

TrackNet

– A Deep Learning Network for Tracking High-speed and Tiny Objects in Sports Applications

To track the tennis ball from broadcast videos in which the ball images are small, blurry, and sometimes with afterimage tracks or even invisible.

The proposed heatmap-based deep learning network is trained to not only recognize the ball image from a single frame but also learn flying patterns from consecutive frames.

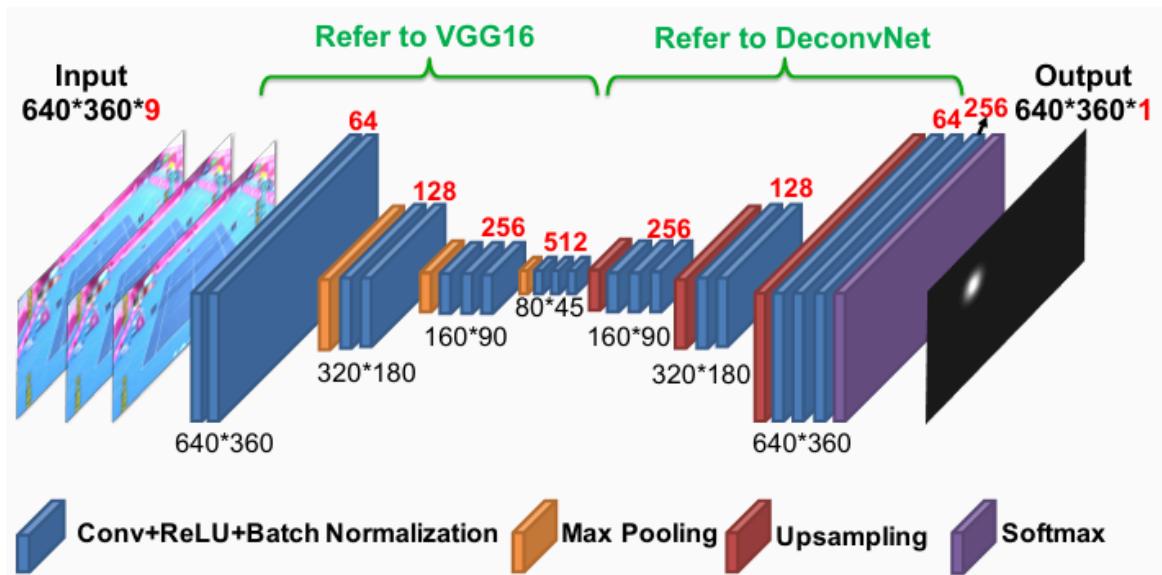
Comparison	TrackNet, Archana (BG sub)
Challenges	Overlapping (Ball&BG, Ball blocked by player), Shutter Speed (Tailing)
Track mode	Flying, Being hit, Bouncing
Background	Grass, Clay(红土), Hard court
Label	FrameName, Visibility Class, X, Y

The Task Aim determined how TrackNet and TSM work and process input differently:

- **TrackNet:** Its goal is to estimate the instantaneous velocity ($\max(\partial(x, y, t)/\partial t)$) of a specific object. This is a regression task in spatiotemporal coordinates.
- **TSM:** Its goal is to classify an action ($P(\text{action}|\mathbf{f}(t-k, t+k))$). This is a classification task over a temporal window.
- Thus determined that **TSM** does not need local features in detail to make classification while **TrackNet** needs to preserve to give a specific coordinate.
- Reflect to loss function, **TrackNet** used pixel-wise regression loss, directly penalizes spatial error; **TSM** used cross-entropy (CE) loss, standard classification loss, measuring difference between predicted probability distribution over action classes and one-hot encoded ground truth label.

