Object Tracking

Vehicle count: 8

Vehicle count: 16
Limits in Object Tracking?
Method: 
Action Decision Network

Outcome: 
light computation 
satisfactory tracking accuracy
Other approach investigated:

- Early approaches Suffer from video tracking data deficiency for training

- No online updating: caused loosing target

- Precise approaches suffer from calculation time

- Sliding windows or candidate are slow
Framework

Frame $F_l$

Bounding Box $b_l t$

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

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

- **Action**: $a_t$, defined by discrete actions
  - Translation moves
  - Scale changes
  - Stop

  one-hot encoding

  **Example**: $\downarrow = [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$

\[
\begin{align*}
  b_t &= [x^{(t)}, y^{(t)}, w^{(t)}, h^{(t)}] \\
  \Delta x^{(t)} &= \alpha w^{(t)} \quad \text{and} \quad \Delta y^{(t)} = \alpha h^{(t)} \\
  [x^{(t)} - \Delta x^{(t)}, y^{(t)}, w^{(t)}, h^{(t)}]
\end{align*}
\]
Action: $a_t$, defined by discrete actions

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

Image patch: \( p_t \in \mathbb{R}^{112 \times 112 \times 3} \)

State: \( s_t = (p_t, d_t) \)

Action: \( a_t \), defined by discrete actions

Translation moves

Scale changes

Stop

Action dynamics: \( d_t \in \mathbb{R}^{110} \)
Framework
Framework
Framework
Framework
Framework

\[ p_{t+1} = f_p(p_t, a_t) \]

\[ d_{t+1} = f_d(d_t, a_t) \]
Framework
Training Method

• pre-training the policy networks with Supervised Learning (SL) before employing policy gradient can improve the performance.

• Step 1: Train the NN with supervised learning
  • learn the appearance characteristics of the target objects

• Step 2: Train NN with Reinforcement Learning (RL)
  • train action dynamics of the tracking target
Step 1: Supervised Learning

- Generate training samples of **state-action** pairs
- The training samples consist of: image patches, action labels and class labels
- Action dynamics is set to zero
- Train policy network as multiclassification with softmax
Step 1: Supervised Learning

- $f(p, a)$ denotes the moved patch from $p$ by action $a$.

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

- The class label corresponding to $p_j$ is defined by:

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

- The multi-task loss function is defined by minimizing the loss

$$L_{SL} = \frac{1}{m} \sum_{j=1}^{m} L(o_j^{(act)}, \hat{o}_j^{(act)}) + \frac{1}{m} \sum_{i=j}^{m} L(o_j^{(cls)}, \hat{o}_j^{(cls)})$$
Value based vs Policy Based

• Value Based (DQN)
  • Value Function learnt
  • arrive at the optimal policy indirectly
  • taking actions that maximizes the value functions

• Policy Based (Policy Gradient)
  • Simply input the state and out comes an action
  • Then increase the probability of desiring action
  • learn a policy function directly (instead of a Q function)
  • directly optimizing for the long term reward (here immediate)
  • Policy Gradient: gradient of the expected reward w.r.t. the parameters of the policy
Step 2: Reinforcement Learning

- train policy network to maximize the expected tracking rewards
- initially RL uses SL Weights
- randomly pick a piece of training sequence \( \{F_t\}_{t=1}^L \) and the ground truths \( \{G_t\}_{t=1}^L \)
- run tracking to a piece of \( \{s_{\downarrow t, l}, a_{\downarrow t, l}, r_{\downarrow t, l}\} \)
- compute tracking score \( z_{t, l} = r(s_{T_t, l}) \) based on \( \{G_t\}_{t=1}^L \)
- Update \( W_{RL} \) by tracking score

\[
\Delta W_{RL} \propto \sum_{l} \sum_{t} T_t \frac{\partial \log p(a_{t, l}|s_{t, l}; W_{RL})}{\partial W_{RL}} z_{t, l}.
\]

- Reward:
  \[
r(s_t) = \begin{cases} 1, & \text{if } IOU(p_t, G) > 0.7 \\ -1, & \text{otherwise} \end{cases}
\]

\[
a_{t, l} = \arg \max_{a} p(a|s_{t, l}; W_{RL})
\]
Algorithm 1 Training ADNet with reinforcement learning (RL).

