Real-Time Detection of Events in Soccer Videos using 3D Convolutional Neural Networks

Olav A. Nergård Rongved*‡, Steven A. Hicks*‡, Vajira Thambawita*‡, Håkon K. Stensland*, Evi Zouganeli†, Dag Johansen¶, Michael A. Riegler*, and Pål Halvorsen*‡§

*SimulaMet, Norway ‡Oslo Metropolitan University, Norway §Forzasys, Norway ¶UIT The Arctic University of Norway

***Abstract*—In this paper, we present an algorithm for automatically detecting events in soccer videos using 3D convolutional neural networks. The algorithm uses a sliding window approach to scan over a given video to detect events such as goals, yellow/red cards, and player substitutions. We test the method on three different datasets from SoccerNet, the Swedish Allsvenskan, and the Norwegian Eliteserien. Overall, the results show that we can detect events with high recall, low latency, and accurate time estimation. The trade-off is a slightly lower precision compared to the current state-of-the-art, which has higher latency and performs better when a less accurate time estimation can be accepted. 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.**

***Index Terms*—Event detection, deep learning, sports analysis, soccer.**

I. Introduction

Today, we have access to billions of hours of content through services like YouTube and Netflix, not to mention hundreds of TV channels that run 24/7. In sports, televised events are broadcast around the clock, where a single event can last for hours before concluding. In this context, summarization techniques have become popular as a means to compress video content, be it news or sports, into only the most interesting parts. 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. 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 in real-time for live services.

Although event detection is useful in many scenarios, one especially interesting use-case is producing game highlights while the game is ongoing. However, for 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 [1] or require complex setups that use expensive equipment [2]. 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 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. 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 built a prototype and used multiple datasets to evaluate our method against a baseline comparison from SoccerNet [3] in terms of both accuracy and tolerance for delays. The results indicate that the approach presented in the current state-of-the-art [4] gives a higher detection accuracy if there is a higher tolerance for accurate time estimation, whereas our approach is competitive for lower tolerance on accurate time estimation, and superior when real-time detection is required.

II. 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 see how it contributes.

A. Sport Analytics

In the rising trend of machine learning-based methods, automatic sports analytics has been growing over the last few years. This field is wide, encompassing sports ranging from soccer [3] to curling [5], [6] and everything in-between [7], [1], [8], [9]. One popular technique is using computer vision to analyze sports broadcasts to produce statistics or generate highlight reels for games. In 2003, Ekin et al. [10] presented a framework to automatically summarizing soccer videos using cinematic and object-based features. Later, Giancola el at. presented SoccerNet [3], which is an open dataset for soccer video analysis. As a benchmark, they performed some preliminary experiments using I3D [11], C3D [12], and ResNet [13] pre-trained on ImageNet [14] as fixed feature extractors, followed by dimensionality reduction with principal component analysis (PCA). These features are sampled every 0.5s throughout the soccer videos. Afterward, based on these features, the authors use convolutional layers to capture temporal information, followed by pooling layers and a fully connected layer. This process is done locally around

DOI: https://doi.org/10.1109/ISM.2020.00030

an annotated event where the final trained model predicts events on full videos by a sliding-window approach and post-processing of the predictions. The authors tested this approach by using temporal windows ranging from 5-60 seconds, where 5-second windows performed best given strict requirements for distance between predicted events, while 20-second windows performed best overall. Cioppa et al. [4] improved the results on SoccerNet. The authors introduced a new loss function for temporal segmentation that varies the loss based on whether a prediction is located far before, just before, just after, or far after an event.

B. Action Detection

Action detection is a problem that has also been getting more attention over the last decade. Action detection aims to find what actions occur, at what time in videos. Videos have challenges such as change of camera view, cluttered backgrounds, and motion blur. There are also practical challenges due to high-computational costs, high storage requirements, and costly annotation for datasets.

Action Recognition is a classification task of short, trimmed clips of video. Datasets such as UCF101 [15] and HMDB-51 [16] have been important additions to the field in terms of making the action recognition task more accessible and creating a common benchmark to measure performance. In earlier works, features such as Histogram of Gradients (HOG), Histogram of Flow (HOF), Motion Boundary Histogram (MBH) [17], dense trajectories [18], [19] showed promising results. In 2014, Karpathy et al. [20] explored the use of CNNs using two streams, cropping the center of the image and down-sampling it as input. Furthermore, Simonyan et al. [21] introduced a Two-Stream CNN architecture related to the *Two-Stream hypothesis* [22], which is an idea of humans possessing two distinct visual systems. They used two separate CNNs, where a spatial CNN [23], pre-trained on ImageNet, which takes RGB input by sampling frames from video, and, the temporal stream, which uses optical flow fields as input. Carreira et al. [11] added 3D convolution to the Two-Stream structure, and Feichtenhofer et al. [24] explored the fusion process with 3D convolution and 3D pooling. The Two-Stream architecture is also extended with ST-ResNet [25], which adds residual connections [13] in both streams and from the motion stream to the spatial stream. Research into a combination of hand-crafted features and deep-learned features has also shown promising results [26]. Wang et al. [27], [28] proposed Temporal Segment Networks (TSN), arguing that existing models mostly focused on short-term motion, rather than long-range temporal structures. TSN takes a video input, separates it into multiple snippets, and makes a prediction using two-stream networks for each snippet. C3D [12] explored 3D convolution learning spatio-temporal features. When comparing with existing 2D convolution solutions, 3D convolution is indeed beneficial by adding temporal information such as motion.