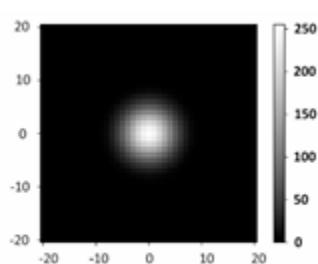
	TrackNet	TSM
Task Aim	$\max(\partial(x,y,t))/\partial t$	$P(\text{action} \mathbf{f}(t-k, t+k))$
Channel Mean	Observation of a pixel in time t	Different Feature (Abstract), LLM Context
Loss Function	$ \text{Heatmap_pred} - \text{Heatmap_gt} $	$\text{CE}(\text{action_pred}, \text{action_gt})$

[TrackINetV1](#)



Input: $W * H * (3Frames(t, t - 1, t - 2) * RGB)$

Output: $W * H * 1$



For output (coordinate, intensity)
 $L(i, j, k), (0, 0) \leq (i, j) \leq (639, 359), k \in [0, 255]$

Probability of depth k at (i, j) :

$$P(i, j, k) = \frac{e^{L(i, j, k)}}{\sum_{l=0}^{255} e^{L(i, j, l)}}$$

Softmax given by:

Heatmap of each pixel: $\text{argmax}_k P(i, j, k)$

Binary heatmap \rightarrow Circle Hough

Figure 6. An example of the detection heatmap.

Loss function:

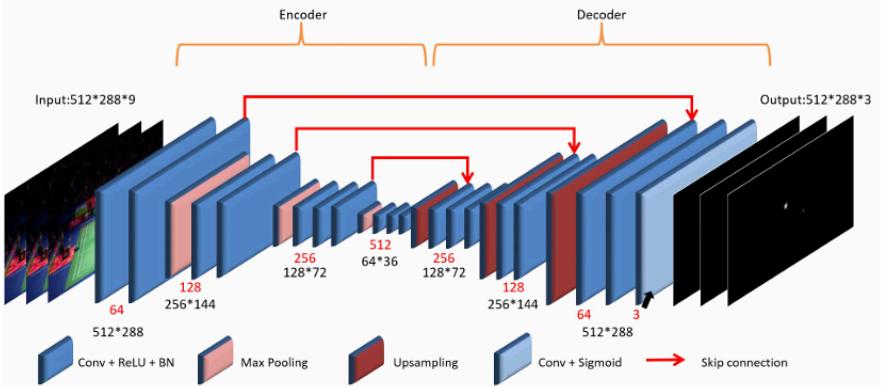
$$H_Q(P) = - \sum_{i, j, k} Q(i, j, k) \log P(i, j, k) .$$

, where GT $Q(i, j, k) = \begin{cases} 1, & \text{if } G(i, j) = k; \\ 0, & \text{otherwise.} \end{cases}$

TrackNetV2

Layer	Details	Output size
input	npy file	512 × 288 × 9
conv2d_1	3 × 3 × 64; relu; batch norm	512 × 288 × 64
conv2d_2	3 × 3 × 64; relu; batch norm	512 × 288 × 64
max_pooling_1	2 × 2 max pool; stride 2	256 × 144 × 64
conv2d_3	3 × 3 × 128; relu; batch norm	256 × 144 × 128
conv2d_4	3 × 3 × 128; relu; batch norm	256 × 144 × 128
max_pooling_2	2 × 2 max pool; stride 2	128 × 72 × 128
conv2d_5	3 × 3 × 256; relu; batch norm	128 × 72 × 256
conv2d_6	3 × 3 × 256; relu; batch norm	128 × 72 × 256
conv2d_7	3 × 3 × 256; relu; batch norm	128 × 72 × 256
max_pooling_3	2 × 2 max pool; stride 2	64 × 36 × 256
conv2d_8	3 × 3 × 512; relu; batch norm	64 × 36 × 512
conv2d_9	3 × 3 × 512; relu; batch norm	64 × 36 × 512
conv2d_10	3 × 3 × 512; relu; batch norm	64 × 36 × 512
up_sampling_1	2 × 2	128 × 72 × 512
concatenate_1	with conv2d_7; axis = 1	128 × 72 × 768
conv2d_11	3 × 3 × 256; relu; batch norm	128 × 72 × 256
conv2d_12	3 × 3 × 256; relu; batch norm	128 × 72 × 256
conv2d_13	3 × 3 × 256; relu; batch norm	128 × 72 × 256
up_sampling_2	2 × 2	256 × 144 × 256
concatenate_2	with conv2d_4; axis = 1	256 × 144 × 384
conv2d_14	3 × 3 × 128; relu; batch norm	256 × 144 × 128
conv2d_15	3 × 3 × 128; relu; batch norm	256 × 144 × 128
up_sampling_3	2 × 2	512 × 288 × 128
concatenate_3	with conv2d_2; axis = 1	512 × 288 × 192
conv2d_16	3 × 3 × 64; relu; batch norm	512 × 288 × 64
conv2d_17	3 × 3 × 64; relu; batch norm	512 × 288 × 64
conv2d_18	1 × 1 × 3; relu	512 × 288 × 3

TABLE I
TRACKNETV2 MODEL STRUCTURE.

