Universität Bielefeld

Technische Fakultät

Masterarbeit

im Studiengang Intelligente Systeme

zum Thema:

Deep Learning for Event Detection from Sports Broadcast Videos

vorgelegt von

Ferdinand Schlatt

Matrikel-Nr: 3129159

1. Gutachterin	Prof. Dr. Barbara Hammer Technische Fakultät, AG Machine Learning
2. Gutachter	Fabian Hinder Technische Fakultät, AG Machine Learning

Bielefeld, im März, 2020

Abstract

Detecting relevant events from long video sequences is an important task in many domains. Examples include video surveillance or traffic monitoring. Reviewing video footage manually is often time-consuming and expensive. In these cases, automatic event detection systems can free human resources for more important tasks. Recently, artificial neural networks have shown great potential in video processing tasks. However, most work focuses on classifying short video clips into different classes. This motivates the development of a neural network architecture that is able to detect events from a continuous stream of frames, thereby allowing arbitrary length videos. Player-ball interactions in soccer broadcast videos are used as the test domain. These interactions provide an ideal testbed as they establish a clearly defined event structure in a visually consistent environment. Next to the introduction of a novel data imbalance sensitive loss function, different recurrent and attention augmented convolutional networks are tested on the task. In the end, the best model succeeds in detecting soccer events in general but often fails for chaotic play phases or when the ball is occluded.

Contents

1 Introduction

Detecting relevant events from long video sequences is an important task in many domains. For example, segmenting surveillance videos [54] or monitoring traffic to report live accidents or congestions [34]. Reviewing video footage manually is often time-consuming, tedious and expensive, but in many cases necessary. In these cases, automatic event detection can help by increasing efficiency and accessibility. Either fully automatic systems or semi-automatic systems which remove irrelevant video segments could reduce the manual labor time and free human resources for more difficult tasks.

In the professional sports industry, data and video analysis are becoming an increasingly important field. Especially soccer, as the world's most popular sport, is developing a need for smarter and faster analysis. For example, broadcasters want to produce high quality and informative content for the user, visualizing and integrating important statistics about a game as quickly as possible. Scouts want to review video footage of prospective talents and the post-game video analysis has become a vital part of coaching.

A growing number of tools and techniques have been developed to aid in these processes. As examples: simple events can be extracted using ball and player trajectories [53], audio and camera cues have been used to determine important event segments [28, 44] and neural network ensembles were trained to detect set piece situations [9]. The drawbacks of all previous approaches can be split into two categories. In the first case, additional data on top of or instead of the broadcast video is required. Multi-camera systems, GPS tracking or other means are used to augment the visual data. In the other, the type of detected events is confined to an extremely small and easily detectable set. A system working only on a broadcast video that can detect a wide variety of events has yet to be developed and is the topic of this thesis.

Specifically, the goal is to develop a model that is capable of discerning if a frame of a soccer broadcast video contains an event. In essence, a video should be segmented into alternating groups of frames with and without event presence. Because of the length of soccer broadcast videos, and also to enable online analysis, frames are tagged sequentially. That is, a model does not have access to the full video at once. An event may be any basic interaction of a player with the ball, for instance, passes, shots, receptions or headers. Many relevant events from a soccer game can then be extracted with an adequately accurate model and further measures

for event classification can be applied.

This thesis compares the performance of several different neural network architectures for soccer event detection. Neural networks have been pushing the state of the art in computer vision tasks rapidly in the recent past. AlexNet's win in the 2012 ImageNet competition sparked a quick succession of advances in image processing [32]. ResNets were able to show that training extremely deep convolutional networks is feasible [21] and Mask RCNNs set an unprecedented standard in object segmentation [22]. The principles learned from image processing have also successfully been applied to video processing. For example, C3D networks expanded 2D convolutional networks to a third time dimension for video action detection [55].

The prominent challenge when working with video data is combining spatial and temporal features. The transition from detecting static features in a single frame to features describing movement over multiple frames is far from trivial. Multiple different approaches to tackle this problem have been used in the past. They can be coarsely divided into three categories. First, passing a stack of frames into a 3D convolutional network [66]. Second, using a so-called two-stream approach, by applying two separate convolution networks on the RGB frames and the optical flow, and combining these in a fully connected layer [50]. Lastly, applying frame-wise 2D convolutions and passing the image feature vector into a recurrent network [4, 51]. Different combinations of the methods are also possible [18, 61]. However, all the previous frameworks were applied to entire videos as a whole. Because of the length of soccer broadcast videos, only the latter framework is a viable option and is tested in this thesis.

Next to the spatiotemporal information, to correctly detect interactions between players and the ball, a model must be able to determine the relations between different objects within an image. By only considering a local neighborhood, the 2D convolution operator has a major drawback in that regard. Recent advances in natural language processing have shown that self-attention is an effective method for capturing global relations between words [60]. Self-attention has since been used to augment convolutional layers, granting them the capability of capturing local features and setting these into a global context [6].

Furthermore, because soccer events are spread sparsely throughout a broadcast video, a large majority of frames are tagged as not containing an event. Neural networks have difficulties when working with imbalanced data [62]. To counteract the data imbalance, imbalance sensitive loss functions can be used. Next to the popular weighted binary cross entropy, it is also possible to augment the F1 score to use it directly as the loss function [41]. Because of the F1 score's bias for the positive class, an alternative loss based on the Matthew's correlation coefficient is suggested in this thesis.

All these developments motivate the specific network architectures tested in this thesis. Starting from a default ResNet-50 architecture, iteratively different archi-

tectural enhancements are tested for performance improvement on the soccer event detection task. First, a range of image preprocessing methods is tested. Next, identical networks are trained using the three aforementioned loss functions. Finally, the ResNet is expanded to include attention and a recurrent LSTM layer. Because no comparison models exist, all models are compared to a random baseline. To conclude, the best model is evaluated in finer detail and potential deficits discussed.

In summary, this thesis is considered to make several contributions: 1) attention augmented convolution is combined with a recurrent network for video event detection 2) instead of video level labels, the model is trained to detect events from an arbitrary length video continuously 3) the Matthew's correlation coefficient loss is introduced as an unbiased data imbalance sensitive loss function 4) event detection is applied to live sports broadcasts on a level of unprecedented detail.

In the end, none of the preprocessing methods were able to improve soccer event detection and were not applied in any of the subsequent experiments. For the data sensitive loss functions, the F1 and Matthew's correlation coefficient loss were able to show significant performance improvements over the commonly used weighted binary cross entropy. Attention augmentation was unable to provide any additional value. On the other hand, adding recurrence did lead to measurable performance increases across all evaluation scores. The final model then showed to successfully detect clearly separated events but struggled for events in more chaotic phases of play with frequent occlusions or deflections.

The remainder of the thesis is structured as follows. First, related work in the fields of sports and deep neural network event detection is summarized. Next, the video and event datasets, the respective measures to clean the data and the different levels of preprocessing are described. Details about the video processing and event classification network architectures follow. Subsequently, the different experiments are evaluated. The results and potential future improvements are discussed next. Finally, a summary concludes the thesis.

2 Related Work

2.1 Soccer Visual Analysis & Event Detection

A large portion of research in automatic visual analysis in soccer revolves around tracking the players and ball [38, 48, 58, 67]. Notably, in [68], Zhu et al. use Gaussian mixture models to mask out the field and an SVM, in turn, to detect and track players in the non-masked candidate regions. Beetz et al. [5] use a similar approach but add color matching as well as compactness and height constraints on the non-masked player blobs to improve player detection. Additionally, they add tracking by matching a field model to detected field lines and thereby estimate the camera parameters. Sabirin et al. improve on player tracking further by adding an extra occlusion handling step [46].

This tracking data has been used in multiple studies for event detection and classification. Stein et al. develop a framework to hierarchically combine events parsed from tracking data [53]. 3D positional player and ball data is combined with a video in [57] by Tsunoda et al. to detect pass, dribble and shot events in futsal videos using hierarchical LSTMs. Imai et al. evaluate several different machine learning techniques on a professionally tracked soccer game to detect ten different event classes [26]. Next to tracking data, Pradeep et al. have also used audio to detect significant events in sports videos [44]. It is assumed that the amplitude of the audio track is highest around important events when the commentator is most excited. Subsequently, frames are extracted around audio peaks from the video.

Many detection systems also rely solely on visual input. Early attempts usually used hand-crafted visual features combined with a range of machine learning techniques. Chen et al. use a combination of color ratio, camera angle and temporal features with a small neural network to detect corner kicks, throw-ins and free kicks close to the box [9]. Bag of visual words is used by Baccouche et al. in [3] to extract features that are passed into an LSTM to detect different set pieces and shots on goal. In [47], Sarkar et al. detect passes from player and ball ROIs using a minimum cost flow network.

More recently, deep neural networks have been used to extract spatiotemporal features from soccer videos. Cioppa et al. train a ResNet to extract the player and field line segmentation masks [11]. The mask is then used to compute the camera angle and position of players to heuristically classify a frame into four different game

phases. To train a model end to end, Cioppa et al. then propose a novel loss function for extremely sparse events [12]. The loss function is evaluated on the SoccerNet dataset [19], improving on the baseline model by 12.8% mAP. Albeit, SoccerNet only contains goal, card and substitution events.

Finally, Tsagkatakis et al. [56] implement a two-stream network to classify two to three second video clips into goal and no goal classes. To conclude, previous literature commonly focuses only on the task of classifying short sequences. When the temporal location of an event is being detected, the set of events is extremely limited. This is usually due to the lack of an appropriate dataset. In contrast to this, in this thesis, the temporal location of a large set of basic soccer events is detected from an extensive professionally tagged soccer event dataset.

