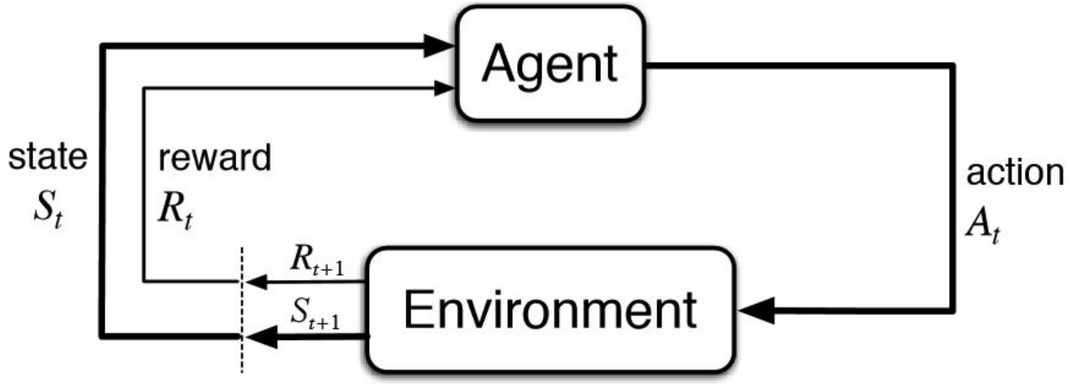
Active Object Localization with Deep Reinforcement Learning

Juan C.Caicedo, Svetlana Lazebnik

Presenter: Chongruo Wu

Reinforcement Learning

Problem involving an agent interacting with an environment, which provides reward signals

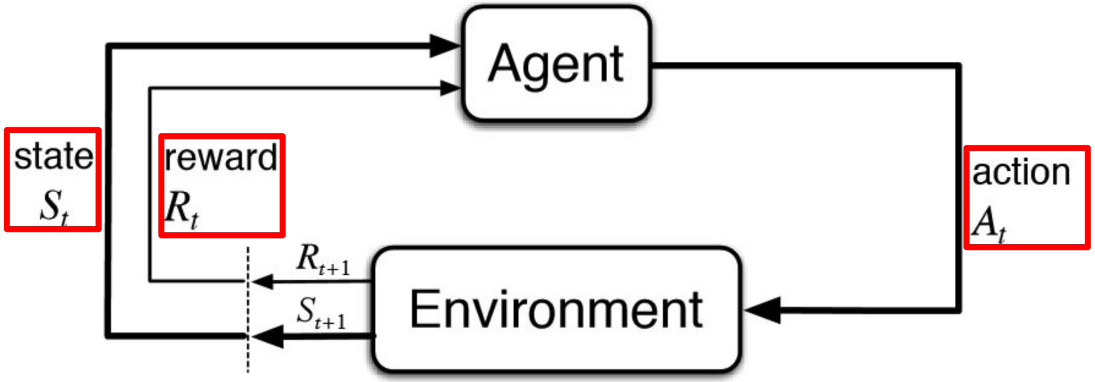


Goal: Learn how to take actions in order to maximize total future rewards

Stanford cs231n.

Reinforcement Learning

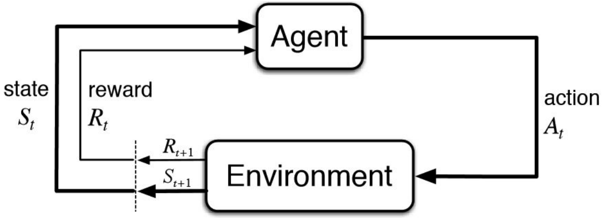
Problem involving an agent interacting with an environment, which provides reward signals



Goal: Learn how to take actions in order to maximize total future reward

Stanford cs231n.

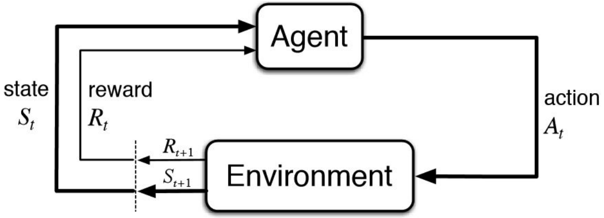
Reinforcement Learning

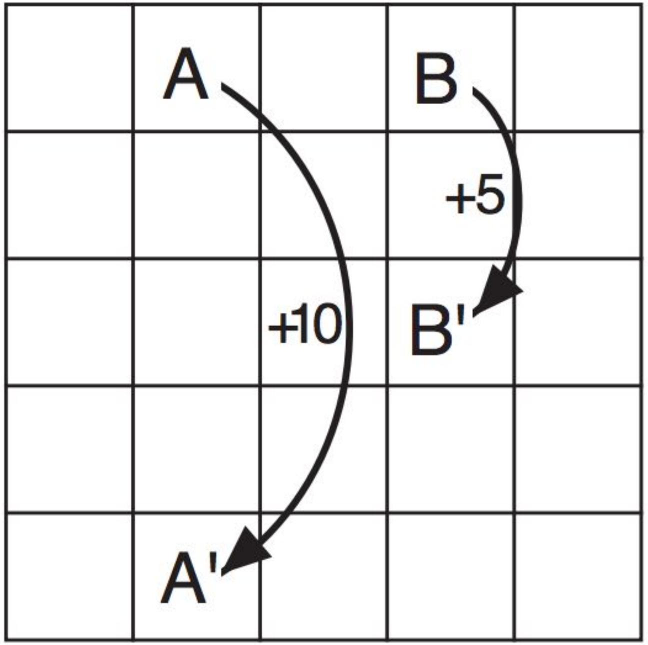


What makes reinforcement learning different from other machine learning paradigms?

