Using 3D Convolutional Neural Networks for Real-Time Detection of Soccer Events

Olav A. Nergård Rongved, Steven A. Hicks, Vajira Thambawita

SimulaMet, Norway & Oslo Metropolitan University, Norway

Håkon K. Stensland

Simula Research Laboratory, Norway

Evi Zouganeli

Oslo Metropolitan University, Norway

Dag Johansen

UIT The Arctic University of Norway

Cise Midoglu, Michael A. Riegler

SimulaMet, Norway

Pål Halvorsen

SimulaMet, Norway & Oslo Metropolitan University, Norway & Forzasys AS, Norway

Developing systems for the automatic detection of events in video is a task which has gained attention in many areas including sports. However, there are still a number of shortcomings with current systems, such as high latency and determining proper timing boundaries for events detected, making it challenging to operate at the live edge. In this paper, we present an algorithm to detect events in soccer videos in real time, using 3D convolutional neural networks. We run and evaluate our algorithm based on on three different real-world soccer data sets from SoccerNet, the Swedish elite series Allsvenskan, and the Norwegian elite series Eliteserien. Overall, the results show that we can detect highly relevant events with high recall, low latency, and accurate time estimation. Rapid response matters most for us, but we compare our results with current state-of-the-art that has less strict timing requirements. We conclude that our algorithm can detect most events in real-times, but still can be improved with slightly better precision. In addition to the presented algorithm, we perform an extensive ablation study on how the different parts of the training pipeline affect the final results.

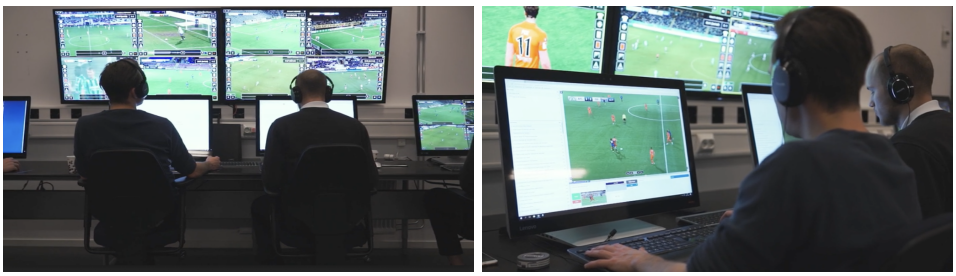
1. Introduction

Video streaming is the most prominent type of traffic in the Internet today, and is expected to grow 9-fold between 2017 and 2022 [1]. Today, we have access to

This is a peer reviewed, accepted postprint-version of the following journal article: Using 3D Convolutional Neural Networks for Real-time Detection of Soccer Events. Olav A. Nergård Rongved, Steven A. Hicks, Vajira Thambawita, Håkon K. Stensland, Evi Zouganeli, Dag Johansen, Cise Midoglu, Michael A. Riegler and Pål HalvorsenInternational Journal of Semantic Computing, 2021; 15:02, 161-187, https://doi.org/10.1142/S1793351X2140002X.

billions of hours of content through services like YouTube and Netflix, not to mention hundreds of TV channels that run 24/7. Within this ecosystem, the prevalence of digital sports streaming is continuously growing. Televised sports events are broadcast around the clock, and almost no aspect of the spectator sports sector is "offline". The global sports industry is currently estimated to be worth over $500 billion, with just under half of this turnover being generated by the spectator sports sector. Across different sports disciplines, association football (soccer) by far has the most market share of this $250 billion turnover (with 45%)[a]. The popularity of this sport is so immense that a game can spark discussions almost immediately upon completion, with dedicated shows and programs all around the world where scene replays and statistics are widely used for game-related discussions.

In this context, summarization techniques have become quite important as a means to compress the video content. For soccer, or sports in general, this summarization would typically consist of game highlights such as goals, bookings (cards), goal attempts, and penalties. Currently, the gold-standard for creating these highlight reels is through manual annotations [b] as shown in Figure 1. This is a time-consuming, tedious, and expensive operation. Automating this process, or parts of it, would go a long way in providing fast game highlights at a much lower cost. Furthermore, automatic detection and annotation of these events could be used for statistical purposes, which in turn could provide value to fans, teams, or the broadcasts themselves. The challenge is to develop a system that is accurate enough for important events and fast enough to be used for live services in real-time.



(a) One person can follow multiple games. (b) Adding metadata.

Fig. 1: A tagging center in live operation. Several persons involved and a lot of buttons to press. It is a cumbersome, error-prone and tedious manual process.

Event detection is useful in many scenarios. In particular, one especially interesting use-case is producing game highlights while the game is ongoing. However, for

[a] https://www.torrens.edu.au/blog/business/why-the-sports-industry-is-booming-in-2020-and-which-key-players-are-driving-growth

[b] https://forzasys.com/videos/forzify-tagging-pluss-small.mp4

this to work, the detection algorithm has to process frames at the pace of the videos' frame-rate (typically 25 frames-per-second). Real-time detection of sports events is nothing new, but previous pursuits rely on textual information that may not always be available [2] or require complex setups that use expensive equipment [3]. With the recent advances in deep learning-based action recognition and detection, it may be possible to create a data-driven model that generalizes to multiple sports and events. In this paper, we extend our work in [4] and explore how state-of-the-art deep learning models perform in soccer event detection. Our goal is to automatically annotate soccer events in videos as close to the actual event (in real-time) as possible. One approach to evaluate automatic event detection is through the task of spotting measuring the distances between the actual events and the predicted events. We build a prototype and use multiple datasets to evaluate our method against a baseline comparison from SoccerNet [5] in terms of both accuracy and tolerance for delays. We first provide an in-depth analysis of the performance of different models, such as ResNet 3D (R3D), ResNet Mixed Convolution (MC3) and ResNet (2+1)D (R(2+1)D), in terms of precision, recall, and F1 score in the scene classification task for soccer videos. Then, following the results of the comparison, using three different datasets, we focus on R3D, and explore and dissect the performance with respect to event categories such as "Goal", "Card", "Substitution, and "Background". The results indicate that the approach presented in [6], as a representative of the current state-of-the-art, gives a higher detection accuracy if there is a higher tolerance for accurate time estimation. In contrast, our approach is competitive for lower tolerance on accurate time estimation and superior when real-time detection is required. Finally, we include an ablation study to assess different parts of the system where we for example use class activation maps (CAMs) see model activation signals and analyse miss-classifications.

The rest of this paper is organized as follows: In Section 2, we give an overview of and discuss related work. We describe our methodology and implementations in Section 3, and outline our experiments and results in Section 4. We present an ablation study in Section 5, before we conclude the paper in Section 6.

2. Related Works

Automatic event detection is a broad field spanning multiple different areas such as sports. In this section, we cover the most relevant works using automatic analysis of sports broadcasts. We split this coverage into two parts, *sports analytics* and *action detection*, and insert our work to show how it contributes.

2.1. *Action Detection*

Action detection aims to detect what action occurs at certain times in a video, and it is a challenge that has gotten much attention over the last decade. There are several challenges when processing videos. There is often cluttered background, the camera view can change, and there can also be motion blur in the video. There are

also practical challenges such as high computational cost, storage requirements, a costly and time-demanding annotation process for the datasets.

Action Recognition is the classification of short and trimmed clips of video. Datasets such as UCF101 [7] and HMDB-51 [8] have been influential in making the action recognition task more accessible and creating a standard benchmark to measure the performance of the algorithms.

In previous works, features such as histogram of oriented gradients (HOG), histogram of flow (HOF), motion boundary histograms (MBH) [9], dense trajectories [10, 11] showed good results. In 2014, Karpathy et al. [12] explored the use of convolutional neural networks (CNNs) using two streams, cropping the center of the image and down-sampling it as input. Moreover, Simonyan et al. [13] introduced a two-stream CNN architecture related to the *two-stream hypothesis* [14], which is an idea of humans possessing two distinct visual systems. The authors used two separate CNNs, where a spatial CNN [15], pre-trained on ImageNet, takes RGB input by sampling frames from video, while the temporal stream uses optical flow fields as input. Carreira et al. [16] further extended this approach by adding 3D convolution to the Two-Stream structure, and Feichtenhofer et al. [17] explored the fusion process by combining 3D convolution and 3D pooling. The Two-Stream architecture was extended with ST-ResNet [18], which adds residual connections [19] in both streams. Combining hand-crafted features and deep-learned features have also shown promising results [20].