2.2 Neural Network Action Classification

Thanks to many challenging and extensive datasets [1, 30, 33, 49, 52], especially the task of action classification has been advancing the state of the art in neural network video processing. Karpathy et al. explored different techniques to fuse the feature maps of frame-wise applied pretrained convolutional networks [29]. So-called slow fusion in which temporal convolutions are applied in different levels within the convolutional hierarchy proved to be most effective but showed only little improvement over a static classification model. In [50], Simonyan et al. first proposed the two-stream approach, training two separate convolutional networks on RGB frames and a stack of optical flow maps. Ballas et al. meanwhile proposed an architecture leveraging recurrent neural networks [4]. The hidden layers within GRU RNN cells were replaced by 2D convolutions to create recurrent spatial neural networks, which were applied to feature maps from a pretrained network. Another alternative was proposed by Tran et al., where 3D convolutional networks were applied to stacks of RGB frames to learn spatiotemporal features [55].

Following this, many prominent publications relied on a mix of the previously mentioned work but improved the frame sampling techniques or pretrained convolutional kernels. I3D networks, proposed by Carreira et al., expand pretrained 2D convolutional kernels to 3D, circumventing the need to train 3D convolutions from scratch [7]. Wang et al. then proposed an improved sampling technique, coined TSN, in which frames are subsampled from different video segments and a class score computed from the aggregated sample features [61]. Lastly, the SlowFast framework by Feichtenhofer et al. used two different pathways, alike to two-stream networks, to achieve state of the art accuracy on many prominent action classification datasets [18]. One pathway uses high temporal resolution to capture motion, with a low channel granularity to improve performance. The second pathway samples frames at a low resolution, capturing spatial semantics.

2.3 Neural Network Event Detection

While action classification is the task of finding which action is present in a video, this thesis examines action detection, i.e. locating the time at which an action has occurred. The two most prominent datasets for this task are ActivityNet [16] and THUMOS 14 [25]. Both include a large set of untrimmed action videos from various action classes, with annotated temporal boundaries of the actions.

Xiong et al. improved on the initial benchmark scores by using a TSN network to detect actionness of sampled video segments [65]. The actionness scores were used to stitch together temporal proposals, which were subsequently passed into an action classification network. In contrast to this top-down approach, Lin et al. employed a bottom-up approach. A two-stream network is trained to detect actionness, as well as starting and end probability sequences for an entire video [36]. These vectors are then passed to a so-called boundary sensitive network to compute proposals and proposal confidences to obtain finer control over temporal boundaries.

The concept is improved upon in [35] by matching the spatiotemporal features from a two-stream network to a matrix of densely sampled proposal boundaries. Specifically, a boundary matching network computes probabilities for pairs of proposal start times and durations. Finally, while just computing action boundaries indirectly, Xu et al. use C3D features with a segment proposal network to create temporal sub action proposals [66]. These are passed into an LSTM to generate captions for videos in the ActivityNet dataset.

2.4 Neural Network Visual Attention

Regarding visual attention, a couple of different important contributions have been made in the recent past. In essence, visual attention attempts to solve the issue that the output of a convolutional network only depends on the local neighborhood in the input. Dynamic filter networks, by Jia et al., augment the classical convolution kernel to dynamically change with the input [27]. That is, the input that a kernel is being applied to depends itself on that input. Hu et al. on the other hand add a channel-wise learnable weight to the output of a convolutional layer [24]. Adding these so-called squeeze and excitation layers to a ResNet-50 network improves its score to that of a plain ResNet-101 architecture, while only adding little additional computational cost. However, these layers do not allow for inferences across a single channel, i.e. pixel-wise attention. For this, Bello et al. borrow the self-attention concept made successful by the Transformer [60] architecture for natural language translation tasks [6]. Self-attention is expanded into two dimensions to allow for pixel-wise global attention within a convolution kernel, further improving on the result of squeeze and excitation networks.

3 Data

In the following, an overview of the data is given. The entire dataset encompasses a total of 183 fully tagged videos from the 2018/2019 English Premier League season, provided by matchmetrics GmbH [20]. A tagged video consists of the TV broadcast video, as well as a set of timestamps and ids marking the events. Superfluous parts like halftime and pre-/postgame segments are cut from the video. In total, 10% of data was set aside for testing and validation. Of all videos, twelve were randomly selected to create the test set, six videos are used as the validation set and the remaining 165 comprise the training set. Events were exhaustively tagged by human operators. The number of events per game ranges from 1717 to 3338, with a total of 479,218 tagged across all videos. The following sections provide further details about the video and event processing pipeline, as well as a short remark about the data imbalance between video frames and events.

3.1 Videos

To train a neural network efficiently, a data loader needs to be able to quickly transfer training data from storage to GPU memory. In this regard, working on long video files poses additional challenges compared to other data modalities. The first major issue is that video files are usually quite large. Accordingly, loading an entire video into memory is often not feasible or practical. At the same time, to reduce the size in storage, videos are encoded. This makes random sampling of frames extremely inefficient. To sample a single frame, multiple preceding frames need to be decoded first. A solution is needed which allows for efficient and randomized sampling.

In a soccer match, play is frequently disrupted by short breaks. The ball travels out of bounds or a player is fouled which creates a longer eventless sequence. These breaks in play give natural time points at which to cut a video and divide it into a subset of smaller clips. By randomizing the clip order, a semi-random but efficient sampling can be achieved. A video decoder is opened for each video file and decodes the first frame of a random clip. Because decoders are designed to efficiently access successive video frames, the following frame can be grabbed quickly. Each decoder then parses successive frames until the end of a clip is reached. The decoder then seeks to the first frame of the next random clip.

The timestamps of the events are used to set the video clip bounds. A video clip

is defined as a set of consecutive frames that start and end on an event timestamp. A clip may contain multiple other event timestamps, with a maximum time difference of three seconds between two events. This definition allows for extremely short clips if, for example, two isolated events happen in quick succession. However, as events are usually grouped in longer chains, this case is exceedingly rare. A random number of up to twelve frames are added to the front and end of a clip to ensure that event on- and offsets are not correlated with clip on- and offsets.

Usually, when training a recurrent neural network, complete sequences are passed through the network. However, the video clips are most often still too long to fit into GPU memory. Instead, another solution is to pack individual frames from multiple video clips into a single batch. The model can then update its internal memory frame by frame. It is therefore additionally important to guarantee that a batch does not contain more than one frame from the same clip. Otherwise, one memory state may be overwritten by another from the same clip. Further details on exactly how the recurrent network's memory is computed and stored can be found in Section 4.1.3. In summary, a batch from the video clip dataset then consists of multiple frames from pairwise different randomized video clips.

A couple of further preprocessing measures were applied to the videos to increase decoding speed and also to decrease GPU memory usage. First, all videos were standardized to 15 frames per second to reduce the total number of frames. Next, to decrease the memory footprint, the resolution was decreased to 400 by 226 pixels. In a final step, the image is normalized channel-wise to zero mean and one variance by the RGB means and standard deviations. In the end, the clip dataset consists of 35,102 clips with 7,144,079 frames. Figure 3.1 provides a couple of example frames.

These examples also nicely demonstrate some of the challenges for a model to reliably detect events. Soccer broadcast videos make use of different techniques to

Figure 3.1: Example frames from the video dataset.

make the viewing experience more enjoyable. The camera pans and zooms with the action. Different camera angles are used to highlight certain events. A model must, therefore, be able to cope with different camera angles, close up shots or visual overlays occluding parts of the frame. Additionally, different lighting conditions can drastically alter the appearance of the field. In all cases, a model must be able to extract robust spatiotemporal features to be able to detect events reliably.

3.2 Events

The focus of this thesis is to detect events with an interaction between a player and the ball. Soccer events are therefore defined as elemental action points. That is, every event is described by a single time point and an action type. For example, these include a player passing the ball, executing a throw-in or tackling the player with the ball. It is important to note that these do not include events that span over a longer period as well as events not involving player-ball interaction. Among others, the excluded events include goals, the ball leaving the playing area, referee actions or complex events like dribbling the ball.

The rationale behind excluding these types of events is that player-ball interaction events define a clear but difficult task. A model must be able to extract spatiotemporal features from an image to be able to set a player and ball into context. Since only the player and ball are of interest though, it is a strictly defined task that is comparably simple to diagnose for issues. This is especially important for an initial study such as this one, as no previous work could be found to compare performance with. By including the other event types, the set of required spatiotemporal features would increase. For example, to detect fouls, inferences on player-player interactions and player allegiance are required. This adds further variability and increases the difficulty of validly evaluating a model. Therefore, the final set of event types only includes events involving player-ball interactions.

The final set includes 17 event types in total. Table 3.1 gives an overview of the different event types and their respective quantities. Aerial duels include all events where two opposing players attempt to touch a high ball. A touch of the ball that is only intended to bring the ball away from a player's goal is defined as a clearance. Control events are small dribbling touches by a player. A contested ball is another type of duel where two players reach the ball at the same time and no clear ball possession could be defined before the event. Involuntary or uncontrolled touches by a player are counted as deflections. There are two different categories for goalkeeper actions based on the body part they are executed with. Goalkeeper kicks are long-range kicks where the goalkeeper kicks the ball directly from the hands. A goalkeeper throw is when the goalkeeper releases the ball directly from the hands. Kick-offs are the initial touch of the ball to start a half or to reinitiate play after a goal. Passes

include any touch of the ball that is directed at a teammate. Ball recoveries are the uncontested variant of contested balls. That is, a ball is located in open space and collected by a player without opponent interference. Set pieces include a variety of different events that all reinitiate play because of different reasons. These include corner kicks, goal kicks and throw-ins for example. If a player actively blocks another player from touching the ball to stay in possession, this is considered a shield. Shots are any ball touches intended to put the ball past the goal line. Tackles are defensive actions in which a player without possession of the ball attempts to gain possession from another player. Finally, take-ons are ball touches to move past an opposing player.

The event counts show that the large majority of events are passes and completions. They make up over 67% of all events. The other 15 events make up the remaining third. Because the task is to detect events and not to classify them, this data imbalance doesn't pose an immediate problem.

