Learning Policies for Adaptive Tracking with Deep Feature Cascades Supplementary Material

Chen Huang Simon Lucey Deva Ramanan
Robotics Institute, Carnegie Mellon University
chenh2@andrew.cmu.edu {slucey,deva}@cs.cmu.edu

1. Algorithmic Details

Network architecture: We use the exact convolutional architecture of SiamFC [1]. The convolutional layers $C1-C5$ and their parameter details are given in Table 1. Note max-pooling is employed for the convolutional layers $C1$ and $C2$. We use the nonlinear ReLU function [4] after every convolutional layer except for $C5$. Batch normalization [3] is inserted after every linear layer.

Deep Q-learning: During deep Q-learning [6], the optimal action-value function $Q(S_l, A_l)$ obeys the Bellman equation: it is optimal to select the action A' that maximizes the expected reward

$$Q(S_l, A_l) = R + \gamma \max_{A'} Q(S', A'), \tag{1}$$

where $Q(S', A')$ is the future reward and γ the discount factor. Since the action-value function is approximated by a Q-Net with weights θ, the Q-Net can be trained by minimizing the loss function $V(\theta_l)$ at each iteration l,

$$V(\theta_l) = \mathbb{E}\left[\left(R + \gamma \max_{A'} Q(S', A'; \theta_{l-1}) - Q(S_l, A_l; \theta_l)\right)^2\right]. \tag{2}$$

The gradient of this loss function with respect to the network weights θ_l is as follows:

$$\nabla_{\theta_l} V(\theta_l) = \mathbb{E}\left[\left(R + \gamma \max_{A'} Q(S', A'; \theta_{l-1}) - Q(S_l, A_l; \theta_l)\right) \cdot \nabla_{\theta_l} Q(S_l, A_l; \theta_l)\right]. \tag{3}$$

2. Discussions and Results

The main idea of our EArly-Stopping Tracker (EAST) is to track easy frames using only early layers of a deep feature cascade, *e.g.* pixel values, while hard frames are processed with invariant but expensive deep layers when needed.

An attached video **demo.mp4** exemplifies such tracking policies in video sequences. To further validate the advantages of EAST in both accuracy and speed, we compare with the top 3 trackers on OTB-50 [8] in terms of speed, and AUC score for One-Pass Evaluation (OPE), Temporal Robustness Evaluation (TRE) and Spatial Robustness Evaluation (SRE). Table 2 shows that EAST achieves the highest scores under all evaluation metrics, while maintaining fast tracking speed.

Table 3 shows the detailed ranks of accuracy R_A and ro-

Table 1. Network architecture and convolutional layer specifics.

Layer		Support	Stride	Template activation	Search activation	Chans.
	Input			127×127	255×255	3
	conv1	11×11	2	59×59	123×123	96
C1	pool1	3×3	2	29×29	61×61	96
	conv2	5×5	1	25×25	57×57	256
C2	pool2	3×3	2	12×12	28×28	256
C3	conv3	3×3	1	10×10	26×26	192
C4	conv4	3×3	1	8×8	24×24	192
C5	conv5	3×3	1	6×6	22×22	128

Table 2. The Area Under the Curve (AUC) score for One-Pass Evaluation (OPE), Temporal Robustness Evaluation (TRE) and Spatial Robustness Evaluation (SRE), and speed (fps, * indicates GPU speed, otherwise CPU speed) on the OTB-50 dataset. The best results are shown in bold.

Method	LCT [5]	SiamFC [1]	SINT [7]	EAST
AUC-OPE	0.612	0.612	0.625	**0.638**
AUC-TRE	0.594	0.621	0.643	**0.662**
AUC-SRE	0.518	0.554	0.579	**0.591**
Speed	**27**	86*	4*	23/**159***

Table 3. The accuracy R_A, robustness R_R and average R ranks under baseline and region noise experiments in VOT-14. R_o is the overall (averaged) ranking for both experiments, which is used to rank the 38 trackers in the main paper. Our CPU/GPU speeds are reported in fps, while the speeds for the top 3 trackers are in EFO units, which roughly correspond to fps (*e.g.* the speed of the NCC baseline is 140 fps and 160 EFO).