- There is no supervisor, only a *reward* signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data) ⟸ Sequential Decision Making
- Agent's actions affect the subsequent data it receives

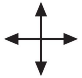
Example





Agent: robot
Environment: grid world

State: each grid

Actions: 

Reward:

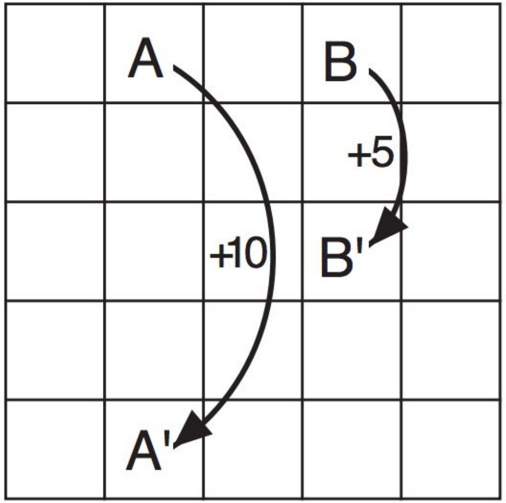
- -1 for any transition
- +10 from A to A’
- +5 from B to B’

Problem:

find the **policy (state -> action)** to maximum the total reward

Example

State Value, V(s): - Estimation of total future reward if starting from state s
- How good it is for the agent to be in state s



Example

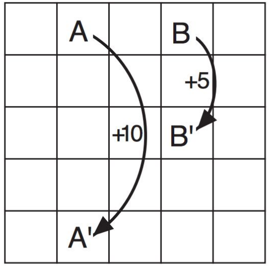
State Value, V(s): - Estimation of total future reward if starting from state s
- How good it is for the agent to be in state s

Future Rewards, G_t

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

discount rate, [0,1]

Example



State Value, V(s): - Estimation of total future reward if starting from state s
- How good it is for the agent to be in state s

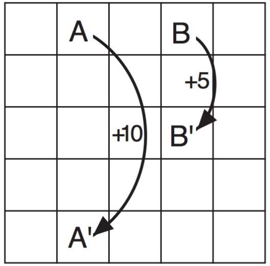
Future Rewards, G_t

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

Bellman Equation

$$v_*(s) = \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a]$$

Example



State Value, V(s): - Estimation of total future reward if starting from state s
- How good it is for the agent to be in state s

Future rewards, G_t

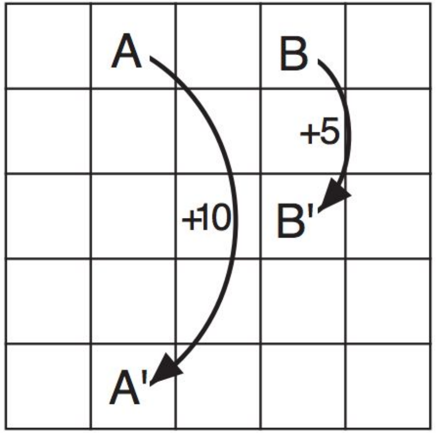
$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

Bellman Equation

$$\begin{aligned} v_*(s) &= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\ &= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r \mid s, a) \big[r + \gamma v_*(s') \big]. \end{aligned}$$

Example

State Value, V(s): - Estimation of total future reward if starting from state s
- How good it is for the agent to be in state s

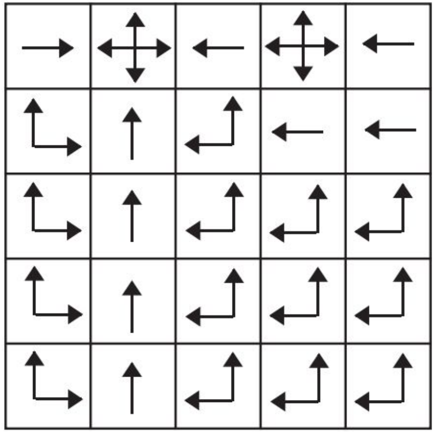


Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

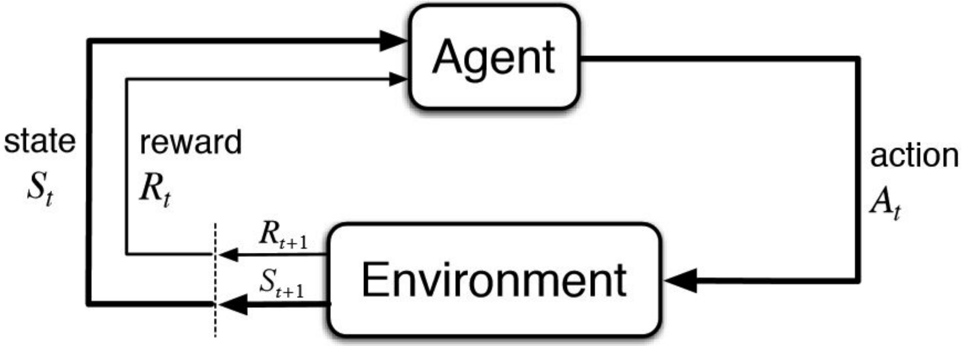
v_*

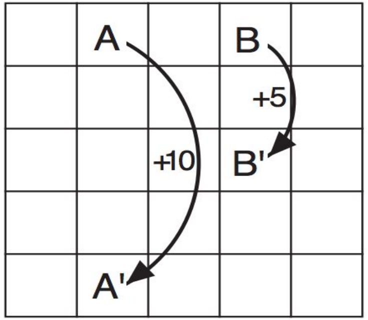
State Value



π_*

Optimal Policy
(state -> action)