Together, these event types constitute an almost exhaustive set, such that close to any interaction between a player and the ball can be classified into one of these categories. An event is linked to the video dataset by its timestamp. For every event, the frame with the closest timestamp to the event timestamp is considered the event frame. Due to human error and frame discretization, the timestamp isn't

Event Type	Number of Events	Ratio
Aerial Duel	7392	01.99%
Clearance	11509	03.09%
Completion	97535	26.23%
Control	28180	07.58%
Contested Ball	1647	00.44%
Deflection	12389	03.33%
Goalkeeper Kick	449	00.12%
Goalkeeper Throw	881	00.24%
Interception	4400	01.18%
Kick-Off	431	00.12%
Pass	152945	41.12%
Recovery	17149	04.61%
Set Piece	12096	03.25%
Shield	2071	00.56%
Shot	4315	01.16%
Tackle	8764	02.36%
Take-On	9760	02.62%
Total	371913	100%

Table 3.1: Overview of event types with number of occurrences and ratios

always exact. It may be offset by a couple of frames in front of or behind the actual event. As a solution, instead of marking a single frame as the event frame, a window of frames around the event frame was therefore defined as all containing that event.

Manual testing showed a window of an additional three frames before and after an event to be a suitable window to capture the majority of all events. Figure 3.2 shows an example of video frames around an event and how the individual frames are classified. An event hence spans seven frames in total, except in cases where two events occur in quick succession. Here the event frame windows may overlap and are then combined into a larger event window.

3.2.1 Data Imbalance

Many machine learning algorithms, including neural networks, have difficulty working with imbalanced data. Because of the frequent inactive play phases, soccer events are spread sparsely throughout a game. On the complete video dataset, the event to frame ratio is about 1:44. The aforementioned methods for cleaning the data fortunately also help to solve the data imbalance. Cutting the video into clips reduces the number of frames by more than half and improves the ratio to about

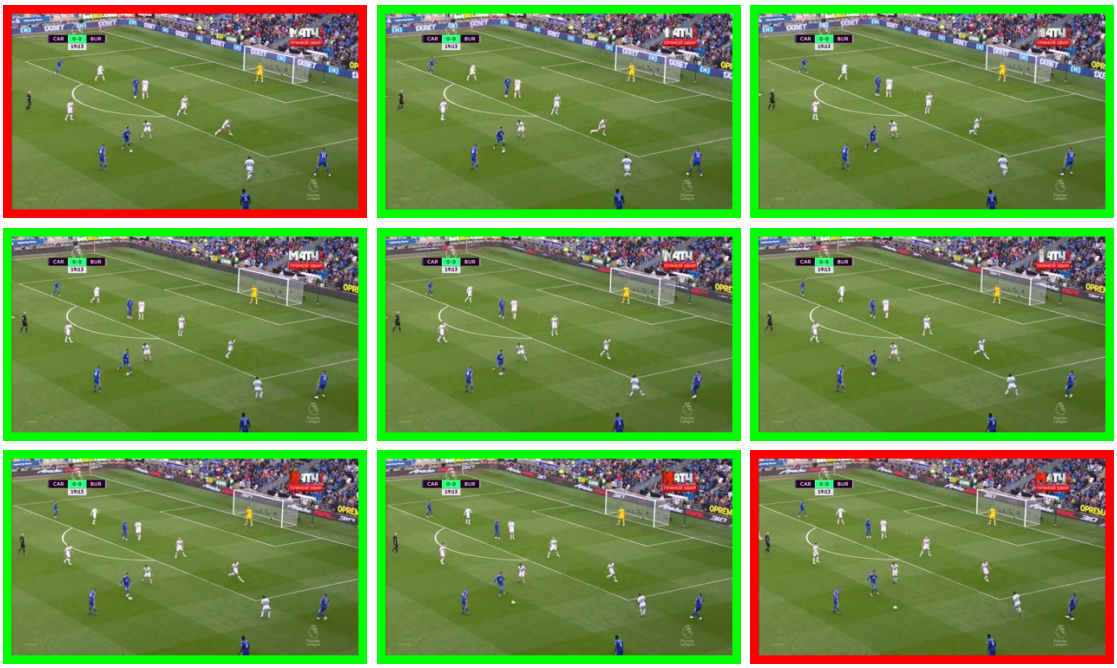


Figure 3.2: Example of an event frame window around a pass event. The frames progress from left to right and top to bottom. A green border denotes the frame is classified as an event. A red border signifies a frame is eventless.

1:19. The event frame window leads to a further improvement of about 1:5. Of all 7,144,079 frames, 1,439,854 are classified as events. This minor imbalance can be solved by augmenting the loss function. Details can be found in Section 4.4.

4 Methodology

To detect events in a video, a system must have a video processing pipeline capable of extracting features describing movement. In this thesis, a convolutional neural network is proposed as a backbone to extract spatial features from video frames. Next, a recurrent neural network sets the spatial features into a temporal context. The spatiotemporal features can then, in turn, be passed into a detector to determine the eventedness of frames. This section describes the architecture for the convolutional backbone, recurrent and detector networks. Figure 4.1 gives a visual overview of the entire model structure. Each design decision is discussed with the event detection task in mind. Next, several different image preprocessing techniques are presented. These aim to improve event detection by masking irrelevant content in a frame. The various loss functions used to cope with the data imbalance are described next. Finally, an overview of the hyperparameters and other training details for the range of experiments concludes this section.

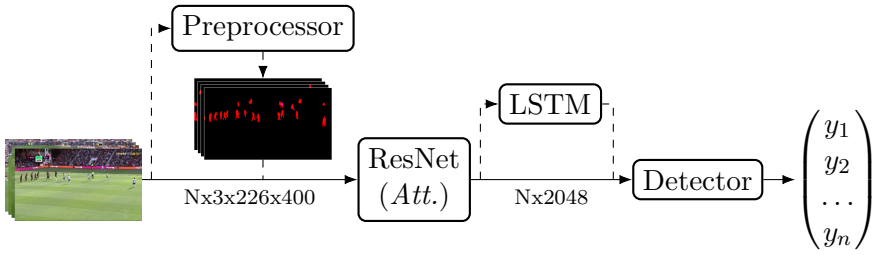


Figure 4.1: Graphical overview of the neural network architecture. Dashed lines show alternative paths which are experimentally evaluated for effectiveness. Furthermore, the ResNet is tested with and without attention augmentation.

4.1 Frame Processing

As mentioned before, three main architectures have been used in neural network video processing. The first is to expand 2D convolutions to three dimensions, jointly learning spatial and temporal features in a single network. Secondly, spatial and temporal features are learned separately by two different convolutional networks and combined. Lastly, spatial features are passed into a recurrent network, which can output temporal features by having access to past frames in its memory.

Even though the former two frameworks have produced better results on video classification tasks [7], the main drawbacks are that a video, or at least the relevant parts, must be passed to the model in one piece. Detrimentally, for the task of detecting soccer events, these methods cannot be applied to lengthy videos. Memory constraints prevent passing long videos into a model. An argument can be made that a video can be split into shorter sequences and processed separately. This raises the question of where best to cut the videos and how to smoothly combine the output. In conclusion, using a recurrent neural network on spatial features is the most natural approach to processing long videos and is hence the architecture used in this thesis. For spatial feature extraction, a ResNet-50 architecture is used. Spatial features are then passed into an LSTM network to add temporal context. Finally, a simple fully connected network predicts a scalar eventedness value of a frame.

4.1.1 ResNet

Previous work has shown that convolutional neural networks are effective at extracting spatial features from images [8, 10, 64]. Specifically in this thesis, a ResNet is used. ResNets are especially well-known for their great performance while also being lightweight. They were first introduced in 2016 by He et al. [21], who added skip connections to bypass blocks of convolutional layers. The idea being for information to flow past a block to combat the vanishing gradient problem. Instead of learning a mapping $\mathcal{F}$ from input to target output $\mathcal{F}(x) = \mathcal{H}(x)$, these blocks then learn the residual between the input and output $\mathcal{F}_R(x) = \mathcal{H}(x) - x$. This residual mapping earns the architecture its name. Residual layers enabled comparably simple training of extremely deep networks, increasing model capacity while keeping model complexity low. In the original paper, a range of networks varying in size were trained on the ImageNet dataset [13]. For this thesis, the ResNet-50 architecture, featuring 50 parameter layers, is chosen as a good middle ground between model complexity and efficiency.

Another concept that has advanced neural network image processing performance is transfer learning [40]. Commonly used in scenarios where data is extremely scarce, a model is first pretrained on a related but larger dataset. Afterward, the network can be fine-tuned on the target task.

This is especially applicable for image processing, as the learned convolutional kernels are often universally pertinent. Early convolutional layers often learn Gabor filters or edge detectors. The further in the convolutional hierarchy, the more complex and specialized kernels become. For example, the lowest level kernels may even react to specific textures or shapes [45]. These pretrained kernels, therefore, provide a value for virtually any image processing task.

Even when data is not scarce, fine-tuning off of a pretrained network provides

many benefits. Training time is cut and often a pretrained network can result in higher evaluation scores. Therefore, a ResNet pretrained on the COCO dataset is used. The COCO dataset is an extensive image object detection dataset [37]. It contains, among others, images with people and soccer balls. For this reason, it is especially well suited for soccer event detection.

Finally, the ResNet outputs a $y \in \mathbb{R}^{[2048\times8\times13]}$ spatial feature vector. The height and width dimensions are squashed by computing the average. The 2048 dimensional spatial feature vector is fed through a final fully connected layer of the same dimension. The final output vector is then passed on for further processing.

4.1.2 Attention Augmented Convolution

A recent promising addition to convolutional neural networks was adding self-attention to the convolutional kernel [6]. The strict locality of a convolutional layer prevents any long-range inferences between image features. This work was able to show that by adding visual attention, this constraint can be lifted. Attention augmented ResNets were shown to provide major improvements in object detection, elevating the performance of a ResNet-50 to the performance of a ResNet-101 architecture.

In soccer event detection, it is especially important to determine the spatial relations between players, the ball and the field. Is the ball at the foot of a player, high up in the air or out of play? To make these kinds of inferences, a model may greatly benefit from the capability of setting local features into a global context.