When the kinetics-400 [29] was released, the Two-Stream Inflated 3D ConvNet (I3D) [11] model was introduced, using both 3D convolution and the inception architecture [30]. I3D works much like the two-stream networks, using one network for the RGB stream input and one for the optical flow. To enable 3D convolution, they inflated filters on a pre-trained 2D CNN, i.e., using transfer learning to initialize the 3D filters.

To avoid relying on only RGB or optical flow as input, PoTion [31] used a pose detection algorithm [32] to find spatial locations frame by frame for joints and key parts of the human body. Combined with RGB input, they could now use a much smaller CNN. However, a challenge with this approach is the reliance on good results from the pose detection model.

Later, Tran et al. [33] introduced Res (2+1)D, which is based on (2+1)D convolutions 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 [34] introduced the SlowFast architecture. The model is based on the use of two different frame rates as input, where the idea is to have a high-capacity *slow* pathway that sub-samples the input heavily, and a *fast* pathway that has less capacity but significantly higher frame rate.

Temporal action detection aims to find a temporal interval in an untrimmed video, along with a given action that occurs. Common datasets for this task are THUMOS [35] and ActivityNet [36]. Some success has been seen with sliding window approaches [35], however, this is computationally expensive and lacks flexibility due to fixed window sizes. Some works focus on generating temporal proposals, which can then be used with a classifier [37], [38]. Inspired by Faster R-CNN [39], Xu et al. [40] created an end-to-end model for temporal action detection that generates temporal region proposals, followed by classification.

Spatio-temporal action detection attempts to not only find a temporal interval for a given action, but also spatially. Finally, spotting [3] focuses on detecting sparse events in untrimmed videos. In contrast to temporal action detection, there is no temporal interval in which an event occurs, rather it is defined as an instant point in time.

III. Methodology

Building on the ideas presented above, we aim for a system both accurate and fast. In a real-time setting, latency is important. Therefore, models that use small temporal windows are useful since it is often necessary to gather contextual features both from the future and the past. If a model relies on information from the future, it will need to buffer the video, creating a delay. While a sliding-window approach is computationally expensive, current hardware can still run CNN's fast enough in many cases. This can also be mitigated in many cases by using a higher temporal stride.

A. Data and Pre-processing Pipeline

To detect events in untrimmed videos, we use a sliding window approach with a CNN classifier. Therefore, we recast the problem as a classification problem locally around each event. We sample N frames locally with the temporal anchor as the center. This approach leaves us with a biased model considering all the background data due to the sparsity of events. We consider video frames that are not near an annotated event as background.

Contextually, the scenes in soccer videos may also be very similar for different events, such as a goal attempt versus a goal, or a close up view on the referee pointing versus a yellow card. Therefore, we try to represent full videos by generating background samples to encourage the model to learn meaningful features. The borders between background and events have some inherent ambiguity, as seen in Sigurdsson et al. [41], which 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 $\frac{a_t+b_t}{2}$. At a minimum 180 seconds away from the closest annotated event, we have cleared a temporal distance large enough such that information from events should not be contained within a background sample. While it is possible to sample a replay of an event, we find it 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 I: The number of samples per class.

Table I 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 other 98.83% seconds containing something else. Therefore, the vast majority of a soccer match is background in relation to these three classes. There are some weaknesses as well since we automatically generate new samples. *Background* may be annotated during replays of any of the three events, while it is likely rare, we expect some 'bad' samples to 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. Secondly, we randomly crop a 112×112 clip. Finally, we randomly flip each frame with a probability of 0.5. Considering that soccer fields are symmetric around that center, therefore this may make the model more robust to events on either side. In Figure 1, we show some example frames included in the three classes.

B. Model Architecture

We use an 18-layered 3D ResNet as in Tran et al. [33]. 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 [29]. Due to the pooling layer, an arbitrary number of video frames can be used as input.

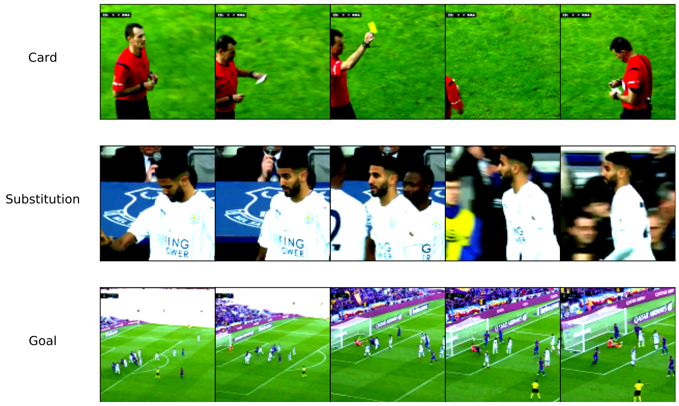


Fig. 1: Sample frames visualized for a sample of 128 frames. The middle frame is at the annotated time.

