Activity Recognition

JUSTIN LIANG
MARCH 27, 2016
Agenda


• Detecting Events and Key Actors in Multi-Person Videos. V. Ramanathan, J. Huang, S. Abu-El-Haija, A. Gorban, K. Murphy and L. Fei-Fei. CVPR 2016.
What is Activity Recognition

• Idea is to be able to detect what event occurs in a video
  • Ex. diving, successful layup, failed layup, successful slam dunk, blocking, setting, standing

• Different sub domains to do activity recognition:
  • Individual activity recognition
  • Group activity recognition
  • Temporal activity recognition

[Ibrahim et al. CVPR 2016]
End-to-end Learning of Action Detection from Frame Glimpses in Videos
End-to-end Learning of Action Detection from Frame Glimpses in Videos

• Paper from Serena Yeung, Olga Russakovsky, Greg Mori, Li Fei-Fei in CVPR 2016.

• Objective:
  • Predict actions and their temporal bounds: how long and where they occur in a video clip. Video clips used are untrimmed.

• Key Contributions:
  • End-to-end approach to action detection and temporal localization in videos
  • Train an agent policy to skip video frames to find where the actions are in the video
  • Show that this method can outperform state of the art results
Approach

• Action detection is a process of observation and refinement. Effectively choosing a sequence of frame observations allows us to quickly narrow down when the baseball swing occurs.
Approach (Pipeline)

- $o_n$: observation feature vector
- $h_n$: internal hidden state
- $d_n$: candidate detection
  - $s_n$: action starts
  - $e_n$: action ends
  - $c_n$: action confidence level
- $p_n$: indicator to emit action
- $l_{n+1}$: location of next observation, $l_n \in [0,1]$
Observation Network

• Both the location $l_n$ and video frame $v_{l_n}$ are mapped to a hidden space and then combined with a fully connected layer to produce the observation vector $o_n$

• $v_{l_n}$ is mapped using the VGG16 network and fc7 features are extracted from it
Recurrent Network

• Observation features \( o_n \) and previous internal hidden state \( h_{n-1} \) are inputs to the recurrent network \( f_h \) which is parameterized by \( \theta_h \) to produce \( h_n \).
Recurrent Network

- Observation features $o_n$ and previous internal hidden state $h_{n-1}$ are inputs to the recurrent network $f_h$ which is parameterized by $\theta_h$ to produce $h_n$

- Candidate detection $d_n$:
  - $d_n = f_d(h_n; \theta_d)$, $f_d$ is a fully connected layer
Recurrent Network

- Observation features $o_n$ and previous internal hidden state $h_{n-1}$ are inputs to the recurrent network $f_h$ which is parameterized by $\theta_h$ to produce $h_n$

- Candidate detection $d_n$:
  - $d_n = f_d(h_n; \theta_d)$, $f_d$ is a fully connected layer

- Prediction Indicator $p_n$:
  - $p_n = f_p(h_n; \theta_p)$, $f_p$ is a fully connected layer
  - During training, $f_p$ is used to parameterize a Bernoulli distribution from which $p_n$ is sampled. At test time MAP estimate is used.
Recurrent Network

- Observation features $o_n$ and previous internal hidden state $h_{n-1}$ are inputs to the recurrent network $f_h$ which is parameterized by $\theta_h$ to produce $h_n$.
- Candidate detection $d_n$:
  - $d_n = f_d(h_n; \theta_d)$, $f_d$ is a fully connected layer.
- Prediction Indicator $p_n$:
  - $p_n = f_p(h_n; \theta_p)$, $f_p$ is a fully connected layer.
  - During training, $f_p$ is used to parameterize a Bernoulli distribution from which $p_n$ is sampled. At test time MAP estimate is used.
- Location of next observation $l_{n+1}$:
  - $l_{n+1} = f_l(h_n; \theta_l)$, $f_l$ is a fully connected layer.
  - During training, $l_{n+1}$ is sampled from a Gaussian distribution with mean $f_l(h_n; \theta_l)$ and fixed variance. At test time MAP estimate is used.
Training

• Goal is to train three outputs: candidate detection $d_n$, prediction indicator $p_n$, location of next observation $l_{n+1}$
  • This is difficult due to the challenges of designing suitable loss and reward functions and handling non-differentiable model components

• We use backpropagation to train $d_n$ and REINFORCE to train $p_n$ and $l_{n+1}$
Training (Candidate Detection $d_n$)