To augment a convolutional layer with self-attention, some of the convolutional channels are replaced by a two-dimensional version of the dot product attention popularized by the Transformer architecture [60]. Within the original publication, the effects of replacing all channels with attention augmented convolutions were tested. However, it was found that a half and half split was more effective. To accommodate image data, the positional encoding, used to determine the relative position of inputs to one another, needs to be expanded to two dimensions. This is solved by adding a fully connected layer to learn embeddings for relative width and height differences between pixels. The output A of a visual attention head is therefore defined as:

$$W = \text{Softmax}\left(\frac{QK^T + S_H^{rel} + S_W^{rel}}{\sqrt{d_k}}\right)$$
$$A = WV$$

$Q \in \mathbb{R}^{[d_q\times H\cdot W]}$, $K \in \mathbb{R}^{[d_k\times H\cdot W]}$ and $V \in \mathbb{R}^{[d_v\times H\cdot W]}$ represent the query, key and value matrices. They are computed by passing half of the input image features

$X \in \mathbb{R}^{[C/2 \times H \times W]}$ through a convolutional layer to obtain attention features $Y \in \mathbb{R}^{[d_q+d_k+d_v \times H \times W]}$. These are then flattened along the height and width dimension and split for each of the matrices. $W \in (0,1)^{[H \cdot W \times H \cdot W]}$ is the attentional weight matrix, dictating how much a pixel $v_{ci} \in V$ attends to another pixel $v_{cj} \in V$.

Up to this point, the definition is equivalent to the dot product attention defined for Transformers. However, S_H^{rel} and S_W^{rel} are pixel-wise relative position weight matrices, giving the attention weights W additional 2D spatial context. Note that the batch dimension, as well as the splitting of attention over multiple heads, have been omitted for simplicity. The attention output is then reshaped back into a 2D pixel representation and passed through a final 1×1 convolutional layer. It can then be concatenated with the output from the convolutional layer that was applied to the other half of the input channels.

For the dimensions, d_q, d_k and d_v of the query, key and value matrices, the recommendations of the original paper are adopted. The same holds for the number of attention heads n_h. d_q and d_k are set to double the number of input channels to the attention layer. For example, the final bottleneck block of the ResNet-50 architecture receives a 1024 channel input. As only half of the channels are passed to the attention layer, the dimensions are therefore set to $d_q = d_k = 1024$. d_v determines the number of output channels of the attention layer. Again, since half of the output should be attention augmented and the final block has a 512 channel output, d_v is set to 256. Finally, n_h is set to 8.

GPU memory constraints prevent applying attention augmentation to all ResNet blocks. Attention augmentation adds an additional $O(HWd_k^h)$ memory cost on top of a plain ResNet. This increased memory cost is mainly due to the pairwise multiplication of the flattened pixel query Q and key K matrices. As a consequence, doubling the size of an image increases the memory footprint by a factor of four. Fortunately, the lower in the convolutional hierarchy, the smaller a processed image becomes. In consequence, the memory footprint also decreases and attention augmentation applied to lower-level blocks requires less GPU memory.

This memory constraint requires making a trade-off between batch size and the number of layers attention could be applied to. In the end, attention augmentation is only applied to the final layer of the ResNet. Additionally, no pretrained attention augmented ResNet models could be found and time constraints prevented pretraining a model for this thesis. To take advantage of transfer learning to the fullest, all blocks preceding the attention block use pretrained weights. The attention block and the output layer are trained from scratch.

4.1.3 Recurrence

After spatial feature extraction, an LSTM network is used to add temporal context. LSTM networks are a type of recurrent neural network in which information from

past inputs is saved in an internal memory cell, also called cell state [23]. Three input dependent gates augment this cell state. For each new input sample, the input is first combined with the output from the previous step (called hidden state). This expanded input vector is fed into each of the gates to update the internal cell state. The first gate, named forget gate, decides how much of the cell state should be forgotten. Next, an update or input gate determines how much of the new input information is transferred to the cell state. Finally, an output gate is used to combine the cell state with the expanded input vector to obtain the output vector. This output vector then also acts as the hidden state for the next time step. For a more concise visual overview, see Figure 4.2.

Normally, an entire sequence is given to an LSTM at once and the cell and hidden state can be reused for each time step. As mentioned in Section 3.1, this is not feasible for the length of videos used in this thesis. Instead, a single input batch consists of a set of frames from multiple different videos. To accumulate an informative cell and hidden state, on receiving the spatial features $x_i^{\langle t \rangle}$ from the t'th frame of video number i, the LSTM also has to receive the cell state $c_i^{\langle t-1 \rangle}$ and hidden

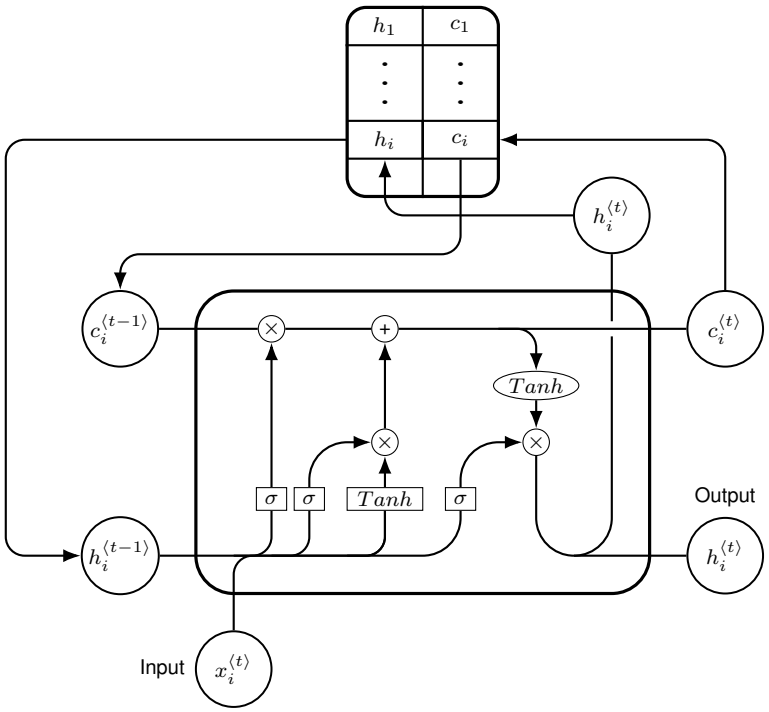


Figure 4.2: Structure of an LSTM cell. Large parts of the graphic were taken from [59]. Cell and hidden states are saved in a separate buffer for each video. When the end of a clip is reached, the cell and hidden state are reset to random vectors for that video.

state $h_i^{\langle t-1 \rangle}$ from the previous frame for that particular video. To accommodate this, the LSTM cell is expanded to include a buffer of cell and hidden states. The buffer has a separate slot for both the cell and hidden state for every video. As soon as a video decoder reaches the end of a video clip, the buffered states for that video are randomly reset and spatiotemporal information needs to be reaccumulated for the new clip.

In summary, the capability of LSTMs to retain information over multiple timesteps is used to add temporal context to previously extracted spatial features. By interpolating over the change of similar spatial features, general spatiotemporal movement features can be extracted. In essence, a non-recurrent network is only able to make event predictions based on a single image, while a recurrent network can utilize movement information across multiple images.

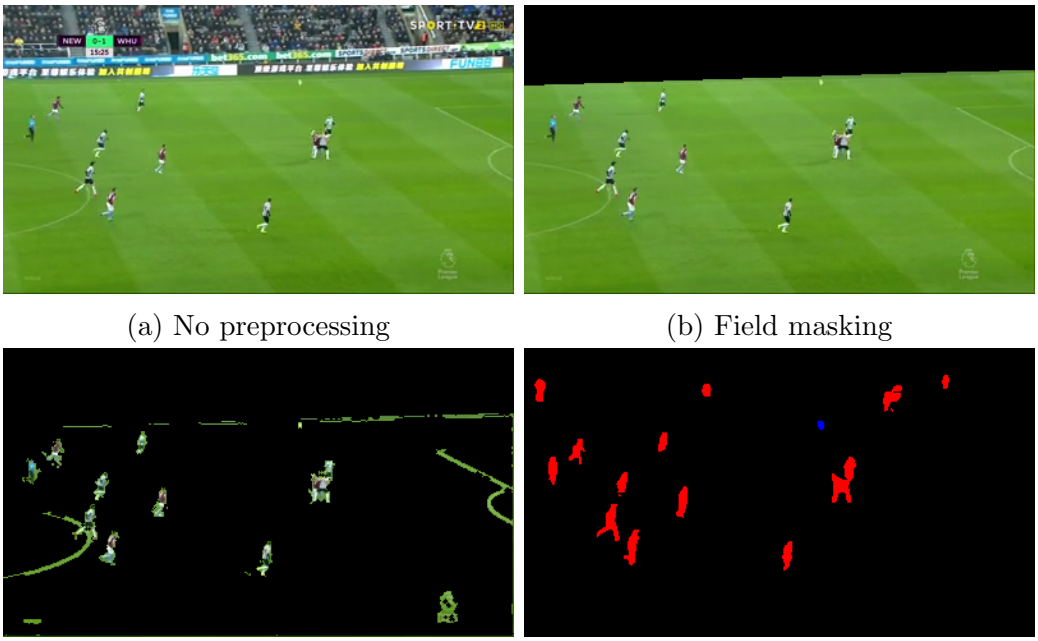
4.2 Detector

As long as the spatiotemporal features extracted from the video processing networks are adequate, a comparably simple network architecture suffices to detect events. Subsequently, the detector is made up of a fully connected three-layer network with ReLu activation functions. For regularization purposes, each layer is followed by batch normalization and a dropout layer with $p = 0.25$. Each layer has the same number of hidden units as the input size of 2048. A final output layer then maps the feature vector to a single scalar value.

4.3 Preprocessing

In an attempt to aid the neural network in extracting relevant image features, different types and levels of preprocessing are applied to a frame. The goal is to mask out irrelevant parts of the image and highlight important objects like the ball and players. In the first step, the stands and fans can be masked out. In a typical soccer video frame, the field will take up most of the image. Using a hue histogram, the color of the field is obtained and a contour detection algorithm determines the field edges. Anything outside of the field is masked. Figure 4.3b shows an example with the field extracted. An edge detection algorithm can then be used on the field masked image to detect the players, ball and lines. An example of the player masking is given in Figure 4.3c.