Wang et al. [21, 22] proposed Temporal Segment Networks (TSN), arguing that existing models mostly focused on short-term motion rather than long-range temporal structures. TSN predicts by taking a video input, separating it into multiple snippets, and using a two-stream network for each snippet. C3D [23] explored 3D convolution learning Spatio-temporal features. They showed that 3D convolutions are beneficial by adding temporal information such as motion compared with existing 2D convolutions.

Carreira et al. [16] introduced the Two-Stream Inflated 3D ConvNet (I3D) model, tested on the kinetics-400 dataset [24], using both 3D convolution and the inception architecture [25]. I3D works much like the two-stream networks, using one network for the RGB stream input and a separate optical flow network. To enable 3D convolution, they use transfer learning to initialize the 3D filters, i.e., they inflated filters on a pre-trained 2D CNN.

PoTion [26] used a pose detection algorithm [27] to find spatial locations frame by frame for joints and critical parts of the human body, i.e., avoiding to rely on only RGB or optical flow as input. Combined with RGB input, they could now use a much smaller CNN. However, a challenge with this approach is the reliance on promising results from the pose detection model.

Tran et al. [28] introduced Res (2+1)D, where (2+1)D convolutions are separating the 3D convolution into two steps. The idea is that it may be easier for the network to learn spatial and temporal features separately. Finally, SlowFast [29] introduced the SlowFast architecture. This model is using two different frame rates

as input. The idea is to have a high-capacity *slow* pathway that sub-samples the input heavily and a *fast* pathway that has a significantly higher frame rate at the cost of the capacity.

Temporal action detection aims to find a temporal interval in an untrimmed video, together with a given action that occurs. Common datasets for this challenge are THUMOS [30] and ActivityNet [31]. One of the successful approaches has been the sliding windows approach [30]. However, this approach is computationally expensive and lacks flexibility due to fixed window sizes. There has been some focus on generating temporal proposals for use with a classifier [32, 33]. Inspired by Faster R-CNN [34], Xu et al. [35] created an end-to-end model for temporal action detection that generates temporal region proposals, followed by classification.

Spatio-temporal action detection attempts to find a temporal interval for a given action and spatially. Finally, spotting [5] focuses on detecting sparse events in untrimmed videos. In contrast to temporal action detection, there is no temporal interval in which an event occurs. Instead, it is defined as an instant point in time.

2.2. *Sports Analytics*

In Sports Analytics, the use of machine learning has become increasingly popular over the last few years. This field is wide, encompassing sports ranging from soccer [5] to curling [36, 37] and everything in-between [2, 38, 39, 40]. Before the advent of machine learning, computer vision was a popular technique to analyze sports broadcasts to generate highlight reels and produce statistics. In 2003, Ekin et al. [41] presented a framework to summarize soccer videos using cinematic and object-based features automatically. Later, Giancola et al. presented SoccerNet [5], which is an open dataset for soccer video analysis. As a benchmark, the authors performed some preliminary experiments using I3D [16], C3D [23], and ResNet [19] pre-trained on ImageNet [42] as fixed feature extractors, followed by dimensionality reduction with principal component analysis (PCA). Every 0.5 seconds throughout the soccer video, the features are sampled. Afterward, the authors use convolutional layers based on these features to capture temporal information, followed by pooling laters and a fully connected layer. When the final trained model predicts an event, this process is executed around the annotated event by a sliding-windows approach and post-processing the predictions. To test this approach, the authors use a temporal window ranging from 5-60 seconds. The results showed that 5-second windows performed best given the strict requirements for distance between predicted events. However, 20-second windows gave the best overall performance. Cioppa et al. [6] improved the results on SoccerNet. A new loss function for temporal segmentation is introduced. This loss function varies the loss based on whether a prediction is located far before, just before, or far after an event.

3. Methodology

Building on the ideas that are presented above, we aim for a system that is both accurate and fast. In a real-time setting, latency is important, and therefore, a reduction of the temporal window is highly desirable. It is often necessary to gather contextual features both from the past and the future. When the model requires information from the future, the video will need to be accordingly buffered, thus introducing a delay. While a sliding-window approach is computationally expensive, current hardware can still run CNNs fast enough for acceptable performance. In many cases, the delay can be further reduced by using a higher temporal stride, without compromising precision.

3.1. *Data and Pre-processing Pipeline*

To detect events in untrimmed videos, we use a sliding window approach with a CNN classifier. To that end, we recast the problem as a classification problem locally around each event. We sample N frames locally, centred at the temporal anchor. Video frames that are not near an annotated event are considered as background. Hence, this approach leaves us with a biased model due to the sparsity of events compared with background.

Contextually, the scenes in soccer videos may also be rather similar for different types of events, consider for example a goal attempt versus a goal, or a close-up view on the referee pointing versus a yellow card. Therefore, we try to represent all parts of a full videos by generating background samples to encourage the model to learn meaningful features even when there is no typical soccer events. The borders between background and events have some inherent ambiguity, as seen in Sigurdsson et al. [43], who explored how the temporal extent of an event varies between different annotators. We annotate a background set using the following rule: If the distance between two consecutive events a and b is greater than 180 seconds, then we annotate a new event labeled *Background* at $\frac{a_t+b_t}{2}$. With minimum 180 seconds away from the closest annotated event, we have cleared a large enough temporal distance to ensure that information from events is not contained within a background sample. While it is still possible to sample a replay of an event, we find it is unlikely since replays are typically played closer to events.

Class	Train	Validation	Test
Card	1296	396	453
Substitution	1708	562	579
Goal	961	356	326
Background	1855	636	653
Total	5820	1950	2011

Table 1: The number of samples per class.

Fig. 2: Sample frames visualized for a sample of 128 frames. The middle frame is at the annotated time.

Table 1 shows our new dataset that contains *Background*. SoccerNet has $6,637$ annotations across about 784 hours of video. If we assume that a given event lasts 5 seconds, we have annotations for 5×6637 seconds of the total $784 \times 60 \times 60$seconds. This adds up to 1.17%, with the remaining 98.83% seconds containing something else. Therefore, with these three classes, the vast majority of a soccer match is background. Another weakness is that since we automatically generate new samples, *Background* may in principle be annotated during replays of any of the three events. Although highly unlikely, some 'bad' samples may still be present.

During training, we pre-process the clips on the fly. First we resize clips to a resolution of 112×199, followed by normalization. Subsequently, we randomly crop a 112×112 clip. Finally, we randomly flip each frame with a probability of 0.5. Since soccer fields are symmetric around the center, this will presumably make the model more robust to events on either side. In Figure 2, we show some example frames for the three classes.

3.2. *Model Architecture*

To successfully detect events, we want models that can capture both spatial and temporal information. ResNet 3D (R3D), ResNet Mixed Convolution (MC3) and ResNet (2+1)D (R(2+1)D) [28] are some strong models that rely only on RGB input. These models all have 18 layers and are available pre-trained on Kinetics-400 using $16 \times 112 \times 112$ inputs from PyTorch [44]. Many models use optical flow [45, 13] in order to capture temporal information. This approach has some disadvantages, such as needing to pre-process optical flow. We also tested SlowFast networks [29]. However, potentially due to our slightly different implementation and configuration with respect to[c], as well as the lack of pre-trained weights, SlowFast performed worse than the ResNet models. We therefore decided to focus on the ResNet models for detection. Below, we provide a brief description of each ResNet model:

- R3D uses a stem block with a single-layered 3D convolutional block with a $3 \times 7 \times 7$ kernel and stride $1 \times 2 \times 2$ followed by multiple two-layered blocks with $3 \times 3 \times 3$ kernels throughout the network before global average pooling and a linear output layer. This enables the model to learn spatio-temporal features.
- R(2+1)D uses a (2+1)D convolutional stem followed by multiple two-layered 2+1D blocks, resulting in higher cost during training mainly due to double the number of convolutions, batch normalization, and ReLU when compared to R3D. Tran et al. [28] argue that it may be easier to learn spatial features and temporal features separately.
- MC3 uses both 3D and 2D convolution blocks. It should be noted that the 2D convolution blocks in this context perform 3D convolution with $1 \times S \times S$ kernels. This model uses the basic stem as in R3D, followed by a layer of two 3D convolutional blocks. Subsequently, the model contains three consecutive 2D convolutional blocks are used before we perform global average pooling and a linear output layer. Tran et al. [28] noted that the idea here is that it may be best to get temporal features in the early layers, focusing on spatial features deeper in the network.

