Leaving Some Stones Unturned: Dynamic Feature Prioritization for Activity Detection in Streaming Video

Yu-Chuan Su and Kristen Grauman

University of Texas at Austin

The supplementary materials consist of:

- Qualitative examples showing recognition episodes in streaming environment. **Also see our project webpage for video examples.** (A)
- Streaming recognition results with different detector speeds (B).
- Un-trimmed detection results with different object detector speeds (C).
- State feature and reward function design (D).
- Details for policy iteration (E).
- Details for observation imputation in batch environment (F).
- Implementation details (G).
- Details for un-trimmed video generation (H).

A Example Recognition Episodes

Table 1 shows example excerpts of learned policies with objects. Here we see, for example, how our approach learns to detect objects that can verify current activity hypothesis or differentiate ambiguous activities, e.g., tap does not co-occur with TV, so seeing tap rules out "Watch-TV." It also demonstrates detailed memory such that it looks for objects that have been observed before but in a different status (actively being used by the recorder vs. passively sitting there).

See the project webpage for more video examples.

$a^{(k)}$	Result	Observed Obj.	Possible Activities	$a^{(k+1)}$
TV	+	None	Watch-TV	TV-remote
	-	Kettle	Watch-TV/Make-tea	Tea-bag
	-	Bottle	Drink-water	Fridge
Tap	+	Dent-floss	Brush-teeth	Soap-passive
	-	Dish	Wash-dish/Watch-TV	Tap
	-	Soap-passive	Wash-hand	Soap-active

Table 1: Excerpts of policies learned from ADL in the streaming case. "+" and "-" indicate whether the object is detected at step-k. Observed objects are those observed before $a^{(k)}$, and possible activities are the most likely activities predicted at step-k.

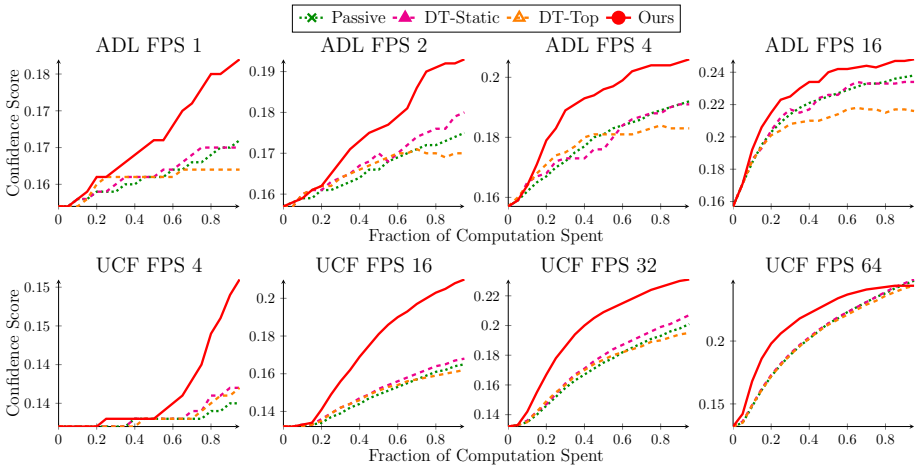

Fig. 1: Streaming recognition accuracy under different object detector speed. These plots go with the one in Figure 3 of the main text.

B Streaming Recognition Result

We show the confidence score improvement during recognition episodes with an 8 fps object detector speed in Section 4.2 of the main paper. For other object detector speeds, please refer to Figure 1. The results are consistent with that of 8 fps, where our method performs better than others under all object detector speeds, and the performance of different methods become more similar as the detector speed becomes faster. We do not show the results of 1 and 2 fps on UCF, because UCF videos are on average shorter, and for detectors that slow the recognition episodes consist of single action for videos shorter than the buffer size, making the curves meaningless.

Note the number of object N=26 for ADL and N=75 for UCF, and using object detector speed that exceed the number of object will reduce the problem to full observation of the video. Therefore, we show 32 fps and 64 fps results only for UCF.

C Un-trimmed Video Activity Detection Result

In the paper, we show AMOC under 4 fps object detector speed due to space limit. For the complete result, please refer to Figure 2 which shows AMOC under all other object detector speeds. Similar to the result in the paper, our method achieves excellent timeliness under all object detector speeds. Also, we can see more clearly how our method reduces computational cost under a high object detector speed. It uses only half of the computation on UCF under 64 fps object detector speed while remaining the best performing method.