• Match each candidate detection $D = \{d_n | n = 1, ..., N\}$ from recurrent network to ground truth $g_1, ..., g_M$

• Matching function:
  • $\gamma_{nm} = \begin{cases} 
    1 & \text{if } m = \arg\min_{j=1,...,M} \text{dist}(l_n, g_j) \\
    0 & \text{otherwise}
  \end{cases}$
  • $g_j = (s_j, e_j)$
  • $\text{dist}(l_n, g_j) = \min(|s_j - l_n|, |e_j - l_n|)$
Training (Candidate Detection $d_n$)

- Match each candidate detection $D = \{d_n | n = 1, ..., N\}$ from recurrent network to ground truth $g_1, ..., M$

- Matching function:
  - $y_{nm} = \begin{cases} 1 & \text{if } m = \arg\min_{j=1, ..., M} \text{dist}(l_n, g_j) \\ 0 & \text{otherwise} \end{cases}$
  - $g_j = (s_j, e_j)$
  - $\text{dist}(l_n, g_j) = \min(|s_j - l_n|, |e_j - l_n|)$

- Loss function:
  - $\sum_n L_{\text{cls}}(d_n) + \gamma \sum_n \sum_m \mathbb{I}[y_{nm} = 1] L_{\text{loc}}(d_n, g_m)$
  - $L_{\text{cls}}(d_n)$: cross entropy loss on detection confidence $c_n$
  - $L_{\text{loc}}(d_n, g_m)$: L2 loss to further minimize distance $\| (s_n, e_n) - (s_m, e_m) \|$

- Optimize loss using backpropagation
Training (Location $l_{n+1}$ and Prediction Indicator $p_n$)

• Use REINFORCE to learn observation and emission policies

REINFORCE:
• Objective: $J(\theta) = \sum_{a \in \mathcal{A}} p_\theta(a) r(a)$
  • $\mathcal{A}$: space of action sequences
  • $p_\theta(a)$: probability of action
  • $r(a)$: reward
Training (Location $l_{n+1}$ and Prediction Indicator $p_n$)

• Use REINFORCE to learn observation and emission policies

• REINFORCE:
  • Objective: $J(\theta) = \sum_{a \in \mathcal{A}} p_\theta(a) r(a)$
    • $\mathcal{A}$: space of action sequences
    • $p_\theta(a)$: probability of action
    • $r(a)$: reward
  • Gradient: $\nabla J(\theta) = \sum_{a \in \mathcal{A}} p_\theta(a) \nabla \log p_\theta(a) r(a)$
    • This is a non trivial optimization problem due to the high dimensional space of possible action sequences!
    • Instead we can use Monte Carlo to take the expectation
Training (Location $l_{n+1}$ and Prediction Indicator $p_n$)

• Use REINFORCE to learn observation and emission policies

• REINFORCE:
  • Objective: $J(\theta) = \sum_{a \in A} p_\theta(a) r(a)$
    - $A$: space of action sequences
    - $p_\theta(a)$: probability of action
    - $r(a)$: reward
  • Gradient: $\nabla J(\theta) = \sum_{a \in A} p_\theta(a) \nabla \log p_\theta(a) r(a)$
    - Use Monte Carlo to approximate:
      - $\nabla J(\theta) \approx \frac{1}{K} \sum_{K} \sum_{n=1}^{N} \nabla \log \pi_\theta (a_n^i | h_{1:n}, a_{1:n-1}^i) R_n^i$
        - $K$ interaction sequences
        - $N$ RNN time steps
        - $\pi_\theta$: agent’s policy
        - $a_n$: current action ($l_{n+1}$ or $p_n$)
        - $R_n$: cumulative reward from current timestep onward
        - $h_n$: hidden state
  • Optimize by maximizing objective
Training (Location $l_{n+1}$ and Prediction Indicator $p_n$)

• Reward function:
  • Want high precision and recall
  • $r_N = \begin{cases} R_p & \text{if } M > 0 \text{ and } N_p = 0 \\ N_+R_+ + N_-R_- & \text{otherwise} \end{cases}$
  • $N_p$: # predictions emitted by agent
  • $N_+, R_+$: # true positive predictions and reward
  • $N_-, R_-$: # false positive predictions and reward
  • $R_p$: penalty for not emitting prediction when # ground truth $M > 0$

• Prediction is correct if its overlap with ground truth is greater than a threshold and higher than any other prediction
Strengths/Weaknesses of Approach