Because the algorithmic field and player masking are prone to fail for different lighting conditions, camera angles or close up shots, a third preprocessing method is tried. Mask-RCNNs are a state of the art method for image object segmentation [22]. A Mask-RCNN pretrained on the COCO dataset is used to extract all objects



(a) No preprocessing (b) Field masking

(c) Player masking (d) Mask-RCNN

Figure 4.3: An example of the different types of preprocessing applied to soccer video frames to attempt to improve event detection.

within a frame. Any detected people and sports balls with a confidence greater than 50% are then pasted into a black image using the computed masks. The players are pasted in red and the ball in blue. See Figure 4.3d for an example. Even though this method is not affected by the camera angle, the missing color information of the field and stands may harm event detection performance. Furthermore, the Mask-RCNN preprocessing method also occasionally produces unwanted artifacts. Round objects are occasionally falsely detected as a sports ball. Colored blobs at the edge of the pitch or in the stands sporadically are mistakenly taken to be people. Evaluating the preprocessing methods on longer video sequences by hand showed only a small amount of mistakes on the large majority of frames though. An example where each preprocessing method made a mistake is given in Figure 4.4.

Since the field masking is the least intrusive, it is hypothesized to only lead to minor differences in performance. Player masking, on the other hand, masks out large portions of the image and should produce larger performance differences. Lastly, the image produced by the Mask-RCCN masking is thought to improve performance especially in cases where the ball and players can be clearly detected. Since all methods are error-prone in several situations though, a general decrease in performance is also plausible.

Figure 4.4: Examples where the preprocessing methods fail to capture essential details of the image. In the first case, a close-up shot of the field disturbs the color detection for the field masking. In the second, lighting conditions create many shadows leading to many detected edges. Lastly, motion blur tricks the Mask-RCCN to detect multiple soccer balls within an image.

4.4 Loss Function

Two main solutions exist to combat data imbalance when training neural networks [14][62]. Different sampling techniques can be used to artificially inflate the size of the minority or deflate the size of the majority set [15]. However, because the video frames need to be sampled sequentially, specialized sampling techniques cannot be applied. Alternatively, the loss function can be adapted to be sensitive to the data imbalance [41].

In this thesis, three different loss functions are considered and tested experimentally for effectiveness: weighted binary cross entropy (wBCE), F1 loss and Matthew's correlation coefficient (MCC) loss. The three loss functions were selected with several different hypotheses in mind. First, it is hypothesized that models trained using the F1 and MCC loss will score better in data imbalance sensitive evaluations scores. Furthermore, as they are specifically designed to minimize their respective scores, the F1 loss trained models are expected to produce the best F1 scores. Identical behavior is expected for the MCC loss. Furthermore, the F1 loss is expected to produce models with a greater bias for positive classification compared to the MCC loss. As the MCC weights positive and negative samples equally, it should achieve a better balance between classification recall and specificity. The following sections present the three loss functions in detail.

4.4.1 Weighted Binary Cross Entropy

The binary cross entropy function (Equation 4.1) has a simple extension to handle imbalanced data. By adding a weight w to the positive samples, incorrect classifications of the positive class can be penalized or incentivized. By scaling the weight by the degree of imbalance, the wBCE (Equation 4.2) can be artificially biased to eliminate the imbalance. For this, the weight should be equal to the ratio of positive to negative samples $w = \frac{N_P}{N_N}$. In other words, the loss then behaves as if the dataset had an equal amount of samples for both classes [2].

$$BCE(o,t) = t \cdot log(o) + (1-t) \cdot log(1-o) \tag{4.1}$$
$$wBCE(o,t) = -w(t \cdot log(o) + (1-t) \cdot log(1-o)) \tag{4.2}$$

4.4.2 F1 Loss & MCC Loss

The straightforward method for evaluating a binary classifier is to compute the precision. Precision only regards positively classified samples though. This especially is a problem when working with imbalanced data, since always predicting the majority class scores a high precision, while the minority class is usually the class of interest. For these cases, the F1 score and the Matthews correlation coefficient (MCC) provide a solution [43]. The F1 score (Equation 4.5) is computed as the harmonic mean between precision (Equation 4.3) and recall (Equation 4.4):

$$P = \frac{TP}{TP + FP} \tag{4.3}$$
$$R = \frac{TP}{TP + FN} \tag{4.4}$$
$$F1 = \frac{2 \cdot P \cdot R}{P + R} = \frac{2 \cdot TP}{2 \cdot TP + FN + FP} \tag{4.5}$$

where TP, FP and FN are the number of true positive, false positive and false negative samples respectively. Thereby, the F1 score also takes false negatives, i.e. incorrectly marked negative samples, into consideration. It ranges between 0 and 1, where 1 is a perfect classification and a score of 0 is achieved, when no positive samples are classified correctly.

A major drawback of the F1 score is its bias for positive samples. It completely ignores true negative samples. This is especially apparent for two datasets with different ratios of positive samples. Assume a random classifier scores a precision $P_A = \frac{TP_A}{TP_A + FP_A}$ and recall $R_A = \frac{TP_A}{TP_A + FN_A}$ on a dataset A. Data set B is an extension to A where $N_B > 0$ negative samples are added. As no positive samples are added,

$TP_B = TP_A$, $FP_B = FP_A$ and as a consequence $P_B = P_A$, i.e. the precision stays the same. The false negative count, on the other hand, is very likely to change $FN_B = FN_A + FN_{B \setminus A}$. Therefore, $R_B \leq R_A$ and finally $F_{1_B} \leq F_{1_A}$. In summary, a smaller ratio of positive to negative samples automatically leads to a lower F1 score for uninformed classifiers. This can make it difficult to compare the F1 score across datasets and disqualifies it as a measure where both negative and positive classifications are important. The MCC is an alternative and improved measure which eliminates this bias for positive samples [39]. It as defined as:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

where TN is the number of true negative samples. The MCC ranges between -1 and 1. An MCC of 1 denotes a perfect classification, -1 a perfect inverse classification and 0 a completely random classification.

Computing the gradient of both functions is not possible as they use the output label of a classifier to compute the respective scores. To convert them to compatible loss functions, they need to be converted to their soft variants. Instead of using the classification label, the classification probability can be used for the true/false positive/negative scores [41]:

$$TP_{soft} = \sum_{i=0}^{n} o_i \cdot t_i \qquad TN_{soft} = \sum_{i=0}^{n} (1 - o_i) \cdot (1 - t_i)$$

$$FP_{soft} = \sum_{i=0}^{n} o_i \cdot (1 - t_i) \qquad FN_{soft} = \sum_{i=0}^{n} (1 - o_i) \cdot t_i$$

$$o_i = \sigma(h(x_i)) = \frac{1}{1 + e^{-h(x_i)}}$$

where o_i is the output of a classifier h for input x_i. t_i is the target label and σ is the sigmoid function. $F1_{soft}$ (Equation 4.6) and MCC_{soft} (Equation 4.7) are then identical to their normal variants, except that all true/false positive/negative score are substituted by their soft variants.

$$F1_{soft} = \frac{2 \cdot TP_{soft}}{\sum_{i=0}^{n} o_i + t_i} \tag{4.6}$$

$$MCC_{soft} = \frac{TP_{soft} \cdot TN_{soft} - FP_{soft} \cdot FN_{soft}}{\sqrt{\sum_{i=0}^{n} o_i \cdot \sum_{i=0}^{n} t_i \cdot \sum_{i=0}^{n} (1 - o_i) \cdot \sum_{i=0}^{n} (1 - t_i)}} \tag{4.7}$$

Furthermore, the loss functions need to be minimized and therefore the values need to inverted. The F1 score ranges from 0 to 1, so the score only needs to be

flipped. The MCC ranges between -1 and 1, so its range needs to be flipped and the minimum value brought to zero. In the end, the F1 (Equation 4.8) and MCC (Equation 4.9) losses are defined as:

$$L_{F1} = 1 - F1_{soft} \tag{4.8}$$
$$L_{MCC} = -1 \cdot MCC_{soft} + 1 \tag{4.9}$$

It is important to note that unlike most loss functions, the F1 loss and MCC loss are not batch independent. The loss for a sample depends on the other samples within a batch because the false/true positive/negative scores are computed over all samples within a batch. Furthermore, if there are no positive samples within a batch, the soft F1 score automatically becomes zero. This leads to a loss of 1, even though the model had no influence. The derivative of the F1 loss (Equation 4.10) shows however that if all t_i are equal to 0, the derivative is also 0. In turn, the weights do not receive updates for that batch.

$$\frac{\partial L_{F1}}{\partial o_j} = \frac{-2t_j}{\sum_{i=0}^{n} o_i + t_i} + \frac{2\sum_{i=0}^{n} o_i t_i}{\left[\sum_{i=0}^{n} o_i + t_i\right]^2} \tag{4.10}$$

The same properties do not hold for the MCC loss. If a batch contains only positive or negative samples the denominator of the soft MCC is equal to 0 and MCC loss is undefined. The loss is set to zero for these invalid batches, mimicking the behavior of the F1 loss. In consequence, it is important to choose an adequately large batch size. Not only does a larger batch size give a better approximation of the true MCC and F1 score, but it also lowers the probability of obtaining invalid batches. Under the assumption that the batch contains at least one negative and positive sample, as well as at least one prediction probability $o_i > 0$, the derivative of the MCC loss is given in Equation 4.11. Both derivatives nicely show the batch intra-dependence, since the sums over all samples in a batch do not cancel out.

$$\begin{aligned} \frac{\partial L_{MCC}}{\partial o_j} &= \frac{-2 \cdot \sqrt{n_1} \cdot n_2}{\sum_{i=0}^{n} 2o_i - 1} \\ n_1 &= \sum_{i=0}^{n} o_i^3 \sum_{i=0}^{n} (1-o_i)^3 \sum_{i=0}^{n} t_i \sum_{i=0}^{n} (1-t_i) \\ n_2 &= 2o_j t_j + \sum_{i=0}^{n} (1-o_i)t_i - o_i t_i \end{aligned} \tag{4.11}$$

4.4.3 Experiment Details & Hyperparameters

Three successive experiment groups are tested for this thesis. First, a plain ResNet model without attention or recurrence is trained for each of the different preprocessing options using wBCE. The loss functions are tested next, again training a static ResNet without attention augmentation for each function separately. Based on the two prior experiments, exhaustive combinations of networks with and without recurrence and attention are trained. In one final experiment, the event classification capabilities of the model are tested. For this, the best architecture from the detector tests is expanded to output a probability for each event type. All networks are implemented using pytorch [42] and experiments run and logged using pytorch-lightning [17]. Training is done on a single NVIDIA Geforce GTX 1080Ti.

