Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning

Presentation by:

Naji Khosravan

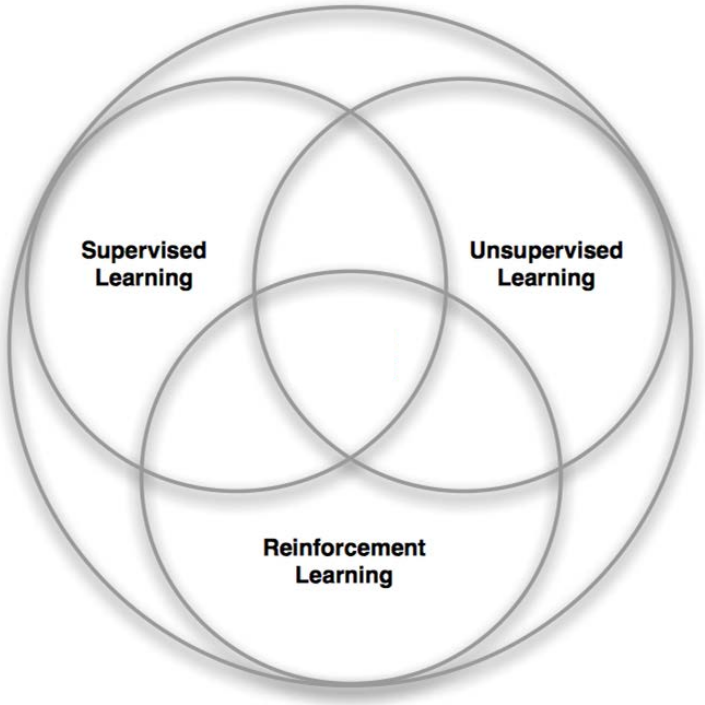
Outline

- Background
  - Different types of learning
  - Deep learning in a nutshell
  - Reinforcement learning in a nutshell
    - Policy
    - Value function
    - Model
    - Approaches to Reinforcement Learning
    - Deep Reinforcement Learning
- Proposed method
  - Action-driven object tracking
  - Problem definition (RL setting)
  - Training:
    - Supervised learning
    - Reinforcement learning
    - Online adaptation
- Results

Background

Different types of learning

- Supervised learning:
  - Labeled data.
  - Learning based on input-output pairs.
- Unsupervised learning:
  - Unlabeled data.
  - Learning based on input data similarity.
- Reinforcement learning:
  - An interactive process.
  - Learning based on states, actions and rewards.

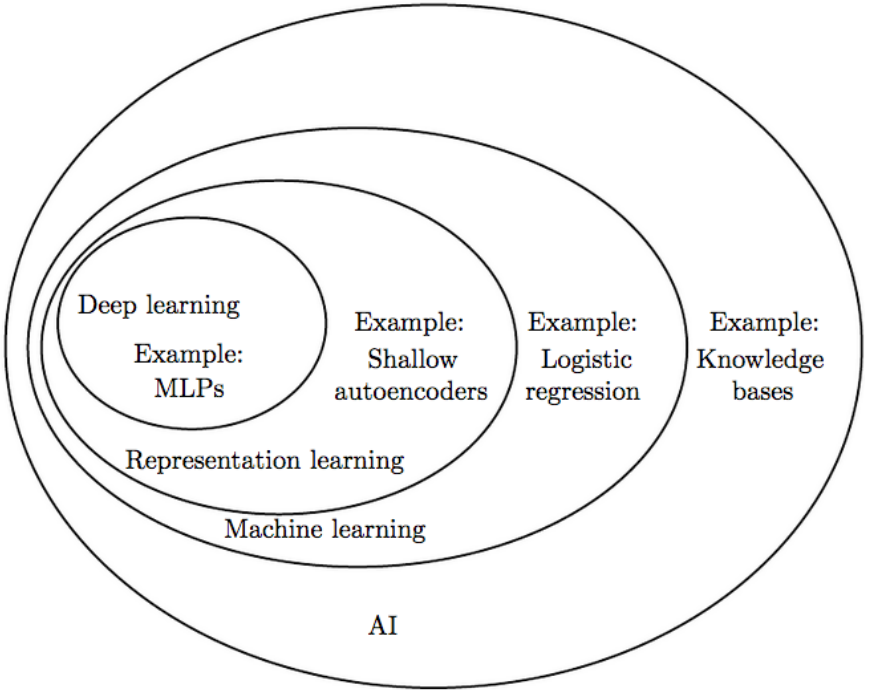


Machine learning

Deep learning in a nutshell

DL is a general-purpose framework for representation learning.

