Activity Recognition

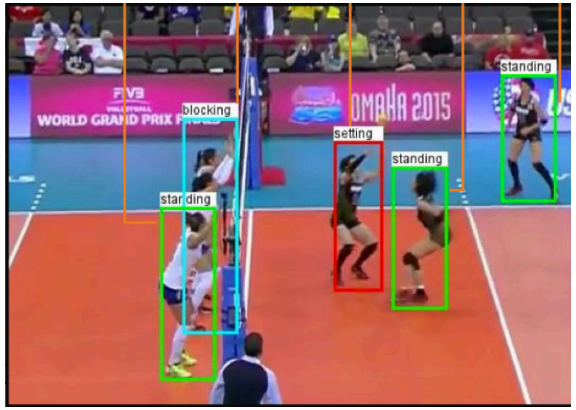
JUSTIN LIANG
MARCH 27, 2016

Agenda

- End-to-end Learning of Action Detection from Frame Glimpses in Videos. S. Yeung, O. Russakovsky, G. Mori, L. Fei-Fei. CVPR 2016.
- 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]

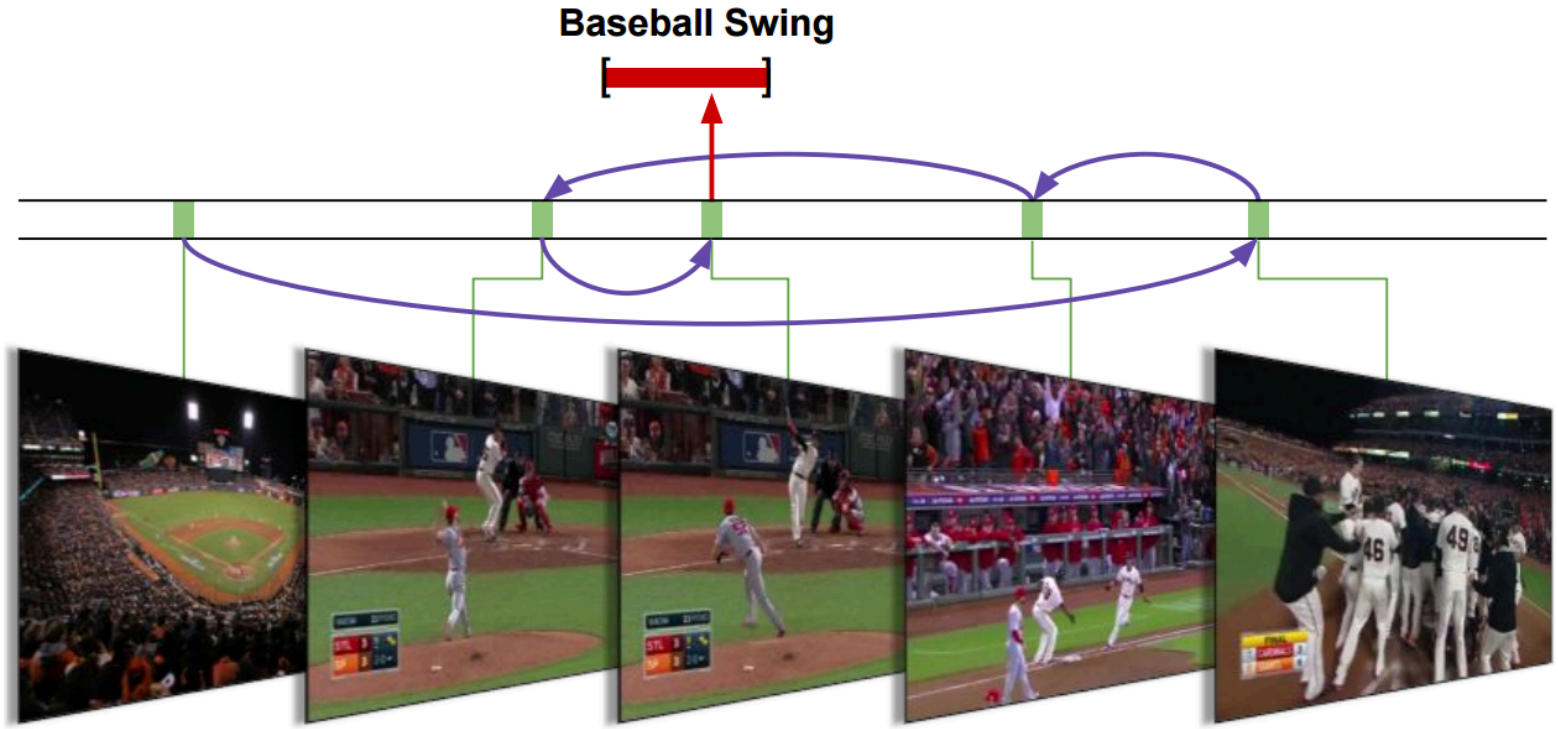
End-to-end Learning of Action Detection from Frame Glimpses in Videos

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 out perform state of the art results

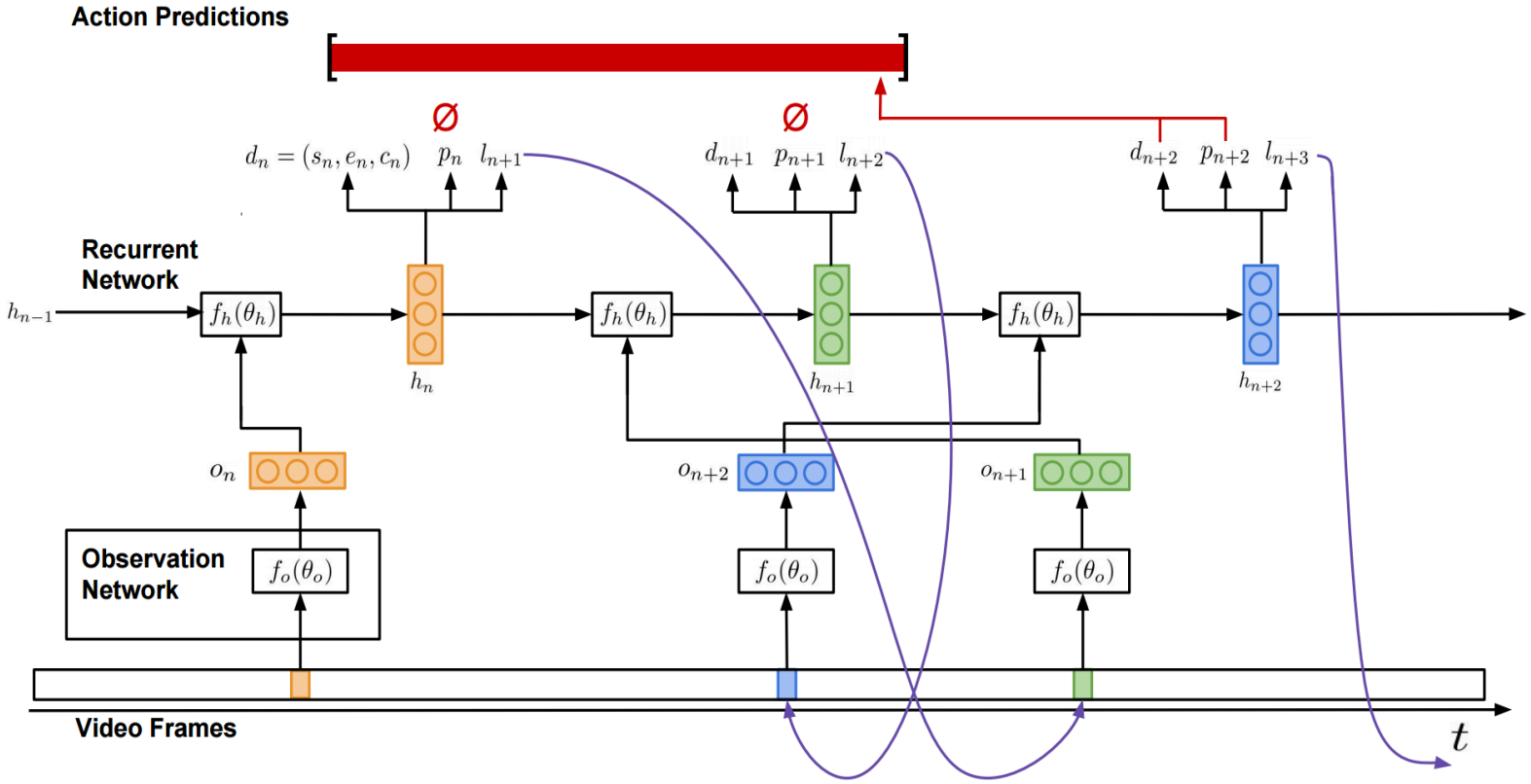
Approach

- 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.