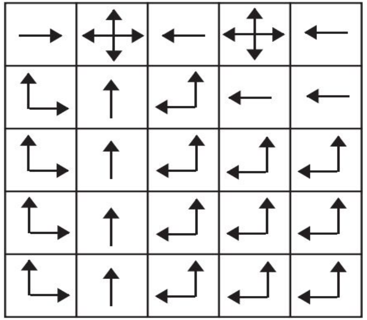


Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*

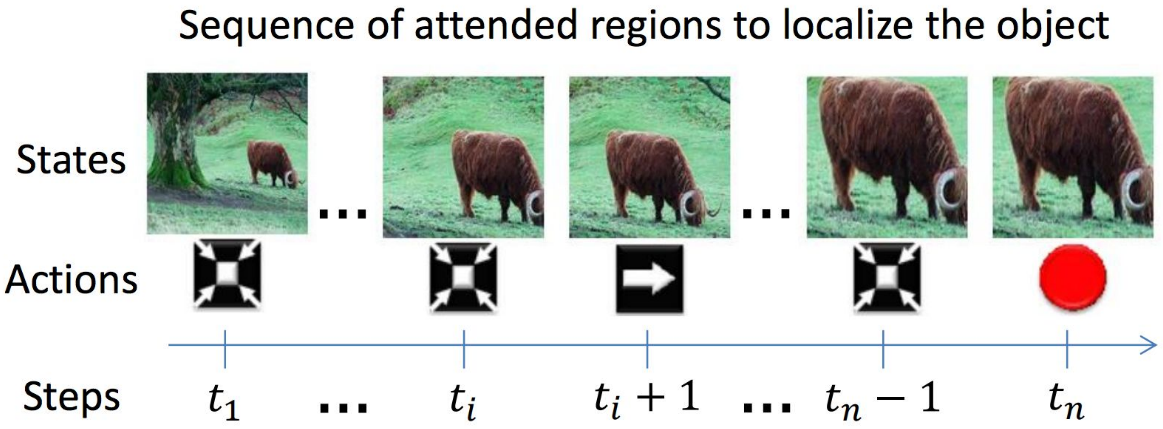
State Value

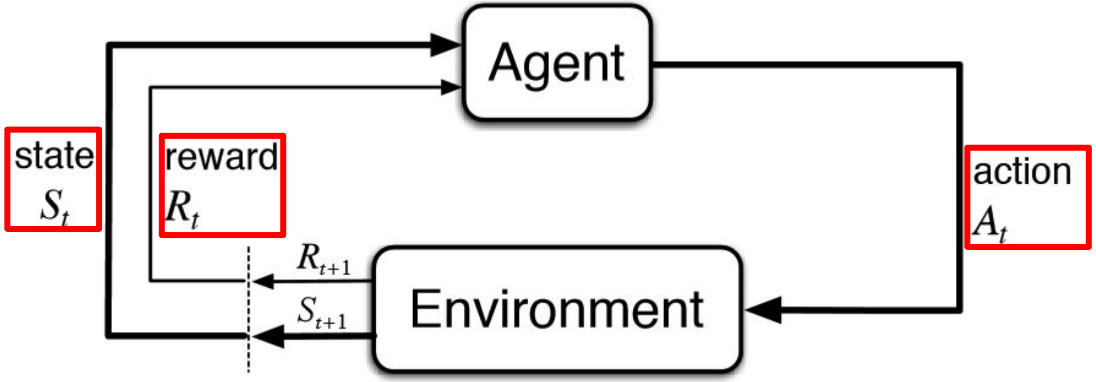


π_*

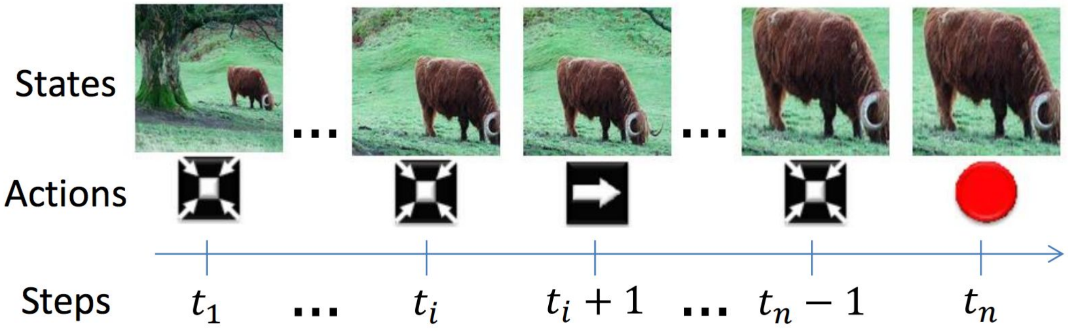
Optimal Policy

Active Object Localization

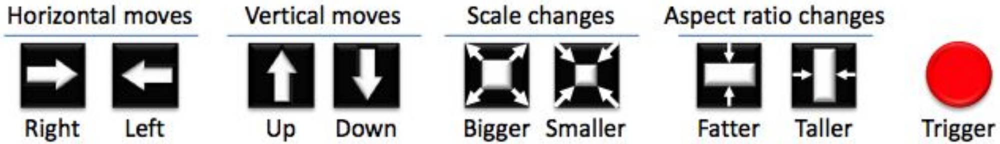




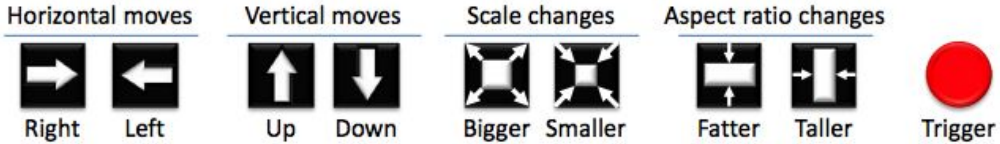
Sequence of attended regions to localize the object



Actions


Horizontal moves
Vertical moves
Scale changes
Aspect ratio changes
Right
Left
Up
Down
Bigger
Smaller
Fatter
Taller
Trigger

Actions



Bounding Box

$$b = [x_1, y_1, x_2, y_2].$$

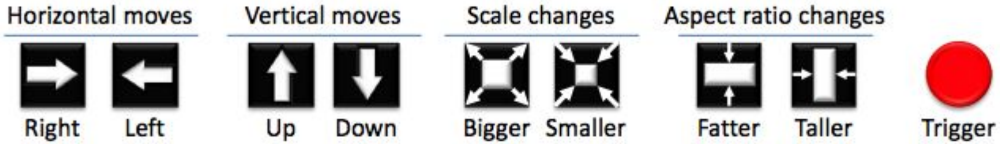
Changes:

0.2

$$\alpha_w = \alpha * (x_2 - x_1) \qquad \alpha_h = \alpha * (y_2 - y_1)$$

a good trade-off between speed and localization accuracy.

Actions



Bounding Box

$$b = [x_1, y_1, x_2, y_2].$$

Changes:

0.2

$$\alpha_w = \alpha * (x_2 - x_1) \qquad \alpha_h = \alpha * (y_2 - y_1)$$

a good trade-off between speed and localization accuracy.

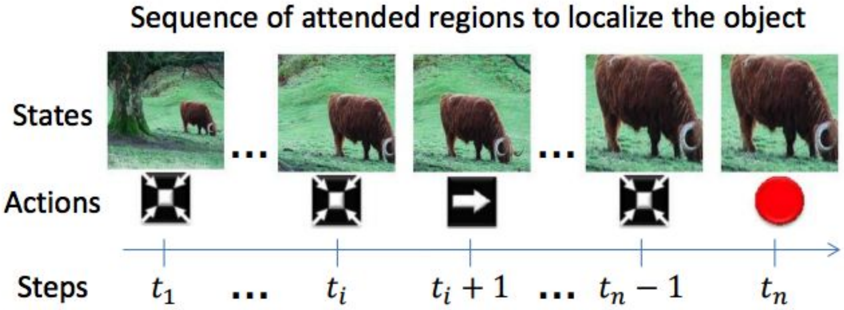
Examples:

Left: $[x_1 - \alpha_w,\ y_1,\ x_2 - \alpha_w,\ y_2]$

Bigger: $[x_1 - \alpha_w,\ y_1 - \alpha_h,\ x_2 + \alpha_w,\ y_2 + \alpha_h]$

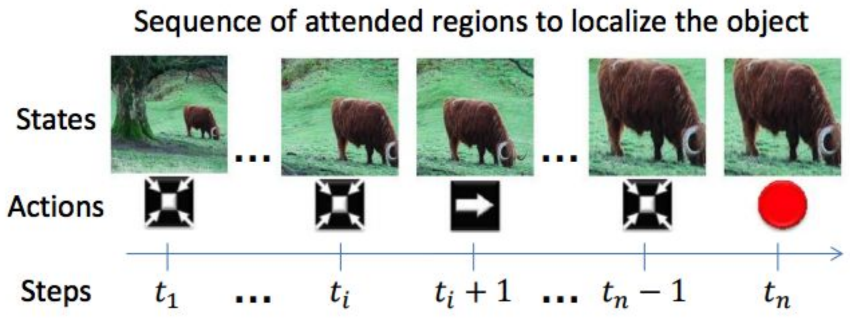
Taller: $[x_1 + \alpha_w,\ y_1,\ x_2 - \alpha_w,\ y_2]$

States



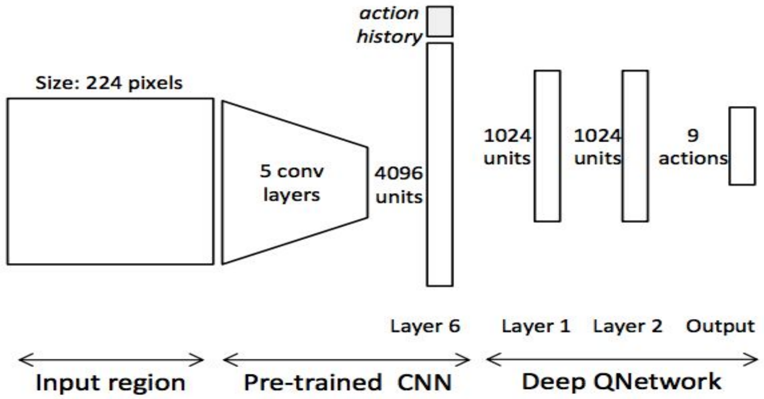
State = feature of the patch + action history

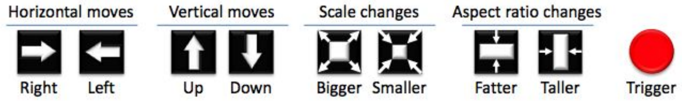
States



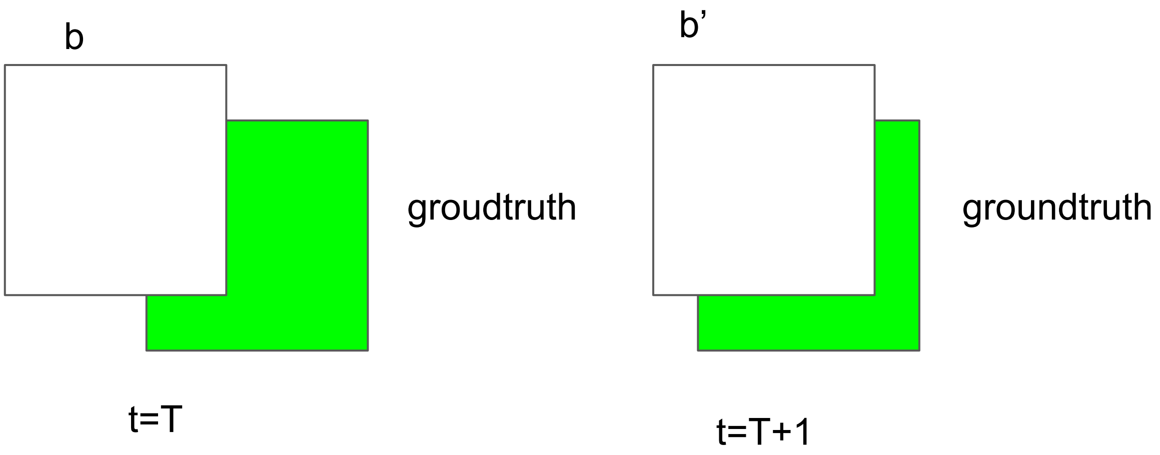
State = feature of the patch + action history

4096 9 x 10





Rewards

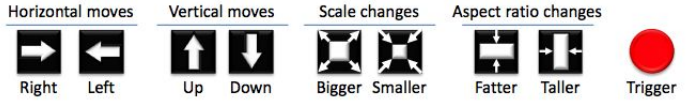


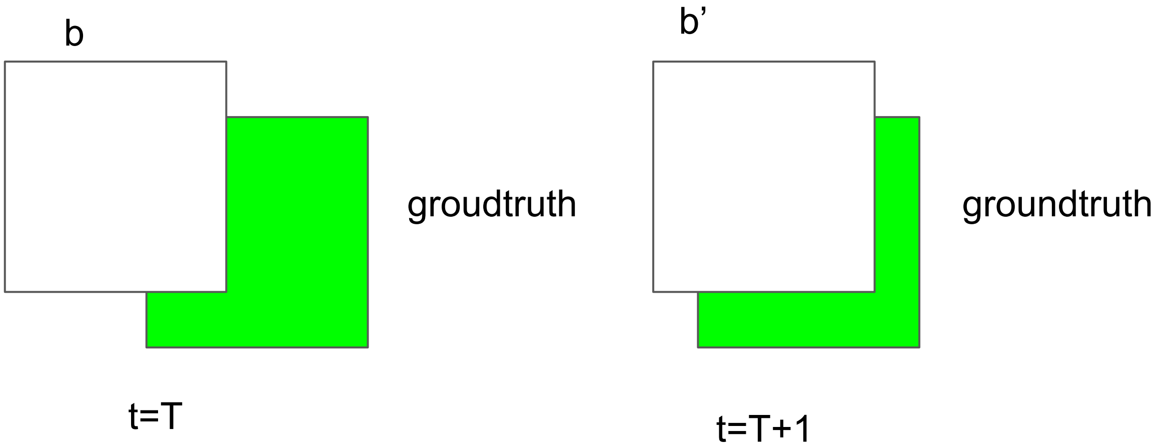
8 actions:

$$R_a(s, s') = sign\left(IoU(b', g) - IoU(b, g)\right)$$

$$r \in \{-1, +1\}.$$

Rewards





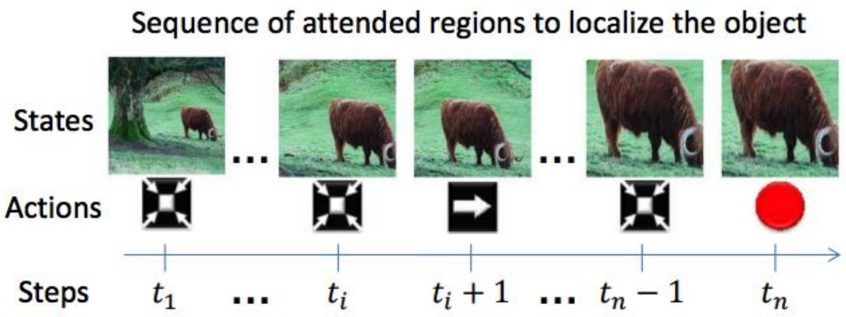
8 actions:

$$R_a(s, s') = sign\left(IoU(b', g) - IoU(b, g)\right) \qquad r \in \{-1, +1\}$$

Trigger:

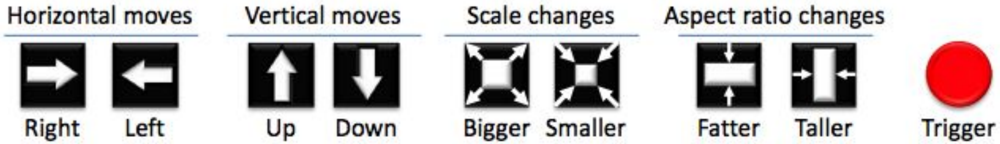
$$R_\omega(s, s') = \begin{cases} +\overset{3.0}{\eta} & \text{if } IoU(b, g) \geq \tau \\ -\eta & \text{otherwise} \end{cases} \qquad \tau = 0.6$$

States



State = feature of the patch + action history

Actions



Bounding Box

$$b = [x_1, y_1, x_2, y_2].$$

Changes:

0.2

$$\alpha_w = \alpha * (x_2 - x_1) \qquad \alpha_h = \alpha * (y_2 - y_1)$$

a good trade-off between speed and localization accuracy.

Examples:

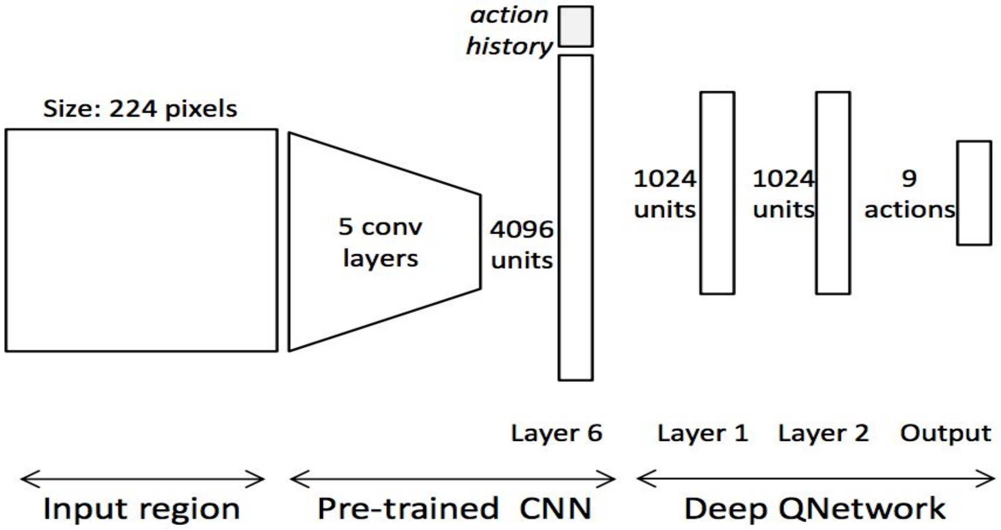
Left: $[x_1 - \alpha_w,\ y_1,\ x_2 - \alpha_w,\ y_2]$

Bigger: $[x_1 - \alpha_w,\ y_1 - \alpha_h,\ x_2 + \alpha_w,\ y_2 + \alpha_h]$

Taller: $[x_1 + \alpha_w,\ y_1,\ x_2 - \alpha_w,\ y_2]$

Policy: State -> Action

Action value, Q(s,a) : Estimation of future reward after taking action a in state s

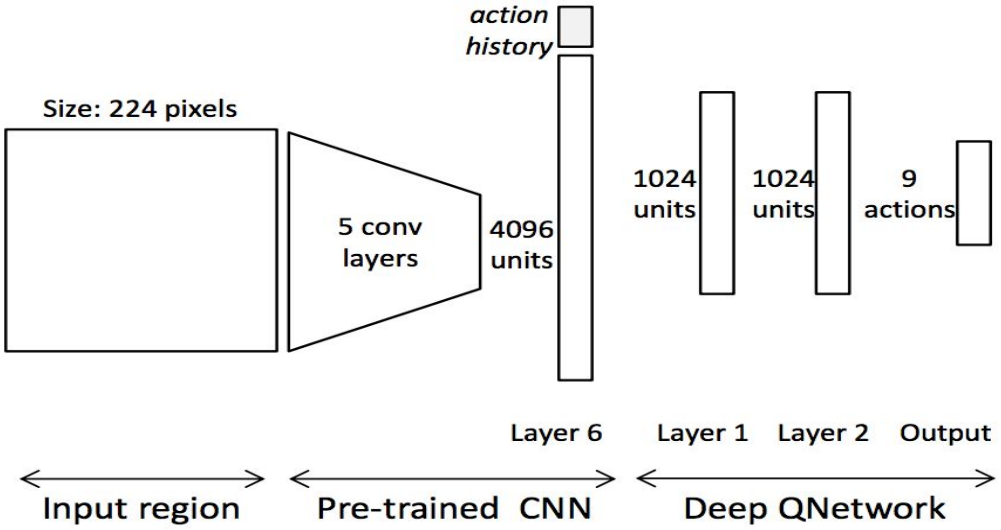


Action value
Q(s,a)

State Value, V(s): - Estimation of total future reward if starting from state s
- How good it is for the agent to be in state s

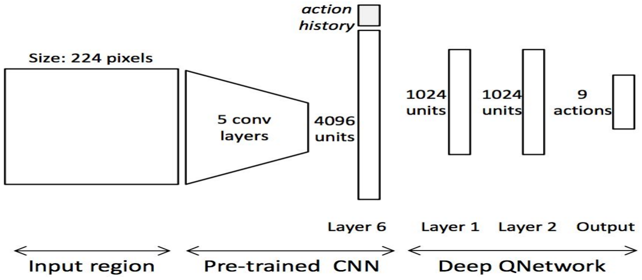
Policy: State -> Action

Action value, Q(s,a) : Estimation of future reward after taking action a in state s