To compare the model performance, we evaluated and compare all three models. We have varied various parameters, such as different temporal input sizes, spatial input sizes, trained versus pre-trained configurations, and so on. We use 32-frame inputs due to high memory requirements, and experimented with a spatial resolution of 112×112 and 224×224. The results showed negligible difference between the two. Given the high cost of using a higher resolution, we opted for a spatial resolution of 112×112. In Table 2, we give an overview with the best configurations for each model highlighted in bold. We observe that the models generally improve with more input frames. All models seem to greatly benefit from 8 frames to 16 frames with

[c] `https://github.com/facebookresearch/SlowFast`

about 2% accuracy improvement. The same jump is seen from 16 frames to 32 frames, with about 3% improvement for R(2+1)D and R3D. 64 frames perform better, however interestingly, it seems that R(2+1)D does not scale as well with more frames when compared to R3D and MC3. With almost 8% higher validation accuracy from 8 frames to 128 frames, it seems that R3D scales best with a longer temporal window. With 128 frames at 25 frames per second (FPS), we have a 5.12-second input. As we increase our window, we will likely include too much irrelevant information. Intuitively, around 5 seconds seems sufficient for a human annotator to understand what event has taken place. However, with a 20, 30, or 40 second window, more than one event may be present, making it less discernible. During validation, we use the same number of frames as the model was trained on. It may be that a larger window during validation simplifies the task for the model. In summary, based on all these tests (Table 2), we proceed using the R3D model pre-trained on Kinetics-400 using a 112×112 frame resolution and a 128-frame temporal resolution.

Thus, our best-performing model is an 18-layered 3D ResNet as in Tran et al. [28]. It uses 17 3D-convolution layers followed by global average pooling and an output layer. The model is composed by using several residual blocks that contain 3D convolutions, with batch-normalization and ReLU. Additionally, the model has been pre-trained on Kinetics-400 [24]. Due to the pooling layer, an arbitrary number of video frames can be used as input.

3.3. *Training and Implementation Details*

We implemented the model in PyTorch [44] and trained on a computer consisting of 16 Nvidia Tesla V100 GPUs, which combined has a total memory capacity of 512 Gigabytes.

Table 3 shows the time spent with our method to create the prediction signal. Samples are 128 frames with resolution 224×398, which is transformed through resizing and center cropping to 112×112 resolution, and then normalized. These are sampled at a stride of 1 second in a single game. Most of the time is spent reading in video frames, followed by the time for transforms. Moreover, we use SGD with 0.9 momentum for each model and a learning rate scheduler that reduces the learning rate by a multiplicative factor of 0.1 every 10 epochs. We use an initial learning rate of 0.001 and a mini-batch size of 64.

Finally, as discussed in Section 3.2 and Table 2, in order to find a good configuration we have experimented with resolutions of 112×112 and 224×224, and the number of frames ranging between 8 and 128. The best results are achieved with the 18-layer R3D pre-trained on Kinetics-400 [24] with 128 frame inputs using 112×112 resolution. The number of frames and the use of a pre-trained model had significant effects on the result. This configuration of the R3D model achieves 88.4% accuracy for classification on the validation set samples, and is hence used in the experiments below.

Results based on number of frames as input				
Model	**Class**	**Precision**	**Recall**	**F1-score**
R(2+1)D 8f	Card	0.700	0.795	0.745
	Substitution	0.854	0.769	0.809
	Goal	0.887	0.840	0.863
	Background	0.741	0.766	0.753
R(2+1)D 16f	Card	0.744	0.770	0.757
	Substitution	0.851	0.804	0.827
	Goal	0.881	0.913	0.897
	Background	0.766	0.770	0.768
R(2+1)D 32f	Card	0.742	0.874	0.803
	Substitution	0.888	**0.849**	0.868
	Goal	0.896	**0.947**	**0.921**
	Background	**0.853**	0.766	0.807
R(2+1)D 64f	Card	**0.815**	0.859	**0.836**
	Substitution	0.914	0.815	0.862
	Goal	**0.940**	0.885	0.912
	Background	0.785	**0.860**	0.821
R(2+1)D 128f	Card	0.768	**0.909**	0.832
	Substitution	**0.938**	0.810	**0.869**
	Goal	0.928	0.907	0.918
	Background	0.826	0.841	**0.833**
MC3 8f	Card	0.817	0.755	0.785
	Substitution	0.828	0.865	0.846
	Goal	0.859	0.927	0.892
	Background	0.777	0.748	0.762
MC3 16f	Card	0.825	0.808	0.816
	Substitution	0.867	0.890	0.878
	Goal	0.869	0.916	0.892
	Background	0.802	0.769	0.785
MC3 32f	Card	**0.881**	0.806	0.842
	Substitution	0.856	**0.911**	0.883
	Goal	0.865	0.919	0.891
	Background	0.819	0.788	0.803
MC3 64f	Card	0.860	**0.871**	0.866
	Substitution	0.880	0.909	0.894
	Goal	**0.878**	0.949	**0.912**
	Background	**0.847**	0.777	0.811
MC3 128f	Card	0.869	**0.871**	**0.870**
	Substitution	**0.909**	0.884	**0.896**
	Goal	0.860	**0.952**	0.904
	Background	0.840	**0.808**	**0.824**
R3D 8f	Card	0.789	0.758	0.773
	Substitution	0.827	0.852	0.840
	Goal	0.856	0.916	0.885
	Background	0.772	0.741	0.756
R3D 16f	Card	0.811	0.793	0.802
	Substitution	0.837	0.870	0.853
	Goal	0.858	0.952	0.903
	Background	0.812	0.745	0.777
R3D 32f	Card	0.836	0.823	0.830
	Substitution	0.861	0.915	0.887
	Goal	0.894	0.947	0.920
	Background	0.838	0.772	0.804
R3D 64f	Card	0.903	**0.846**	0.874
	Substitution	0.873	**0.920**	0.896
	Goal	0.893	**0.963**	0.927
	Background	**0.856**	0.811	0.833
R3D 128f	Card	**0.917**	0.838	**0.876**
	Substitution	**0.880**	0.916	**0.898**
	Goal	**0.912**	0.955	**0.933**
	Background	0.852	**0.844**	**0.848**

Table 2: Precision, recall, and F1-score per class for the tested models pre-trained on the Kinetics-400 dataset with different number of consecutive frames as input.

Component	Time (seconds)
Avg. Video Read/Load + Transforms	0.605930s
Avg. Transforms	0.144913s
Avg. Video Read/Load	0.460952s
Avg. GPU load	0.004735s
Avg. GPU foward pass	0.004633s
Avg. GPU load + forward pass	0.009372s
Avg. Total	0.615302s

Table 3: Average runtime over 50 samples on a single Nvidia Tesla V100 GPU.

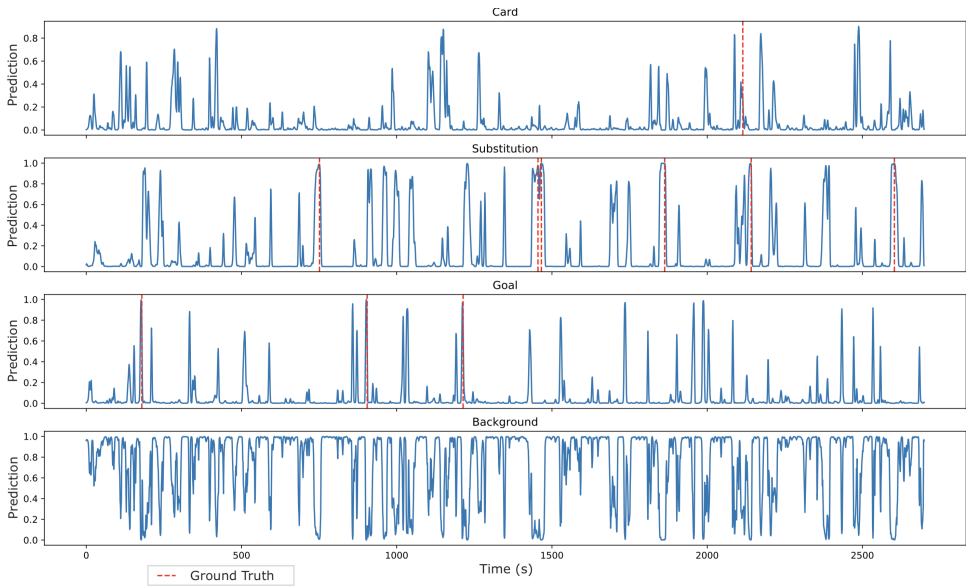


Fig. 3: Softmax confidence for each class over 45-minutes with ground truth.