C. Training and Implementation Details

We implemented the model in PyTorch [42] and trained on a machine consisting of 16 Nvidia Tesla V100 GPUs, which combined have a total memory capacity of 512 Gigabytes.

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

1) Runtime: Table II shows the time spent with our method to create our prediction signal. Samples are 128 frames with resolution 224×398, that 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 transforms.

2) Hyperparameters: We use SGD with 0.9 momentum for each model and use 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 minibatch size of 64.

3) Configuration: We evaluate results on samples from the validation split. To find a good configuration for the classification model, we experiment with resolutions of 112×112 and 224×224, and the number of frames ranging between 8 and 128. The input data consists of video at 25 frames-per-second with 224×398 resolution. We achieve the best results with R3D-18 pre-trained on Kinetics-400 [29] with

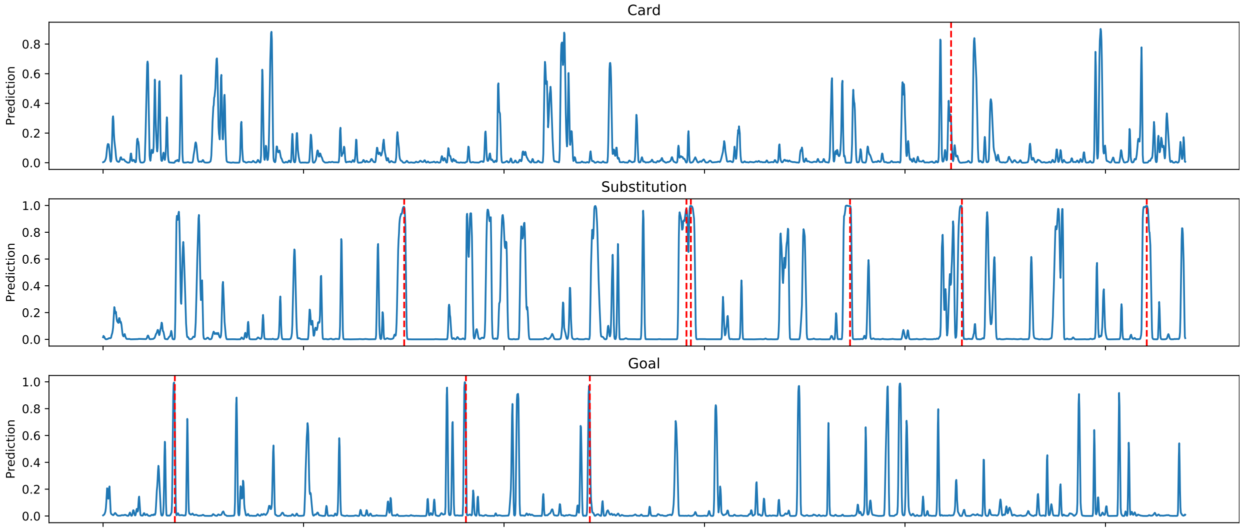


Fig. 2: Softmax confidence for each class over 45-minutes with ground truth.

128 frame inputs using 112×112 resolution. The difference between 112×112 resolution and 224×224 was negligible. The number of frames and the use of a pre-trained model had significant effects on the result. The video used is run at 25 frames-per-second, therefore, a 128 frame model has a temporal footprint of 5.12 seconds. The model achieves 88.4% accuracy for classification on the validation set samples.

D. 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.

IV. Results and Discussion

To test the proposed model, we have performed various experiments to compare the approach to the state of the art.

A. Metrics

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 higher confidence than our threshold. Formally, we use the condition in Equation 1:

$$|gt_{spot} - p_{spot}| < \frac{\delta}{2} \tag{1}$$

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 1 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, is considered a false positive. If, for a given gt_{spot}, there are no predictions made where this condition holds, we consider it a false negative. We use the condition in Equation 1 to calculate the average precision (AP) for each class, and subsequently we calculate the mean average precision (mAP). This is done for tolerances δ ranging between 5 and 60 seconds. Finally, we use the mAP scores calculated for different δ to calculate the AUC to get the Average-mAP score which gives us some insight into the models total performance in the range of 5 - 60 seconds.

B. Results

1) SoccerNet: Figure 2 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 3 shows our final predictions. After applying a threshold of 0.9, we observe that we have removed most of the noise. For the event *Card*, we get a false positive long after the event itself. Looking at both *Substitution* and *Goal*, we observe that we get reasonable predictions close to the ground truth. However, we also end up with multiple false positives that are entirely unrelated.

In Figure 4, 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 is used. Therefore, results at higher tolerance are of less interest. The threshold used is found by estimating the threshold that optimizes f1-score at tolerance δ = 5.

