**Abstract**

Most modern object detection algorithms (YOLO, SSD) are prone to bounding box jitter. Our project explores the feasibility of attaching a recurrent neural network at the end of a YOLO detector to de-noise/stabilize a jittery bounding box trajectory.

**Bounding box jitter?**

**Causes**
- Inherent pixel noise in camera sensors
- Improper aggregation of proposed bounding boxes

**We care because**
- Problematic in applications such as in surveillance where the behavior of an object depends on the bounding box movement
- Distracting!

**Example**
- Two bounding box trajectories are shown below, blue and orange. Notice how the center position of the orange box is not fixed on the person frame by frame.

**Methods**
- Use Kalman Filters to “predict” bounding box trajectory and filter noisy bounding box predictions
- Fix improper aggregation in YOLO/SSD by using weighted Non-Max Suppression
- Use an RNN as a filter and achieve improvement in performance!

**Approach**

**Architecture:**

- LSTM
- YOLO
- Stabilized bounding box

**Training:**
- Trained on MOT2015 bounding box trajectories, each ranging anywhere from 100-600 frames
- YOLO retrained for single-class detection
- Custom stability loss function, 20k epochs

**LSTM inputs:**
- Bounding boxes
- Flattened higher layer CNN feature maps

**Evaluation Metrics:**
- Center position error
  \[ e^c_t = \frac{e^x_t}{\text{bbox size}}, \quad \sigma_x = \text{std}(e^x), \quad \sigma_y = \text{std}(e^y) \]
  \[ E^c_t = \sigma_x + \sigma_y \]
- Scale and ratio error
  \[ e^s_t = \sqrt{\frac{e^x_t^2}{e^x_t} \cdot \frac{e^y_t^2}{e^y_t}}, \quad \sigma_e = \text{std}(e^s), \quad \sigma_s = \text{std}(e^s) \]
  \[ E^s_t = \sigma_e + \sigma_s \]

**Training Loss:**

<table>
<thead>
<tr>
<th>Pure YOLO</th>
<th>YOLO + SORT</th>
<th>YOLO + RNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center Position Error</td>
<td>93.01</td>
<td>71.11</td>
</tr>
<tr>
<td>Scale Ratio Error</td>
<td>163.27</td>
<td>323.97</td>
</tr>
</tbody>
</table>

- Results show YOLO-RNN improves performance by 5-8% with initial training
- With better training, confident in producing better performance
- Want to also try and implement different RNN architectures to figure out what is the best for bounding box stability

**References**