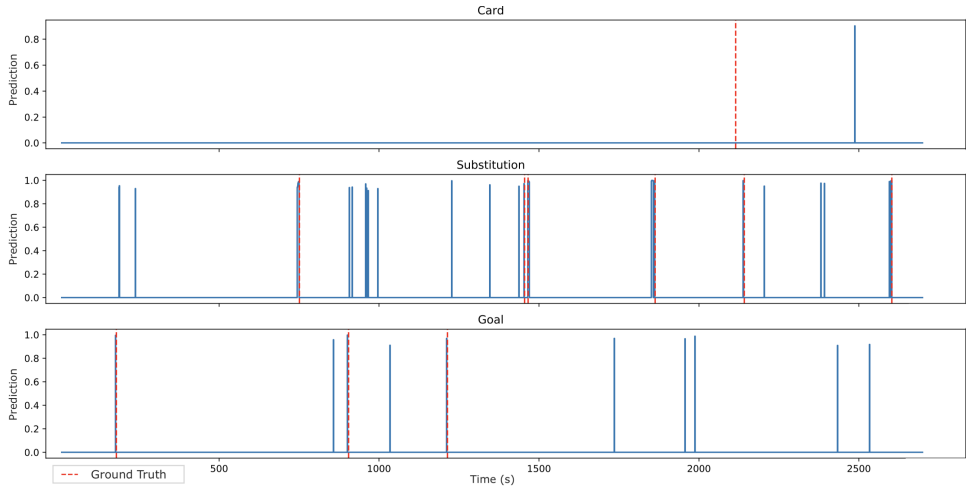


Fig. 4: Prediction for each second after mean filtering and thresholding. Red dotted line shows ground truth for each class.

3.4. *Post-processing Output*

We apply the model in a sliding window fashion with a stride of 1 second. We use a moving average filter with a kernel size of 3 on the output signal, followed by non-maximum suppression (NMS) for windows of 8 seconds, such that there are no

predictions within 4 seconds of each other. Lastly, we apply a threshold in order to remove low-confidence predictions.

4. Results and Evaluation

We have performed various experiments to test and evaluate the proposed model and compare its performance to the state of the art.

4.1. *Metrics*

In our multi-class *classification* experiments, we have used standard metrics like precision, recall, and F1-score. Here, the classification problem can be interpreted as a one-vs-all binary classification for each class in which *true positive* (TP) is when a model predicts the correct class, a *false positive* (FP) is when a class is incorrectly predicted, a *true negative* (TN) when a class is correctly rejected, and a *false negative* (FN), where a class was incorrectly rejected. Hence, the precision is the ratio of correct instances among the retrieved instances and is calculated as follows:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

The recall is the fraction of the correct instances that were retrieved:

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Finally, the F1 score is the harmonic mean of precision and recall:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3}$$

For *spotting*, we look at each class separately as a one-vs-all binary problem and consider a positive prediction as a possible true positive if it is within a tolerance δ of the ground truth event with confidence equal or higher than our threshold. Formally, we use the condition in Equation 4:

$$|gt_{spot} - p_{spot}| < \frac{\delta}{2} \tag{4}$$

where gt_{spot} is a ground truth spot, and p_{spot} a predicted spot in seconds. We take predictions that match the criteria in Equation 4 and create unique pairs of predicted spots and ground truth spots. These are matched in a greedy fashion, where each ground truth spot is matched with the closest prediction. Predicted spots that have no match are considered a false positive. For a given gt_{spot}, when no predictions are made where this condition holds, we consider it a false negative. We use the condition in Equation 4 to calculate the average precision (AP) for each class:

$$AP = \sum_{n} (R_n - R_{n-1}) P_n \tag{5}$$

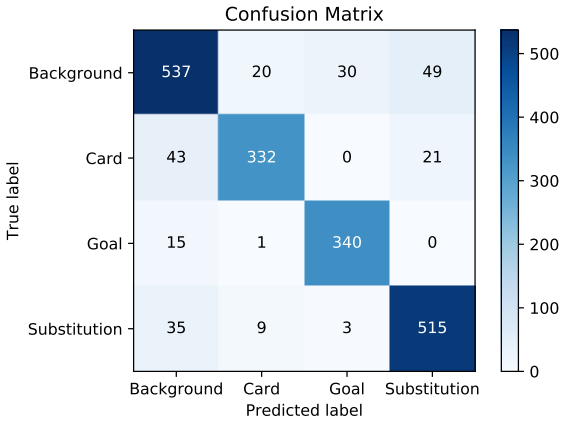


Fig. 5: Confusion matrix for validation results using R3D with 128 frames.

where R_n and P_n is the recall and precision at the n'th threshold. AP is related to the precision-recall and can be calculated as the area under the curve. This is useful as it reduces the PR-curve to a single numeric value. Subsequently, we calculate the mean average precision (mAP):

$$mAP = \frac{\sum_{i=1}^{C} AP_i}{C} \tag{6}$$

where AP_i is AP calculated for the i'th class for C classes, mAP is the mean AP calculated over all classes. This is then calculated for tolerances δ ranging between 5 and 60 seconds. Finally, we use the mAP scores calculated for different δ to calculate the area under the ROC curve (AUC) and the Average-mAP score, which provides some insight into the model's overall performance in the range of 5 - 60 seconds.

4.2. *SoccerNet Classification*

Our first experiment tries to classify an event, given that we know there is an event. Using the validation dataset described in Table 1 for testing, Figure 5 shows the confusion matrix for our R3D model with 128 frames. Good accuracy is attained, as most clips are correctly classified. However, there is still room for improvements. For the events *Card*, *Goal* and *Substitution*, most of the errors are predictions for the *Background* event type, i.e. nothing happening. We also consider the cases where *Background* is wrongly predicted as one of the other three classes. The challenge here is that *Background* events may contain replays of goal, goal attempts and several conceptually similar scenes to other classes. Hence, it is a matter of definition whether this is in fact a wrong classification or not. However, in practice this type of error ought to be eliminated.

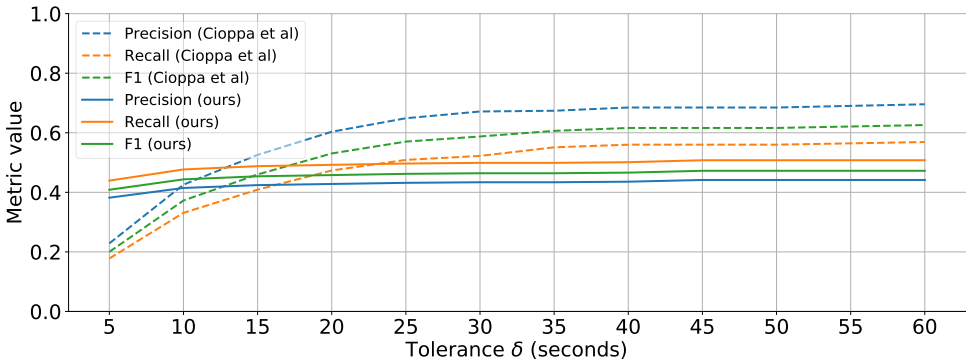


Fig. 6: Recall, precision and f1-score with respect to the tolerance δ for class *Card*, comparison with Cioppa et al. [6].

4.3. *SoccerNet Spotting*

Figure 3 depicts the softmax confidence for each class separately for a soccer half-game of 45 minutes. The background signal dominates most of the time. However, the signals are noisy and include multiple high responses at the wrong time.

Figure 4 shows our final predictions. After applying a threshold of 0.9, we observe that most of the noise is removed. For the event *Card*, we get a false positive long after the event itself. Looking at both *Substitution* and *Goal*, we observe that reasonable predictions are obtained close to the ground truth. However, we also end up with a number of false positives that are entirely unrelated.

In Figure 6, we can observe that our method reaches a plateau at about δ=10. Our method has a small temporal receptive field of 8 seconds after all post-processing steps, including NMS. Therefore, results at higher tolerance are of less interest. The appropriate threshold level is estimated by the threshold that optimizes f1-score at tolerance $\delta = 5$.