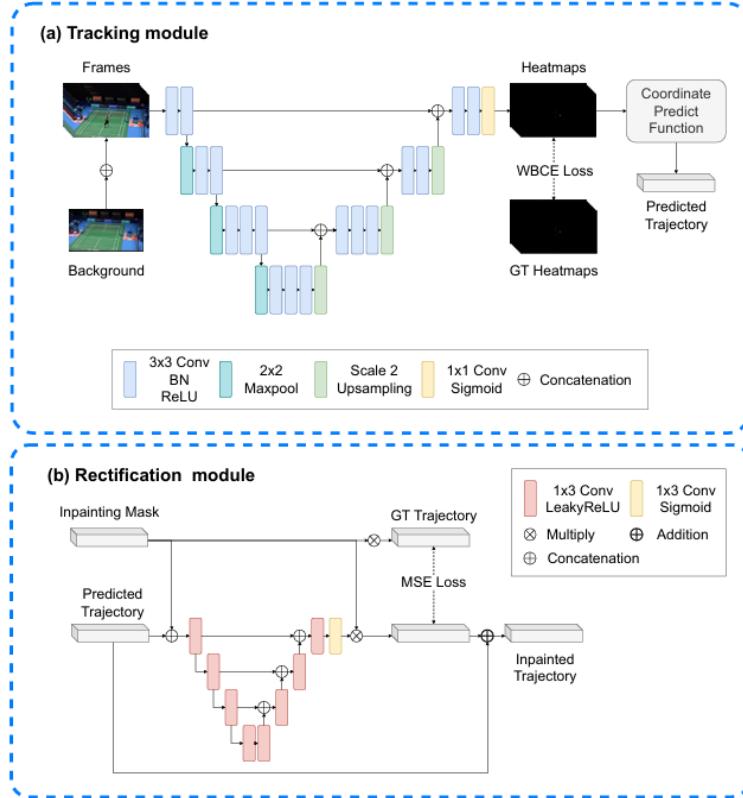


Key Changes:

- VGG Based Encoder-Decoder to U-Net Skip connection
 - Less FP, Trajectory jittering
- Output $W * H * 1$ to $W * H * InputFrames$
 - Better expression with smoother trajectory
 - Forcing continuous position prediction, better handle motion blur
- Longer warm-up (more stable training)

V1	V2
GT Heatmap tends to like Hard Binary	Smoother Gaussian heatmap (Soft label like)
Loss (BCE) is unstable to motion blur frame	(Weighted BCE) Adjustable σ based on size / resolution. No longer one pixel peak
Aggressive upsampling at decoder	Better gradual decoder
Only 1 Channel (Frame) output	3 Channel (Frame) output
Lack of BG sample, too less FP (Data Imbalance)	Added BG frame, more sensitive FP (More balancing, ball vs. non-ball)

TrackNetV3



Key Changes:

- Added Background Image in Input
- Mixup is used during training
- Added (b) Rectification module to rectify track misalignment
 - Aim to fix object overlapping, Visual blocking/hard identify

Rectification module

Height threshold: For a video frame $i, f < i < b$ not detecting shuttlecock, tracking interval $[f + 1, b - 1]$, generate inpainting masks according to:

$$M_i = \begin{cases} 1 & \text{if } p_y^f < \delta \text{ and } p_y^b < \delta \\ 0 & \text{otherwise} \end{cases}$$

where p_y^f and p_y^b represent the position of the shuttlecock at frames f and b , the threshold is controlled by $\delta, \delta = 30$ pixels. Otherwise if detected, $M_i = 0$.

Experiment

For a 13s Video:	TrackNetV2	1st 14.3s	2nd 12.6s
	TrackNetV3	1st 22s	2nd 22s