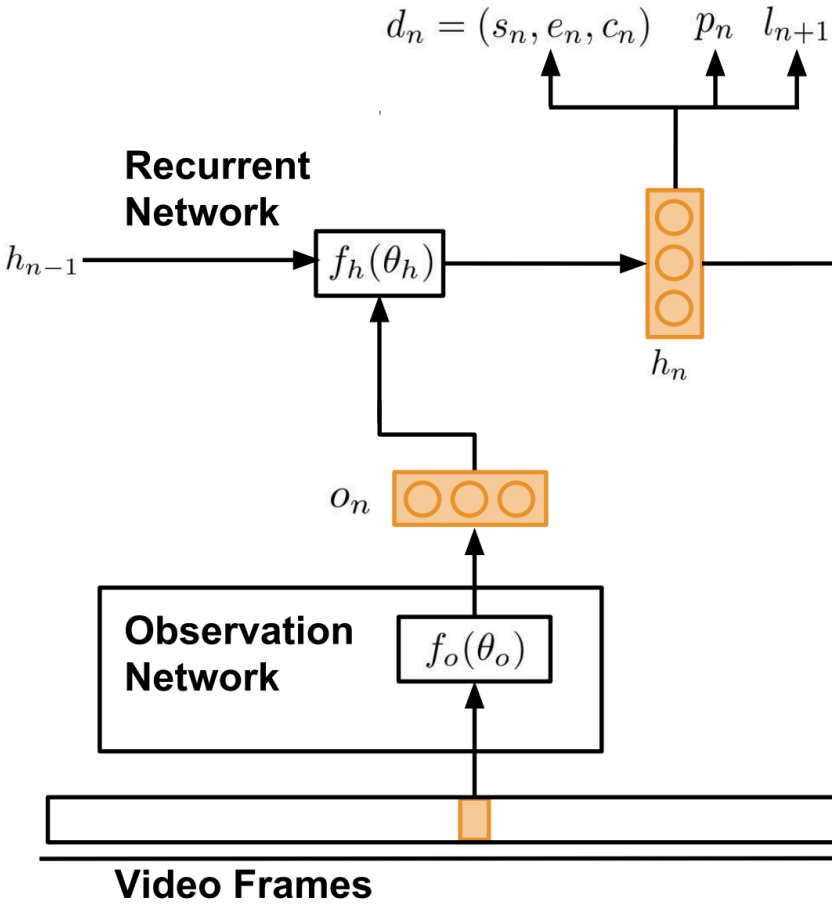
Approach (Pipeline)