4.4. *Cross-dataset Evaluation*

Training a model on one particular dataset does not automatically mean that the same model generalizes well to other datasets that contain the same classes. To investigate how our approach generalizes to other soccer videos, we download an additional 617 short clips combined from Norwegian Eliteserien and Swedish Allsvenskan. 533 of these clips contain a goal, while 84 contain goal Attempts. As we do not explicitly train our model on goal attempts, we expect false positives as we test clips that are contextually similar to goals. This dataset contains clips with duration ranging between approximately 60 to 90 seconds, where each clip includes an event at about 25 seconds into the clip. In practice, the event generally lies within 20 to 30 seconds into the clip. Results from randomly sampled goals from Eliteserien are shown in Figure 7a using a softmax predicted score. Since all samples are similar in

that the event occurs at the same temporal spot, we look at the average prediction signal over all samples. In the figure, we mark the expected goal at 25 seconds, and the maximum prediction produced by the model. We generally see a peak of around 25 seconds, as expected. Furthermore, the model seems to produce false positives 30-60 seconds after the goal. This is consistent with replays, and highlights a limitation of our model in that it cannot inherently separate the actual goal from a replay. The results imply that we can expect the model to produce false positives for most goals, depending on factors such as replay speed and camera angle. It may be applicable to remove these predictions and hence errors by assuming that there are no overlapping goals within a temporal window of length L centered at the max prediction. In Figure 7b, we also find that there is a trend of high responses around 25 seconds. However, we note that, the prediction score from these samples is lower compared to actual goals. This indicates that lower thresholds lead to an increased number of false positives, hence it is crucial to use a high enough threshold.

Dataset	N
Goal Allsvenskan	233
Goal Eliteserien	300
Goal attempt Allsvenskan	84
Total	617

Table 4: Statistics for Allsvenskan and Eliteserien clips.

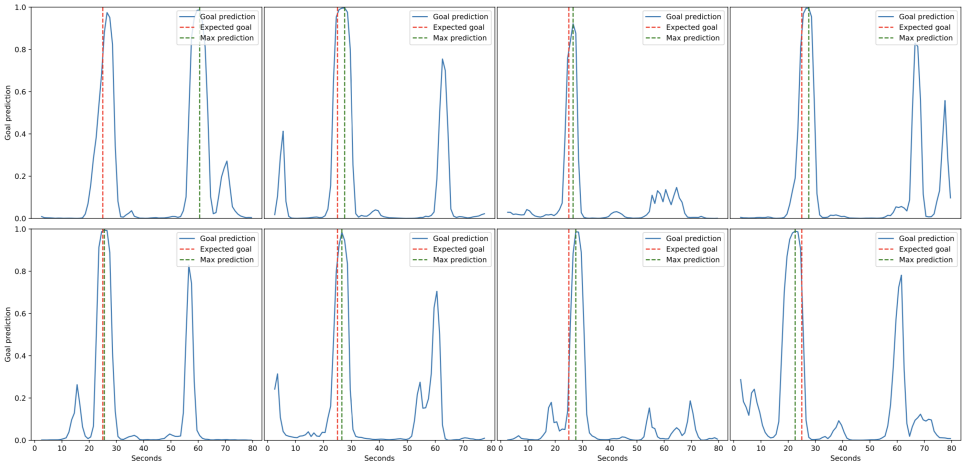
Table 4 shows the number of goals and the number of goal attempts, and Figure 8 depicts the average prediction for two event types. In Figure 8a, we observe how our approach generally responds to goals. This is further exemplified in Figure 8b for the goal attempts. The general conclusion is that the prediction is very accurate in predicting the time of the different events.

4.5. *Comparison to State-of-the-art*

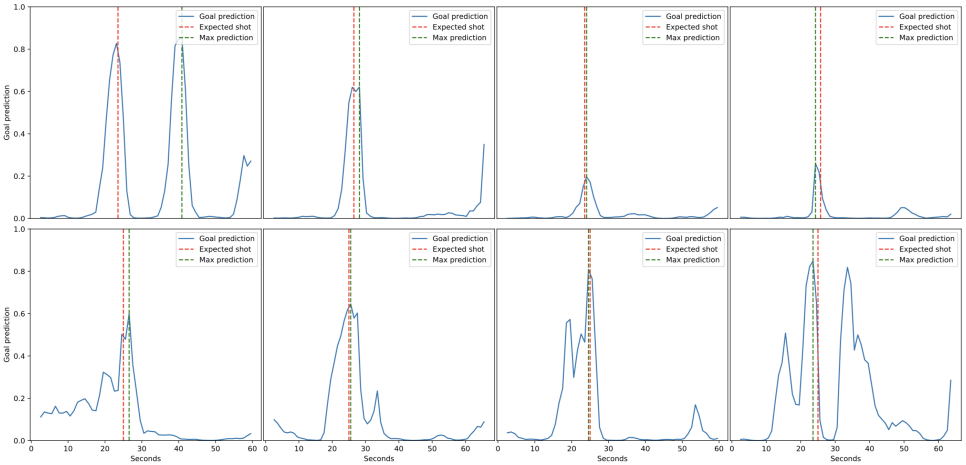
Method	mAP
SoccerNet baseline 20s [5]	49.7
Cioppa et al. [6]	**62.5**
Ours	32.0

Table 5: Results for the spotting task in SoccerNet.

Table 5 shows that our method scores low for the Average-mAP. However, since our approach relies on a small temporal window of 8 seconds after post-processing, the values calculated for tolerances $\delta > 8$ can be misleading. Average-mAP uses a range of tolerances from 5 to 60 seconds. Since our model relies on local information, rather than long-range contextual information regarding events, it is expected that increasing the tolerances will only result in finding spots that are false positives. In a



(a) *Goal* clips from Eliteserien.



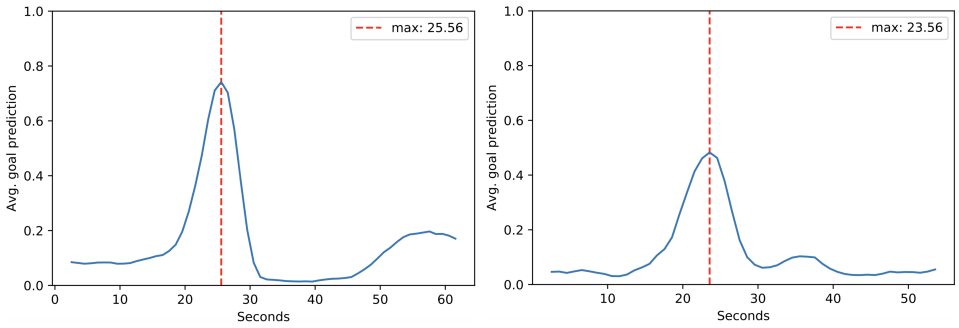
(b) *Goal attempt* clips from Allsvenskan.

Fig. 7: Predictions on random samples.

real-time setting, it may be important to have as little prediction delay as possible. With our approach, a live prediction will have a delay of about 4 seconds. This includes both buffering future frames and computation. The baseline model would introduce about 10 seconds delay, and the current work in [6] about 100 seconds delay. Evidently, long-range contextual features can boost performance, and there is a trade-off between acceptable delay and detection performance.

5. Ablation Study

To assess different parts of the system, we have performed an ablation study.



(a) *Goal* clips. (b) *Goal attempt* clips.

Fig. 8: The average predictions for the 'goal' and 'goal attempts' clips.

5.1. *Local Behavior with Sliding Window*

The sliding window approach was performed with a stride of 1 second. The videos' frame rate is 25 FPS, and we could therefore sample more often, which might be valuable. However, sampling densely at 25 times per second would be computationally expensive.

To better understand the local behavior of our model and how densely we need to sample, we perform the following experiment. First, we randomly sample 8 correctly predicted event samples from the validation set. Next, we pad the input with 130 zero frames before and after, hence, we now have a $3 \times 386 \times 112 \times 112$ tensor. Finally, we take a sliding-window approach, where we use a stride of 1 frame, densely sampling predictions. It may be that some classes require longer temporal windows for correct predictions.

In Figure 9, we can observe how the model prediction is generally strong while close to a perfect overlap with the event, especially when within 15 frames of a perfect overlap. Depending on the class, we also notice differences. The class *Substitution* has a higher AUC in our samples compared to the class *Goal*. Intuitively, this may be due to the underlying length of the different events. A goal will often contain rapid changes, while a substitution may be a more slow process.