All training details and hyperparameters are identical across all experiments except for the batch size. Batch sizes are chosen such that GPU memory utilization is maxed out for the experiment with the highest memory requirement in an experiment group. For the preprocessing experiments, the Mask-RCNN requires the most memory and the batch size is set to 16. For the network architecture experiments the batch size can be doubled to 32. The loss experiments also use a batch size of 32 for comparability. Otherwise, the learning rate is initialized to 0.00005 and weight updates are done using the ADAM optimizer [31]. All networks are trained until the validation loss (F1 loss) does not improve for three consecutive epochs. The model with the best validation score is saved and used for further evaluation. Because the validation loss converges rather quickly, only 2% of the training data is used per epoch.

5 Evaluation

This section gives a detailed report of the results obtained from the different evaluations. In order, the performance of the preprocessing techniques, data imbalance sensitive loss functions and different model architectures are investigated. The model architecture evaluations include a closer look at the convolutional and attentional feature maps, as well as the effect of recurrence on the detection of events early in a video clip. Next, the final model is evaluated on the full video dataset. A summary of observations when viewing examples of successfully and unsuccessfully detected events is reported next. Finally, the model is expanded to enable event classification. The evaluation scores per event id provide some interesting insights into the capabilities of detecting different event types.

5.1 Model Comparisons

As part of the network architecture decision process, multiple configurations are tested and evaluated sequentially. Starting from a non-recurrent ResNet-50 trained using wBCE, the prevailing network architecture from one test is carried over to the next. First, different preprocessing techniques are tested. Next, models are trained using different data imbalance sensitive loss functions. Finally, attention and recurrence are added. Since no baseline comparison models exist, all models are also compared against a random baseline. Events are spread randomly across all frames of the clip dataset, based on the ratios of the different event types. Applying the event frame window to the random events then grants a semi-informed random and comparable baseline prediction.

A range of scores is used to evaluate the different model architectures and loss functions. These can be split into two categories; based on frames and based on events. The per-frame scores consider every video frame independently. This creates a binary classification task to detect if a frame is an event or not. As evaluation scores, the accuracy, F1 score and MCC are computed. In connection with the F1 score, the accuracy provides a gauge of how well a model is able to learn the distribution and ratio of events. A classifier that tags every frame negatively would achieve an accuracy of 65%, but an F1 score of 0. Any classifier achieving an accuracy lower than 65% and a positive F1 score is classifying a disproportionate number of frames as events. Thereby choosing to detect more event frames correctly as a

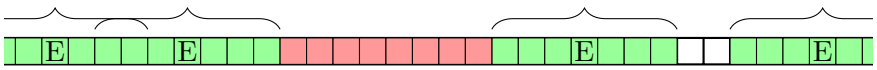


Figure 5.1: Figure showing examples of positive and negative evaluation event frame windows. An example sequence of frames with four events (E) and corresponding individual event frame windows (brackets) is shown. A positive evaluation event frame window (green) can consist of multiple overlapping individual event frame windows. Negative evaluation event frame windows (red) are any event free frame sequences at least seven frames long.

trade-off for detecting more frames without an event incorrectly. The MCC lastly provides a balanced value about a model's detection performance on both positive and negative samples simultaneously.

The per-event scores, on the other hand, group several frames together. To determine if an event is detected correctly, a fraction of t frames of an event frame window need to be classified as events. That is, if $t = 1$ all events from an event frame window need to be classified as an event, for $t = 0.5$ only half, etc. When multiple individual event frame windows overlap, they are fused into one event frame window. As a result, an event frame window may encompass more than the original seven frames. To obtain negative samples, the chains of frames between two event

		Per Frame			Per Event		
Experiment	Model	Acc	F1	MCC	Rec	Spec	μ_H
Baseline	Random	55.78	32.28	-00.34	24.01	52.30	32.91
Preprocessing	None	**51.77**	53.90	**17.80**	77.86	**21.64**	**33.87**
	Mask Field	46.88	**54.19**	17.03	90.01	11.02	19.64
	Mask Players	41.56	52.11	09.02	**93.00**	02.77	05.37
	Mask-RCNN	46.78	54.10	16.74	92.24	06.82	12.70
Loss	wBCE	54.10	55.15	21.12	**77.54**	25.90	38.83
	F1	59.90	**56.43**	**25.66**	68.65	41.60	51.80
	MCC	**60.96**	55.11	24.41	60.87	**47.04**	**53.07**
Architecture	Static	59.90	56.43	25.66	68.65	**41.60**	51.80
	Static Att.	56.58	56.29	24.13	76.06	32.21	45.25
	LSTM	**61.41**	**58.28**	**29.20**	72.69	40.92	**52.36**
	LSTM Att.	55.39	56.70	24.63	**82.31**	25.31	38.71

Table 5.1: Comparison table for different model architectures and experiments. All values given in percent. For each experiment the accuracy, F1 score and MCC are reported on a per frame basis. For per event metrics, frames are grouped together based on event frame windows and recall and specificity was computed. The harmonic mean μ_H of both recall and specificity is also reported.

frame windows are used. Any set of consecutive non-event frames, with a length greater or equal to the size of a minimal event frame window, i.e. seven frames, are considered as negative event sequences. Figure 5.1 provides a visual description of the positive and negative evaluation event frame sequences.

In the following evaluations, the recall and specificity are computed for $t = 0.7$. Additionally, alike to the usage of the harmonic mean in the F1 score, the harmonic mean over recall and specificity is also reported. A good model should be able to predict both positive and negative event frame sequences well. By weighting recall and specificity scores equally, but penalizing extremely low values, the harmonic mean gives a good gauge over how well a model can detect both event types. Both the per frame and per event values are recorded in Table 5.1.

5.1.1 Preprocessing

In total, three different pre-processing methods are tested: masking everything but the field, masking everything but field markings and players and using a Mask-RCNN to extract a channel-wise binary player and ball image. See Section 4.3 for more detailed information.

The evaluation scores clearly show that none of the preprocessing methods have any positive effect (Table 5.2). In per frame evaluation, the model trained on the raw frames scores better than all other models in precision and MCC. Only the field masking and Mask-RCNN preprocessing are able to score marginally better in the F1 score. Both also scoring fairly similarly and only slightly worse than the no preprocessing model. The player masking in contrast, scores worse than all other preprocessing methods in all three per frame evaluation scores. With an MCC score that is almost 50% lower than than the no preprocessing model, both event and non-event frame detection are shown to suffer from this preprocessing method. A closer look at the accuracy of 41.56% and F1 score of 52.11% shows that a model trained with player masking probably classifies an increased number of frames as positive.

This claim is supported by the per event evaluation scores. The model trained on player masking is able to achieve the highest recall at 93%, but also the lowest specificity at 2.77%. This also results in the worst harmonic mean of any model. Subsequently, a bias for positive classification can be deduced. With varying degrees of severity, similar behavior can be seen for all other preprocessing methods. The positive classification bias incrementally drops from the Mask-RCNN over the field masking to the model without any preprocessing.

This is especially interesting, because of the similar frame evaluation scores of the Mask-RCNN and field masking models. A similar score in one evaluation category is expected to equate to similar scores in the other category. This may be caused by a higher variance of predicted events for the Mask-RCNN model. That is, the ratio

of positive labels between the two models is the same, but the model trained with field masking clusters positive labels closer together. The Mask-RCNN model, on the other hand, has a more uniform distribution. However, a further investigation of this issue goes beyond the scope of this thesis.

In summary, none of the preprocessing methods are able to improve event frame detection. The results match expectations in so far that the field masking and no preprocessing models did achieve the most similar scores. In per frame detection, all models improve on the random baseline with an increased MCC. In event-based detection, only the model with no preprocessing is able to score a higher harmonic mean between recall and specificity compared to the random baseline. Albeit, the models are not specifically trained for this task. Ultimately, this result leads to the abandonment of the preprocessing techniques for all future model architectures.

	Per Frame			Per Event		
Model	Acc	F1	MCC	Rec	Spec	μ_H
Random	55.78	32.28	-00.34	24.01	52.30	32.91
No Preproc.	**51.77**	53.90	**17.80**	77.86	**21.64**	**33.87**
Mask Field	46.88	**54.19**	17.03	90.01	11.02	19.64
Mask Players	41.56	52.11	09.02	**93.00**	02.77	05.37
Mask-RCNN	46.78	54.10	16.74	92.24	06.82	12.70

Table 5.2: Comparison table for the preprocessing experiment of per frame and per event evaluation scores (%). See Table 5.1 for an overview of the evaluation scores across all experiments.

5.1.2 Loss

Again, three different variants are tested for the data imbalance loss functions. Next to the commonly used wBCE, losses directly optimizing for the F1 score and MCC are also tested. Section 4.4 provides further details on the specific loss functions.

The evaluated scores show three particularly interesting results (Table 5.3). First, wBCE is the worst loss function for this particular task and network architecture by a significant margin. It features the worst scores in every category except for event recall, which is counteracted by a significantly worse event specificity and subsequent harmonic mean. Like the player masking preprocessing, with a characteristic low accuracy, moderate F1 score and high event detection recall, the wBCE trained model exhibits a positive classification bias in comparison to the other models.

Second, even though the MCC loss should optimize the MCC score, the model using the F1 loss outperforms the MCC loss model in both the F1 score and MCC.

In terms of accuracy, the MCC trained model scores slightly higher though. Third, despite an overall better performance in per frame evaluation, the F1 loss model performs worse than the MCC model when detecting event sequences. The MCC model features a lower recall but higher specificity, with a smaller difference between both. This improved balance leads to a better harmonic mean.