- 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, $l_n \in [0,1]$



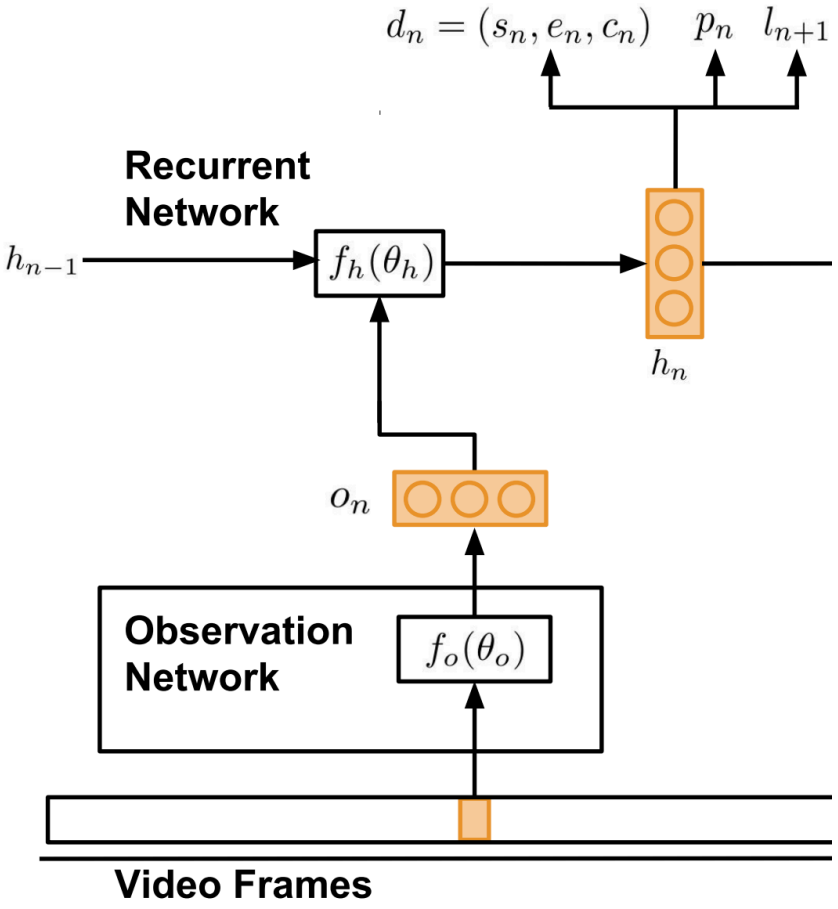
Observation Network

- 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



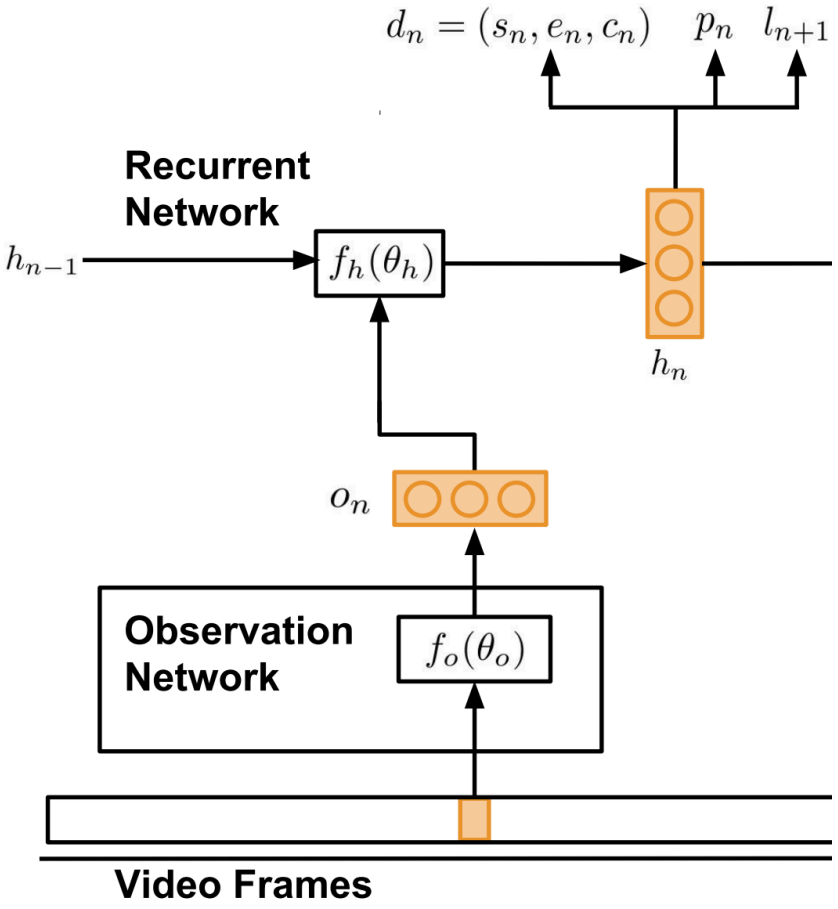
Recurrent Network

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



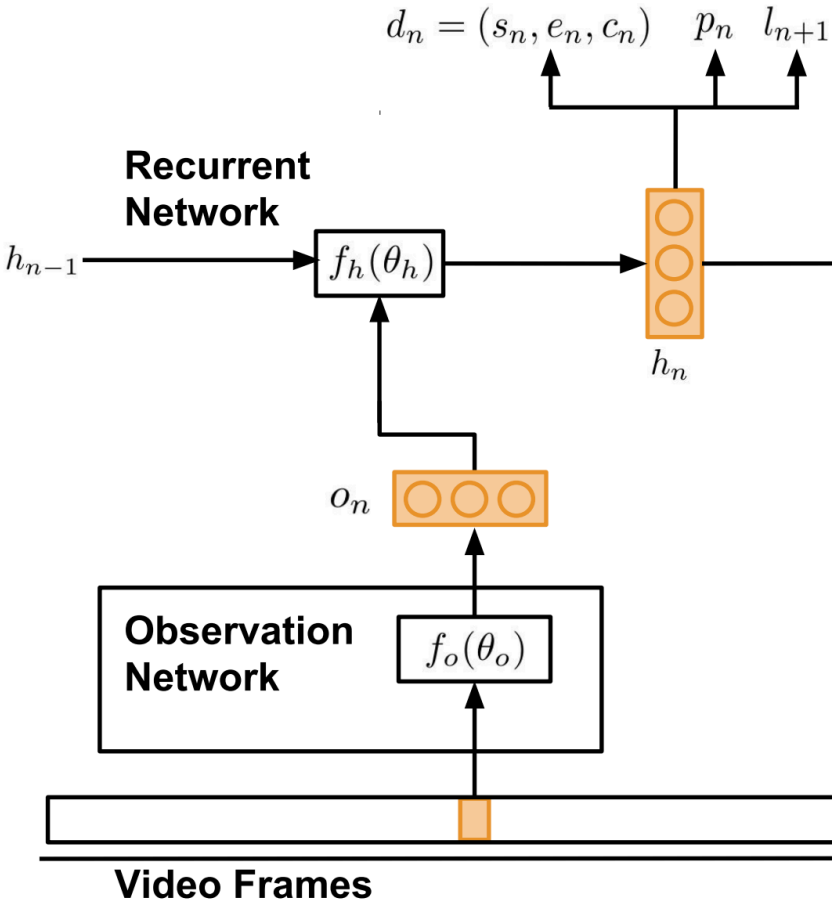
Recurrent Network

- Observation features o_n and previous internal hidden state h_{n-1} are inputs to the recurrent network f_h which is parameterized by θ_h to produce h_n
- Candidate detection d_n:
  - $d_n = f_d(h_n; \theta_d)$, f_d is a fully connected layer



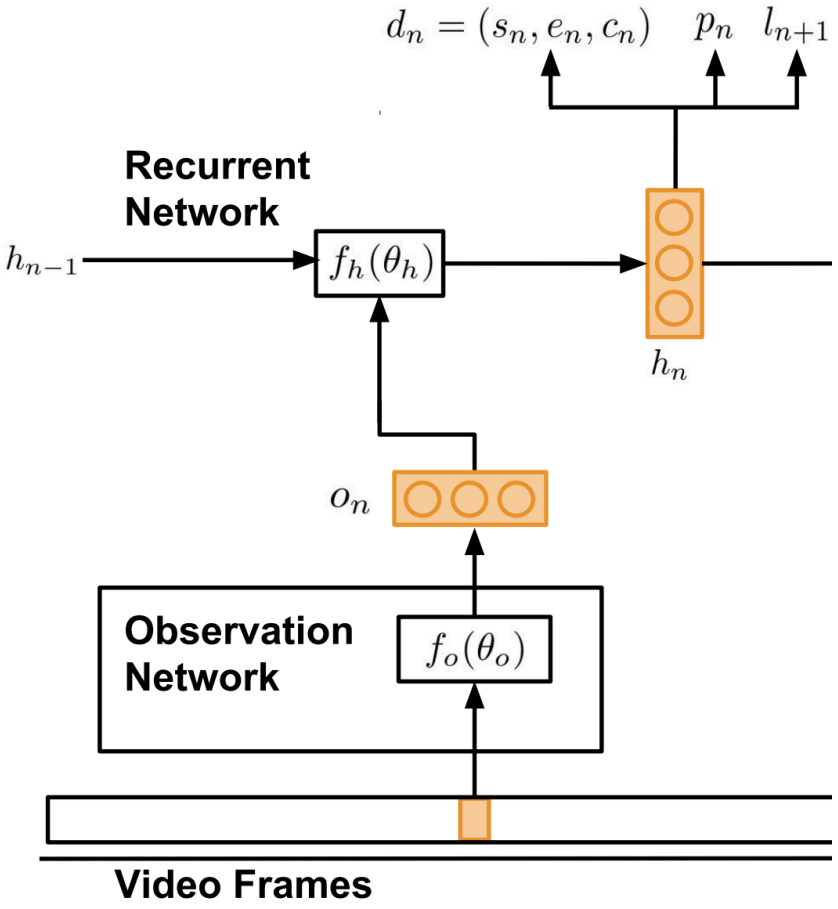
Recurrent Network

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

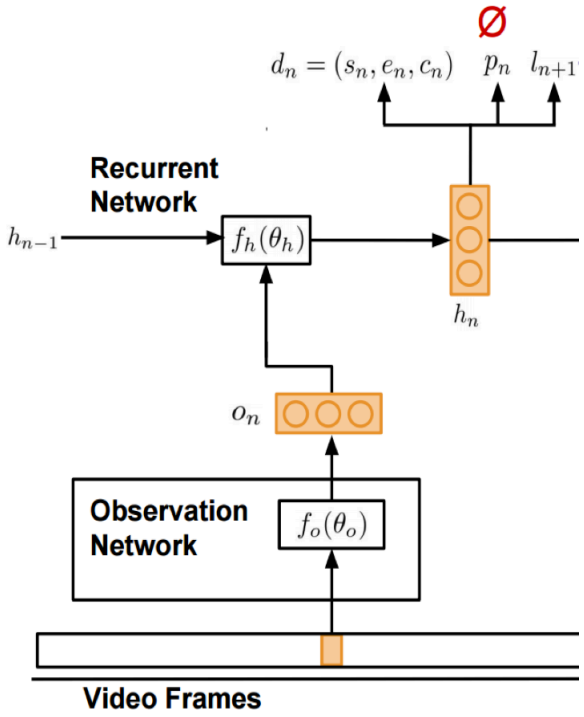
- Goal is to train three outputs: candidate detection d_n, prediction indicator p_n, location of next observation l_{n+1}
  - This is difficult due to the challenges of designing suitable loss and reward functions and handling non-differentiable model components
- We use backpropagation to train d_n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

Training (Candidate Detection d_n)

- Match each candidate detection $\mathrm{D} = \{d_n | n = 1, \dots, N\}$ from recurrent network to ground truth $g_{1,\dots,M}$
- Matching function:
  - $y_{nm} = \begin{cases} 1 \; if \; m = argmin_{j=1,\dots,M} dist(l_n, g_j) \\ 0 \; otherwise \end{cases}$
  - $g_j = (s_j, e_j)$
  - $dist(l_n, g_j) = \min(|s_j - l_n|, |e_j - l_n|)$

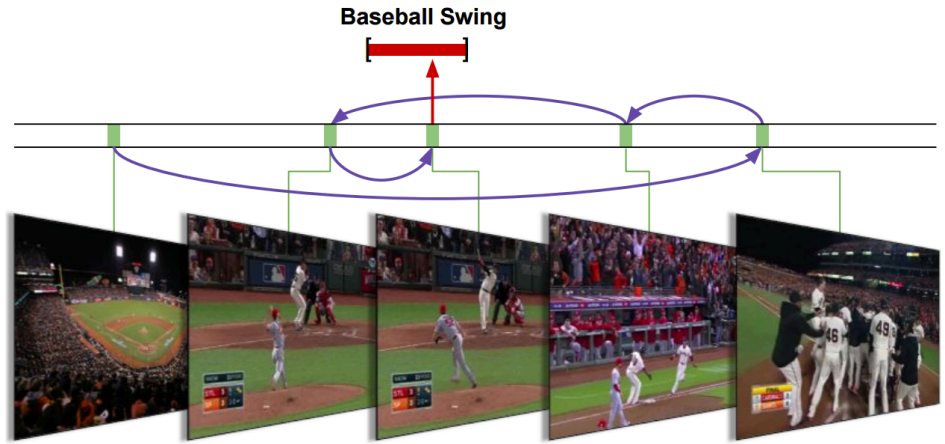
Training (Candidate Detection d_n)