2) Cross dataset: 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 from Norwegian Eliteserien and Swedish Allsvenskan,

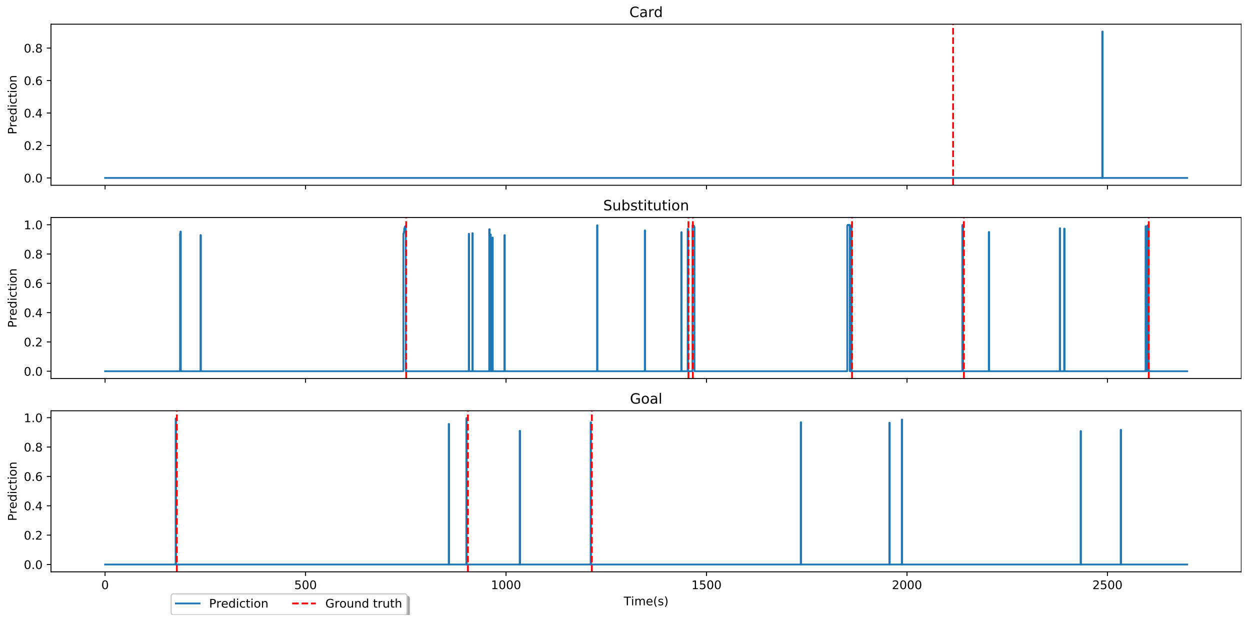


Fig. 3: Prediction for each second after mean filtering and thresholding. Red dotted line shows ground truth for each class.

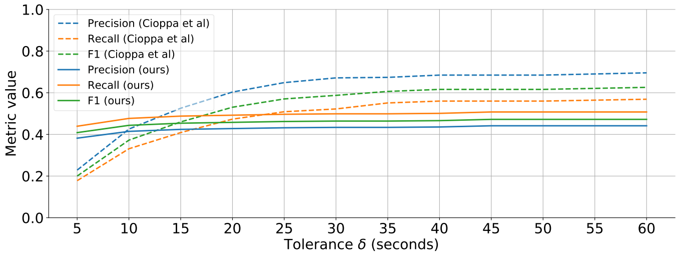


Fig. 4: Comparison of recall, precision and f1-score relative to a tolerance δ for the class '*Card*'. Code used to generate figure was adapted from: https://github.com/cioppaanthony/context-aware-loss [4].

ranging from 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. In Figure 5, we find that there is a trend of high responses around 25 seconds. However, we note that based on these samples, the prediction score is lower compared to actual goals. This indicates that we should expect many false positives at lower thresholds and that it is crucial to have a high enough threshold.

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

TABLE III: Statistics for Allsvenskan and Eliteserien clips.

Table III shows the number of goals and the number of goal attempts. In Figure 5, we observe how our approach generally seems to react to goal attempts. This is further exemplified in Figure 6.

Method	mAP
SoccerNet baseline 20s [3]	49.7
Cioppa et al. [4]	**62.5**
Ours	32.0

TABLE IV: Results for the spotting task in SoccerNet.

3) Comparison to state-of-the-art: Table IV 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 higher tolerances will only result in finding spots that are false positives. In a real-time setting, it may be important to have an as little delay in predictions as possible. For our approach, a live prediction will have a delay of about 4 seconds. This includes buffering future frames and computation. The baseline model would have about 10 seconds delay, while the current state-of-the-art [4] would have about 100 seconds. It seems that long-range contextual features can boost performance, and that there is a trade-off between delay and the practical ability to buffer future video frames.