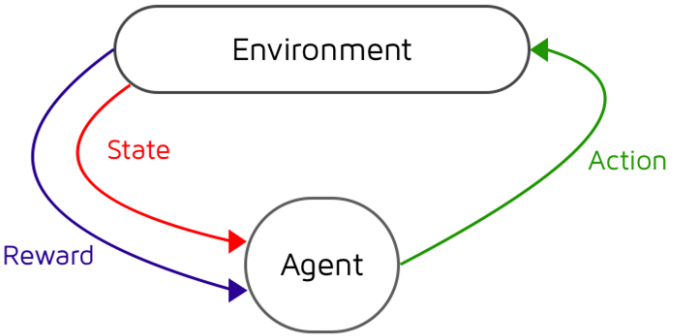
- Given an objective
- Learn representation that is required to achieve objective
- Directly from raw inputs



Reinforcement learning in a nutshell

RL is a general-purpose framework for decision-making

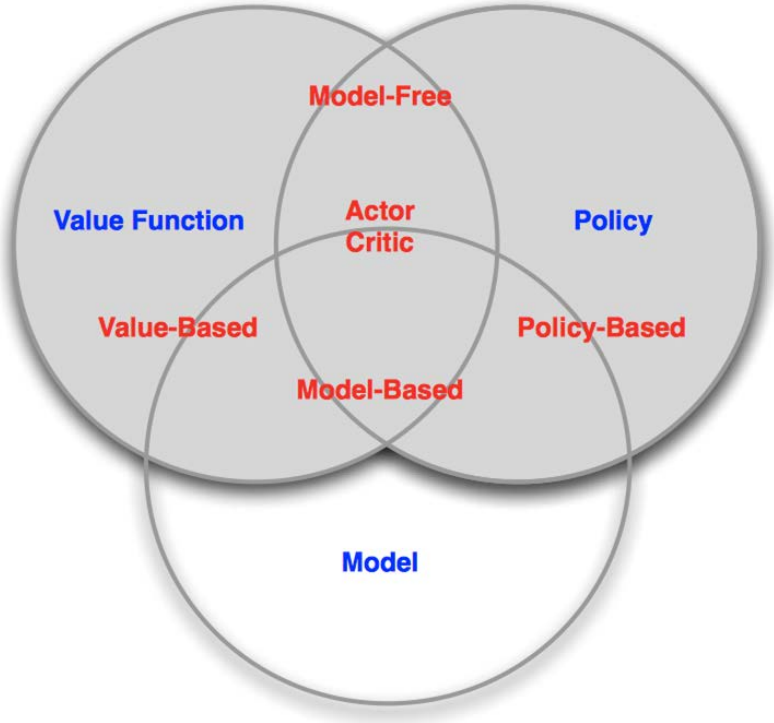
- RL is for an agent with the capacity to act
- Each action influences the agent's future state
- Success is measured by a scalar reward signal
- Goal: select actions to maximize future reward



Reinforcement learning in a nutshell

An RL agent may include one or more of these components:

- Policy: agent's behaviour function
- Value function: how good is each state and/or action
- Model: agent's representation of the environment



Policy

A policy is the agent's behaviour

- It is a map from state to action:
  - Deterministic policy: $a = \pi(s)$
  - Stochastic policy: $\pi(a|s) = P[a|s]$

Value function

A value function is a prediction of future reward

- "How much **reward** will I get from action **a** in state **s**?"

Q-value function gives expected total reward

- From state **s** and action **a** under policy **π** with discount factor **γ**

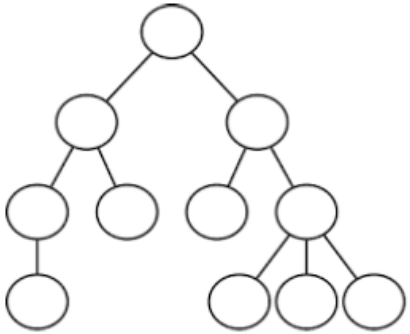
$$Q^{\pi}(s, a) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s, a\right]$$

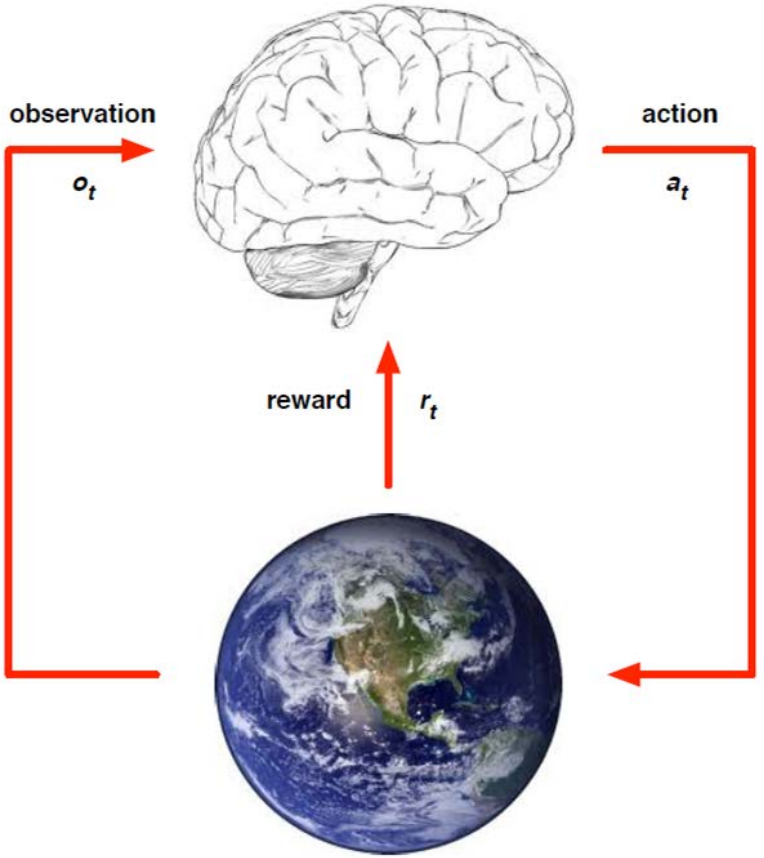
*Image from David Silver tutorial on DRL

