7. Supplementary Material

A supplementary video summarizing our work is available on https://youtu.be/FAeWxs0d4_o.

7.1. Notations

Let us recall the following notations from the paper:

- C is the number of classes in the spotting task.
- N_F is the number of frames in the chunk considered.
- N_{GT} is the number of ground-truth actions in the chunk considered.
- N_{pred} is the number of predictions output by the network for the spotting task.
- f is the number of features computed for each class, for each frame, before the segmentation module (see Figure 9).
- r is the temporal receptive field of the network (used in the temporal convolutions).
- $\hat{\mathbf{Y}}$ regroups the spotting predictions of the network, and has dimension $N_{\text{pred}} \times (2+C)$. The first column represents the confidence scores for the spots, the second contains the predicted locations, and the other are per-class classification scores.
- $\mathbf{Y}$ encodes the ground-truth action vectors of the chunk considered, and has dimension $N_{\text{GT}} \times (2+C)$.
- K_i^c $(i = 1, 2, 3, 4)$ denotes the context slicing parameters of class c.

We also use the following notations for the layers of a convolutional neural network:

- FC(n) is a fully connected layer (*e.g.* in a multi-layer perceptron) between any vector to a vector of size n.
- ReLU is the rectified linear unit.
- Conv($n, p \times q$) is a convolutional layer with n kernels of dimensions $p \times q$.

7.2. Detailed Network Architecture for SoccerNet

The architecture of the network used in the paper for the action spotting task of SoccerNet [21], as depicted in Figure 9, is detailed hereafter.

1. Frame feature extractor and temporal CNN. SoccerNet [21] provides three frame feature extractors with different backbone architectures (I3D, C3D, and ResNet). Each of them respectively extracts 1024, 4096, and 2048 features that are further reduced to 512 features with a Principal Component Analysis (PCA). We use the PCA-reduced features provided with the dataset as input of our temporal CNN.

The aim of the temporal CNN is to provide Cf features for each frame, while mixing temporal information across the frames. It transforms an input of shape $N_F \times 512$ into an output of shape $N_F \times Cf$.

First, each frame is input to a 2-layer MLP to reduce the dimensionality of the feature vectors of each frame. We design its architecture as: FC(128) - ReLU - FC(32) - ReLU. We thus obtain a set of $N_F \times 32$ features, which we note $\mathcal{F}_{\text{MLP}}$.

Then, $\mathcal{F}_{\text{MLP}}$ is input to a spatio-temporal pyramid, *i.e.* it is input in parallel to each of the following layers of the pyramid:

- Conv($8, r/7 \times 32$) - ReLU
- Conv($16, r/3 \times 32$) - ReLU
- Conv($32, r/2 \times 32$) - ReLU
- Conv($64, r \times 32$) - ReLU

producing $8 + 16 + 32 + 64 = 120$ features for each frame, which are concatenated with $\mathcal{F}_{\text{MLP}}$ to obtain a set of $N_F \times 152$ features.

Finally, we feed these features to a Conv($Cf, 3 \times 152$) layer, which produces a set of $N_F \times Cf$ features, noted $\mathcal{F}_{\text{TCNN}}$.

2. Segmentation module. This module produces a segmentation score per class for each frame. It transforms $\mathcal{F}_{\text{TCNN}}$ into an output of dimension $N_F \times C$, through the following steps:

- Reshape $\mathcal{F}_{\text{TCNN}}$ to have dimension $N_F \times C \times f$.
- Use a frame-wise Batch Normalization.
- Activate with a sigmoid so that each frame has, for each class, a feature vector $v \in (0, 1)^f$.
- For each frame, for each class, compute the distance d between v and the center of the unit hypercube $(0, 1)^f$, *i.e.* a vector composed of 0.5 for its f components. Hence, $d \in [0, \sqrt{f}/2]$.
- The segmentation score is obtained as $1 - 2d/\sqrt{f}$, which belongs to $[0, 1]$. This way, scores close to 1 for a class (*i.e.* v close to the center of the cube) can be interpreted as indicating that the frame is likely to belong to that class.