We observe similar results for larger samples around events. Based on these experiments, we conjecture that sampling too densely with a sliding window approach is unnecessary. Since we are not required to densely sample predictions, it is easier to run real-time.

5.2. *Class Activation Maps*

To understand what our model reacts to temporally and spatially, we inspect the class activation maps (CAMs) for correct and wrong predictions. We follow the methodology presented by Zhou et al. [46] to generate CAMs.

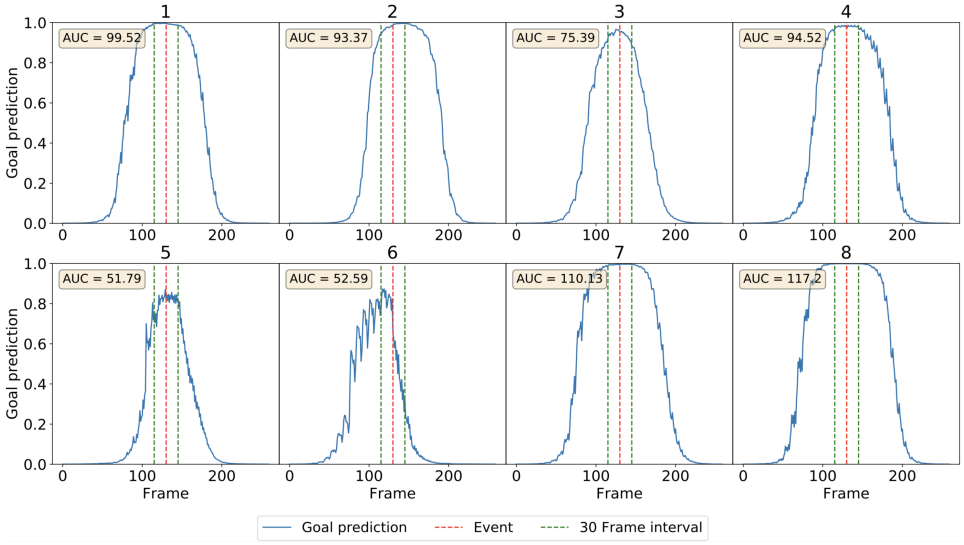


Fig. 9: Softmax output for the event *Goal* with a stride of 1 frame over 260 frames. Samples are correctly predicted validation samples. Red dotted line shows the output prediction for the 128 frames of the event.

The model R3D uses global average pooling before a fuller connected layer after 17 convolutional layers. For R3D, with 128 frames specifically, we have a $512 \times 16 \times 7 \times 7$ tensor that is reduced to $512 \times 1 \times 1 \times 1$ after the global average pooling layer. These are the 512 features that are used to compute the final scores, followed by softmax. Let F_i denote the i-th feature channel for i $\in \{1, 2, 3, ..., N\}$. Then, S_c is the pre-softmax class score, computed as a weighted sum by Equation 7:

$$S_c = B_c + \sum_{i=1}^{N} w_i^c F_i \tag{7}$$

Here, B_c is the bias term, w_i^c are the weights, and c denotes our four different classes *Card*, *Substitution*, *Goal*, and *Background*. F_i is calculated by Equation 8:

$$F_i = \frac{1}{K} \sum_{t,x,y} V_i(t, x, y) \tag{8}$$

V_i holds our N feature volumes with K elements each, which in our case is $16 \times 7 \times 7 = 784$, where t denotes our temporal dimension, and x, y our spatial dimensions. Thereby:

$$\begin{aligned} S_c &= B_c + \sum_{i=1}^{N} w_i^c \frac{1}{K} \sum_{t,x,y} V_i(t,x,y) \\ &= B_c + \frac{1}{K} \sum_{t,x,y} \sum_{i=1}^{N} w_i^c V_i(t,x,y) \end{aligned} \tag{9}$$

In Equation 9, we attempt to show the relationship between the pre-average pool feature volumes and the weights and bias used to compute the class scores S_c. As Zhou et al. [46], we ignore the bias term moving forward as it has little impact on the final results. We define a class activation tube (CAT) as a 3-dimensional equivalent of CAMs as follows:

$$T_c(t,x,y) = \sum_{i=1}^{N} w_i^c V_i(t,x,y) \tag{10}$$

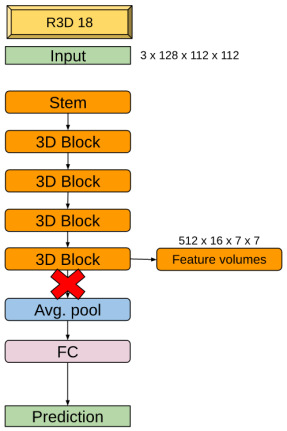
Going from Equation 10 to our class score S_c, we need to calculate the average over all elements and add the bias term. Since this is the case, we may find useful information both temporally and spatially that helps us gain insight into what our model reacts to. Figure 10a illustrates where we get our feature volumes in our model. Additionally, we illustrate how these feature volumes are used to compute CAT's, as in Equation 10, in Figure 10b.

We use CATs to understand spatio-temporal features by considering the spatial information at time t. In order to get a comparison to our input, we interpolate our 7×7 maps in $\{T_c(t=0,x,y), T_c(t=1,x,y)...T_c(t=T,x,y)\}$ using bicubic interpolation. We also use the CATs to compute temporal signals to indicate where in time, our model reacts. This is achieved by average pooling T_c across spatial dimensions, resulting in a 1-dimensional signal. Due to the spatial relationship of the spatio-temporal class activation features at time t, we refer to them as CAMs. Furthermore, we refer to the 1-dimensional temporal signal as *class activation signal*.

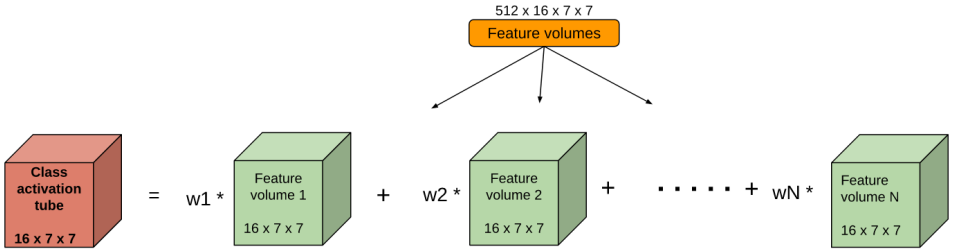
5.2.1. *Sparse CAMs*

In order to gain some insight into the spatial features of the network and what the network base its detection on, we slide our model across each sample with inputs of 128 frames, using a stride of 16 frames. For each location, we save the middle input image and the corresponding middle CAM at $T_c(8,x,y)$. In Figure 11, we observe how the model seems to react strongly to a referee holding a card, i.e., the strong red areas.

As mentioned in Section 4.2, there are also miss-classified events. Again using the CAMs, we looked at several examples and as depicted in the example in Figure 12 could manifest that the model hardly reacts at anything. Hence, the model incorrectly classified the sample as *Background*. There seem to be several causes of miss-classifications. In the given example, we notice that the goal seems to be a



(a) Structure in the R3D model. The feature volumes V_i are extracted prior to global average pooling layer in order to preserve spatial and temporal information.



(b) Calculation of class activation tubes T_c.

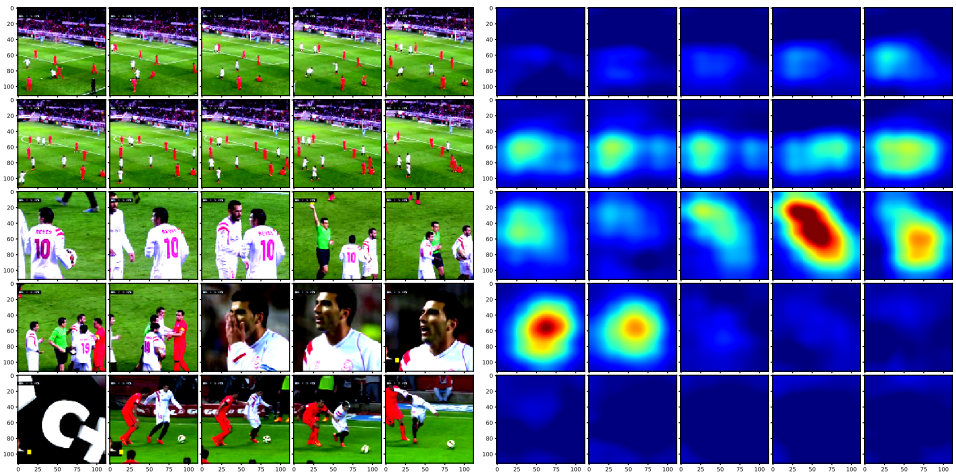
Fig. 10: Extraction of features and calculation of CAT.

penalty, where the camera angle is somewhat unusual. Another factor might also be that much of the scene is filled with audience members, which might be associated with background samples.