Action value
Q(s,a)

Training



Deep Q-Learning Algorithm

- Define objective function by mean-squared error in Q-values

$$\mathcal{L}(w) = \mathbb{E}\left[\left(\underbrace{r + \gamma \max_{a'} Q(s', a', w)}_{\text{target}} - Q(s, a, w)\right)^2\right]$$

http://www0.cs.ucl.ac.uk/staff/d.silver/web/Resources_files/deep_rl.pdf

Training

$$q_*(s,a) \;=\; \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\Big|\; S_t = s, A_t = a\Big]$$

Bellman Equation for action value

Deep Q-Learning Algorithm

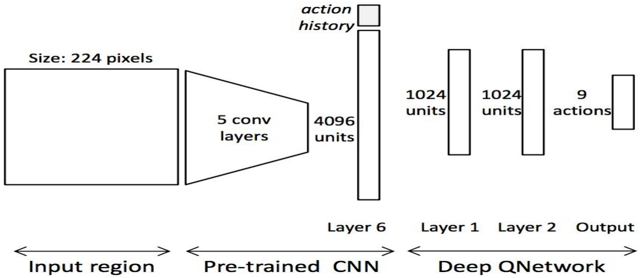
- Define objective function by mean-squared error in Q-values

$$\mathcal{L}(w) = \mathbb{E}\left[\left(\underbrace{r + \gamma \max_{a'} Q(s', a', w)}_{\text{target}} - Q(s, a, w)\right)^2\right]$$

http://www0.cs.ucl.ac.uk/staff/d.silver/web/Resources_files/deep_rl.pdf

Training

Deep Q-Learning Algorithm



- Define objective function by mean-squared error in Q-values

$$\mathcal{L}(w) = \mathbb{E}\left[\left(\underbrace{r + \gamma \max_{a'} Q(s', a', w)}_{\text{target}} - Q(s, a, w)\right)^2\right]$$

- Leading to the following Q-learning gradient

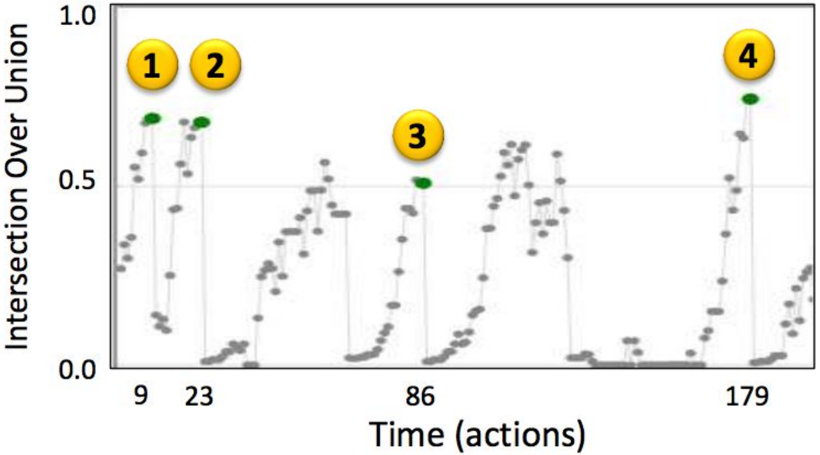
$$\frac{\partial \mathcal{L}(w)}{\partial w} = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w)\right) \frac{\partial Q(s, a, w)}{\partial w}\right]$$

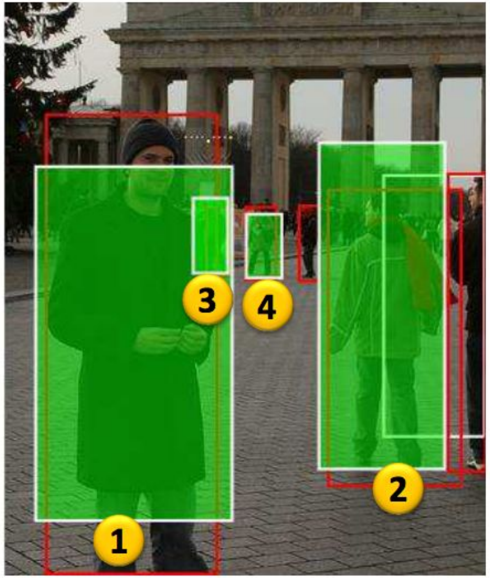
- Optimise objective end-to-end by SGD, using $\frac{\partial L(w)}{\partial w}$

http://www0.cs.ucl.ac.uk/staff/d.silver/web/Resources_files/deep_rl.pdf

Test

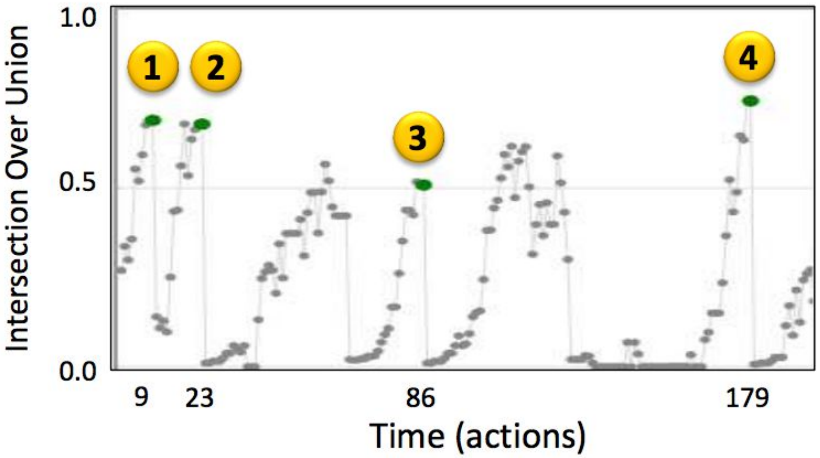
- **200** steps in total
- less than **40** steps for localizing one instance