Input: Pre-trained ADNet ($W_{SL}$), Training sequences $\{F_l\}$ and ground truths $\{G_l\}$

Output: Trained ADNet weights $W_{RL}$

1: Initialize $W_{RL}$ with $W_{SL}$
2: repeat
3: Randomly select $\{F_l\}_{l=1}^L$ and $\{G_l\}_{l=1}^L$
4: Set initial $b_{1,1} \leftarrow G_1$
5: Set initial $d_{1,1}$ as zero vector
6: $T_l \leftarrow 1$
7: for $l = 2$ to $L$ do
8:   $\{a_{t,l}, b_{t,l}, d_{t,l}, T_l\} \leftarrow \text{TRACKING PROCEDURE}(b_{T_l-1,l-1}, d_{T_l-1,l-1}, F_l)$ in Algorithm 2
9: end for
10: Compute tracking scores $\{z_{t,l}\}$ with $\{b_{t,l}\}$ and $\{G_l\}_{l=1}^L$
11: $\Delta W_{RL} \propto \sum_{l=1}^L \sum_{t=1}^{T_l} \frac{\partial \log p(a_{t,l}|z_{t,l})}{\partial W_{RL}} z_{t,l}$ [13]
12: Update $W_{RL}$ using $\Delta W_{RL}$
13: until $W_{RL}$ converges

Algorithm 2 Tracking Procedure of ADNet.

1: procedure TRACKING PROCEDURE$(b_{T_l-1,l-1}, d_{T_l-1,l-1}, F_l)$
2: $t \leftarrow 1$
3: $p_{t,l} \leftarrow \phi(b_{T_l-1,l-1}, F_l)$
4: $d_{t,l} \leftarrow d_{T_l-1,l-1}$
5: $a_{t,l} \leftarrow (p_{t,l}, d_{t,l})$
6: repeat
7:   $a_{t+l} \leftarrow \arg \max_a p(a_{t+l}|W)$
8:   $b_{t+l} \leftarrow f_p(b_{t+l}, a_{t+l})$
9:   $p_{t+l} \leftarrow \phi(b_{t+l}, F_l)$
10:  $d_{t+l} \leftarrow f_d(d_{t+l}, a_{t+l})$
11:  $s_{t+l} \leftarrow (p_{t+l}, d_{t+l})$
12:  $t \leftarrow t + 1$
13: until $s_{t,l}$ is a terminal state
14: Set termination step $T_l \leftarrow t$
15: Return $\{a_{t,l}, b_{t,l}, d_{t,l}, T_l\}$
16: end procedure
RL in Semi Supervised Case

can train ADNet even if the ground truths $\{G_i\}_{i=1}^{\mathcal{F}}$ are partially given
More detail In Model

- Makes the tracking algorithm more robust during appearance changes
- Fine tune Adnet by SL using temporal training sample in tracking process
- Ground truth is tracked patch position determined by network.
- It updates every “10” frame using data sampled last 20 frames.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>ADnet</th>
<th>ADnetFast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online adaptation</td>
<td>10 frame</td>
<td>30</td>
</tr>
<tr>
<td>Number of target position in redaction</td>
<td>256</td>
<td>64</td>
</tr>
<tr>
<td>Number of sample generated at first frame</td>
<td>3000</td>
<td>300</td>
</tr>
</tbody>
</table>
Time to Evaluate!
Test platform

- Used on popular object tracking benchmark (OTB)
- I7-4790k CPU 32 Gb RAM, GTX Titan X CPU
- 3fps, 15 fps
- OTB50, OTB 100
- Used 360 Training Videos

Test Method

- Center location error
- IOU: intersection over Union
- OPE: one pass evaluation
Self Comparison