Tracker	baseline			region noise				
	R_A	R_R	R	R_A	R_R	R	R_o	Speed
EAST	**4.95**	**5.42**	**5.19**	**5.11**	**4.73**	**4.92**	**5.06**	**22/155**
DSST	5.41	11.93	8.67	5.40	12.33	8.86	8.77	7.66
SAMF	5.30	13.55	9.43	5.24	12.30	8.77	9.10	1.69
KCF	5.05	14.60	9.82	5.17	12.49	8.83	9.33	24.23

bustness R_R under baseline and region noise experiments in VOT-14 challenge. The two experiments evaluate trackers with the initial target location from ground truth and that perturbed with random noises. The table also lists the overall rank R_o and running speed to compare EAST with the best 3 trackers out of 38 submitted ones. It is evident that EAST is one of the fastest trackers, while outperforming other top performers in the overall rank.

2.1. Template Update

It is worth noting that, in our feature cascade we explore the pixel and HOG layers before expensive deep layers. When processing the cheap pixel and HOG layers, we make use of fast correlation filters [2]. A correlation filter w with the same size of image patch x is learned by solving the Ridge Regression loss function

$$\min_{w} \sum_{i} \left(w^T x_i - y_i\right)^2 + \lambda \|w\|^2, \tag{4}$$

where y_i is the target response value, and λ is the regularization parameter.

Solving this loss function is fast due to the efficient use of all shifted patches x_i by exploiting the discrete Fourier transform. Besides fast speed, the correlation filter has another benefit of updating the template w over time. However, this adaptive merit is not preserved for deep layers. Recall that for the deep convolutional layers $C1 - C5$, we follow SiamFC [1] to compute the similarity of a template image z to all translated regions in search image x by

$$F_l(z, x) = \varphi_l(z) * \varphi_l(x) + v\mathbb{I}, \tag{5}$$

where φ_l is the convolutional feature embedding at layer l, and $v \in \mathbb{R}$ is an offset value.

Here $\varphi_l(z)$ can be treated as the convolutional template to compute the target responses, but is fixed to $\varphi_l(z_{t=1})$ from the first frame and is never updated during tracking. Then a question naturally arises: can we improve the performance by updating the template for deep layers?

To this end, we conduct the following experiment on OTB-50: we simply update $\varphi_l(z_t)$ as $\varphi_l(z_{t-1})$ from the previous frame, and record the accuracy if we separately update the convolutional layer l from $C1-C5$. Fig.1 shows the AUC score when we update the template for each deep layer. Marginal gains are obtained on lower layers $C1-C2$, suggesting that they are less invariant and so would need to be updated more often. On the other hand, updating the top layer $C5$ leads to no difference, which is actually in line with the observations by SiamFC [1] that always uses this invariant top layer for tracking. In the future, we can consider how to *learn* to update template online rather than just use the previous frame. Another promising direction is to further speedup the deep convolutional process by adopting the Fourier transform techniques.

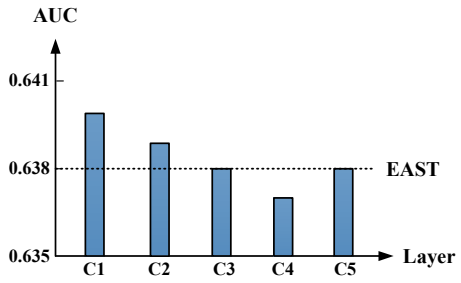

Figure 1. The Area Under the Curve (AUC) score for One-Pass Evaluation (OPE) of template update for deep layers $C1 - C5$.

References

[1] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, 2016.

[2] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *TPAMI*, 37(3):583–596, 2015.

[3] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[5] C. Ma, X. Yang, C. Zhang, and M.-H. Yang. Long-term correlation tracking. In *CVPR*, 2015.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[7] R. Tao, E. Gavves, and A. W. M. Smeulders. Siamese instance search for tracking. In *CVPR*, 2016.

[8] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013.