The segmentation scores ζ_{seg} output by the segmentation module thus has dimension $N_F \times C$ and is assessed through the segmentation loss $\mathcal{L}^{\text{seg}}$.

3. Spotting module. The spotting module takes as input $\mathcal{F}_{\text{TCNN}}$ and ζ_{seg}, and outputs the spotting predictions $\hat{\mathbf{Y}}$ of the network. It is composed of the following layers:

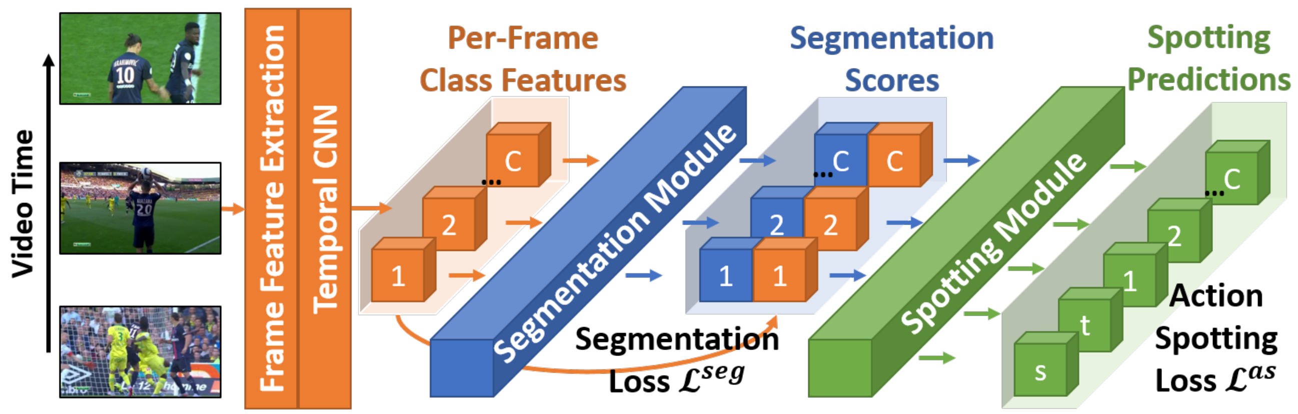


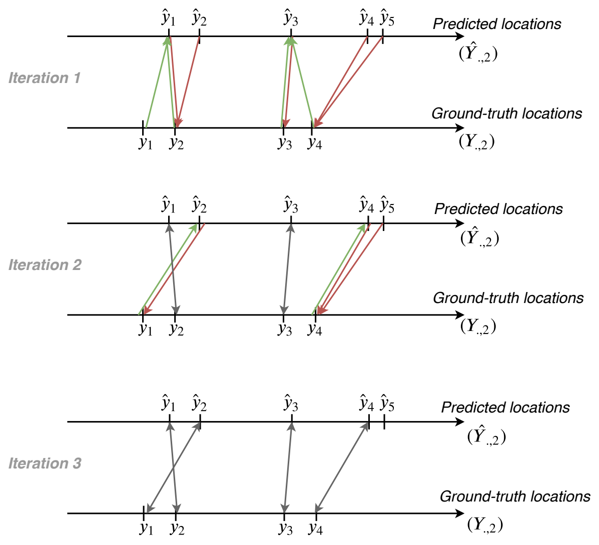
Figure 9. **Pipeline for action spotting.** We propose a network made of a **frame feature extractor** and a **temporal CNN** outputting C class feature vectors per frame, a **segmentation module** outputting per-class segmentation scores, and a **spotting module** extracting $2+C$ values per spotting prediction (*i.e.* the confidence score s for the spotting, its location t and a per-class prediction).