Model

Model is learnt from experience

- Acts as proxy for environment
- Planner interacts with model
  - e.g. using lookahead search





Approaches To Reinforcement Learning

Value-based RL

- Estimate the optimal value function
- This is the maximum value achievable under any policy

Policy-based RL

- Search directly for the optimal policy
- This is the policy achieving maximum future reward

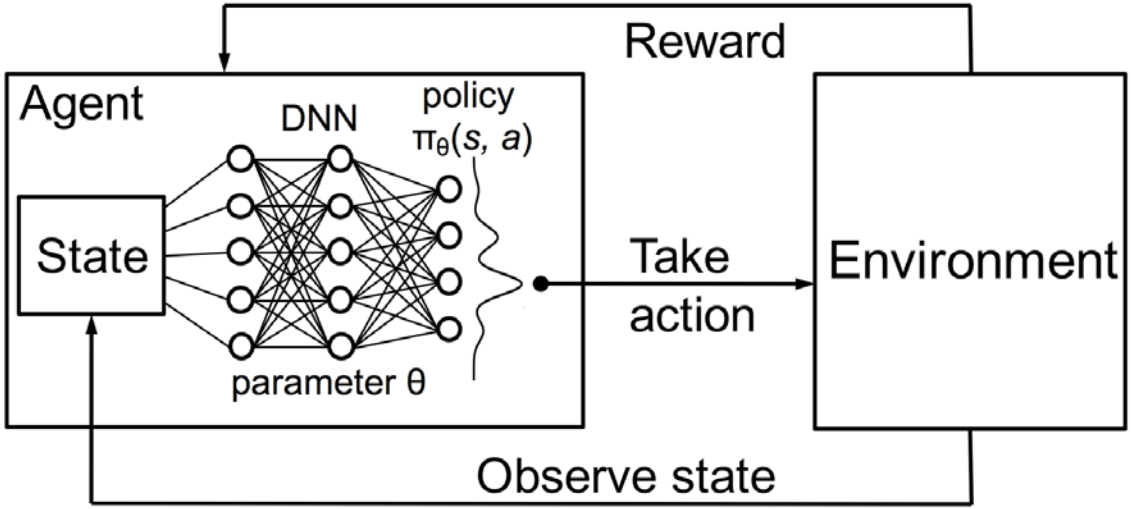
Model-based RL

- Build a model of the environment
- Plan (e.g. by lookahead) using model

Deep RL in a nutshell

RL + DL

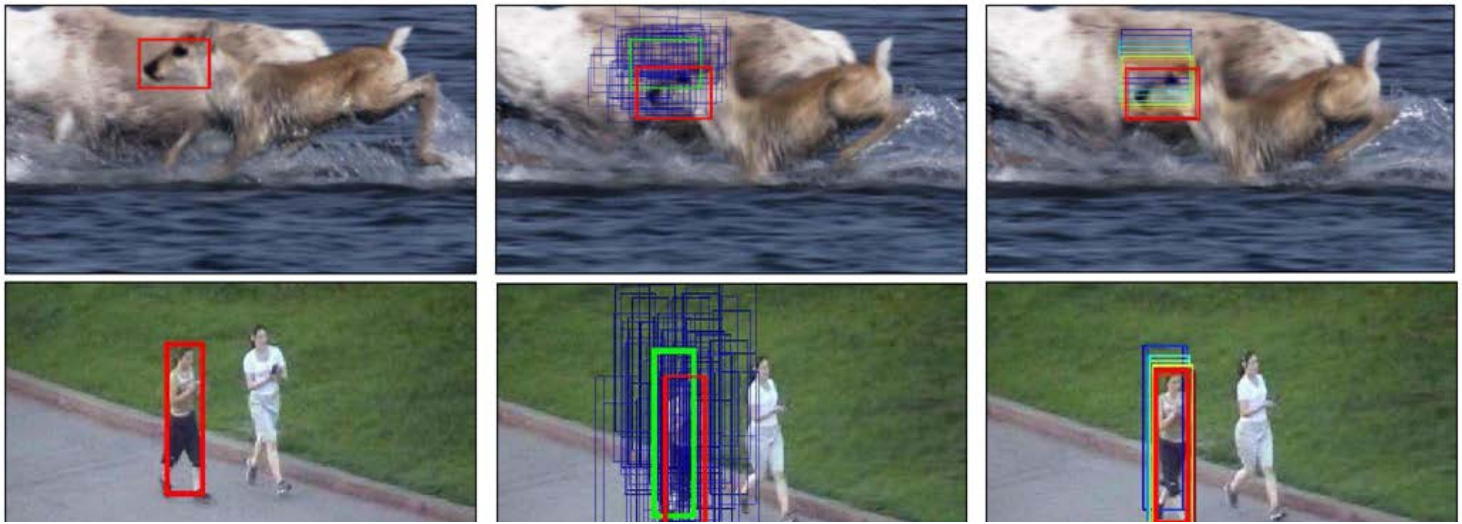
- RL defines the objective
- DL gives the mechanism



Proposed method

Motivation

Efficiency in search space.