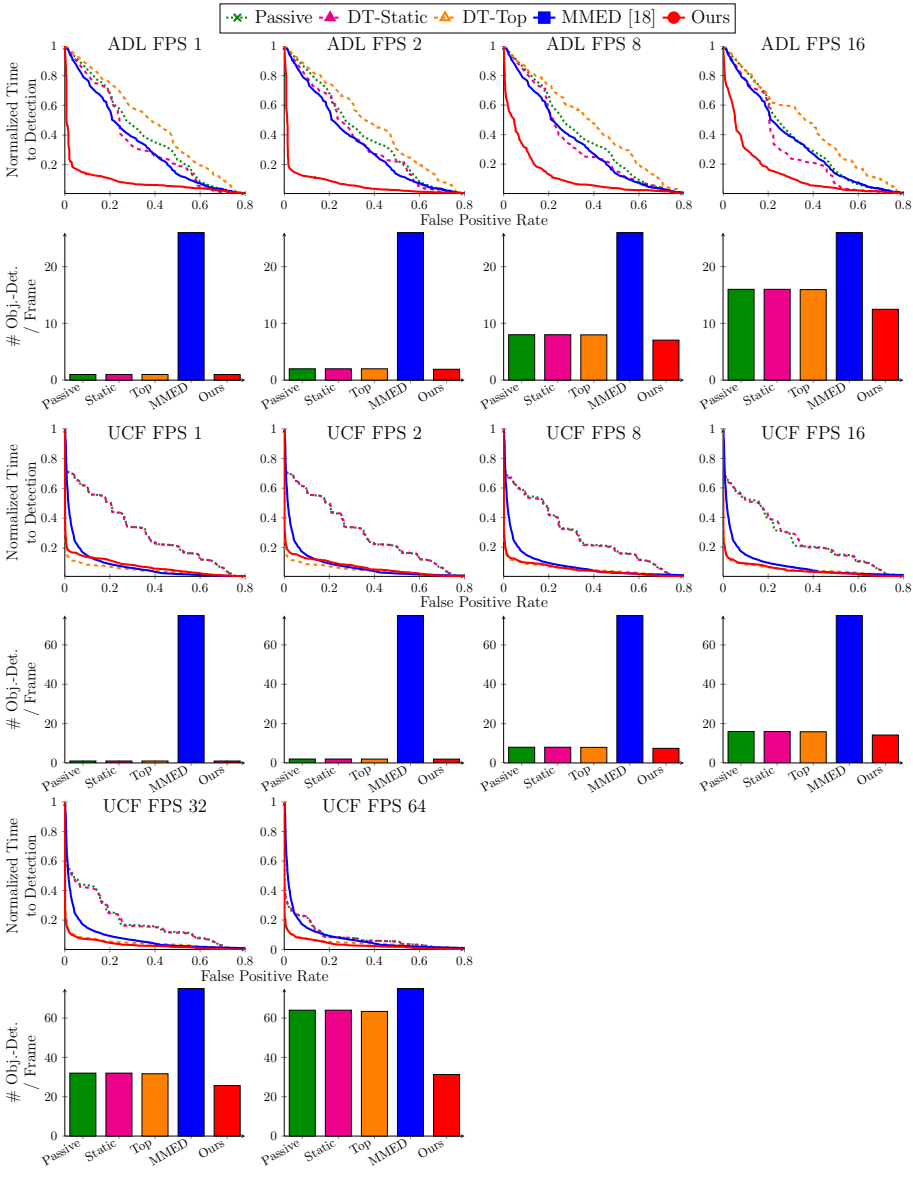

Fig. 2: AMOC under different object detector speed. These plots go with the ones in Figure 4 of the main text.

D State Feature and Reward Function Design

The intuition behind our state feature design is explained in the second paragraph of the *State-Action Features* section in the main paper. It is also motivated by [1, 2], except that the *action history* is no longer a binary indicator function as in the batch setting of [1, 2].

Our reward function design is intuitive: it requires the policy to gradually increase the predicted probability of the correct label. Although learning one specific policy for each budget constraint is a common strategy in reinforcement learning, i.e. learn a policy $\pi^{(N)}$ to perform exactly N actions before making the prediction for $N = 1 \sim \infty$, this strategy is not applicable in the streaming setting because we do not know the length of the recognition episode in advance and cannot switch between different policies since there is no guarantee that the first N steps of $\pi^{(N+1)}$ is exactly the same as $\pi^{(N)}$. Also, empirical results show that our single policy performs similarly to multiple fixed-budget policies in the batch setting.

E Policy Iteration Details

In this section, we describe the details of our policy iteration implementation. Policy iteration is an iterative algorithm that alternates between generating training samples given a policy $\pi^{(i)}$ parametrized by $\theta^{(i)}$ and learning $\theta^{(i+1)}$ given the generated training samples. We describe the steps within one iteration next.

