Efficient Object Detection in Large Images Using Deep Reinforcement Learning

Burak Uzkent, Christopher Yeh, and Stefano Ermon
Department of Computer Science, Stanford University
Detection in Large Images - Sliding Window

- No need to downsample
- Low memory requirement
- Large runtime to process each window
Proposed Method - Adaptive Sliding Window

Small objects require fine-level information whereas large objects can be detected at coarse-level.
Policy network treats sampling each image patch as a Bernoulli random variable.

\[ \alpha_c \in \{0, 1\}^{P_c} \]

\[
\pi_c(\alpha_c | x_L, \theta_p^c) = \prod_{i=1}^{P_c} (s^i_c)^{a^i_c} (1 - s^i_c)^{(1-a^i_c)}
\]

\[
s_c = f_p^c(x_L; \theta_p^c)
\]
**accuracy cost**

\[
R_{acc} = \sum_{i=1}^{P_c} \left( \text{Recall} \left( \hat{Y}_i^f, Y_i \right) - \left( \text{Recall} \left( \hat{Y}_i^c, Y_i \right) + \beta \right) \right) \cdot N_i
\]
Modeling the Policy Networks

Policy network treats sampling each image patch as a Bernoulli variable

\[
\pi_c(a_c|x_L, \theta_p^c) = \prod_{i=1}^{P_c} (s_c^i)^a_c^i (1 - s_c^i)^{(1-a_c^i)}
\]

\[s_c = f_p^c(x_L; \theta_c^c)\]

Policy network is trained with policy gradient method, with advantage function

\[
J_c = \mathbb{E} [R_c(a_c, a_d, Y)]
\]

\[
\nabla_{\theta_p^c} J_c = \mathbb{E} \left[ R_c \cdot \nabla_{\theta_p^c} \log \pi_{\theta_p^c}(a_c|x_L) \right]
\]
Experiments on the xView dataset, consisting of 847 very large images (>3000 x >3000 px).

<table>
<thead>
<tr>
<th>Model/Metric</th>
<th>HR</th>
<th>AP</th>
<th>AR</th>
<th>Run-time</th>
<th>HR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random (5x)</td>
<td>50</td>
<td>24.1</td>
<td>47.1</td>
<td>1408</td>
<td>31</td>
</tr>
<tr>
<td>Entropy (5x)</td>
<td>50</td>
<td>25.4</td>
<td>47.2</td>
<td>1415</td>
<td>31</td>
</tr>
<tr>
<td>Sliding Window-L (5x)</td>
<td>0</td>
<td>26.3</td>
<td>39.8</td>
<td>640</td>
<td>0</td>
</tr>
<tr>
<td>Sliding Window-H</td>
<td>100</td>
<td>39.0</td>
<td>60.9</td>
<td>3200</td>
<td>100</td>
</tr>
<tr>
<td>Gao et al. [7] (5x)</td>
<td>35.4</td>
<td>35.2</td>
<td>55.5</td>
<td>1551</td>
<td>31.6</td>
</tr>
<tr>
<td><strong>Ours (5x)</strong></td>
<td>35.5</td>
<td>38.1</td>
<td>59.7</td>
<td>1484</td>
<td>31.5</td>
</tr>
</tbody>
</table>

Table 1: Results on the xView test set.

Experiments - Caltech Pedestrian

Experiments on the Caltech Pedestrian dataset (>800 x >800 px).

<table>
<thead>
<tr>
<th>Model/Metric</th>
<th>AP</th>
<th>AR</th>
<th>Run-time</th>
<th>HR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random (×5)</td>
<td>30.9</td>
<td>62.1</td>
<td>248</td>
<td>44.4</td>
</tr>
<tr>
<td>Entropy (×5)</td>
<td>34.0</td>
<td>63.9</td>
<td>250</td>
<td>44.4</td>
</tr>
<tr>
<td>Sliding Window-L (×5)</td>
<td>21.2</td>
<td>46.3</td>
<td>90</td>
<td>0</td>
</tr>
<tr>
<td>Sliding Window-H</td>
<td>64.7</td>
<td>74.7</td>
<td>450</td>
<td>100</td>
</tr>
<tr>
<td>Gao et al. [7] (×2)</td>
<td>64.5</td>
<td>73.1</td>
<td>295</td>
<td>7.1</td>
</tr>
<tr>
<td>Gao et al. [7] (×5)</td>
<td>57.3</td>
<td>70.7</td>
<td>309</td>
<td>43.3</td>
</tr>
<tr>
<td>CPNet (×2)</td>
<td>64.4</td>
<td>74.5</td>
<td>267</td>
<td>6.6</td>
</tr>
<tr>
<td>CPNet (×5)</td>
<td>61.7</td>
<td>74.1</td>
<td>270</td>
<td>44.4</td>
</tr>
</tbody>
</table>

Table 2: Results on the Caltech Pedestrian test set.
Code

https://github.com/uzkent/EfficientObjectDetection

Efficient Object Detection in Large Images with Deep Reinforcement Learning

This repository contains PyTorch implementation of our IEEE WACV20 paper on Efficient Object Detection in Large Images with Deep Reinforcement Learning. The arXiv version of the paper can be found here.