- Match each candidate detection $\mathrm{D} = \{d_n | n = 1, \dots, N\}$ from recurrent network to ground truth $g_{1,\dots,M}$
- Matching function:
  - $y_{nm} = \begin{cases} 1 \; if \; m = argmin_{j=1,\dots,M} dist(l_n, g_j) \\ 0 \; otherwise \end{cases}$
  - $g_j = (s_j, e_j)$
  - $dist(l_n, g_j) = \min(|s_j - l_n|, |e_j - l_n|)$
- Loss function:
  - $\sum_n L_{cls}(d_n) + \gamma \sum_n \sum_m \mathbb{I}[y_{nm} = 1] L_{loc}(d_n, g_m)$
  - $L_{cls}(d_n)$: cross entropy loss on detection confidence c_n
  - $L_{loc}(d_n, g_m)$: L2 loss to further minimize distance $\|(s_n, e_n) - (s_m, e_m)\|$
- Optimize loss using backpropagation



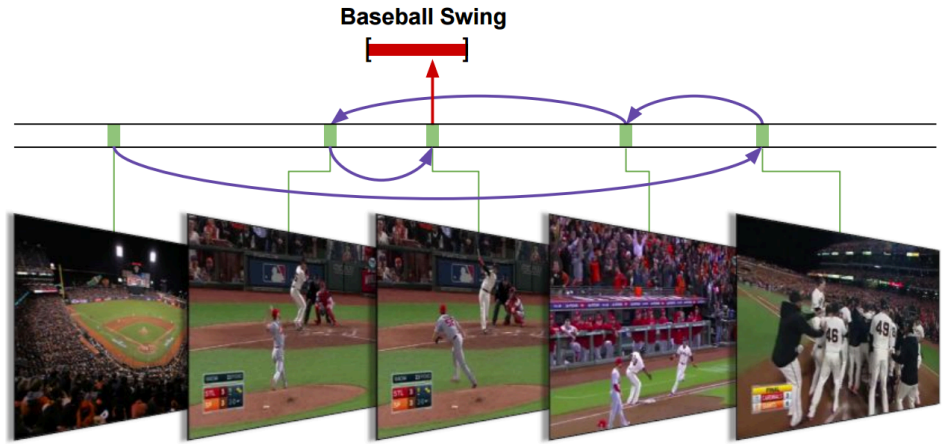
Training (Location l_{n+1} and Prediction Indicator p_n)

- 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)

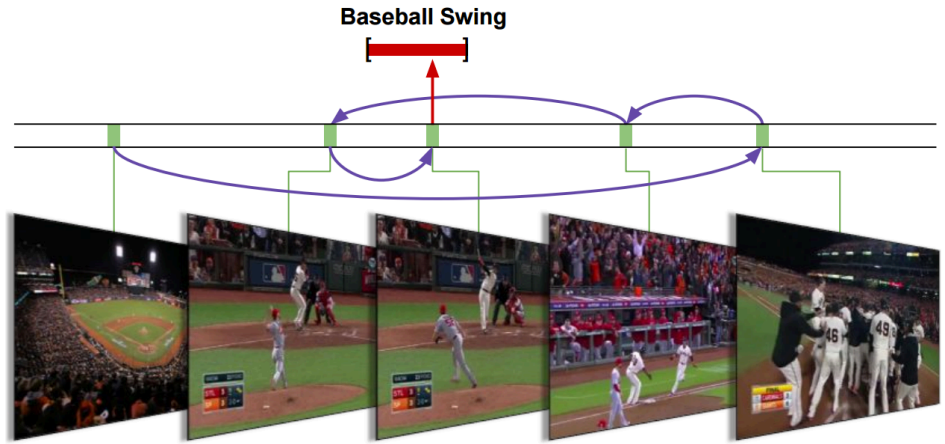
- Use REINFORCE to learn observation and emission policies
- REINFORCE:
  - Objective: $J(\theta) = \sum_{a \in \mathcal{A}} p_\theta(a) r(a)$
    - $\mathcal{A}$: space of action sequences
    - $p_\theta(a)$: probability of action
    - $r(a)$: reward
  - Gradient: $\nabla J(\theta) = \sum_{a \in \mathcal{A}} p_\theta(a) \nabla \log p_\theta(a) r(a)$
    - 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: $J(\theta) = \sum_{a \in \mathcal{A}} p_\theta(a) r(a)$
    - $\mathcal{A}$: space of action sequences
    - $p_\theta(a)$: probability of action
    - $r(a)$: reward
  - Gradient: $\nabla J(\theta) = \sum_{a \in \mathcal{A}} p_\theta(a) \nabla \log p_\theta(a) r(a)$
    - Use Monte Carlo to approximate:
      - $\nabla J(\theta) \approx \frac{1}{K} \sum_{i=1}^{K} \sum_{n=1}^{N} \nabla \log \pi_\theta \left(a_n^i | h_{1:n}^i, a_{1:n-1}^i\right) R_n^i$
        - K interaction sequences
        - N RNN time steps
        - π_θ: 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
  - $r_N = \begin{cases} R_p \; if \; M > 0 \; and N_p = 0 \\ N_+ R_+ + N_- R_- \; otherwise \end{cases}$
    - N_p: # predictions emitted by agent
    - N_+, R_+: # true positive predictions and reward
    - N_-, R_-: # false positive preditions 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

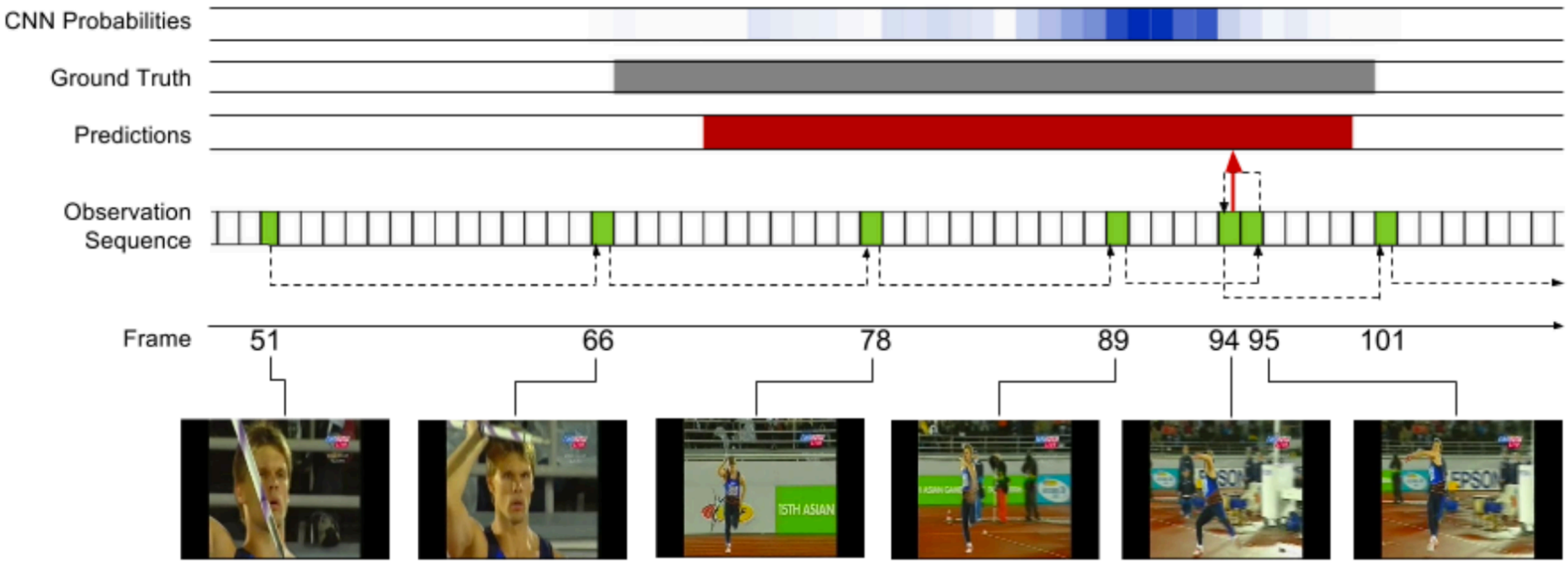
- Strengths:
  - Do not need to look at all the frames
  - End-to-end learning
- Weaknesses:
  - Need all the frames in a clip (cannot do online detection)
  - Can be difficult to learn observation policy if event contains less discriminative movements