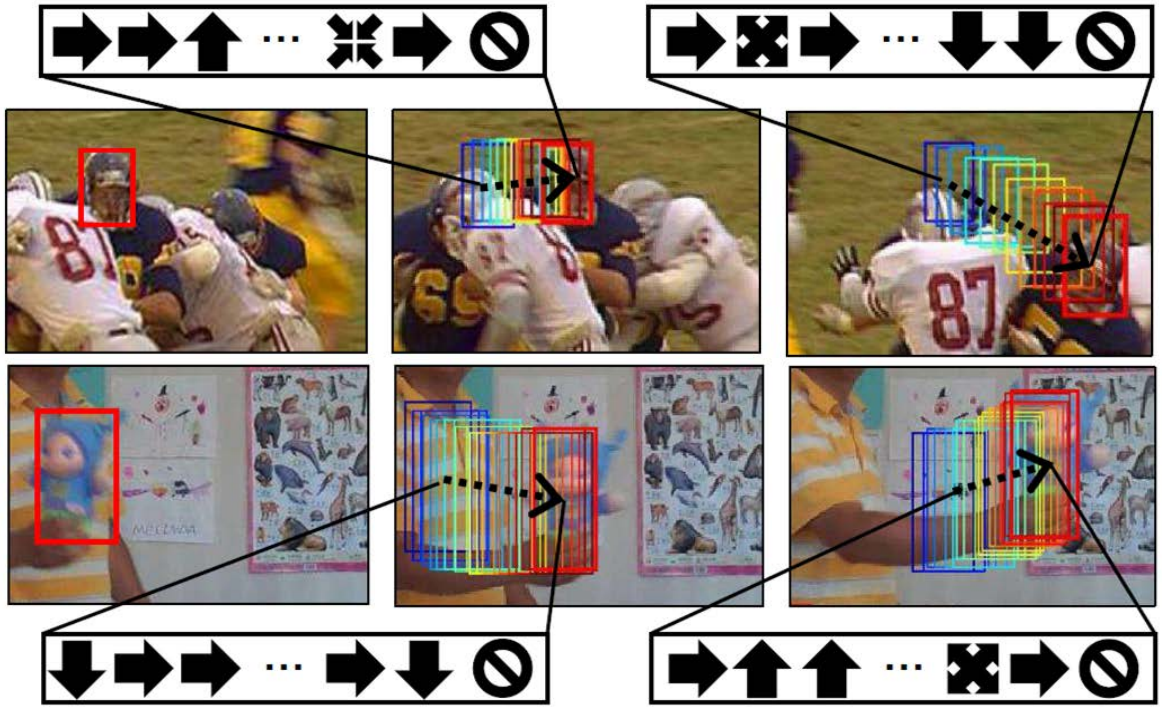
(a) Previous frame

(b) Tracking-by-Detection method

(c) Our method

Action-driven object tracking

Dynamically track the target by selecting sequential actions.



Problem definition (RL setting)

Action: A set of 11 discrete actions:

- **Translation moves**
  - 4 directional moves, *{left, right, up, down}* and also have their two times larger moves.
- **Scale changes**
  - *{scale up, scale down}* which maintains the aspect ratio of the tracking target
- **Stop**



Problem definition (RL setting)

State: The state st is defined as a tuple (p_t, d_t)

- **ϕ** denotes the pre-processing function which **crops** the patch pt from *F*.

Image patch: $p_t \in \mathbb{R}^{112*112*3}$

Action dynamics: $d_t \in \mathbb{R}^{110}$

$$p_t = \phi(b_t, F)$$

$$b_t = [x^{(t)}, y^{(t)}, w^{(t)}, h^{(t)}]$$

State transition:

Where **α = 0.03**

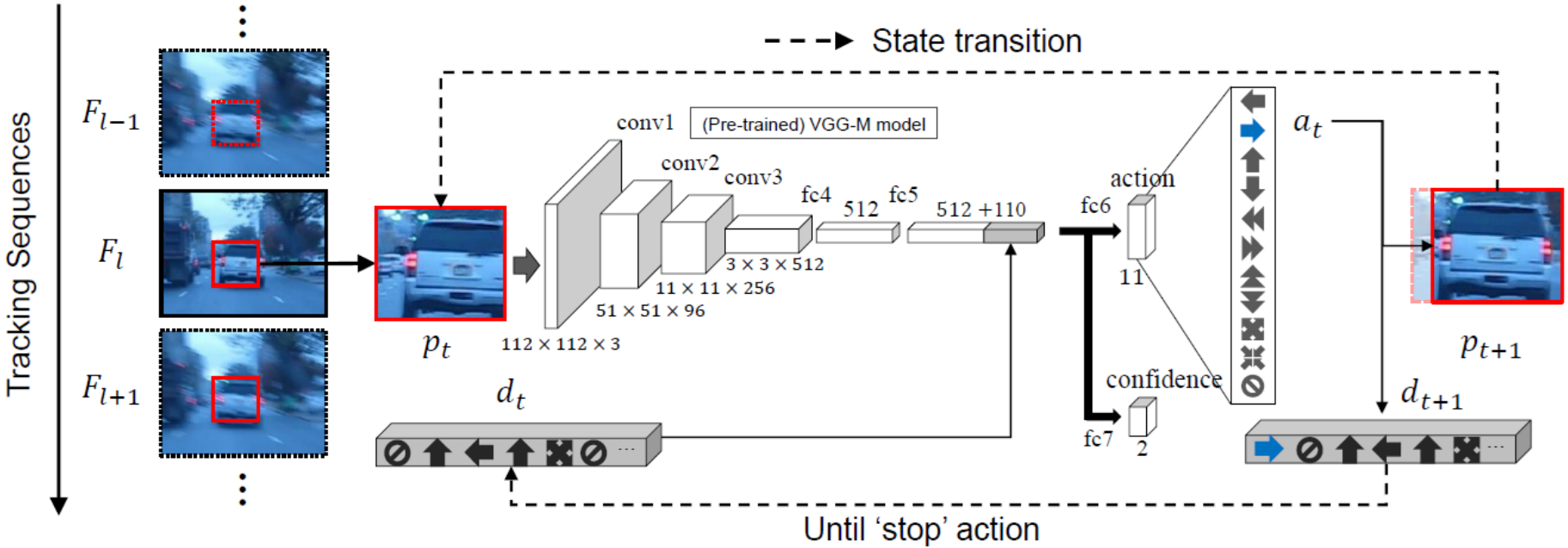
$$\Delta x^{(t)} = \alpha w^{(t)} \quad \text{and} \quad \Delta y^{(t)} = \alpha h^{(t)}$$

Problem definition (RL setting)

Reward: $IoU(b_T, G)$ denotes overlap ratio of the terminal patch position b_T and the ground truth G of the target with intersection-over-union criterion.

$$r(s_t) = \begin{cases} 1, \text{if } IOU(p_t, G) > 0.7 \\ -1, \text{otherwise} \end{cases}$$

Action-decision network

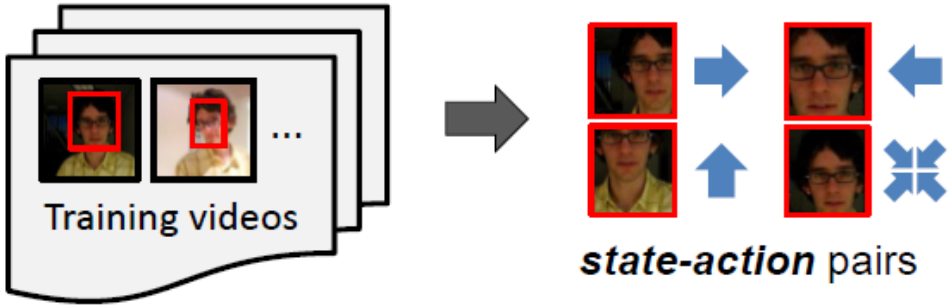


Training: Supervised learning

Generate state-action pairs.

$$o_j^{(act)} = \arg\max_a IoU(\bar{f}(p_j, a), G)$$

$$o_j^{(cls)} = \begin{cases} 1, & \text{if } IoU(p_j, G) > 0.7 \\ 0, & \text{otherwise.} \end{cases}$$



Train policy network as multiclass classification with softmax. (**L = cross-entropy**)

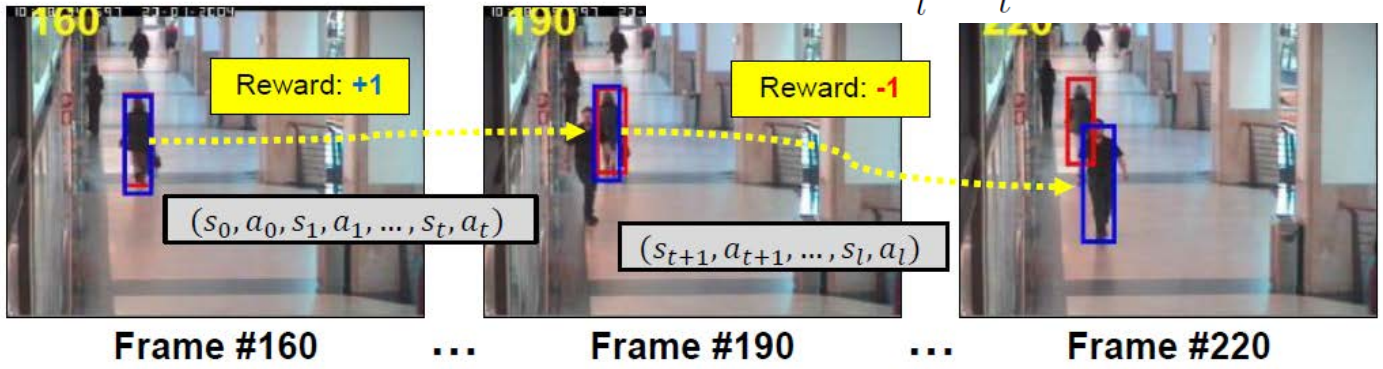
$$L_{SL} = \frac{1}{m}\sum_{j=1}^{m} L(o_j{}^{(act)}, \hat{o_j}{}^{(act)}) + \frac{1}{m}\sum_{i=j}^{m} L(o_j{}^{(cls)}, \hat{o_j}{}^{(cls)})$$

Training: Reinforcement learning

Training ADNet with RL in this section aims to improve the network by policy gradient approach.