- ReLU on $\mathcal{F}_{\text{TCNN}}$, then concatenate with ζ_{seg}. This results in $N_F \times (Cf + C)$ features.
- Temporal max-pooling 3×1 with a 2×1 stride.
- Conv$(32, 3 \times (Cf + C))$ - ReLU
- Temporal max-pooling 3×1 with a 2×1 stride.
- Conv$(16, 3 \times 32)$ - ReLU
- Temporal max-pooling 3×1 with a 2×1 stride.
- Flatten the resulting features, which yields $\mathcal{F}_{\text{spot}}$.
- Feed $\mathcal{F}_{\text{spot}}$ to a FC$(2N_{\text{pred}})$ layer, then reshape to $N_{\text{pred}} \times 2$ and use sigmoid activation. This produces the confidence scores and the predicted locations for the action spots.
- Feed $\mathcal{F}_{\text{spot}}$ to a FC(CN_{pred}) layer, then reshape to $N_{\text{pred}} \times C$ and use softmax activation on each row. This produces the per-class predictions for the action spots.
- Concatenate the confidence scores, predicted locations, and per-class predictions to produce the spotting predictions $\hat{\mathbf{Y}}$ of shape $N_{\text{pred}} \times (2 + C)$.

Eventually, $\hat{\mathbf{Y}}$ is assessed through the action spotting loss $\mathcal{L}^{\text{as}}$.

7.3. Iterative One-to-One Matching

The iterative one-to-one matching between the predicted locations $\hat{\mathbf{Y}}_{\cdot,2}$ and the ground-truth locations $\mathbf{Y}_{\cdot,2}$ described in the paper is illustrated in Figure 10. It is further detailed mathematically in Algorithm 1.



Iteration 1
Iteration 2
Iteration 3
Predicted locations $(\hat{Y}_{\cdot,2})$
Ground-truth locations $(Y_{\cdot,2})$
$\hat{y}_1$ $\hat{y}_2$ $\hat{y}_3$ $\hat{y}_4$ $\hat{y}_5$
y_1 y_2 y_3 y_4

Figure 10. **Iterative one-to-one matching.** Example of the iterative one-to-one matching. At iteration 1, each ground-truth location is matched with its closest predicted location (**green arrows**), and vice-versa (**brown arrows**). Locations that match each other are permanently matched (**gray arrows**), and the process is repeated with the remaining locations at iteration 2. In this case, two iterations suffice to match all the ground-truth locations with a predicted location, as evidenced by the absence of available ground-truth location for iteration 3.

7.4. Details on the Time-Shift Encoding (TSE)

The time-shift encoding (TSE) described in the paper is further detailed below. We note $s^c(x)$ the TSE of frame x related to class c.

We denote s^c_p (resp. s^c_f) the difference between the frame index of x and the frame index of its closest past (resp.

Algorithm 1: Iterative matching between ground-truth and predicted locations.

```
Data: $Y, \hat{Y}$ ground-truth and predicted locations
Result: Matching couples $(y, \hat{y}) \in Y \times \hat{Y}$
Algorithm:
while $Y \neq \emptyset$ do
    $f : Y \rightarrow \hat{Y} : f(y) = \text{argmin}\{|y - \hat{y}| : \hat{y} \in \hat{Y}\}$;
    for $\hat{y} \in \hat{Y}$ do
        if $|f^{-1}(\{\hat{y}\})| \geq 1$ then
            $y_{\hat{y}} = \text{argmin}\{|y - \hat{y}| : y \in f^{-1}(\{\hat{y}\})\}$;
            Save matching couple $(y_{\hat{y}}, \hat{y})$;
            Remove $y_{\hat{y}}$ from $Y$ and $\hat{y}$ from $\hat{Y}$;
        end
    end
end
```

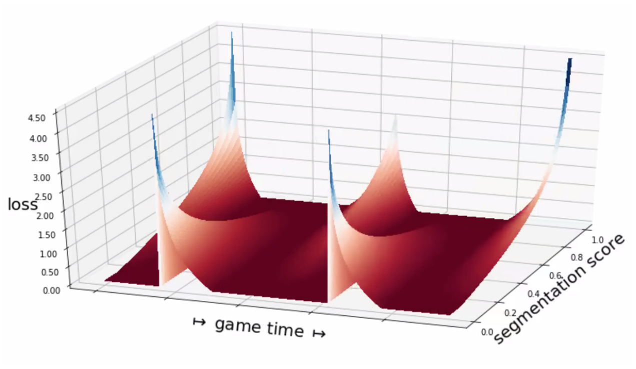


Figure 11. **Context-aware loss function (close actions).** Representation of our segmentation loss when two actions of the same class are close to each other. The loss is parameterized by the time-shift encoding of the frames and is continuous through time, except at frames annotated as actions. A video clip where we vary the location of the second action is provided with this document (*3dloss.mp4*).