• Strengths:
  • Do not need to look at all the frames
  • End-to-end learning

• Weaknesses:
  • Need all the frames in a clip (cannot do online detection)
  • Can be difficult to learn observation policy if event contains less discriminative movements
Results

• Results from THUMOS’14 comparing with top 3 performers. mAP is reported for different IOU thresholds $\alpha$

• Ablation studies show that without localization regression and where to observe next, results are significantly worse

<table>
<thead>
<tr>
<th></th>
<th>$\alpha=0.5$</th>
<th>$\alpha=0.4$</th>
<th>$\alpha=0.3$</th>
<th>$\alpha=0.2$</th>
<th>$\alpha=0.1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karaman et al. [13]</td>
<td>0.9</td>
<td>1.4</td>
<td>2.1</td>
<td>3.4</td>
<td>4.6</td>
</tr>
<tr>
<td>Wang et al. [39]</td>
<td>8.3</td>
<td>11.7</td>
<td>14.0</td>
<td>17.0</td>
<td>18.2</td>
</tr>
<tr>
<td>Oneata et al. [22]</td>
<td>14.4</td>
<td>20.8</td>
<td>27.0</td>
<td>33.6</td>
<td>36.6</td>
</tr>
<tr>
<td>Ours (full)</td>
<td><strong>17.1</strong></td>
<td><strong>26.4</strong></td>
<td><strong>36.0</strong></td>
<td><strong>44.0</strong></td>
<td><strong>48.9</strong></td>
</tr>
</tbody>
</table>
Results (Learned Observation Policy)
Results (Learned Observation Policy)
Future Direction

• Learn joint spatio-temporal observation policies
Detecting Events and Key Actors in Multi-Person Videos
Detecting Events and Key Actors in Multi-Person Videos

• Paper from Vignesh Ramanathan, Jonathan Huang, Sami Abu-El-Haija, Alexander Gorban, Kevin Murphy and Li Fei-Fei in CVPR 2016.

• Objective:
  • Predict events and key actors in videos where multiple people are involved

• Key Contributions:
  • Introduce large-scale basketball event dataset
  • Use attention to decide most relevant people to the action being performed
  • Show that the attention model results in better event recognition
Dataset

• Introduced a large dataset with multi-person action videos. The dataset consists of 257 NCAA games each around 1.5 hours long. 11 different basketball events are densely annotated in the videos.
Approach

• Events in a team sport are performed by a set of key players. It is sufficient to focus only the players participating to recognize an event. For example, a “steal” event in basketball is defined by the action of the player attempting to pass the ball and the player stealing.

• The idea is to focus on key players to predict events.
Approach (Pipeline)

• Each player track is processed by a BLSTM network. The output hidden state is processed by an attention model to identify key players.

• The thickness of the boxes show attention weights.

• Each video frame is processed by a BLSTM network.
Feature Extraction

• Each video frame $t$ is represented as a feature vector $f_t$ from the activation of the last fully connected layer of the Inception7 network.

• Each player $i$ bounding box is represented as a feature vector $p_{ti}$ from Inception7.
Event Classification

- Compute global context vector for each frame $t$:
  - $h_t^f = BLSTM_{frame}(h_{t-1}^f, h_{t+1}^f, f_t)$
Event Classification

- Compute global context vector for each frame $t$:
  - $h_t^f = BLSTM_{frame}(h_{t-1}^f, h_{t+1}^f, f_t)$

- Next compute hidden state of event at time $t$:
  - $h_t^e = LSTM(h_{t-1}^e, h_t^f, a_t)$
  - $a_t$ is the feature vector for the players from the attention model
Event Classification

• Compute global context vector for each frame $t$:
  - $h_t^f = BLSTM_{frame}(h_{t-1}^f, h_{t+1}^f, f_t)$

• Next compute hidden state of event at time $t$:
  - $h_t^e = LSTM(h_{t-1}^e, h_t^f, a_t)$
  - $a_t$ is the feature vector for the players from the attention model

• Predict class label using $w_k^T h_t^e$

• Squared Hinge Loss function:
  - $L = \frac{1}{2} \sum_{t=1}^{T} \sum_{k=1}^{K} \max(0, 1 - y_k w_k^T h_t^e)^2$
  - $y_k$ is 1 if the video belongs to class $k$ and -1 otherwise
Attention

• How do we get the feature vector $a_t$ for the players from the attention model?
Attention Models (with tracking)