Results

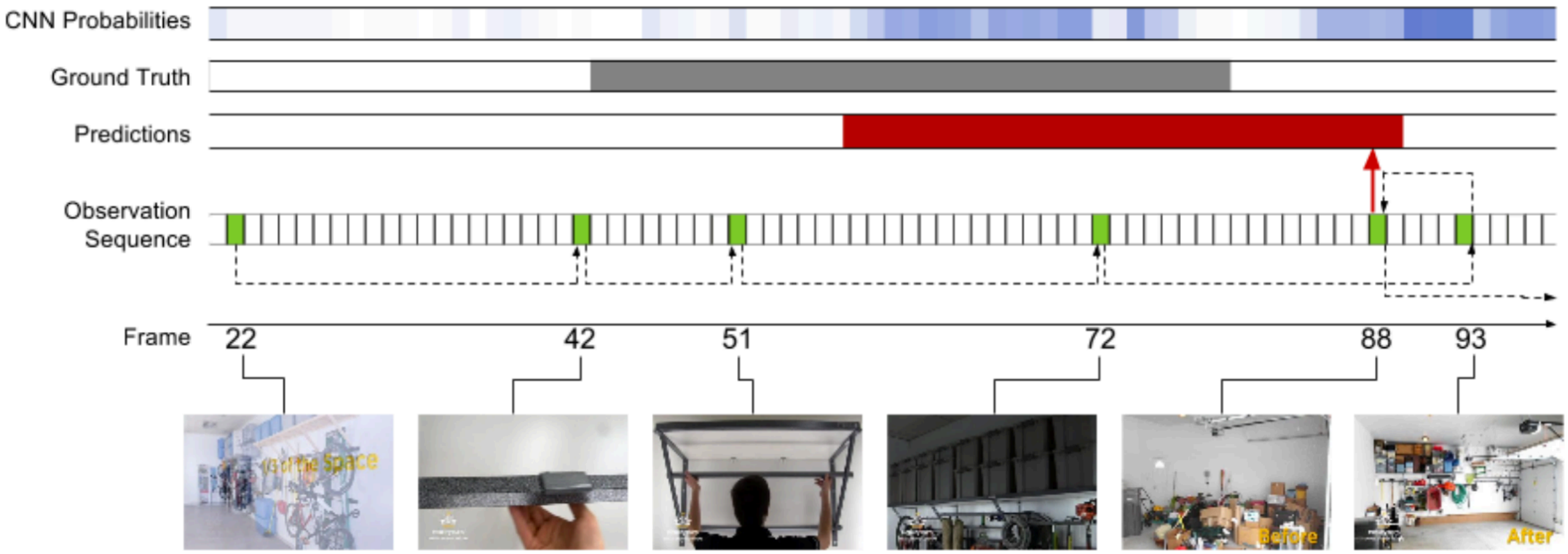
- Results from THUMOS'14 comparing with top 3 performers. mAP is reported for different IOU thresholds α
- Ablation studies show that without localization regression and where to observe next, results are significantly worse

	α=0.5	α=0.4	α=0.3	α=0.2	α=0.1
Karaman et al. [13]	0.9	1.4	2.1	3.4	4.6
Wang et al. [39]	8.3	11.7	14.0	17.0	18.2
Oneata et al. [22]	14.4	20.8	27.0	33.6	36.6
Ours (full)	**17.1**	**26.4**	**36.0**	**44.0**	**48.9**

Results (Learned Observation Policy)



Results (Learned Observation Policy)



Future Direction

- Learn joint spatio-temporal observation policies

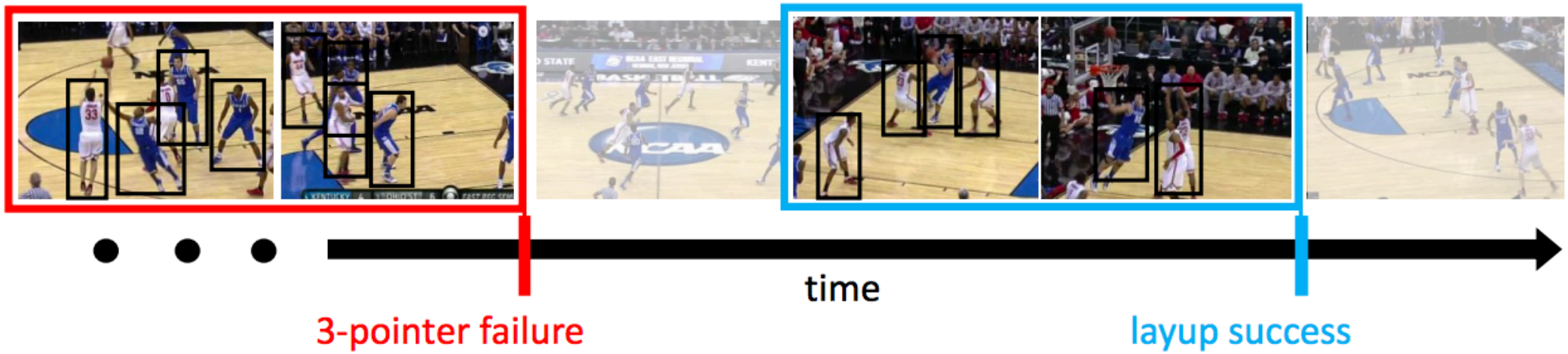
Detecting Events and Key Actors in Multi-Person Videos

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.

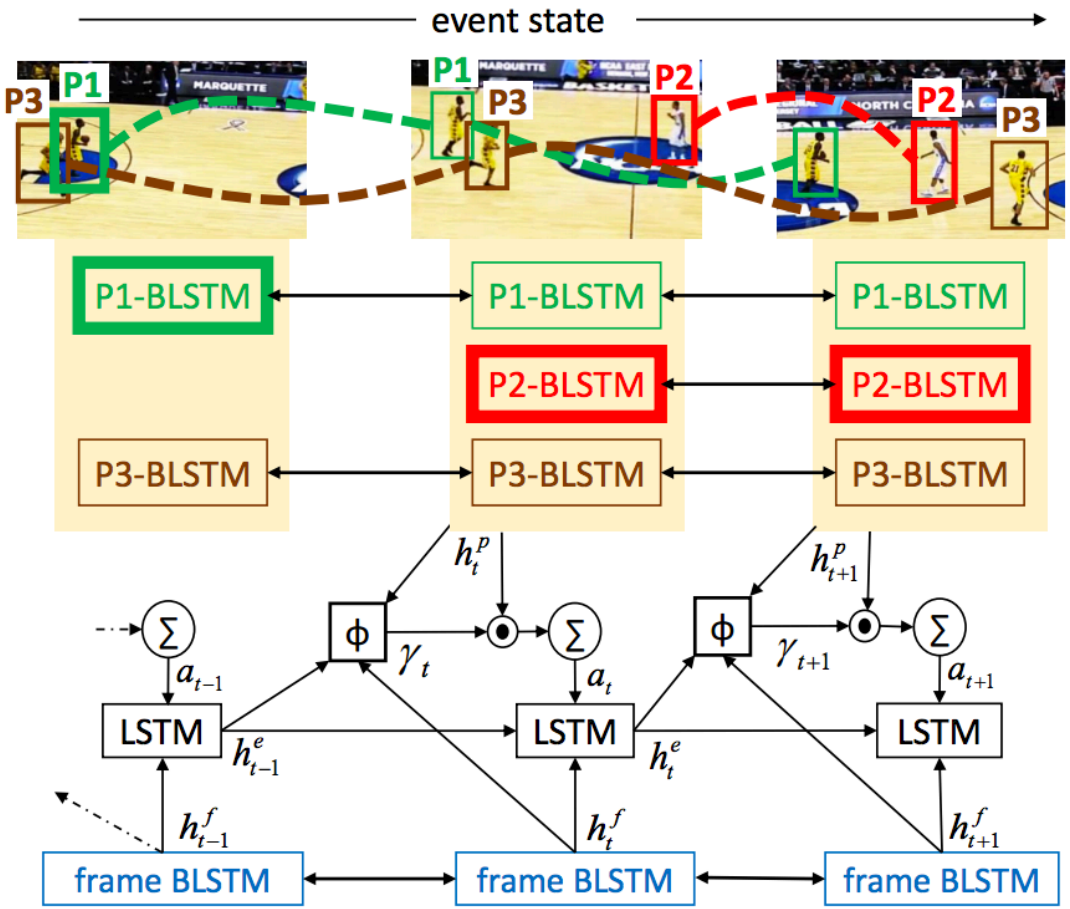


Approach

- 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.