- Adnet-int: Baseline Adnet parameter of convolutional network are trained with VGG-M
- Adnet_SL: ADNET with Supervised learning
- Adnet_SS: SemiSupervised learning
- Adnet_SL_RL: Supervised learning with reinforcement learning
Compare with Other Approaches

(a) OTB-50

(b) OTB-100
# Summary of Experiment

Table 1: Summary of experiments on OTB-100.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Prec.(20px)</th>
<th>IOU(AUC)</th>
<th>FPS</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADNet</td>
<td>88.0%</td>
<td>0.646</td>
<td>2.9</td>
<td>O</td>
</tr>
<tr>
<td>ADNet-fast</td>
<td>85.1%</td>
<td>0.635</td>
<td>15.0</td>
<td>O</td>
</tr>
<tr>
<td>MDNet [24]</td>
<td>90.9%</td>
<td>0.678</td>
<td>&lt; 1</td>
<td>O</td>
</tr>
<tr>
<td>C-COT [9]</td>
<td>90.3%</td>
<td>0.673</td>
<td>&lt; 1</td>
<td>O</td>
</tr>
<tr>
<td>DeepSRDCF [8]</td>
<td>85.1%</td>
<td>0.635</td>
<td>&lt; 1</td>
<td>O</td>
</tr>
<tr>
<td>HDT [25]</td>
<td>84.8%</td>
<td>0.564</td>
<td>5.8</td>
<td>O</td>
</tr>
<tr>
<td>MUSTer [15]</td>
<td>76.7%</td>
<td>0.528</td>
<td>3.9</td>
<td>X</td>
</tr>
<tr>
<td>MEEM [42]</td>
<td>77.1%</td>
<td>0.528</td>
<td>19.5</td>
<td>X</td>
</tr>
<tr>
<td>SCT [5]</td>
<td>76.8%</td>
<td>0.533</td>
<td>40.0</td>
<td>X</td>
</tr>
<tr>
<td>KCF [13]</td>
<td>69.7%</td>
<td>0.479</td>
<td>223</td>
<td>X</td>
</tr>
<tr>
<td>DSST [7]</td>
<td>69.3%</td>
<td>0.520</td>
<td>25.4</td>
<td>X</td>
</tr>
<tr>
<td>GOTURN [12]</td>
<td>56.5%</td>
<td>0.425</td>
<td>125</td>
<td>O</td>
</tr>
</tbody>
</table>
Weakness and Future Work
Experimental Results
Thank you
\[ p_\theta(s_1, a_1, \ldots, s_T, a_T) = \frac{\prod_{t=1}^{T} \pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t)}{\pi_\theta(\tau)} \]

\[ \theta^* = \arg\max_{\theta} E_{\tau \sim p_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \]

\[ J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[ \sum_t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, a_{i,t}) \]

\[ \nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[ \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \left( \sum_{t=1}^{T} r(s_t, a_t) \right) \right] \]

\[ \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left( \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \right) \left( \sum_{t=1}^{T} r(s_{i,t}, a_{i,t}) \right) \]
REINFORCE Algorithm

1. sample \( \{\tau^i\} \) from \( \pi_\theta(a_t|s_t) \) (run the policy)
2. \( \nabla_\theta J(\theta) \approx \sum_i \left( \sum_t \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) \right) \left( \sum_t r(s_t^i, a_t^i) \right) \)
3. \( \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \)
Deep Reinforcement Learning

To learn a policy that decides sequential actions by maximizing cumulative future rewards

**Agent:**
the tracker is defined as an agent of which goal is to capture the target with a bounding box shape. The action is defined in a discrete space and a sequence of actions and states is used to iteratively pursue the resulting bounding box location and size in each frame.

**State:**

**Action:**

**Reward:**
**Problem Setting**

\[ S_t = (P_t, d_t) \]

\[ S_{t+1} \]

**Symbols:**

- \( P_t \): Image Patch
  - \( p_t = \phi(b_t, F) \)

- \( d_t \): Action dynamic vector
  - \( b_t = [x(t), y(t), w(t), h(t)] \)