- Attention model with KLT tracking for player $i$ and frame $t$:
  - $h_{ti}^p = BLSTM_{track}(h_{t-1,i}^p, h_{t+1,i}^p, p_{ti})$
  - $a_{t}^{track} = \sum_{i=1}^{N_t} \gamma_{ti}^p h_{ti}^p$
  - $\gamma_{ti}^{track} = \text{softmax}(\phi(h_t^f, h_{ti}^p, h_{t-1}; \tau))$

- $a_t$: weighted combination over players in frame $t$
- $\gamma_{ti}$: attention weights
- $N_t$: # player detections in frame $t$
- $\phi()$: multilayer perceptron
- $\tau$: softmax temperature
Attention Models (without tracking)

- Attention model without tracking:
  - $a_t^{\text{notrack}} = \sum_{i=1}^{N_t} \gamma_{ti}^{\text{notrack}} p_{ti}$
  - $\gamma_{ti}^{\text{notrack}} = \text{softmax} (\phi(h_t^i, p_{ti}, h_{t-1}^e); \tau)$

- $a_t$: weighted combination over players in frame $t$
- $\gamma_{ti}$: attention weights
- $N_t$: # player detections in frame $t$
- $\phi()$: multilayer perceptron
- $\tau$: softmax temperature
- $p_{ti}$: player feature vector from Inception7
Strengths/Weaknesses of Approach

• Strengths:
  • Attention focuses on key players

• Weaknesses:
  • Need all the frames in a clip (cannot do online detection)
  • Model tends to be reluctant to switch attention between players in a scene
Results (Event Classification)

- Here we compare the ability to classify isolated video clips into 11 classes.
- Attention is particularly good for shot-based events where attending to the shot making person or defenders can be useful.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3-point succ.</td>
<td>0.370</td>
<td>0.428</td>
<td>0.117</td>
<td>0.237</td>
<td>0.462</td>
<td>0.469</td>
<td>0.545</td>
<td>0.583</td>
<td>0.600</td>
</tr>
<tr>
<td>3-point fail.</td>
<td>0.501</td>
<td>0.481</td>
<td>0.282</td>
<td>0.335</td>
<td>0.564</td>
<td>0.614</td>
<td>0.702</td>
<td>0.668</td>
<td>0.738</td>
</tr>
<tr>
<td>fr-throw succ.</td>
<td>0.778</td>
<td>0.703</td>
<td>0.642</td>
<td>0.597</td>
<td>0.876</td>
<td>0.885</td>
<td>0.809</td>
<td>0.892</td>
<td>0.882</td>
</tr>
<tr>
<td>fr-throw fail.</td>
<td>0.365</td>
<td>0.623</td>
<td>0.319</td>
<td>0.318</td>
<td>0.584</td>
<td><strong>0.700</strong></td>
<td>0.641</td>
<td>0.671</td>
<td>0.516</td>
</tr>
<tr>
<td>layup succ.</td>
<td>0.283</td>
<td>0.300</td>
<td>0.195</td>
<td>0.257</td>
<td>0.463</td>
<td>0.416</td>
<td>0.472</td>
<td>0.489</td>
<td>0.500</td>
</tr>
<tr>
<td>layup fail.</td>
<td>0.278</td>
<td>0.311</td>
<td>0.185</td>
<td>0.247</td>
<td>0.386</td>
<td>0.305</td>
<td>0.388</td>
<td>0.426</td>
<td>0.445</td>
</tr>
<tr>
<td>2-point succ.</td>
<td>0.136</td>
<td>0.233</td>
<td>0.078</td>
<td>0.224</td>
<td>0.257</td>
<td>0.228</td>
<td>0.255</td>
<td>0.281</td>
<td>0.341</td>
</tr>
<tr>
<td>2-point fail.</td>
<td>0.303</td>
<td>0.285</td>
<td>0.254</td>
<td>0.299</td>
<td>0.378</td>
<td><strong>0.473</strong></td>
<td>0.442</td>
<td>0.471</td>
<td>0.471</td>
</tr>
<tr>
<td>sl. dunk succ.</td>
<td>0.197</td>
<td>0.171</td>
<td>0.047</td>
<td>0.112</td>
<td>0.285</td>
<td>0.107</td>
<td>0.186</td>
<td>0.210</td>
<td>0.291</td>
</tr>
<tr>
<td>sl. dunk fail.</td>
<td>0.004</td>
<td>0.010</td>
<td>0.004</td>
<td>0.005</td>
<td><strong>0.027</strong></td>
<td>0.006</td>
<td>0.010</td>
<td>0.006</td>
<td>0.004</td>
</tr>
<tr>
<td>steal</td>
<td>0.555</td>
<td>0.473</td>
<td>0.303</td>
<td>0.843</td>
<td>0.876</td>
<td>0.843</td>
<td>0.894</td>
<td>0.886</td>
<td>0.893</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td>0.343</td>
<td>0.365</td>
<td>0.221</td>
<td>0.316</td>
<td>0.469</td>
<td>0.452</td>
<td>0.489</td>
<td>0.505</td>
<td><strong>0.516</strong></td>
</tr>
</tbody>
</table>

Table 2. Mean average precision for event classification given isolated clips.
Results (Event Detection)

- Here we compare the ability to temporally localize events in untrimmed videos using a 4 second sliding window through all the videos.