The event evaluation scores therefore mostly coincide with the initial hypotheses. The performance difference between wBCE and the other two losses is larger than expected. Also, in the per-frame evaluation scores, the higher MCC of the F1 loss model raises questions. No plausible explanation could be found for this effect and exploration of the exact origins is again left for further research.

	Per Frame			Per Event		
Model	Acc	F1	MCC	Rec	Spec	μ_H
Random	55.78	32.28	-00.34	24.01	52.30	32.91
wBCE	54.10	55.15	21.12	**77.54**	25.90	38.83
F1	59.90	**56.43**	**25.66**	68.65	41.60	51.80
MCC	**60.96**	55.11	24.41	60.87	**47.04**	**53.07**

Table 5.3: Comparison table for the loss function experiment of per frame and per event evaluation scores (%). See Table 5.1 for an overview of the evaluation scores across all experiments.

One further important note to make is the performance increase of the wBCE trained model compared to the model without preprocessing from the preprocessing tests. The only difference between both models is the batch size used in training was doubled for the loss experiment model. A deeper investigation into optimal hyperparameters is therefore also an open issue. Finally, it was decided to use the F1 loss during all further experiments. This was mainly due to the higher MCC score, making it the objectively better detector on a per-frame basis.

5.1.3 Model Architecture

In the final experiment, a plain ResNet-50 model is compared with models that integrate attention augmentation, insert a recurrent LSTM layer and a combination of both. The most prominent result from the architecture evaluations is the lower performance across the board of the two attention augmented models compared to the models without attention augmentation (Table 5.4). Adding attention to the model decreases performance in almost all categories. Again, the same behavior as in previous suboptimal experiments can be found: lower accuracy, with moderate F1 score and higher event recall. An attention augmented model, therefore, classifies a higher ratio of frames as events to compensate for its uninformed decision. Because

this performance loss applied to both the recurrent and non-recurrent model, it can be inferred that models with attention augmentation are unable to learn adequate spatial features.

In contrast to this, recurrence did lead to conclusive performance improvements. In all per frame evaluation scores, the recurrent model is able to best the plain ResNet-50 model. An approximately 2.5% performance increase in accuracy, a 3% increase in F1 score and 14% performance increase in MCC is achieved by including recurrence. However, this trend does not directly translate to the event-based evaluation scores. The recurrent model scores a higher recall, but lower specificity and therefore does not balance the two values as well as the non-recurrent model. Still, through a significantly better recall but only marginally worse specificity, the recurrent model is able to achieve a higher harmonic mean.

Compared to the random baseline, all models are able to improve in all major evaluation scores. Performance gains across all per frame evaluation scores and an especially high MCC show that the models learn to detect event frames beyond a random classification. Even though not specifically trained on the task, the models are also able to detect grouped event frames better than the semi-informed random classifier.

In conclusion, the initial hypotheses are partially confirmed. The result for attention augmentation is a major contradiction, especially considering that this extension has been shown to improve performance significantly for object detection tasks [6]. A closer look into why attention augmentation yields no performance improvements is given below.

However, expanding a network by recurrence does lead to improvements. Still, in event-based evaluation scores, in which recurrence is expected to add a large performance boost, it only achieves a slightly better harmonic mean than a model without recurrence. This may be due to lower performance on events early in a video clip. A recurrent network may need to process an initial set of frames to obtain an

	Per Frame			Per Event		
Model	Acc	F1	MCC	Rec	Spec	μ_H
Random	55.78	32.28	-00.34	24.01	52.30	32.91
Static	59.90	56.43	25.66	68.65	**41.60**	51.80
Static Att.	56.58	56.29	24.13	76.06	32.21	45.25
LSTM	**61.41**	**58.28**	**29.20**	72.69	40.92	**52.36**
LSTM Att.	55.39	56.70	24.63	**82.31**	25.31	38.71

Table 5.4: Comparison table for the model architecture experiment of per frame and per event evaluation scores (%). See Table 5.1 for an overview of the evaluation scores across all experiments.

informative internal cell state. This hypothesis is tested and the results reported down below. As a result, the recurrent model without attention is chosen as the final model, based on its improved per frame and event evaluation scores.

Attention Weight & Feature Maps

To diagnose the behavior of the attention augmented convolution, the output of the final attention augmented convolutional block is inspected. Figure 5.2 shows 25 of the 256 feature maps of the convolutional and the attention convolution layer for two different frames. The feature maps are normalized between the range of 0 and 255 and then saved as grayscale images. On top of this, the weight map for each of the attention heads is shown. The weight maps are computed by summing over the first dimension of the attention weights W. They are also mapped to a grayscale image by normalizing the values to range between 0 and 255. This weight map represents the total relative influence a pixel had on all other pixels. In other words, it shows how important a pixel is for computing the output of the attention augmented convolution.

First, it makes sense to take a look at the attention weights w_{ij} for each of the frames. A well-trained attention mechanism is expected to produce diverse weights, while an uninformed attention mechanism would distribute attention weights uniformly across the pixels. A uniform weighting would weight each pixel by $\frac{1}{h \cdot w} = \frac{1}{8 \cdot 13} \approx 00.96\%$. The maximum weight for any pixel from both frames is only $w_{max} \approx 02.20\%$, with a minimum weight of $w_{min} \approx 00.31\%$. This shows that the attention augmented convolution exhibits an almost uniform weighting and does not attend to any pixels in particular.

Another interesting perspective is gained from examining the convolution and attention feature maps. At first glance, both seem to exhibit a varied structure, denoting a rich spatial feature space. On closer inspection, a comparison of the two attention feature maps shows that they are both nearly identical. Prominent pixels with high or low activation are equally distributed in both maps, only differing slightly in luminance. The same observation can be made on the weight attention weight maps. Here, the similarity is even more obvious at a first glance. For both frames, the general structure of the attention weight maps is identical, with only some areas slightly brighter than others. However, this property cannot be found in the convolution feature maps. The convolution feature maps exhibit an inherently different structure for the two different frames.

To quantify this similarity, the structural similarity index (SSIM) can be used [63]. It computes a metric for two grayscale images based on the luminance, contrast and structure. The higher the SSIM value, the more similar images are. The comparison of the convolution and attention feature maps result in an SSIM of 35.27% and 68.87% respectively. The combination of SSIM values and identical similarity

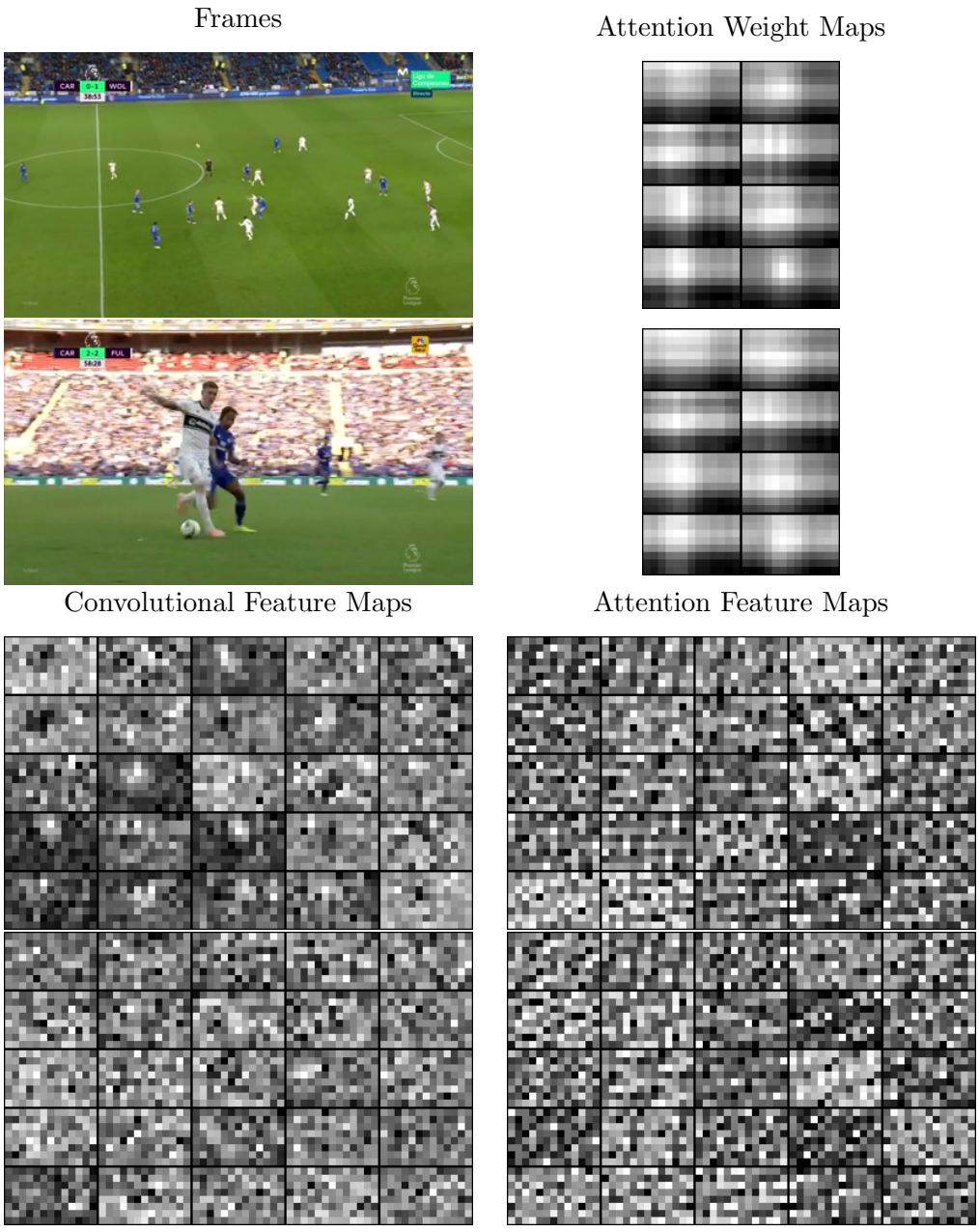


Figure 5.2: Attention and convolutional feature maps for two different frames. The feature maps are extracted from the last block of the attention augmented ResNet. The upper right image shows the weight contribution of each pixel from the attention value matrix.

observations on 100 frames from different videos, lead to a confident conclusion that the attention feature maps feature a constant structure irrespective of the input frame.