V. Ablation Study

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

A. Local Behavior with Sliding Window

The sliding window approach was performed with a stride of 1 second. The videos' frame rate is 25 frames-per-second,

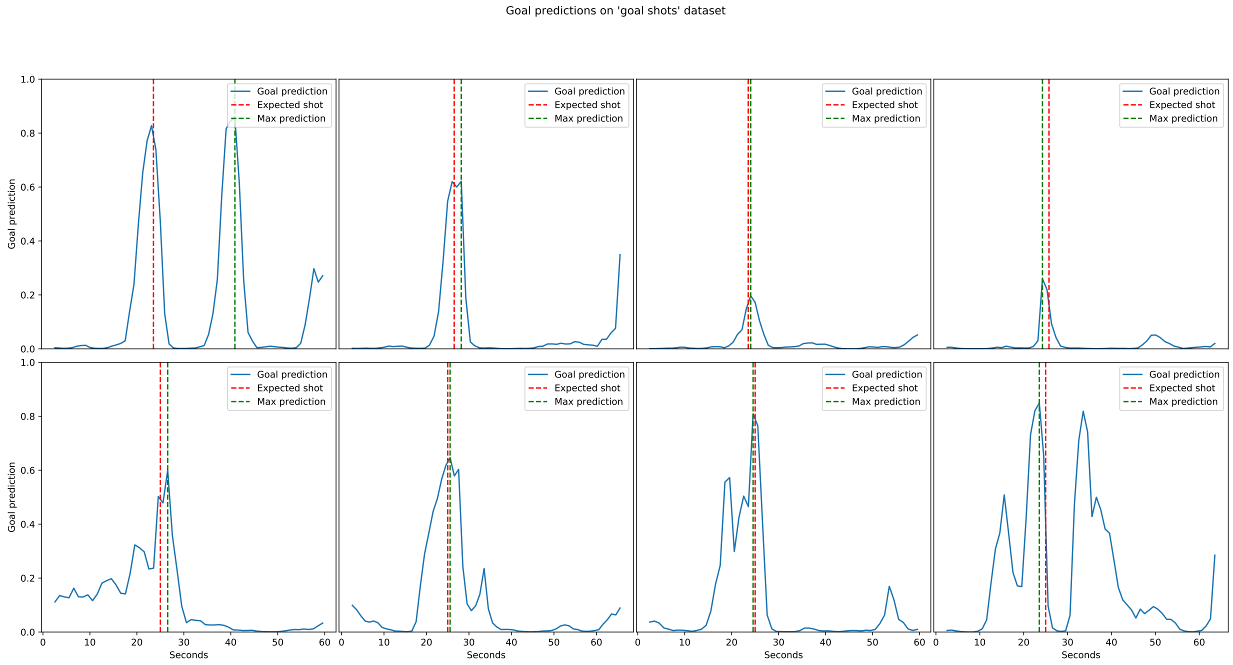


Fig. 5: Randomly sampled from Allsvenskan 'goal attempts' clips.

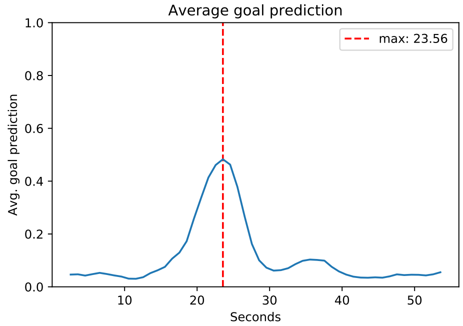


Fig. 6: The average prediction for the 'goal attempts' clips.

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 do 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 7, 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 conjeture 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.

B. 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. [43] to generate CAMs.

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 2:

$$S_c = B_c + \sum_{i=1}^{N} w_i^c F_i \tag{2}$$

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 3:

$$F_i = \frac{1}{K} \sum_{t,x,y} V_i(t, x, y) \tag{3}$$

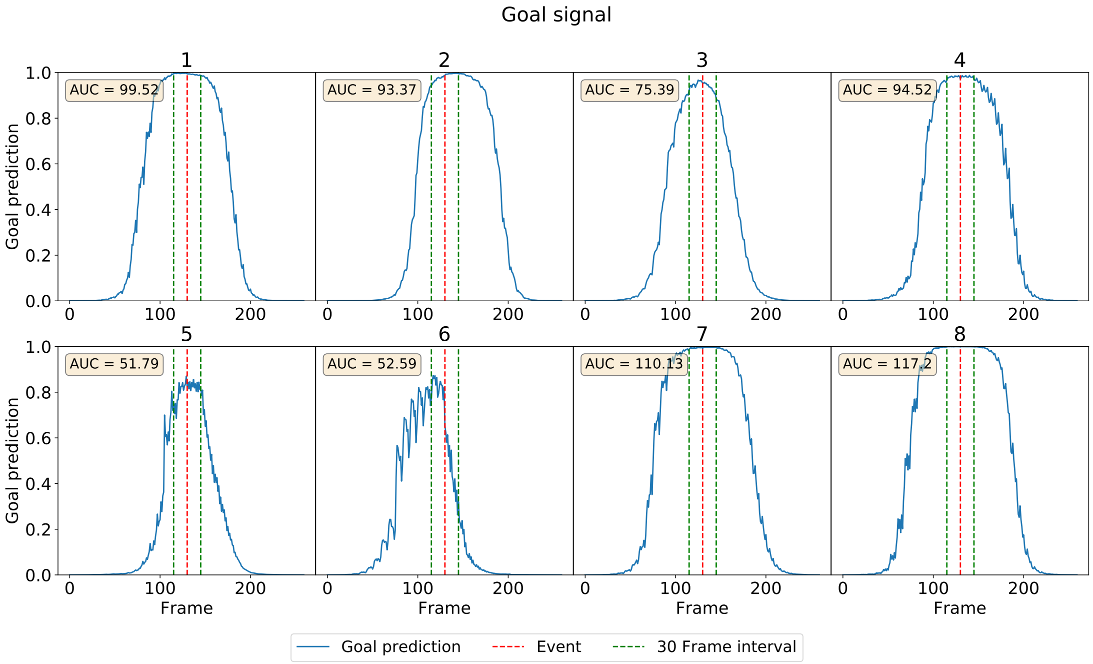


Fig. 7: Softmax output for the event *Goal* with a stride of 1 frame over 260 frames. Samples are correctly predicted validation samples. Red dotted line shows the output prediction for the 128 frames of the event.

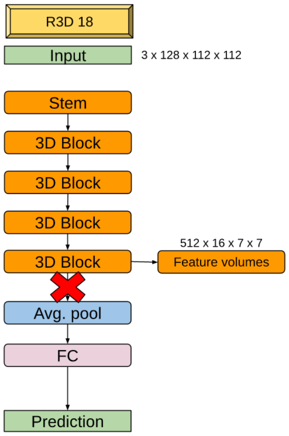
V_i holds our N feature volumes with K elements each, which in our case is $16 \times 7 \times 7 = 784$, where t denotes our temporal dimension, and x, y our spatial dimensions. Thereby:

$$\begin{aligned} S_c &= B_c + \sum_{i=1}^{N} w_i^c \frac{1}{K} \sum_{t,x,y} V_i(t,x,y) \\ &= B_c + \frac{1}{K} \sum_{t,x,y} \sum_{i=1}^{N} w_i^c V_i(t,x,y) \end{aligned} \tag{4}$$

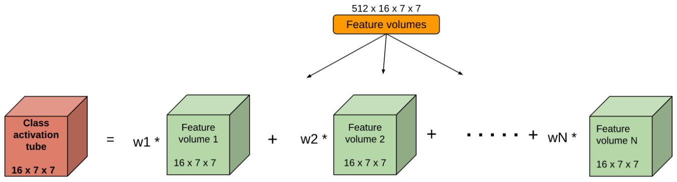
In Equation 4, we attempt to show the relationship between the pre-average pool feature volumes and the weights and bias used to compute the class scores S_c. As Zhou et al. [43], we ignore the bias term moving forward as it has little impact on the final results. We define a class feature tube (CAT) as a 3-dimensional equivalent of class activation maps as follows:

$$T_c(t,x,y) = \sum_{i=1}^{N} w_i^c V_i(t,x,y) \tag{5}$$

Going from Equation 5 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 8a 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 5, in Figure 8b.



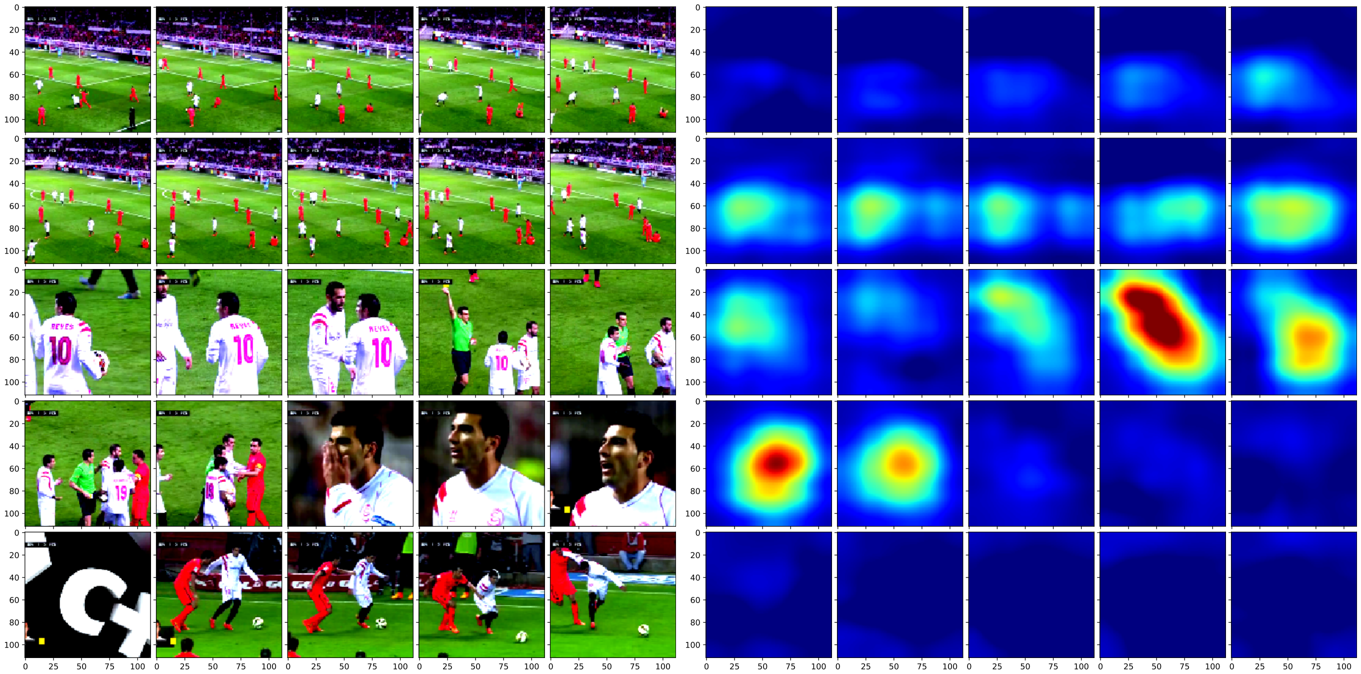
(a) Structure in the R3D model. The feature volumes V_i are extracted prior to global average pooling layer in order to preserve spatial and temporal information.