5.2.2. *Class Activation Signal*

Since we are using video and a 3-dimensional CNN, we have both spatial and temporal information. Therefore, we can focus on the temporal signal alone by averaging across the spatial dimensions, resulting in a 1-dimensional signal.

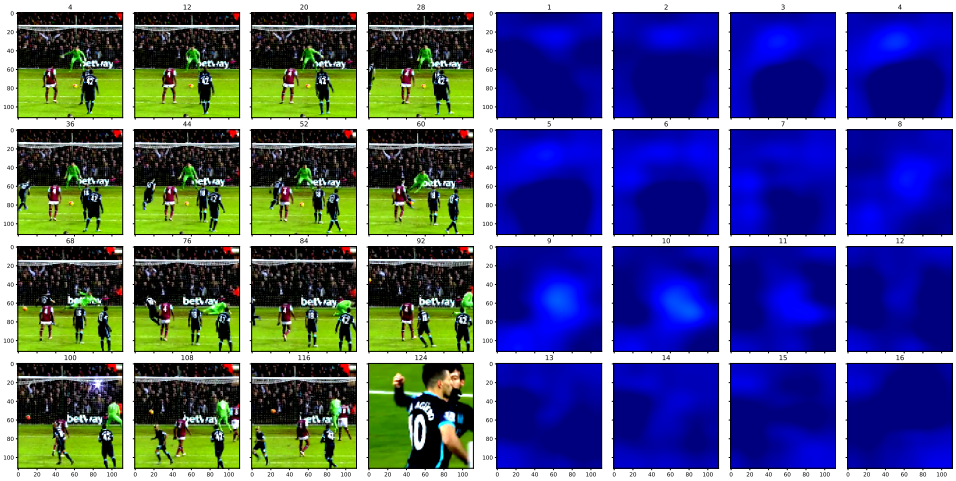
Zhou et al. [46] showed that CAMs could be used for object detection. It may also be possible to use this for more accurate and efficient detection of events. In Figure 13a, we show how our model reacts in the temporal dimension. It seems that the strongest activations occur close to the actual event. Normally, any of these signals would be averaged and used as input into softmax to produce the final predictions. However, this process removes temporal information that could be used



(a) Center input frame between interval $[s \times n, 28 + s \times n]$ of 512 frame sample of event for the n^{th} sample.

(b) Corresponding center CAM.

Fig. 11: Results from CATs spatio-temporally for the event *Card*. Each small picture has the used 112×112 resolution.

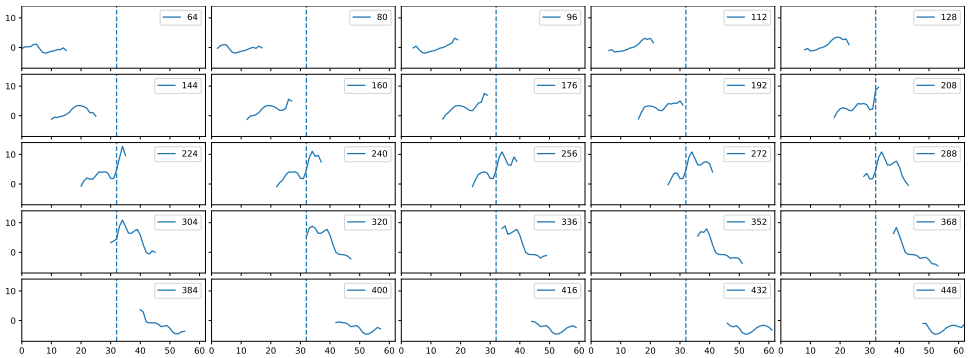


(a) Frames

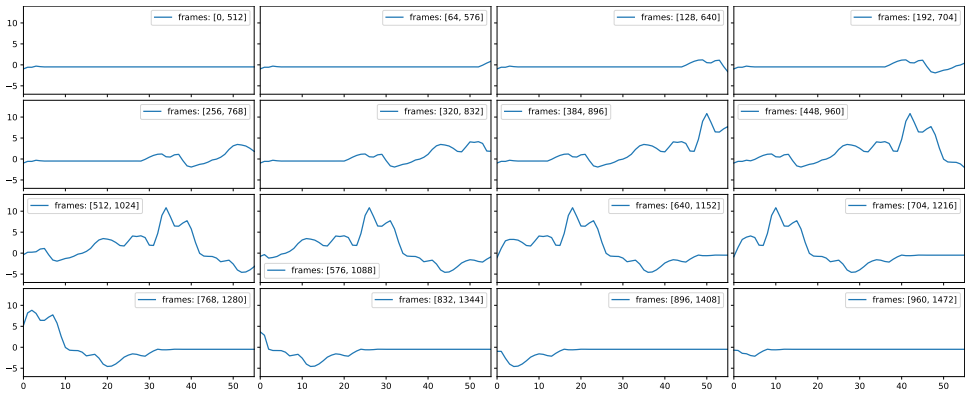
(b) Class activation maps

Fig. 12: Example of erroneously classification of a '*Goal*' as a '*Background*' event. Each small picture has the used 112×112 resolution.

for more accurate annotations. This is seen when increasing the temporal footprint



(a) Class activation signal for the center input frame between interval $[s \times n, 28 + s \times n]$ of 512 frame sample of event for the n^{th} sample.



(b) Corresponding class activation signal with 512 frame input for center CAM $T_{card}(8, x, y)$.

Fig. 13: Results from CATs temporally for the event *Card*.

as well. Figure 13b shows that the class activation signal indicates strong reactions at the actual event.

6. Conclusion

In this paper, we have presented a real-time algorithm to automatically detect and annotate segments of soccer videos using CNNs. The approach uses sliding windows to detect events and classifies them into a set number of categories. To better understand how the algorithm detects events, we performed an extensive ablation study and visualized the network's layers using CAMs. Overall, we achieved an accuracy of 88.4% on trimmed clips in SoccerNet, and an Average-mAP score of 32.0% on the spotting task of SoccerNet [5]. Compared to the results presented in the SoccerNet papers, we achieve a slightly lower detection accuracy compared to the

the general state-of-the-art, which has a higher tolerance for precise and accurate time estimation. Consequently, our approach is competitive when accurate time estimation is important (finding the event's exact time) and better than the state-of-the-art when a low delay is required, for example for real-time event detection.

Ongoing work includes tuning of the system to increase precision while keeping the recall and low delay. Inspired by recent work like [47], we are also researching how audio can be added to state-of-the-art approaches making multi-modal models. Another interesting direction for the future is to test with more data and additional event types in the upcoming SoccerNet-v2 dataset [48]. Finally, we conjecture that the presented approach will scale to other sports as the algorithm is rater application agnostic, which will be investigated in the future.

References

[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.pdf, (2019).

[2] R. Kapela, K. McGuinness, A. Swietlicka and N. E. O'Connor, Real-time event detection in field sport videos, in *Proc. of CVPR* (IEEE Computer Society, 2014).

[3] T. D'Orazio, M. Leo, P. Spagnolo, M.Nitti, N. Mosca and A. Distante, A visual system for real time detection of goal events during soccer matches, *Computer Vision and Image Understanding* **113**(5) 622 – 632 (2009).

[4] O. A. N. Rongved, S. A. Hicks, V. Thambawita, H. K. Stensland, E. Zouganeli, D. Johansen, M. A. Riegler and P. Halvorsen, Real-time detection of events in soccer videos using 3d convolutional neural networks, in *Proc. of ISM* (IEEE Computer Society, 2020), pp. 135–144.

[5] S. Giancola, M. Amine, T. Dghaily and B. Ghanem, Soccernet: A scalable dataset for action spotting in soccer videos, in *Proc. of CVPR Workshops* (IEEE Computer Society, USA, June 2018), pp. 1711–1721.