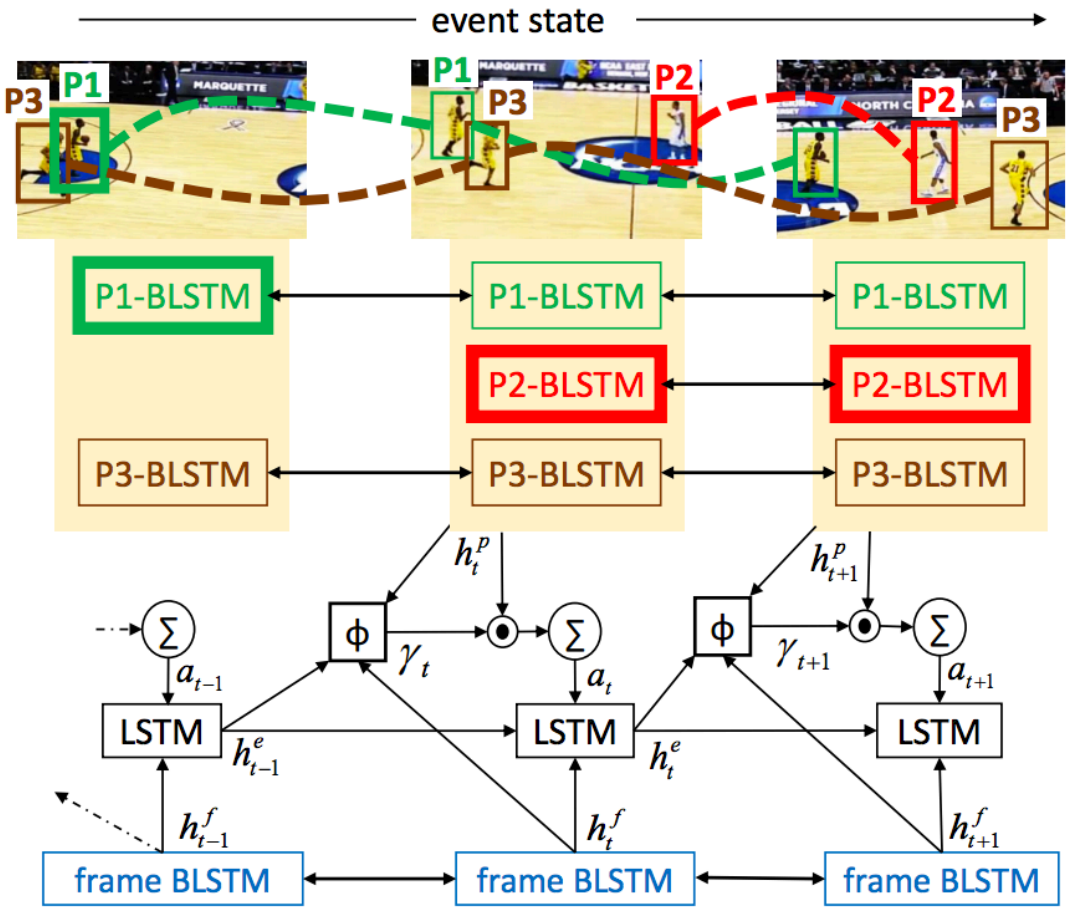
Event Classification

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

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

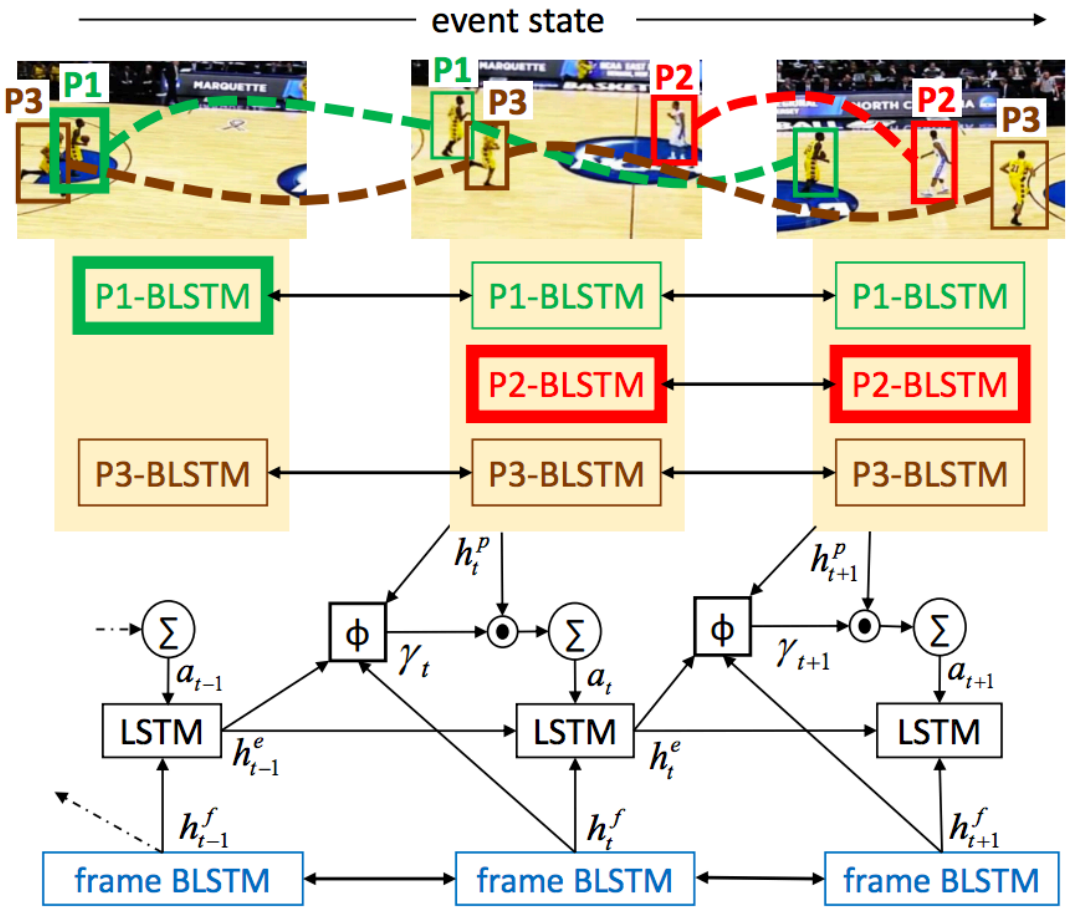
Event Classification

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

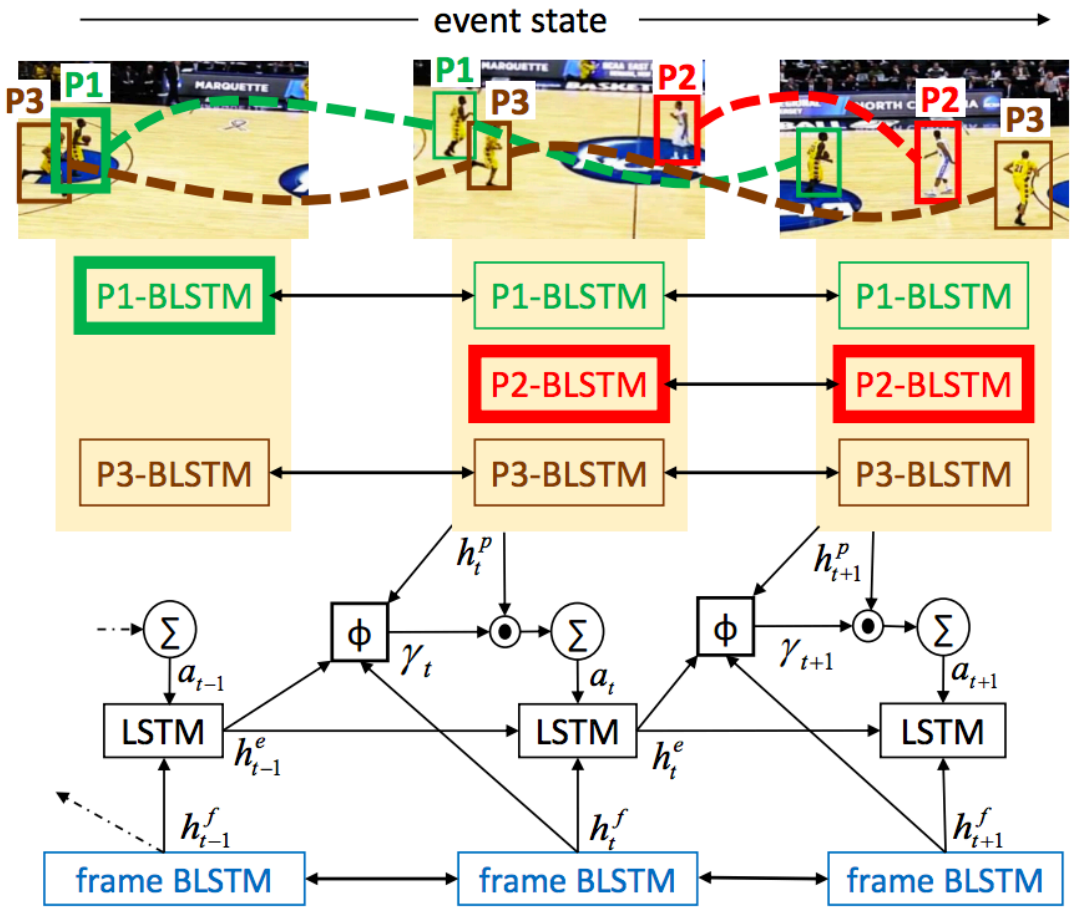


Attention

- How do we get the feature vector a_t for the players from the attention model?

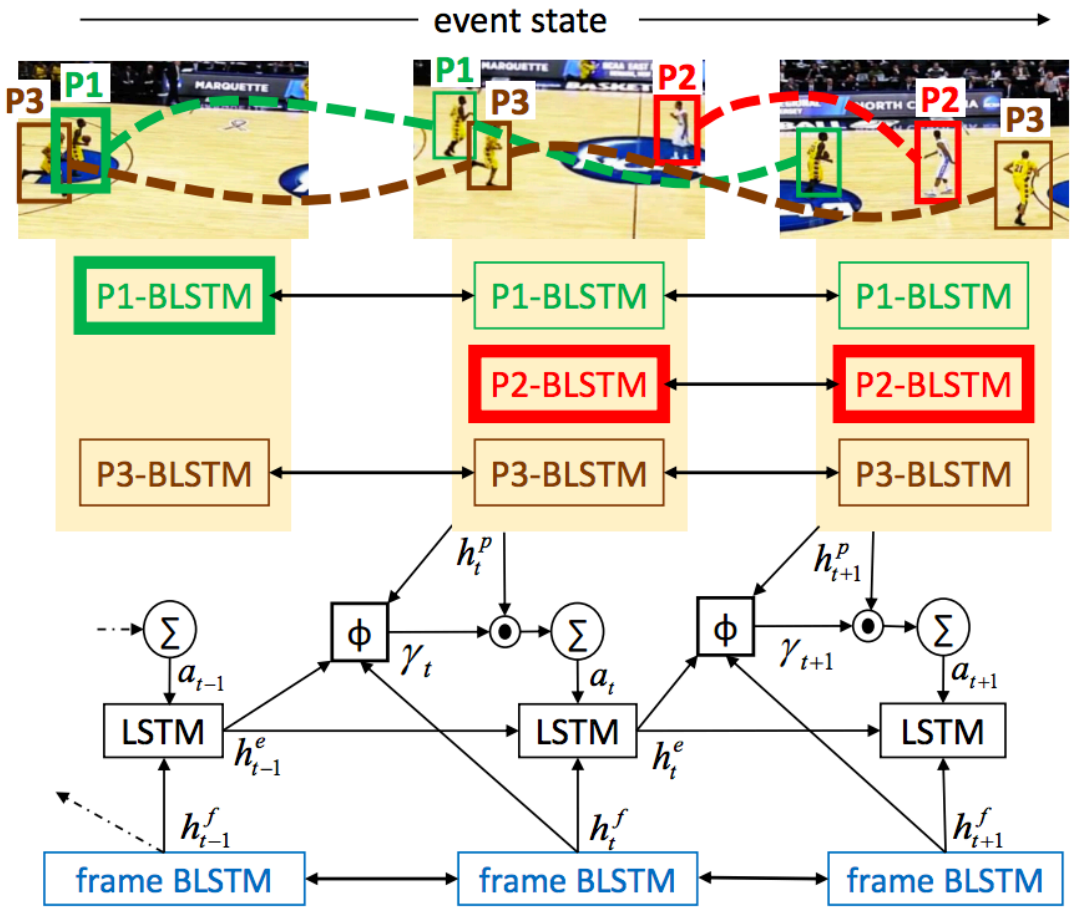
Attention Models (with tracking)

- Attention model with KLT tracking for player i and frame t:
  - $h_{ti}^{p} = BLSTM_{track}(h_{t-1,i}^{p}, h_{t+1,i}^{p}, p_{ti})$
  - $a_t^{track} = \sum_{i=1}^{N_t} \gamma_{ti}^{track} h_{ti}^{p}$
  - $\gamma_{ti}^{track} = softmax(\phi(h_t^f, h_{ti}^p, h_{t-1}^e); \tau)$
- a_t: weighted combination over players in frame t
- γ_{ti} : attention weights
- N_t: # player detections in frame t
- $\phi()$: multilayer perceptron
- τ: softmax temperature



Attention Models (without tracking)

- Attention model without tracking:
  - $a_t^{notrack} = \sum_{i=1}^{N_t} \gamma_{ti}^{notrack} p_{ti}$
  - $\gamma_{ti}^{notrack} = softmax(\phi(h_t^f, p_{ti}, h_{t-1}^e); \tau)$
- a_t: weighted combination over players in frame t