future) ground-truth action of class c. They constitute the time-shifts of x from its closest past and future ground-truth actions of class c, expressed in number of frames (*i.e.* if frames 9 and 42 are actions of class c, then frame 29 has $s_p^c = 29-9 = 20$ and $s_f^c = 29-42 = -13$). We set $s_p^c = 0$ for a frame corresponding to a ground-truth action of class c, thus ensuring the relations $s_f^c < 0 \leq s_p^c$. The TSE $s^c(x)$ is defined as the time-shift among $\{s_p^c, s_f^c\}$ related to the action that has the dominant influence on x. The rules used to determine which time-shift is selected are the following:

- if $s_p^c < K_3^c$: keep s_p^c, because x is located *just after* the past action, which still strongly influences x.
- if $K_3^c \leq s_p^c < K_4^c$: x is in the *transition zone* after the past action, whose influence weakens, thus the decision depends on how far away is the future action:
  - if $s_f^c \leq K_1^c$: keep s_p^c, because x is located *far before* the future action, which does not yet influence x.
  - if $s_f^c > K_1^c$: The future action may be close enough to influence x:
    - if $\frac{s_p^c - K_3^c}{K_4^c - K_3^c} < \frac{K_2^c - s_f^c}{K_2^c - K_1^c}$: keep s_p^c, because x is closer to the *just after* region of the past action than it is to the *just before* region of the future action, with respect to the size of the transition zones.
    - else: keep s_f^c, because the future action influences x more than the past action.
- if $s_p^c \geq K_4^c$: keep s_f^c, because x is located *far after* the past action, which does not influence x anymore.

For completeness, let us recall the following details mentioned in the main paper. If x is both located *far after* the past action and *far before* the future action, selecting either of the two time-shifts has the same effect in our loss. Furthermore, for the frames located either before the first or after the last annotated action of class c, only one time-shift can be computed and is thus set as $s^c(x)$. Finally, if no action of class c is present in the video, then we set $s^c(x) = K_1^c$ for all the frames. This induces the same behavior in our loss as if they were all located far before their closest future action.

The TSE is used to shape our novel context-aware loss function for the temporal segmentation module. The cases described above ensure the temporal continuity of the loss, regardless of the proximity between two actions of the same class, excepted at frames annotated as ground-truth actions. This temporal continuity can be visualized in Figure 11, which shows a representation of $\tilde{L}(p, s)$ (analogous to Figure 1) when two actions are close to each other. It is further illustrated in the video clip *3dloss.mp4* provided with this document, where we gradually vary the location of the second action. For each location of the second action, the TSE of all the frames is re-computed, and so is the loss.

7.5. Extra Analyses

Per-class results. As for the class *goal* in Figure 6 of the main paper, Figures 12 and 13 display the number of TP, FP, FN and the precision, recall and F_1 metrics for the classes *card* and *substitution* as a function of the tolerance δ allowed for the localization of the spots.

Figure 12 shows that most cards can be efficiently spotted by our model within 15 seconds around the ground truth ($\delta = 30$ seconds). We achieve a precision of 66% for that tolerance. The previous baseline plateaus within 20 seconds ($\delta = 40$ seconds) and still has a lower performance.