The action a_t, for the state s_t, is assigned by: $$a_{t,l} = arg\max_{a} p(a|s_{t,l}; W_{RL})$$

Network weights are updated by: $$\Delta W_{RL} \propto \sum_{l}^{L} \sum_{t}^{T_l} \frac{\partial \log p(a_{t,l}|s_{t,l}; W_{RL})}{\partial W_{RL}} z_{t,l}$$



Training: Online adaptation

Training ADNet in supervised manner using generated samples during tracking.

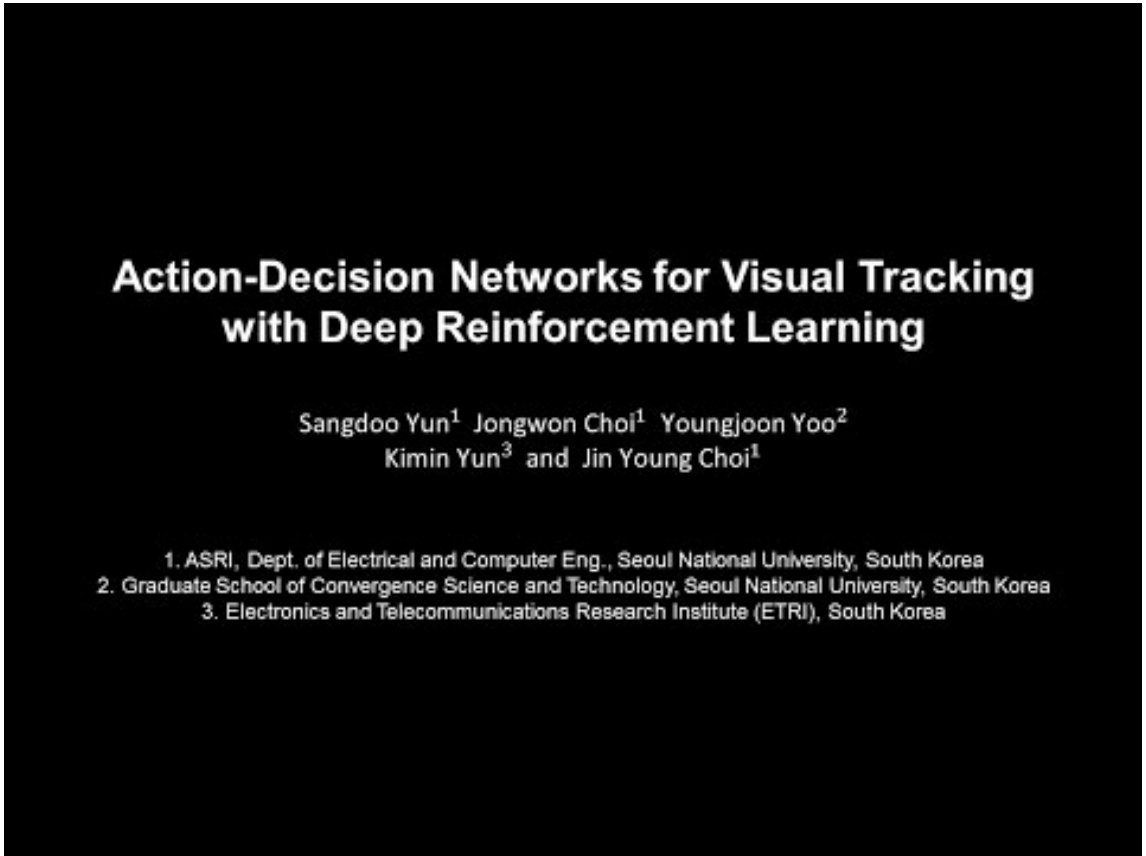
Improves robustness to appearance changes.

Data generation for this step:

- Tracked patch is assumed to be G_T
- Random patches around it to be used for supervised training
- Redetection is performed using random patches around current detected patch:
  (C is class prob.)

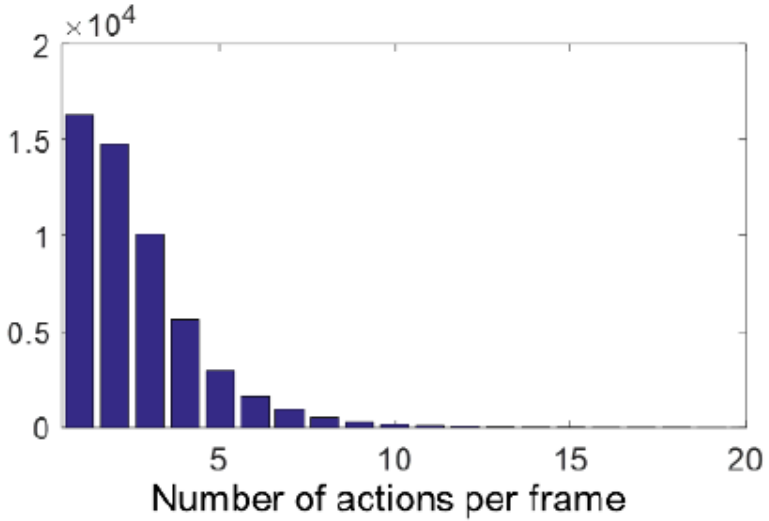
$$b^* = arg\max_{\tilde{b}_i} c(\tilde{b}_i)$$