- γ_{ti}: attention weights
- N_t: # player detections in frame t
- $\phi()$: multilayer perceptron
- τ: 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

Event	IDT[64]	IDT[64] player	C3D [60]	MIL[1]	LRCN [8]	Only player	Avg. player	Our no track	Our track
3-point succ.	0.370	0.428	0.117	0.237	0.462	0.469	0.545	0.583	**0.600**
3-point fail.	0.501	0.481	0.282	0.335	0.564	0.614	0.702	0.668	**0.738**
fr-throw succ.	0.778	0.703	0.642	0.597	0.876	0.885	0.809	**0.892**	0.882
fr-throw fail.	0.365	0.623	0.319	0.318	0.584	**0.700**	0.641	0.671	0.516
layup succ.	0.283	0.300	0.195	0.257	0.463	0.416	0.472	0.489	**0.500**
layup fail.	0.278	0.311	0.185	0.247	0.386	0.305	0.388	0.426	**0.445**
2-point succ.	0.136	0.233	0.078	0.224	0.257	0.228	0.255	0.281	**0.341**
2-point fail.	0.303	0.285	0.254	0.299	0.378	0.391	**0.473**	0.442	0.471
sl. dunk succ.	0.197	0.171	0.047	0.112	0.285	0.107	0.186	0.210	**0.291**
sl. dunk fail.	0.004	0.010	0.004	0.005	**0.027**	0.006	0.010	0.006	0.004
steal	0.555	0.473	0.303	0.843	0.876	0.843	**0.894**	0.886	0.893
Mean	0.343	0.365	0.221	0.316	0.469	0.452	0.489	0.505	**0.516**

Table 2. Mean average precision for event *classification* given isolated clips.