Figure 13 shows that most substitutions can be efficiently spotted by our model within 15 seconds around the ground

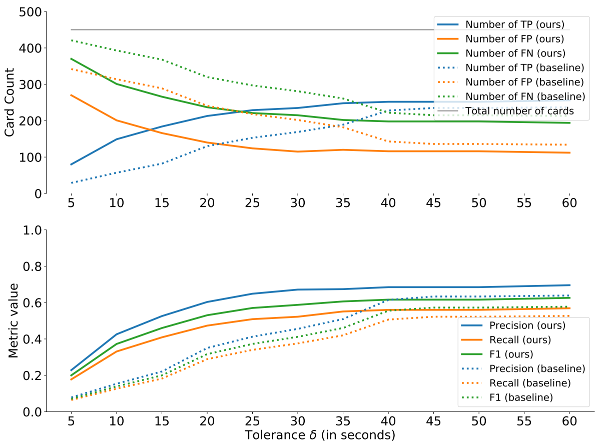


Figure 12. **Per-class results (cards).** A prediction of class *card* is a **true positive (TP)** with tolerance δ when it is located at most $\delta/2$ seconds from a ground-truth card. The baseline results are obtained from the best model of [21]. Our model spots most cards within 15 seconds around the ground truth ($\delta = 30$ seconds).

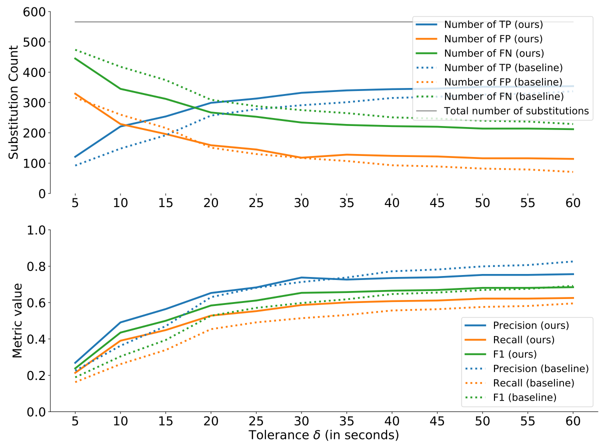


Figure 13. **Per-class results (substitutions).** A prediction of class *substitution* is a **true positive (TP)** with tolerance δ when it is located at most $\delta/2$ seconds from a ground-truth substitution. The baseline results are obtained from the best model of [21]. Our model spots most substitutions within 15 seconds around the ground truth ($\delta = 30$ seconds).

truth ($\delta = 30$ seconds). We achieve a precision of 73% for that tolerance. The previous baseline reaches a similar performance for that tolerance, and reaches 82% within 60 seconds ($\delta = 120$ seconds) around the ground truth.

Except for the precision metric for the substitutions with tolerances larger than 20 seconds, our model outperforms the previous baseline of SoccerNet [21]. As mentioned in the paper, for goals, many visual cues facilitate their spotting, *e.g.* multiple replays, particular camera views, or celebrations from the players and from the public. Cards and substitutions are more difficult to spot since the moment the referee shows a player a card and the moment a new player enters the field to replace another are rarely replayed (*e.g.* for cards, the foul is replayed, not the sanction). Also, the number of visual cues that allow their identification is reduced, as these actions generally do not lead to celebrations from the players or the public. Besides, cards and substitutions may not be broadcast in full screen, as they are sometimes merely shown from the main camera and are thus barely visible. Finally, substitutions occurring during the half-time are practically impossible to spot, as said in the main paper.

Segmentation loss analysis. We provide a supplementary analysis on the λ^{seg} parameter, which balances the segmentation loss and the action spotting loss in Equation 11 of the main paper. We fix different values of λ^{seg} and train a network for each value. We show the segmentation scores on one game for the *goal* class in Figure 14. We also display the Average-mAP for the whole test set for the different values of λ^{seg}.