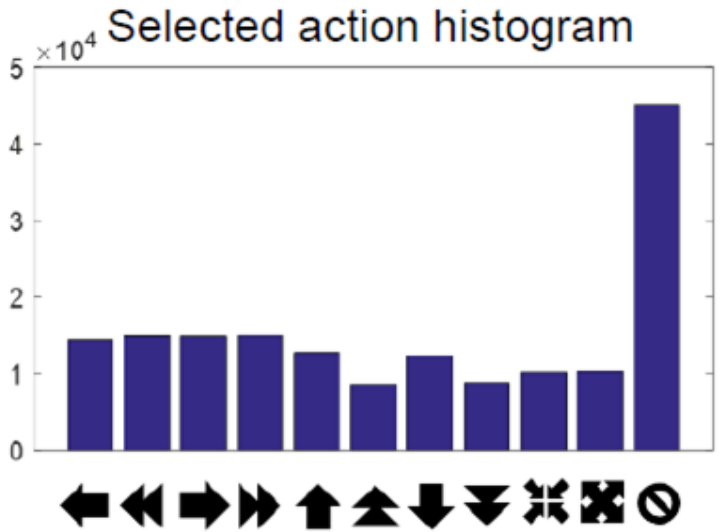
Results


Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning
Sangdoo Yun[1] Jongwon Choi[1] Youngjoon Yoo[2]
Kimin Yun[3] and Jin Young Choi[1]
1. ASRI, Dept. of Electrical and Computer Eng., Seoul National University, South Korea
2. Graduate School of Convergence Science and Technology, Seoul National University, South Korea
3. Electronics and Telecommunications Research Institute (ETRI), South Korea

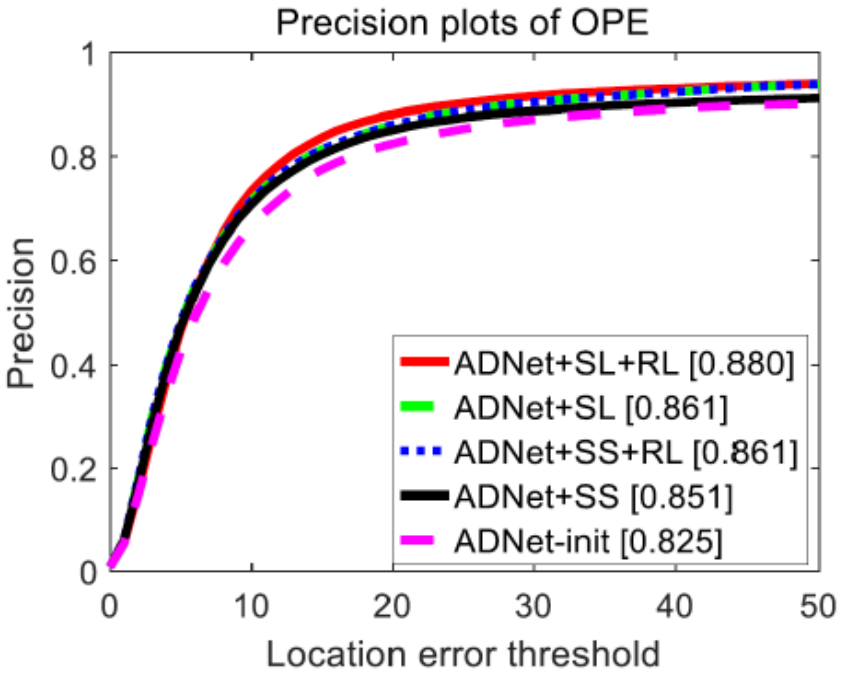
Results

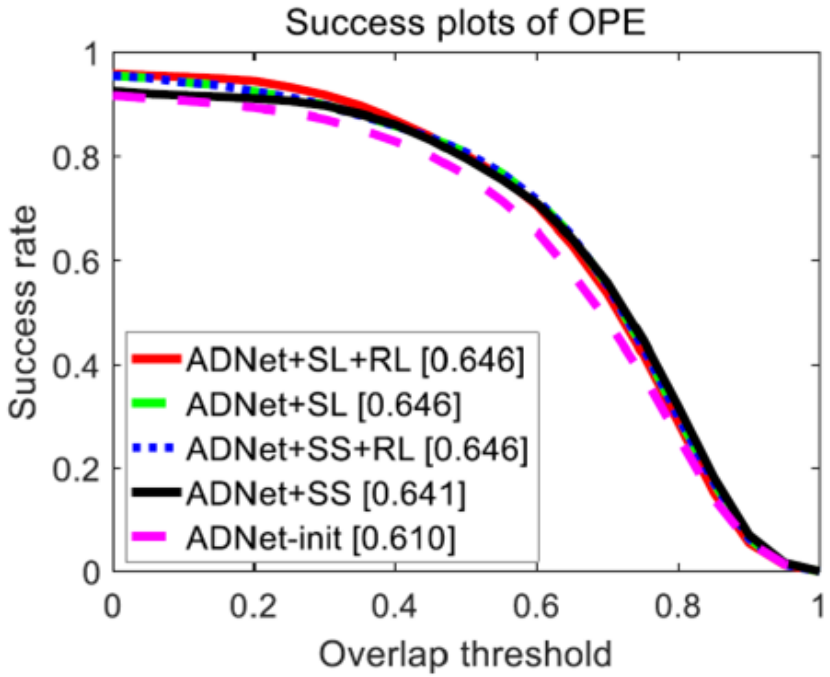
Analysis on actions.