Given the policy $\pi^{(i)}$ learned from the previous iteration, new training samples are generated by running recognition episode on all videos following $\pi^{(i)}$. For each video, the recognition episode will result in a series of three tuple $\{(a^{(k)}, \phi(s_k, a^{(k)}), r_k)\}_{k=1}^{K_j}$, where the length K_j is the number of actions performed when recognizing video v_j. Each three tuple corresponds to one action in the episode, and we collect the corresponding action, state-action-feature and reward during recognition. The target value for $Q^\pi(s, a)$ can be computed as

$$E[R|s_k, a, \pi] = \sum_{k}^{K_j} \gamma^k r_k,$$

following the equation in line 214 after finishing the recognition episode. Therefore, we can transform the three tuples into $(a_k, \phi(s_k, a^{(k)}), E[R|s_k, a^{(k)}, \pi])$. Learning $\theta^{(i+1)}$ from the three tuples becomes a regression problem

$$E[R|s_k, a^{(k)}, \pi] = \theta^T_{a^{(k)}} \phi(s_k, a^{(k)}),$$

where we solve it using ridge regression.

To improve exploration, we apply ϵ-greedy strategy in the recognition episode during data generation. In other words, we pick the action with maximum $Q(s, a)$ with probability $1 - \epsilon$ and a random action with probability ϵ. We use random policy for $\pi^{(0)}$ in the first iteration to generate samples, and we use all the samples generated during iteration $1 \sim i$ to learn $\theta^{(i+1)}$.

F Observation Imputation

In this section, we describe the details of observation imputation in batch recognition environment mentioned in Section 3.2. Let $\tilde{x} \in \mathbb{R}^M$ represent the observation results of all actions on a video, where the m-th dimension $\tilde{x}_m = x_m$ corresponds to the result of m-th action. The vector $\tilde{x}$ represents the object configuration in a video, and we learn its probability $p(\tilde{x})$ on the same data that trains the activity recognizer using a Gaussian Mixture Model:

$$p(\tilde{x}) = \sum_{i=1}^{n} w_i \mathcal{N}(\tilde{x}|\mu_i, \Sigma_i), \tag{1}$$

where we enforce a diagonal Σ_i for computational efficiency. At test time, the model can be partitioned as

$$\tilde{x} = \begin{bmatrix} \tilde{x}_u \\ \tilde{x}_p \end{bmatrix},\ \mu_i = \begin{bmatrix} \mu_{iu} \\ \mu_{ip} \end{bmatrix},\ \Sigma_i = \begin{pmatrix} \Sigma_{iu} & 0 \\ 0 & \Sigma_{ip} \end{pmatrix}, \tag{2}$$

where $\tilde{x}_p$ corresponds to the observation results of performed actions and $\tilde{x}_u$ to un-performed actions. We estimate $\tilde{x}_u$ using its expected value over the conditional probability $p(\tilde{x}_u|\tilde{x}_p)$, i.e.

$$\langle \tilde{x}_u \rangle = \sum_{i=1}^{n} w_i' \mu_{ip}, \tag{3}$$

where

$$w_i' = \frac{w_i \mathcal{N}(\tilde{x}_p|\mu_{ip}, \Sigma_{ip})}{\sum_i w_i \mathcal{N}(\tilde{x}_p|\mu_{ip}, \Sigma_{ip})}. \tag{4}$$

G Implementation Details

We run 8 iterations of policy iteration, with γ=0.4. We initialize ϵ=0.5 for ϵ-greedy exploration, and decrease by 0.1 each iteration with lower bound 0.05. For the streaming case, we use the video framerate inherited from ADL (1 fps), and evaluate over a range of object detector framerates. We fix the buffer size to half the median clip length, 25 seconds. We set the window size upper bound β to one-third of the number of object categories to avoid the model observing all objects within the window. For all methods, we initialize $\Psi(X)$ with features computed in the first frame in the streaming case.

H Generate Un-trimmed Video

We concatenate trimmed video clips to simulate an un-trimmed video stream following [3, 4]. Although concatenation may introduce discontinuity in content, it resembles scenarios in real videos. For example, it is similar to the video

where the recorder walks from one room to another and starts the next activity. We concatenate five trimmed video clips for one un-trimmed video. For each positive clip, we generate five un-trimmed videos by placing the positive clip in different temporal location and drawing four negative clips for other locations randomly. We sort the categories by their trimmed full observation results, and take the top 8 for untrimmed experiments. For all experiments with trimmed data (streaming/batch), we use the datasets as-is and test all 18 (ADL) and 101 (UCF) activity categories.

References

1. Karayev, S., Fritz, M., Darrell, T.: Anytime recognition of objects and scenes. In: CVPR. (2014)
2. Karayev, S., Baumgartner, T., Fritz, M., Darrell, T.: Timely object recognition. In: NIPS. (2012)
3. Hoai, M., la Torre, F.D.: Max-margin early event detectors. In: CVPR. (2012)
4. Chen, C.Y., Grauman, K.: Efficient activity detection with max-subgraph search. In: CVPR. (2012)