It appears that extreme values of λ^{seg} substantially influence both the action spotting performance and the segmentation curves, hence the automatic highlights generation. Small values (*i.e.* $\lambda^{\text{seg}} \leq 0.1$) produce a useless segmentation for spotting the interesting unannotated *goal opportunities*. This is because the loss does not provide a sufficiently strong feedback for the segmentation task as it does not penalize enough the segmentation scores. These values of λ^{seg} also lead to a decrease in the Average-mAP for the action spotting task, as already observed in the ablation study presented in the main paper. Moreover, very large values ($\lambda^{\text{seg}} \geq 100$) penalize too much the unannotated goal opportunities, for which the network is then forced to output very small segmentation scores. Such actions are thus more difficult to retrieve for the production of highlights. These values of λ^{seg} also lead to a large decrease in the Average-mAP for the action spotting task, as the feedback of the segmentation loss overshadows the feedback of the spotting loss. Finally, it seems that for $\lambda^{\text{seg}} \in [1, 10]$, the spotting performance is high while providing informative segmentation scores on *goal opportunities*. These values lead to the spotting of several *goal opportunities*, shown in Figure 14, which might be included in the highlights automatically generated for this match by the method described in the main paper.

Comments on improvements on ActivityNet. In Table 3, we report the averages over samples of 20 results for each metric, that we further analyze statistically below for the Av.-mAP. First, following D'Agostino's normality test, we can reasonably assume that the samples are normally distributed, since we obtain p-values > 0.1 (0.28 for BMN and 0.24 for ours respectively). The standard deviations of the samples are 0.08% and 0.07%. Since the difference between the averages is 0.38%, the normal distributions overlap beyond two standard deviations from their centers,

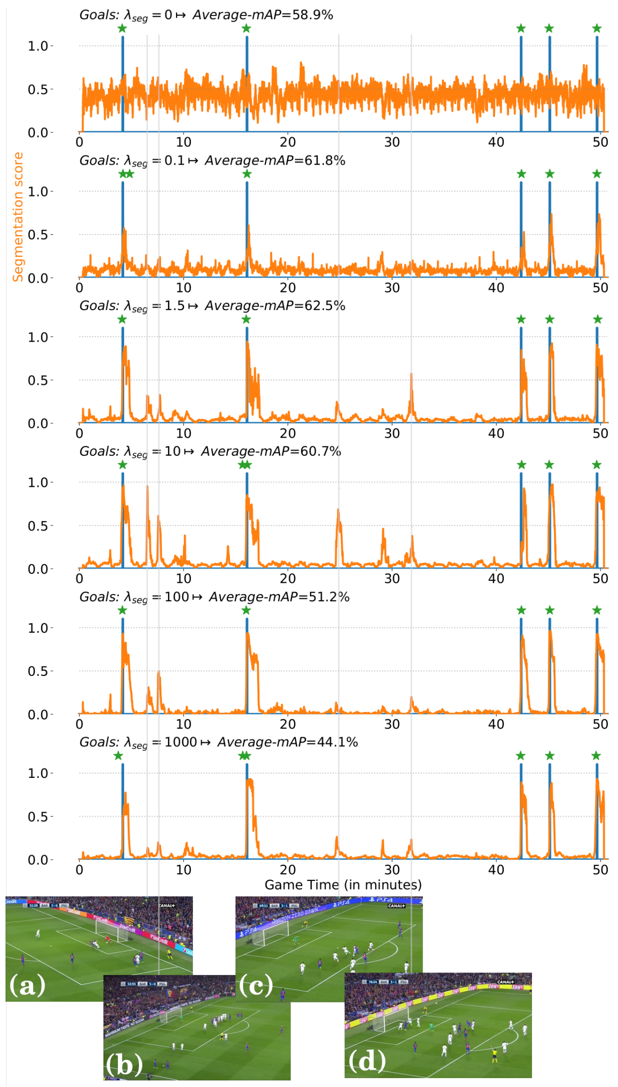


Figure 14. **Influence of λ^{seg}** on the segmentation and spotting results of the second half of the famous "Remuntada" match, Barcelona - PSG, for the class *goal*, for different values of λ^{seg}. The best Average-mAP for the spotting task is located around $\lambda^{\text{seg}} = 1.5$, while the best value for spotting unannotated goal opportunities might be around $\lambda^{\text{seg}} = 10$. For this value, several meaningful *goal opportunities* have a high **segmentation score**: **(a)** a shot on a goal post, **(b)** a free kick, **(c)** lots of dribbles in the rectangle, and **(d)** a headshot right above the goal.