(b) Calculation of class activation tubes T_c.

Fig. 8: Extraction of features and calculation of CAT.

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 com-



(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 $T_{card}(8, x, y)$.

Fig. 9: Results from CATs spatio-temporally for the event *Card*.

pute 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 *class activation maps*. Furthermore, we refer to the 1-dimensional temporal signal as *class activation signal*.

1) Sparse CAMs: In order to gain some insight into the spatial features of the network, 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 9, we observe how the model seems to react strongly to a referee holding a card.

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.

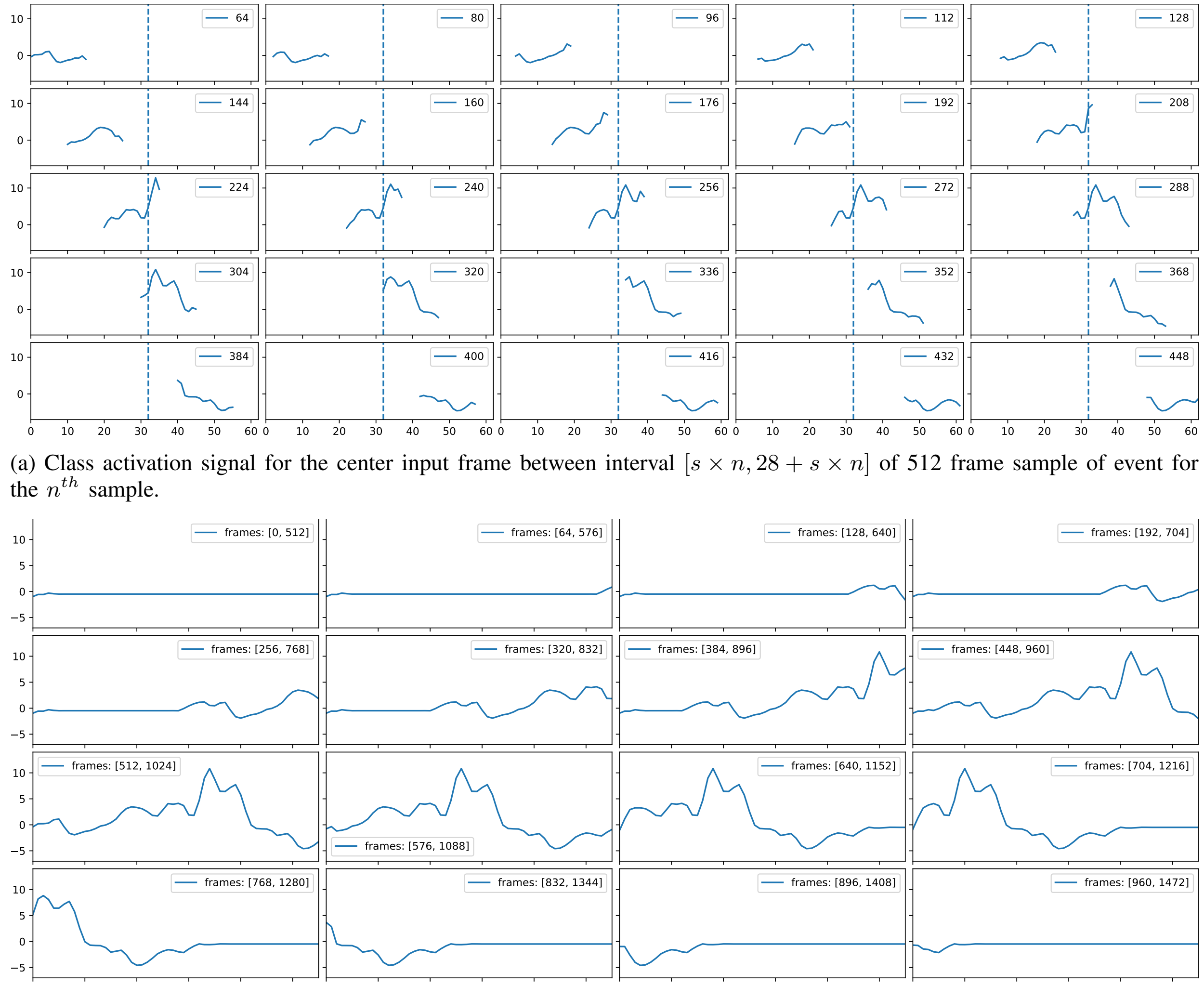
Zhou et al. [43] showed that CAMs could be used for object detection. It may also be possible to use this for more accurate and efficient detection of events. In Figure 10a, we show how our model reacts in the temporal dimension. It seems that the strongest activations occur close to the actual event. 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 for more accurate annotations. This is seen when increasing the temporal footprint as well. Figure 10b shows that the class activation signal indicates strong reactions at the actual event.

