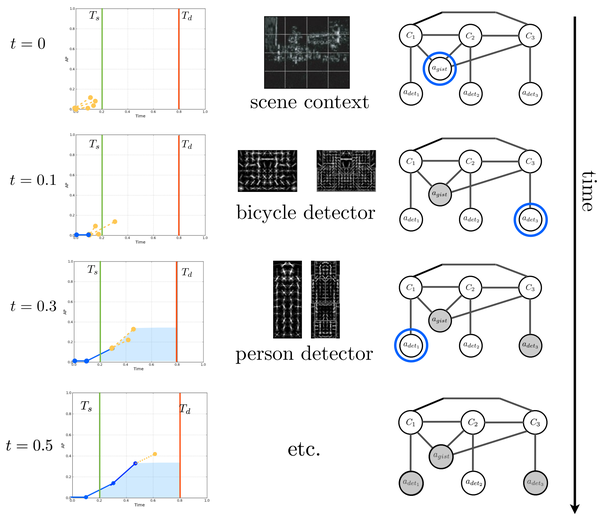
# Timely Object Recognition

Sergey Karayev[1], Tobias Baumgartner[2], Mario Fritz[3], Trevor Darrell[1]
[1]UC Berkeley, [3]RWTH Aachen, [3]MPI Informatics

## Abstract


- Our task is multi-class object detection in a setting where there is not enough time to evaluate all detectors.
- We use reinforcement learning to learn a *dynamic* policy for selecting detectors and classifiers.
- We evaluate detection performance vs. time with the novel *timeliness* measure, and test on the PASCAL VOC challenge.


## Sequential Detection




- Potential actions $a \in \mathcal{A}$ are considered according to their predicted *value*.
- The value $Q(s, a)$ depends on the *belief state* of the system as well as the action considered.
- The selected action returns observations (list of detections, or evaluated feature).
- Observations update the belief state through an inference process.
- The final evaluation is the area of the AP vs. Time curve between the start time $T_s$ and end time $T_d$.
- VOCalues are learned with reinforcement learning, using a reward function derived from this evaluation.

## Learning the Policy

We define our policy as taking an untaken action with maximum value.

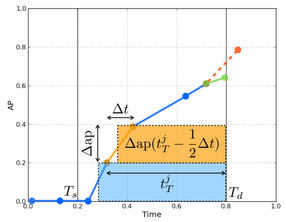
$$\pi(s) = \arg \max_{a_i \in \mathcal{A} \setminus \mathcal{O}} Q(s, a_i) = \theta_\pi^\top \phi(s, a)$$

The Q-function is linearly approximated.

### Reward Function

The final performance evaluation of area under the AP vs. Time curve between $T_s$ and $T_d$ is additive per action. We define the reward function as

$$R(s^j, a) = \Delta\text{ap}(t_T^j - \frac{1}{2}\Delta t)$$




### Reinforcement Learning

To compute $Q^\pi(s^j, a) = \mathbb{E}[R \mid s, a, \pi)]$, where $R = \sum_{i=j}^{J} \gamma^{i-j} R(s^i, a^i)$ and $J$ is the index of the last action before deadline time $T_d$, we run Monte Carlo policy iteration. We evaluate full episodes of the detection task on several thousand images, with a decreasing $\epsilon$-greedy policy. Policy weights $\theta$ are updated with $L_2$ regularization.

We evaluate different values of the reward discount $\gamma$, which at $0$ induces a completely greedy policy, and at $1$ has full lookahead. We find $\gamma = 0.4$ to work best in cross-validation.

### Feature Representation

The features $\phi(s, a)$ are composed of

| | |
|---|---|
| $P(C_a)$ | The prior probability of presence of the class that corresponds to the detector of action $a$ (omitted for the scene-context action). |
| $P(C_0\|\mathbf{o}) \dots P(C_K\|\mathbf{o})$ | The presence probabilities for all classes, conditioned on the current set of observations. |
| $H(C_0\|\mathbf{o}) \dots H(C_K\|\mathbf{o})$ | The entropies for all classes, conditioned on the current set of observations. |
| $t_T^j$ | The time until deadline, and four other time features. |

### Updating the belief state

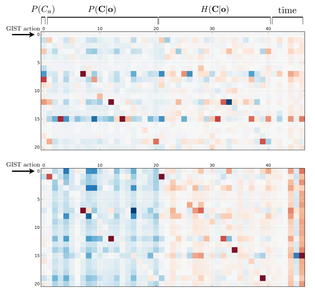
Observations $\mathbf{o}$ of detector for class $k$ update $P(C_k|\mathbf{o})$ either *directly* (using a probability estimated by a classifier trained on the detector output) or through an *MRF*. The MRF is learned with $L_1$-regularization and inference is done with loopy belief propagation.

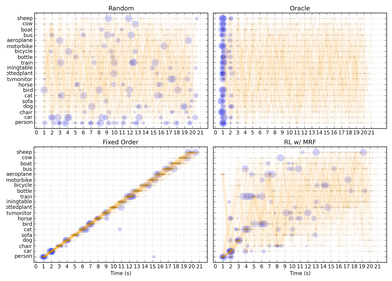
## Results

We evaluated on the PASCAL VOC 2007 detection task, with 20 Deformable Part Model detectors (one per class), and an additional scene context action (GIST feature).

We compared against a random baseline, an optimal static ordering, and an oracle ordering.

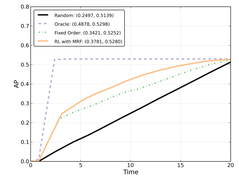
**Weights** Learning with a higher $\gamma$ results in policies more relient on the global scene feature, for example.




**Trajectories** Action selection traces are plotted in orange over many episodes; the size of the blue circles correspond to the increase in AP obtained by the action. Our policy does not stick a static order but selects actions dynamically to maximize the rewards obtained early on.

**Performance vs. Time**
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




| Bounds | Random | Fixed Order | RL | RL 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 |