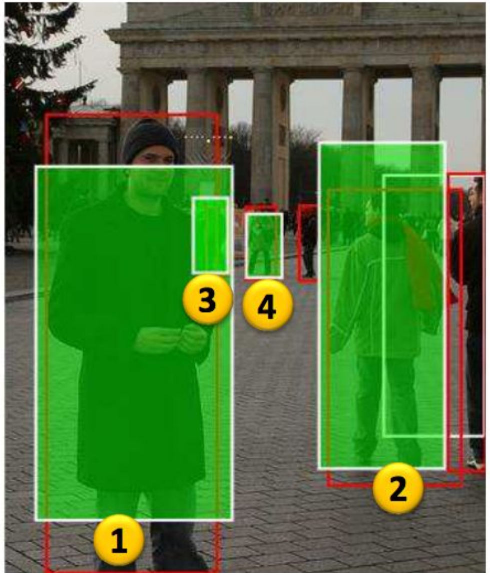


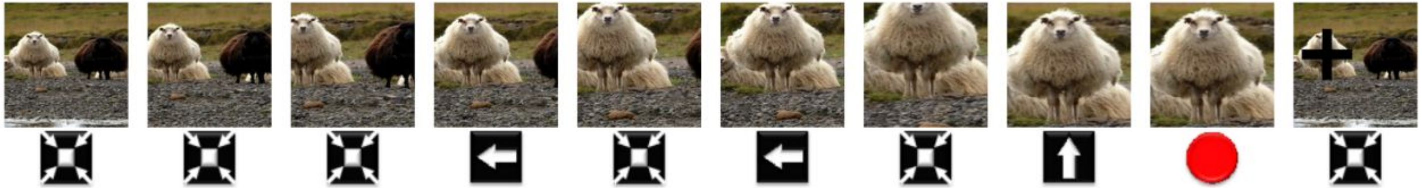


Test

- **200** steps in total
- Less than **40** steps for localizing one instance



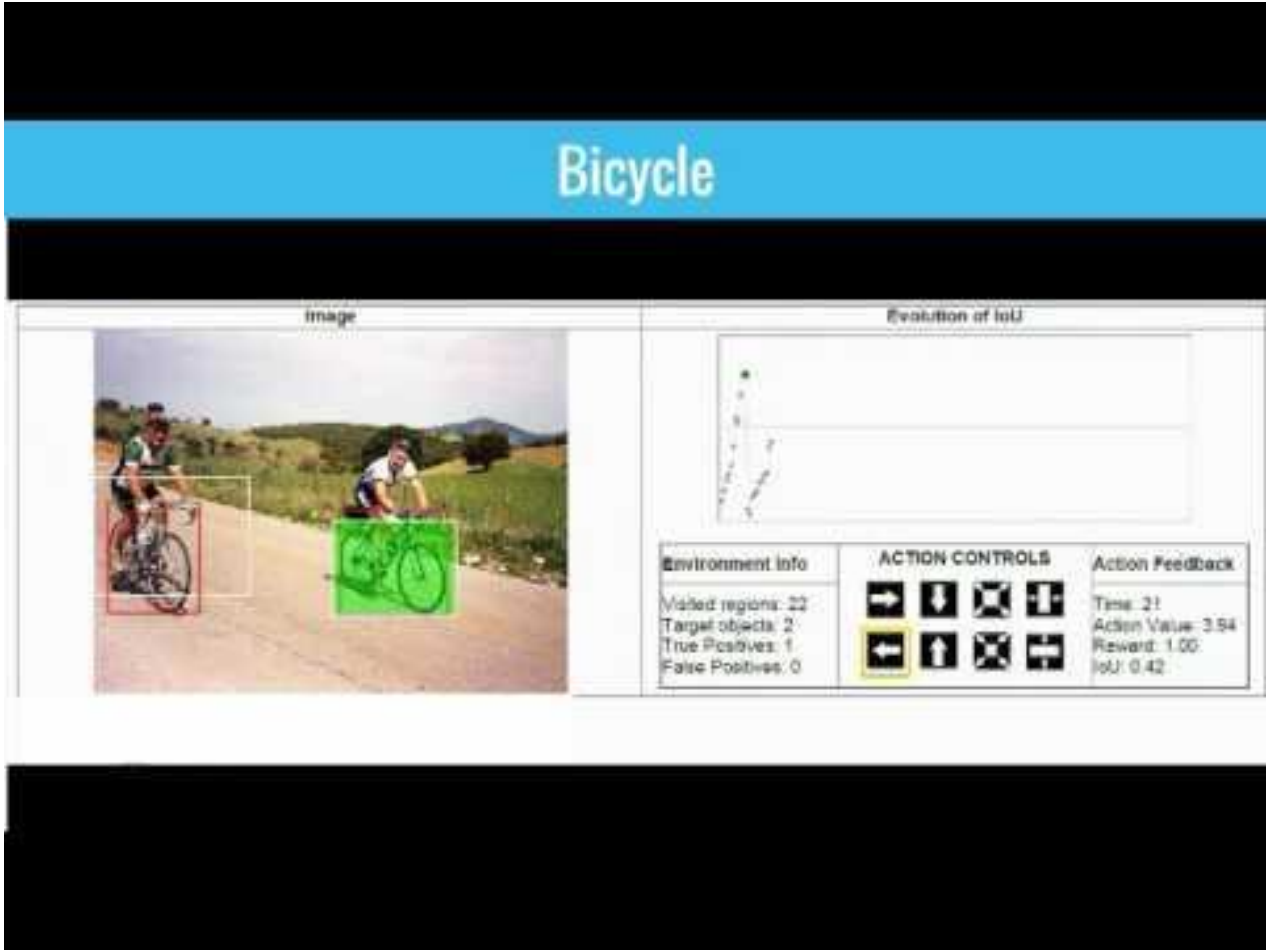




Test

Figure 6. Example sequences observed by the agent and the actions selected to focus objects. Regions are warped in the same way as they are fed to the CNN. Actions keep the object in the center of the box. More examples in the supplementary material. Last row: example Inhibition of Return marks placed during test.