VI. Conclusion

In this paper, we presented an algorithm to automatically detect and annotate segments of soccer videos using convolutional neural networks (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 class activation maps (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 [3]. Compared to the results presented in the SoccerNet papers, we achieve a slightly lower detection accuracy compared to the state-of-the-art if there is a higher tolerance for accurate time estimation. However, our approach is competitive when accurate time estimation is important (finding the event's exact time) and better 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. Finally, we believe that the presented approach will scale to other sports, which will be investigated in the future.

References

[1] R. Kapela, K. McGuinness, A. Swietlicka, and N. E. O'Connor, "Real-time event detection in field sport videos," in *Proc. of CVPR*, 2014.



(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. 10: Results from CATs temporally for the event *Card*.

[2] 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*, vol. 113, no. 5, pp. 622 – 632, 2009.

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

[4] 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*, June 2020.

[5] 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*, vol. 19, no. 2, p. 337, 2020.

[6] J. Bradley, "The sports science of curling: A practical review," *Journal of Sports Science & Medicine*, vol. 8, pp. 495–500, 12 2009.

[7] V. Bettadapura, C. Pantofaru, and I. Essa, "Leveraging contextual cues for generating basketball highlights," in *Proc. of MM*, 2016, p. 908–917.

[8] 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*, vol. 60, no. 3, pp. 641–667, 2012.

[9] N. Homayounfar, S. Fidler, and R. Urtasun, "Sports field localization via deep structured models," in *Proc. of CVPR*, 2017, pp. 4012–4020.

[10] A. Ekin, A. M. Tekalp, and R. Mehrotra, "Automatic soccer video analysis and summarization," *IEEE Transactions on Image Processing*, vol. 12, no. 7, pp. 796–807, 2003.

[11] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Proc. of CVPR*, 2017, pp. 4724–4733.

[12] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proc. of ICCV*, 2015, p. 4489–4497.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of CVPR*, 2016, pp. 770–778.

[14] 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*, 2009, pp. 248–255.

[15] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *CoRR*, vol. abs/1212.0402, Dec. 2012.

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

[17] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Proc. of ECCV*, 2006, p. 428–441.

[18] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *International Journal of Computer Vision*, vol. 103, no. 1, pp. 60–79, 2013.

[19] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proc. of ICCV*, 2013, pp. 3551–3558.

[20] 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*, Jun. 2014, pp. 1725–1732.

[21] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. of NIPS*, 2014, p. 568–576.

[22] M. A. Goodale and A. D. Milner, "Separate visual pathways for perception and action," *Trends in Neurosciences*, vol. 15, no. 1, pp. 20 – 25, 1992.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. of NIPS*, 2012, pp. 1097–1105.

[24] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proc. of CVPR*, Jun. 2016, pp. 1933–1941.

[25] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Spatiotemporal residual networks for video action recognition," in *Proc. of NIPS*, 2016, p. 3476–3484.

[26] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *Proc. of CVPR*, Jun. 2015, pp. 4305–4314.

[27] Z. Shou, D. Wang, and S.-F. Chang, "Temporal action localization in untrimmed videos via multi-stage cnns," in *Proc. of CVPR*, Jun. 2016, pp. 1049–1058.

[28] 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*, 2016, pp. 20–36.

[29] 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*, vol. abs/1705.06950, May 2017.

[30] 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*, 2015, pp. 1–9.

[31] V. Choutas, P. Weinzaepfel, J. Revaud, and C. Schmid, "Potion: Pose motion representation for action recognition," in *Proc. of CVPR*, June 2018.

[32] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proc. of CVPR*, 2017, pp. 1302–1310.

[33] 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*, Jun. 2018, pp. 6450–6459.

[34] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *Proc. of ICCV*, Oct. 2019, pp. 6202–6211.

[35] 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*, vol. 155, pp. 1–23, 2017.

[36] 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*, 2015, pp. 961–970.

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

[38] T. Lin, X. Zhao, H. Su, C. Wang, and M. Yang, "Bsn: Boundary sensitive network for temporal action proposal generation," in *Proc. of ECCV*, 2018.

[39] 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*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 91–99.

[40] H. Xu, A. Das, and K. Saenko, "R-c3d: Region convolutional 3d network for temporal activity detection," in *Proc. of ICCV*, 2017.

[41] 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.

[42] 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*, 2019, pp. 8024–8035.

[43] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proc. of CVPR*, 2016, pp. 2921–2929.