which shows that our improvements are beyond noise domain. Furthermore, Bartlett's test for equal variances gives a p-value of 0.62 (> 0.1), which allows us to use Student's t-test to check whether the two samples can be assumed to have the same mean or not. We obtain a p-value of 2.3×10^{-18}, which strongly indicates that our results are significantly different from those of BMN and hence confirm the significant improvement. For the AR@100 and AUC, similar analyses give final p-values of 7.4×10^{-3} and

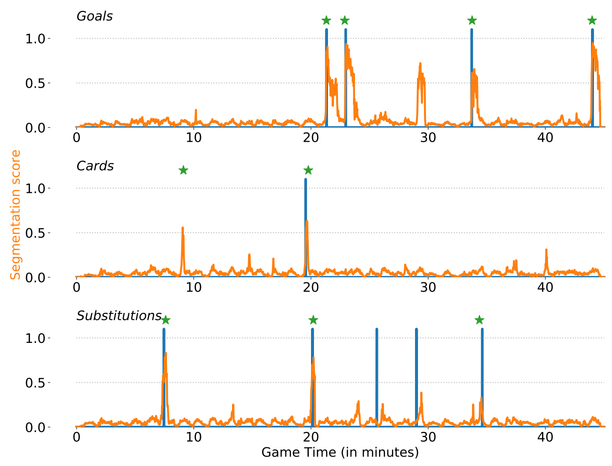


Figure 15. **Extra action spotting and segmentation results.** These results are obtained on the second half of the match Barcelona - Espanyol in December 2016. **Ground truth actions**, **temporal segmentation curves**, and **spotting results (green stars)** are illustrated. Unannotated actions can be identified and included in the highlights using our segmentation. For example, a goal opportunity occurs around the 29th minute. A false positive spot for a card is predicted by our network around the 9th minute. As it corresponds to a severe unsanctioned foul, it is fine for our automatic highlights generator to include it in the summary of the match.

9.8×10^{-2}, which corroborates the statistical significance of our improvements.

7.6. Extra Actions and Highlights Generation

Figure 15 shows additional action spotting and segmentation results. We can identify actions that are unannotated but display high segmentation scores such as goal opportunities and unsanctioned fouls. A goal opportunity around the 29th minute can be identified through the segmentation results. Besides, a false positive spot (green star) for a card is predicted around the 9th minute, further supported by a high segmentation score. A manual inspection reveals that a severe unsanctioned foul occurs at this moment. The automatic highlights generator presented in the main paper would include it in the summary of the match. Even though this foul does not lead to a card for the offender, the content of this sequence corresponds to an interesting action that would be tolerable in a highlights video.

Figure 16 shows a frame for which our network provides a high segmentation score and a false positive spot around the 26th minute (*i.e.* 71st minute of the match) for *substitutions* in Figure 7 of the main paper. We can see that the LED panel used by the referee to announce substitutions is visible on the frame. This may indicate that the network learns, quite rightly, to associate this panel with substitutions. As a matter of fact, at this moment, even the commentator announces that a substitution is probably imminent.

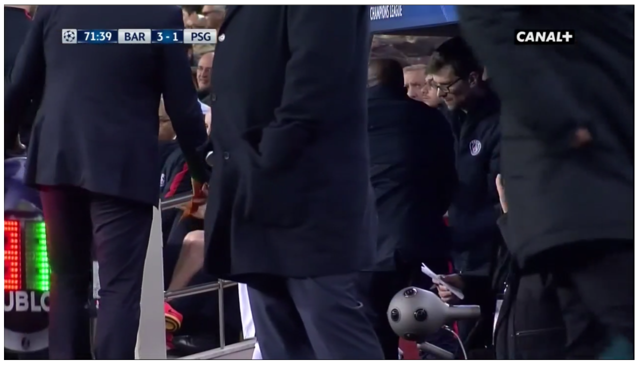


Figure 16. **False positive spot of a substitution** for the second half of the famous “Remuntada” match, Barcelona - PSG, in March 2017. The LED panel used to announce substitutions is visible on the left, which presumably explains why the network predicted the sequence around this frame as a substitution.