- Here, a steal event is particularly challenging as it is often mistaken for a pass.

- Combining the player features by averaging without using attention performs very good as well.
  - Possibly because the algorithm has difficulty changing attention since we are dealing with untrimmed videos.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3-point succ.</td>
<td>0.194</td>
<td>0.203</td>
<td>0.123</td>
<td>0.230</td>
<td>0.251</td>
<td><strong>0.268</strong></td>
<td>0.263</td>
<td>0.239</td>
</tr>
<tr>
<td>3-point fail.</td>
<td>0.393</td>
<td>0.376</td>
<td>0.311</td>
<td>0.505</td>
<td>0.526</td>
<td>0.521</td>
<td>0.556</td>
<td>0.600</td>
</tr>
<tr>
<td>free-throw succ.</td>
<td>0.585</td>
<td>0.621</td>
<td>0.542</td>
<td>0.741</td>
<td>0.777</td>
<td><strong>0.811</strong></td>
<td>0.788</td>
<td>0.810</td>
</tr>
<tr>
<td>free-throw fail.</td>
<td>0.231</td>
<td>0.277</td>
<td>0.458</td>
<td>0.434</td>
<td>0.470</td>
<td>0.444</td>
<td>0.468</td>
<td>0.405</td>
</tr>
<tr>
<td>layup succ.</td>
<td>0.258</td>
<td>0.290</td>
<td>0.175</td>
<td>0.492</td>
<td>0.402</td>
<td>0.489</td>
<td>0.494</td>
<td>0.512</td>
</tr>
<tr>
<td>layup fail.</td>
<td>0.141</td>
<td>0.200</td>
<td>0.151</td>
<td>0.187</td>
<td>0.142</td>
<td>0.139</td>
<td>0.207</td>
<td>0.208</td>
</tr>
<tr>
<td>2-point succ.</td>
<td>0.161</td>
<td>0.170</td>
<td>0.126</td>
<td>0.352</td>
<td>0.371</td>
<td><strong>0.417</strong></td>
<td>0.366</td>
<td>0.400</td>
</tr>
<tr>
<td>2-point fail.</td>
<td>0.358</td>
<td>0.339</td>
<td>0.226</td>
<td>0.544</td>
<td>0.578</td>
<td><strong>0.684</strong></td>
<td>0.619</td>
<td>0.674</td>
</tr>
<tr>
<td>slam dunk succ.</td>
<td>0.137</td>
<td>0.275</td>
<td>0.114</td>
<td>0.428</td>
<td>0.566</td>
<td>0.457</td>
<td>0.576</td>
<td><strong>0.555</strong></td>
</tr>
<tr>
<td>slam dunk fail.</td>
<td>0.007</td>
<td>0.006</td>
<td>0.003</td>
<td><strong>0.122</strong></td>
<td>0.059</td>
<td>0.009</td>
<td>0.005</td>
<td>0.045</td>
</tr>
<tr>
<td>steal</td>
<td>0.242</td>
<td>0.255</td>
<td>0.187</td>
<td><strong>0.359</strong></td>
<td>0.348</td>
<td>0.313</td>
<td>0.340</td>
<td>0.339</td>
</tr>
<tr>
<td>Mean</td>
<td>0.246</td>
<td>0.273</td>
<td>0.219</td>
<td>0.400</td>
<td>0.408</td>
<td><strong>0.414</strong></td>
<td>0.426</td>
<td><strong>0.435</strong></td>
</tr>
</tbody>
</table>

Table 3. Mean average precision for event detection given untrimmed videos.
Results (Attention)

- Attended player is in cyan and ball is in yellow
- Results show that model attends to the player making the shot at the beginning
Results (Attention Heatmap)

- Distribution of attention shows initially attention focuses on shooter and then disperses later in the event.
Wrap Up

• Questions?
• Suggestions?