Results (Event Detection)

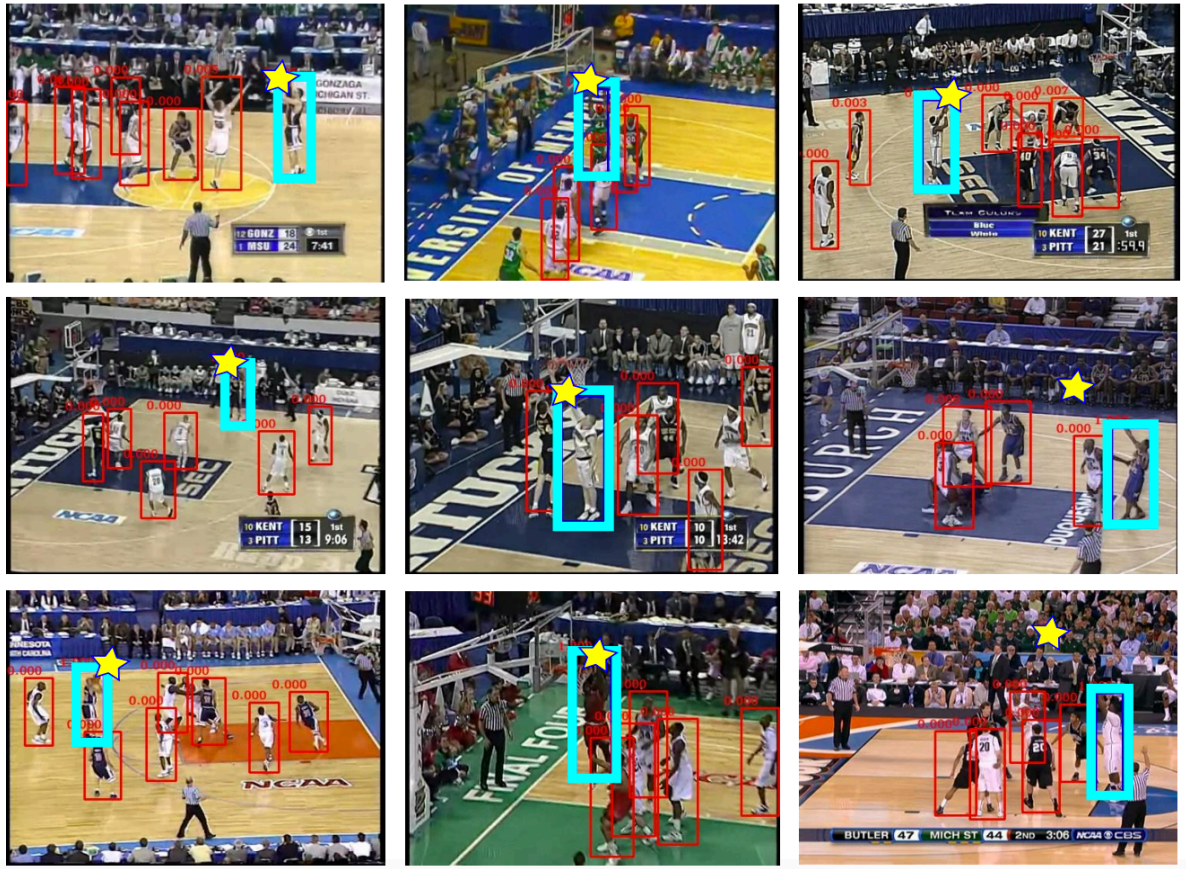
- Here we compare the ability to temporally localize events in untrimmed videos using a 4 second sliding window through all the videos
- Here, a steal event is particularly challenging as it is often mistaken for a pass
- Combining the player features by averaging without using attention performs very good as well
  - Possibly because the algorithm has difficulty changing attention since we are dealing with untrimmed videos

Event	IDT[64]	IDT player[64]	C3D [60]	LRCN [8]	Only player	Avg. player	Attn no track	Attn track
3-point succ.	0.194	0.203	0.123	0.230	0.251	**0.268**	0.263	0.239
3-point fail.	0.393	0.376	0.311	0.505	0.526	0.521	0.556	**0.600**
free-throw succ.	0.585	0.621	0.542	0.741	0.777	**0.811**	0.788	0.810
free-throw fail.	0.231	0.277	0.458	0.434	**0.470**	0.444	0.468	0.405
layup succ.	0.258	0.290	0.175	0.492	0.402	0.489	0.494	**0.512**
layup fail.	0.141	0.200	0.151	0.187	0.142	0.139	0.207	**0.208**
2-point succ.	0.161	0.170	0.126	0.352	0.371	**0.417**	0.366	0.400
2-point fail.	0.358	0.339	0.226	0.544	0.578	**0.684**	0.619	0.674
slam dunk succ.	0.137	0.275	0.114	0.428	0.566	0.457	0.576	**0.555**
slam dunk fail.	0.007	0.006	0.003	**0.122**	0.059	0.009	0.005	0.045
steal	0.242	0.255	0.187	**0.359**	0.348	0.313	0.340	0.339
Mean	0.246	0.273	0.219	0.400	0.408	0.414	0.426	**0.435**

Table 3. Mean average precision for event *detection* given untrimmed videos.

Results (Attention)

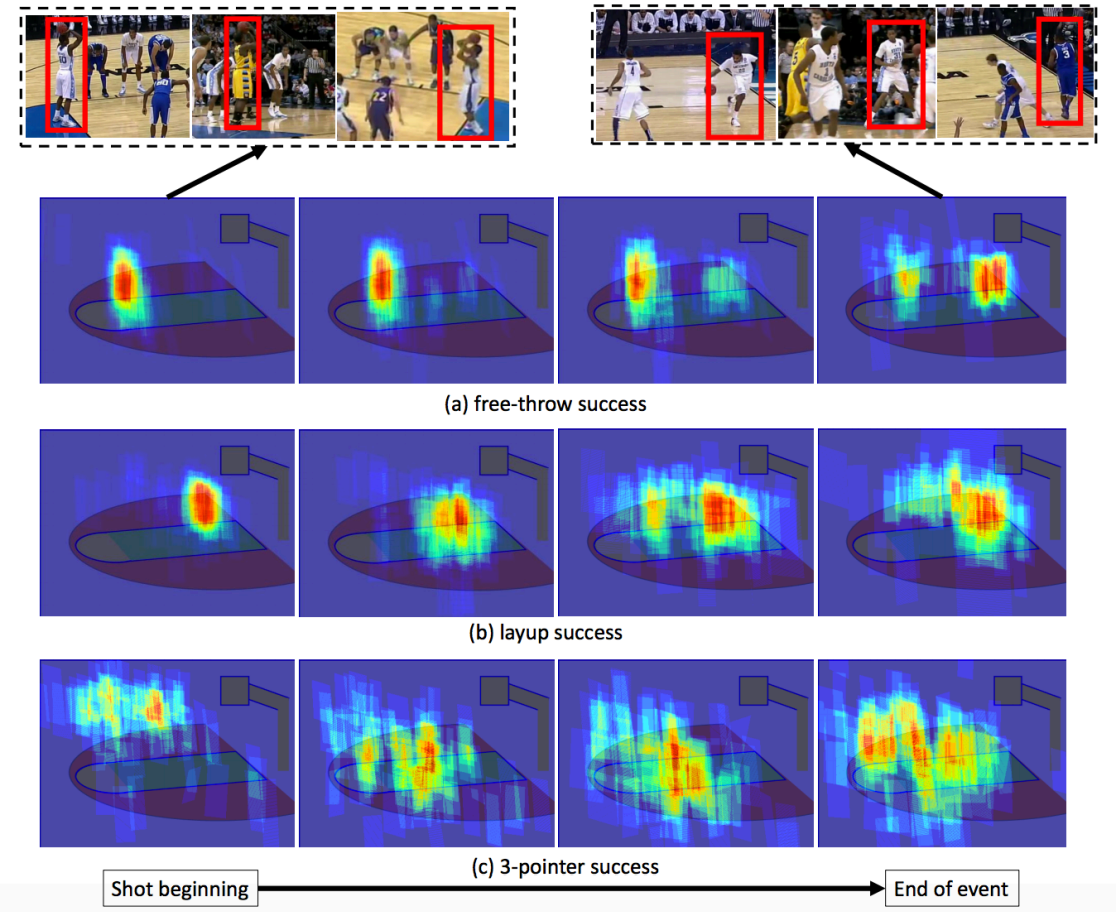
- Attended player is in cyan and ball is in yellow
- Results show that model attends to the player making the shot at the beginning



3-pointer success | slam dunk success | free-throw failure

Results (Attention Heatmap)

- Distribution of attention shows initially attention focusses on shooter and then disperses later in the event



Wrap Up

- Questions?
- Suggestions?