[6] A. Cioppa, A. Deliege, S. Giancola, B. Ghanem, M. V. Droogenbroeck, R. Gade and T. B. Moeslund, A context-aware loss function for action spotting in soccer videos, in *Proc. of CVPR* (IEEE Computer Society, June 2020).

[7] K. Soomro, A. R. Zamir and M. Shah, Ucf101: A dataset of 101 human actions classes from videos in the wild, *CoRR* **abs/1212.0402** (December 2012).

[8] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio and T. Serre, Hmdb: A large video database for human motion recognition, in *Proc. of ICCV* (IEEE Computer Society, 2011), pp. 2556–2563.

[9] N. Dalal, B. Triggs and C. Schmid, Human detection using oriented histograms of flow and appearance, in *Proc. of ECCV* (Springer Berlin Heidelberg, 2006), p. 428–441.

[10] H. Wang, A. Kläser, C. Schmid and C.-L. Liu, Dense trajectories and motion boundary descriptors for action recognition, *International Journal of Computer Vision* **103**(1) 60–79 (2013).

[11] H. Wang and C. Schmid, Action recognition with improved trajectories, in *Proc. of ICCV* (IEEE Computer Society, 2013), pp. 3551–3558.

[12] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar and L. Fei-Fei, Large-scale video classification with convolutional neural networks, in *Proc. of CVPR* (IEEE Computer Society, June 2014), pp. 1725–1732.

[13] K. Simonyan and A. Zisserman, Two-stream convolutional networks for action recognition in videos, in *Proc. of NIPS* (Curran Associates, Inc., 2014), p. 568–576.

[14] M. A. Goodale and A. D. Milner, Separate visual pathways for perception and action, *Trends in Neurosciences* **15**(1) 20 – 25 (1992).
[15] A. Krizhevsky, I. Sutskever and G. E. Hinton, Imagenet classification with deep convolutional neural networks, in *Proc. of NIPS* (Curran Associates, Inc., 2012), pp. 1097–1105.
[16] J. Carreira and A. Zisserman, Quo vadis, action recognition? a new model and the kinetics dataset, in *Proc. of CVPR* (IEEE Computer Society, 2017), pp. 4724–4733.
[17] C. Feichtenhofer, A. Pinz and A. Zisserman, Convolutional two-stream network fusion for video action recognition, in *Proc. of CVPR* (IEEE Computer Society, June 2016), pp. 1933–1941.
[18] C. Feichtenhofer, A. Pinz and R. P. Wildes, Spatiotemporal residual networks for video action recognition, in *Proc. of NIPS* (Curran Associates, Inc., 2016), p. 3476–3484.
[19] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, in *Proc. of CVPR* (IEEE Computer Society, 2016), pp. 770–778.
[20] L. Wang, Y. Qiao and X. Tang, Action recognition with trajectory-pooled deep-convolutional descriptors, in *Proc. of CVPR* (IEEE Computer Society, June 2015), pp. 4305–4314.
[21] Z. Shou, D. Wang and S.-F. Chang, Temporal action localization in untrimmed videos via multi-stage cnns, in *Proc. of CVPR* (IEEE Computer Society, June 2016), pp. 1049–1058.
[22] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang and L. Van Gool, Temporal segment networks: Towards good practices for deep action recognition, in *Proc. of ECCV* (Springer Berlin Heidelberg, 2016), pp. 20–36.
[23] D. Tran, L. Bourdev, R. Fergus, L. Torresani and M. Paluri, Learning spatiotemporal features with 3d convolutional networks, in *Proc. of ICCV* (IEEE Computer Society, 2015), p. 4489–4497.
[24] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman and A. Zisserman, The kinetics human action video dataset, *CoRR* **abs/1705.06950** (May 2017).
[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, Going deeper with convolutions, in *Proc. of CVPR* (IEEE Computer Society, 2015), pp. 1–9.
[26] V. Choutas, P. Weinzaepfel, J. Revaud and C. Schmid, Potion: Pose motion representation for action recognition, in *Proc. of CVPR* (IEEE Computer Society, June 2018).
[27] Z. Cao, T. Simon, S.-E. Wei and Y. Sheikh, Realtime multi-person 2d pose estimation using part affinity fields, in *Proc. of CVPR* (IEEE Computer Society, 2017), pp. 1302–1310.
[28] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun and M. Paluri, A closer look at spatiotemporal convolutions for action recognition, in *Proc. of CVPR* (IEEE Computer Society, June 2018), pp. 6450–6459.
[29] C. Feichtenhofer, H. Fan, J. Malik and K. He, Slowfast networks for video recognition, in *Proc. of ICCV* (IEEE Computer Society, October 2019), pp. 6202–6211.
[30] H. Idrees, A. R. Zamir, Y. Jiang, A. Gorban, I. Laptev, R. Sukthankar and M. Shah, The thumos challenge on action recognition for videos "in the wild", *Computer Vision and Image Understanding* **155** 1–23 (2017).
[31] F. C. Heilbron, V. Escorcia, B. Ghanem and J. C. Niebles, Activitynet: A large-scale video benchmark for human activity understanding, in *Proc. of CVPR* (IEEE Computer Society, 2015), pp. 961–970.

[32] T. Lin, X. Liu, X. Li, E. Ding and S. Wen, Bmn: Boundary-matching network for temporal action proposal generation, in *Proc. of ICCV* (IEEE Computer Society, October 2019).

[33] T. Lin, X. Zhao, H. Su, C. Wang and M. Yang, Bsn: Boundary sensitive network for temporal action proposal generation, in *Proc. of ECCV* (Springer Berlin Heidelberg, 2018).

[34] S. Ren, K. He, R. Girshick and J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in *Advances in Neural Information Processing Systems 28*, eds. C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett (Curran Associates, Inc., 2015), pp. 91–99.

[35] H. Xu, A. Das and K. Saenko, R-c3d: Region convolutional 3d network for temporal activity detection, in *Proc. of ICCV* (IEEE Computer Society, 2017).

[36] H. Pojskic, K. McGawley, A. Gustafsson and D. G. Behm, The reliability and validity of a novel sport-specific balance test to differentiate performance levels in elite curling players, *Journal of Sports Science & Medicine* **19**(2) p. 337 (2020).

[37] J. Bradley, The sports science of curling: A practical review, *Journal of Sports Science & Medicine* **8** 495–500 (12 2009).

[38] V. Bettadapura, C. Pantofaru and I. Essa, Leveraging contextual cues for generating basketball highlights, in *Proc. of MM* (Association for Computing Machinery, 2016), p. 908–917.

[39] H.-T. Chen, W.-J. Tsai, S.-Y. Lee and J.-Y. Yu, Ball tracking and 3d trajectory approximation with applications to tactics analysis from single-camera volleyball sequences, *Multimedia Tools and Applications* **60**(3) 641–667 (2012).

[40] N. Homayounfar, S. Fidler and R. Urtasun, Sports field localization via deep structured models, in *Proc. of CVPR* (IEEE Computer Society, 2017), pp. 4012–4020.

[41] A. Ekin, A. M. Tekalp and R. Mehrotra, Automatic soccer video analysis and summarization, *IEEE Transactions on Image Processing* **12**(7) 796–807 (2003).

[42] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in *Proc. of CVPR* (IEEE Computer Society, 2009), pp. 248–255.

[43] G. A. Sigurdsson, O. Russakovsky and A. Gupta, What actions are needed for understanding human actions in videos?, in *Proc. of ICCV* (IEEE Computer Society, 2017), pp. 2156–2165.

[44] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in *Proc. of NIPS*, eds. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett (Curran Associates, Inc., 2019), pp. 8024–8035.

[45] J. Carreira and A. Zisserman, Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset (February 2018).

[46] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva and A. Torralba, Learning deep features for discriminative localization, in *Proc. of CVPR* (IEEE Computer Society, 2016), pp. 2921–2929.

[47] B. Vanderplaetse and S. Dupont, Improved soccer action spotting using both audio and video streams, in *Proc. of CVPR Workshops* (IEEE Computer Society, June 2020).

[48] A. Deliège, A. Cioppa, S. Giancola, M. J. Seikavandi, J. V. Dueholm, K. Nasrollahi, B. Ghanem, T. B. Moeslund and M. V. Droogenbroeck, Soccernet-v2 : A dataset and benchmarks for holistic understanding of broadcast soccer videos (2020).