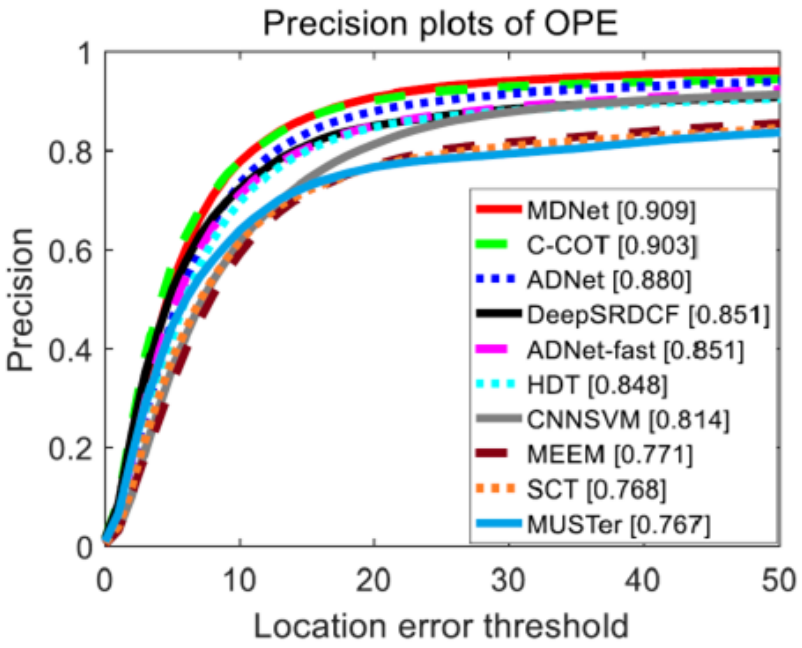


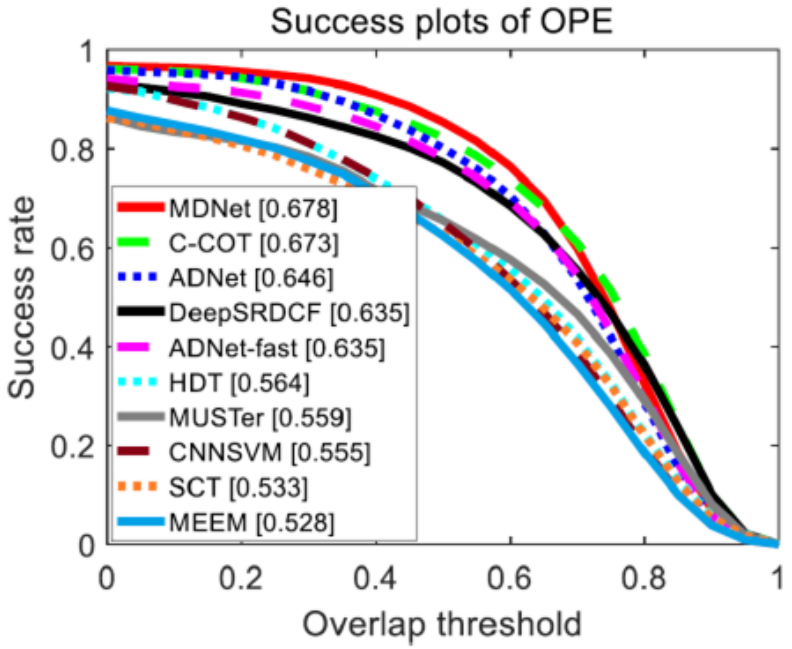
Self comparison





OTB-100 test results





OTB-100 test results

	Algorithm	Prec.(20px)	IOU(AUC)	FPS	GPU
	ADNet	88.0%	0.646	2.9	O
	ADNet-fast	85.1%	0.635	15.0	O
Non real-time	MDNet [24]	90.9%	0.678	< 1	O
	C-COT [9]	90.3%	0.673	< 1	O
	DeepSRDCF [8]	85.1%	0.635	< 1	O
	HDT [25]	84.8%	0.564	5.8	O
	MUSTer [15]	76.7%	0.528	3.9	X
Real-time	MEEM [42]	77.1%	0.528	19.5	X
	SCT [5]	76.8%	0.533	40.0	X
	KCF [13]	69.7%	0.479	223	X
	DSST [7]	69.3%	0.520	25.4	X
	GOTURN [12]	56.5%	0.425	125	O

Thank you !
Question and discussion.