Demo


Bicycle
Image
Evolution of IoU
Environment Info
Visited regions: 22
Target objects: 2
True Positives: 1
False Positives: 0
ACTION CONTROLS
Action Feedback
Time: 21
Action Value: 3.94
Reward: 1.00
IoU: 0.42

Test

Only need 11 - 25 regions to localize a single instance

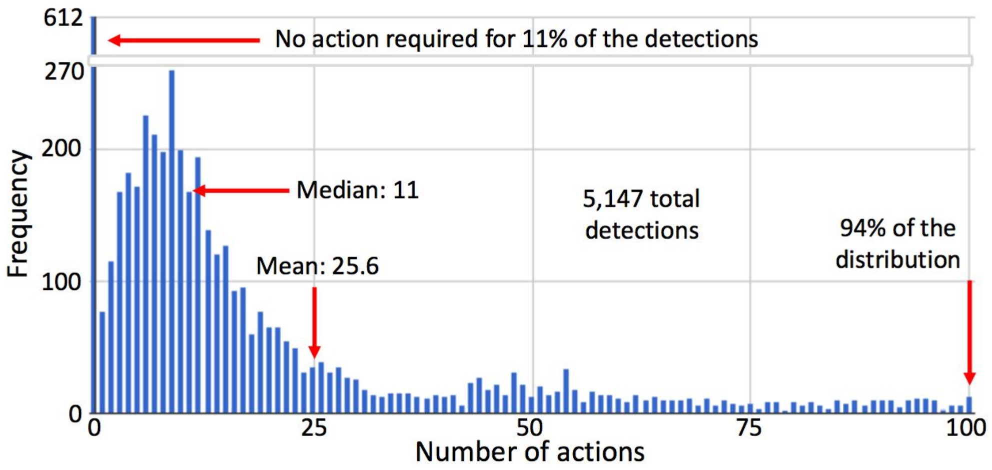


Figure 5. Distribution of detections explicitly marked by the agent as a function of the number of actions required to reach the object. For each action, one region in the image needs to be processed. Most detections can be obtained with about 11 actions only.

Failure Cases

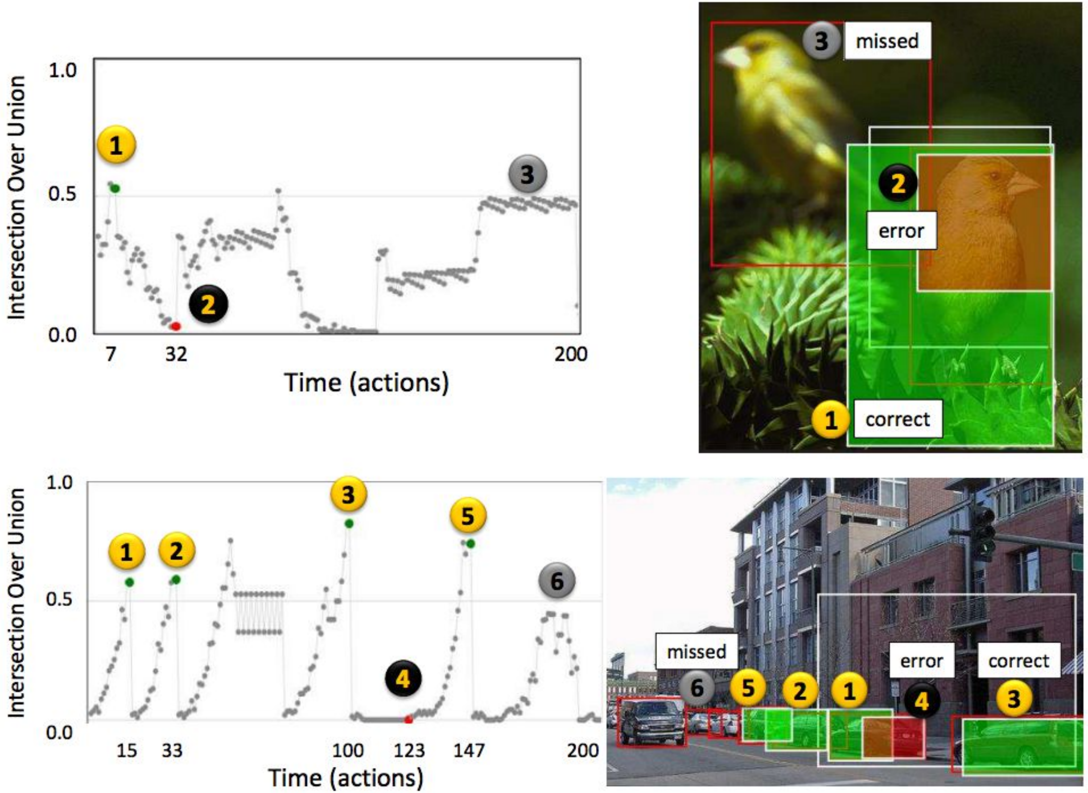


Figure 8. Examples of images with common mistakes that include: duplicated detections due to objects not fully covered by the IoR mark, and missed objects due to size or other difficult patterns.

Evaluation

- All attended regions **(AAR)**
  scores all regions processed by the agent during a search episode

- Terminal Regions **(TR)**
  only consider the final region that is triggered.

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	MAP
DPM [11]	33.2	60.3	10.2	16.1	**27.3**	54.3	58.2	23.0	20.0	24.1	26.7	12.7	58.1	48.2	43.2	12.0	21.1	36.1	46.0	43.5	33.7
MultiBox [9]	41.3	27.7	30.5	17.6	3.2	45.4	36.2	53.5	6.9	25.6	27.3	46.4	31.2	29.7	37.5	7.4	29.8	21.1	43.6	22.5	29.2
DetNet [32]	29.2	35.2	19.4	16.7	3.7	53.2	50.2	27.2	10.2	34.8	30.2	28.2	46.6	41.7	26.2	10.3	32.8	26.8	39.8	47.0	30.5
Regionlets [38]	44.6	55.6	24.7	23.5	6.3	49.4	51.0	**57.5**	14.3	35.9	45.9	41.3	*61.9*	54.7	44.1	16.0	28.6	**41.7**	*63.2*	44.2	40.2
Ours TR	**57.9**	56.7	38.4	33.0	17.5	51.1	52.7	53.0	17.8	39.1	*47.1*	52.2	58.0	**57.0**	45.2	19.3	42.2	35.5	54.8	49.0	43.9
Ours AAR	55.5	**61.9**	**38.4**	**36.5**	21.4	**56.5**	**58.8**	55.9	**21.4**	**40.4**	46.3	**54.2**	56.9	55.9	**45.7**	**21.1**	**47.1**	41.5	54.7	**51.4**	**46.1**
R-CNN [12]	*64.2*	*69.7*	*50.0*	*41.9*	*32.0*	*62.6*	*71.0*	*60.7*	*32.7*	*58.5*	**46.5**	*56.1*	**60.6**	*66.8*	*54.2*	*31.5*	*52.8*	*48.9*	**57.9**	*64.7*	*54.2*

Table 1. Average Precision (AP) per category in the Pascal VOC 2007 test set. The DPM system is the only baseline that does not use CNN features. R-CNN is the only method that uses object proposals. Our system is significantly better at localizing objects than other recent systems that predict bounding boxes from CNN features without object proposals. Numbers in bold are the second best result per column, and underlined numbers are the overall best result.

Thanks