Finally, the range of activation of the attention feature maps is significantly lower than that of the convolution feature maps. The convolution features map activations range between -0.8201 and 0.8892, whereas the attention feature maps only range between -0.1763 and 0.1955 for the two different frames. This most likely leads to a greater impact of the convolution feature maps on subsequent layers and in turn event prediction.

In the end, these results show that the attention mechanism was unable to learn to attend to any pixels in particular. It outputs a mostly constant feature map, irrespective of the input frame, indicating that it provides little to no contribution to the image processing pipeline. In essence, it acts as a bottleneck by removing half of the feature channels, without providing any additional value. This hypothesis could be tested by removing the attention augmented channels entirely, but retaining the reduced channel size. However, this experiment is again left for future research.

Three different explanations for this behavior are considered to be plausible. Either, the attention augmented layers were just trained unsuccessfully. To obtain adequate weights, more training time, a different set of hyperparameters or pre-training on a more extensive and diverse dataset is necessary. Also possible is that the hypothesis of an additional benefit for detecting soccer events does not hold. Perhaps plain convolutional layers are better suited for this kind of task. Lastly, an incorrect implementation of the attention augmentation also cannot be ruled out.

Early Clip Events

A possible issue with a recurrent detection model could be the necessity to process an initial set of frames, to accumulate an informative internal cell state. This issue may reflect itself in the detection recall of events early within a video clip. To test this, all events within the first 15 frames of a video clip are considered as early events. Any other events are categorized as late. Table 5.5 summarizes the detection recall for a model with and without recurrence.

Because of the cell and hidden buffer resets on a new clip, the recurrent model may be aware of clip transitions. Measures were taken to ensure that event onsets after clip transitions are randomized. Still, the clip transition may have a priming effect on the recurrent model. This would explain a higher recall on early events for the recurrent model. Also, because the videos were split between longer eventless sequences, in most cases, the initial event in a clip is clearly separated from other events. This should make early clip events inherently easier to detect.

As both the recurrent and non-recurrent models exhibit similar improvement for early clip events, it can be assumed that early events are easier to detect. However,

at the same time, no conclusive evidence can be found to support the hypothesis that the recurrent model requires initial startup input.

Model	Early	Late	Total
Static	75.23	67.80	68.65
LSTM	79.21	71.84	72.69

Table 5.5: Recall (%) comparison tables of the recurrent and non recurrent model for detecting events at different time points.

5.2 Detector Evaluation

Following the initial tests to determine the model architecture, in this section, several tests are summarized which evaluate the behavior of the final model. To begin with, the performance of the model on an entire unseparated game is tested. Especially of interest are long non-active play sequences and the model's ability to detect these. An overview of observations on several correctly incorrectly detected events is given next. This is followed by a small section about inference speed of the model. Lastly, the model's capabilities of classifying events are analyzed.

5.2.1 Full Video Evaluation

On top of the evaluation of the clip dataset, the recurrent model is also evaluated on the full video dataset. The full video gives insights on a model's ability to detect inactive play phases. Inactive play phases are longer sequences in which no events occur, usually due to the ball leaving the area of play. These are frequently bridged either by replays of the past action or close-ups of the bench to provide interesting visual content for a viewer.

Replays are especially difficult to discern from normal play. Often, the only visual distinction is an overlay faded in shortly before a replay, to cue the viewer that the following scene is a replay. Moreover, the model may also struggle with close-up shots. The clip dataset mostly eliminates these and the model, therefore, has limited training experience. In summary, two possible scenarios were hypothesized. In any case, a slight degradation in evaluation scores was expected due to the difficulty of detecting replays as non-event sequences. In the first case, only a slight loss in performance occurs. Because the model is unable to detect interactions between a player and ball during close up filler shots, it doesn't detect these frames as events. A second plausible possibility is that the model outputs close to random probabilities and a major loss in evaluation performance can be recorded.

In general, the evaluation scores (Table 5.6) support the latter hypothesis. The per-frame accuracy, F1 and MCC scores drop considerably more than is to be expected. While the random baseline gains in accuracy, stays stable in MCC and drops by about 53% in the F1 score, the model performance drops by about 18%, 45% and 47% in accuracy, F1 score and MCC respectively. The accuracy shows that the model does not adapt to the more imbalanced ratio of event and non-event frames in the full video dataset. The random baseline predicts fewer event frames because of a higher event sparsity and in turn scores a higher accuracy. In contrast, the model predicts events with the same distribution on a full video as on a video clip. Too many events are predicted in non-active phases and the accuracy scores drop accordingly.

The same conclusions can be drawn from the F1 score. The random baseline adapts its prediction probabilities and predicts less positive events. As a consequence, both the precision and recall drop and the F1 score drops drastically. The model, in contrast, does not change the positive event distribution. As a result, the precision stays the same while the recall suffers from a higher amount of false negative samples. This leads to a less dramatic drop in the F1 score. In summary, the model classifies more non-event frames as events than the random baseline when transitioning from clips to full videos. The MCC further supports the hypothesis. It drops by almost half, showing that the model classifies with greatly increased randomness.

The per-event metrics paint a similar picture. The model scores worse in both recall and specificity and logically also in the harmonic mean. Surprisingly, the performance difference is a lot smaller than the difference in per frame evaluation scores would suggest. This is most likely due to the length of the non-event frame windows which were removed in the video clip dataset. Even though the number of frames more than doubled from 455,146 to 1,053,192 from the video clip to the full video test dataset, the number of positive and negative event frame sequences only increased from 21517 and 16129 to 22169 and 18048 respectively.

The sequences removed from the video dataset are most often very long frame

		Per Frame			Per Event		
Data Set	Model	Rec	F1	MCC	Rec	Spec	μ_H
Baseline	Clip	55.78	**32.28**	-00.34	**24.01**	52.30	**32.91**
	Full	**74.43**	15.00	**-00.04**	10.71	**80.08**	18.89
LSTM	Clip	**61.41**	**58.28**	**29.20**	**72.69**	**40.92**	**52.36**
	Full	50.62	31.84	15.56	69.66	40.69	51.37

Table 5.6: Comparison table of per frame and per event evaluation scores (%) from the full video and video clip dataset.

sequences without events, which are grouped for the event-based evaluation. Even though these sequences are often classified incorrectly, they only have a marginal impact on the event evaluation scores because of the event frame grouping.

5.2.2 Manual Event Detection Evaluation

Many interesting soccer event properties are not directly inferable from the event chains. These include properties like close-up shots, ball occlusion by players or speed of play. Therefore, it is necessary to manually examine some examples of correctly and incorrectly detected event and non-event frame sequences to analyze model behavior for these kinds of scenes. By computing the MCC score per video clip, event chains that are detected especially poorly or well can be investigated. Also of interest are long non-event frame sequences, which can be filtered out by taking the frames from the full video dataset not contained in the video clip dataset.

The successfully detected event chains show that the model is especially good at detecting clearly separated events. That is events where a player receives a ball and directly emits it again into another direction, without any opponent interference. This kind of behavior is often found when a team keeps possession of the ball through long pass chains. The model's predicted event frame window if often offset by one or two frames, but covers the bulk of the target event frame window.

In contrast to this, the model exhibits problems for chaotic play phases, where multiple players are close to the ball and the ball trajectory is fast and unpredictable. It also has difficulties predicting the event frame window when a stationary player is in possession of the ball, i.e. the ball is kept close at a players' feet but doesn't move. Lastly, the model also struggles with events where the ball is either occluded or difficult to detect. This is frequently the case when the ball is struck high and blends in with the stands or a player is positioned between the camera and the ball. In all of these cases, the model predicts nearly every frame to contain an event. This intuitively makes sense for chaotic play phases or stationary possessions, as it is extremely difficult to predict when a player will touch the ball and the F1 loss biases the model for the positive class. The occluded ball examples show this bias carries over to when the model receives familiar input images but has to make a completely uninformed decision.

Many examples of unfamiliar frame sequences can be found when taking a look at long non-event frame sequences. These often contain close up filler shots the model did not see during training. In many of these shots, no clear pattern for the event predictions can be found. This is especially prevalent for shots where no players or the ball are displayed, for example, if the bench or stands are highlighted. This supports the hypothesis from the full video evaluation that the model outputs random probabilities for unfamiliar input.

It should be noted that this deficit does not translate to all close-up shots though.

Often, if the action is on the far side of the field, a close-up shot is used to highlight the action. In most of these close up shots, the model shows no problems detecting the event frame window correctly. As a result, no conclusive evidence for any camera angle bias of the model could be found.

5.2.3 Speed

It is important to consider inference speed when evaluating the viability of a model for real-world applications. Different tasks set different requirements regarding inference speed. For example, online video processing will need to be able to run in real-time, so usually around 25 to 30 frames per second. Offline processing usually loosens the speed requirement and a model has more time to evaluate an entire video.

The proposed architecture in this thesis is extremely lightweight. It can process a single video at 116 frames per second on an NVIDIA Geforce GTX 1080Ti. Through parallelization, this can be further increased. Processing all 12 test videos in parallel pushes the frames per second to around 226. Depending on the scenario, this leaves lots of headroom for further developments and improvements, while still enabling real-time execution.

5.2.4 Classification

As a final test, the model's capability for classifying events is evaluated. To enable classification, the detector output layer is expanded to output a prediction value for each event type. Two different models are trained, both using the recurrent network architecture without attention. One model uses pretrained weights from the detection model and the other model is trained completely from scratch. Both models are compared to one another and a random baseline classifier. The random classification is generated identically to the way described in Section 5.1, except also expanded to the 17 event types. Table 5.7 reports all per frame evaluation scores. The number of frames tagged for each event type is also reported for context.

One clear trend is immediately evident. The models struggle to classify rare event types. That is, event types with comparably low frame counts also have, in general, low evaluation scores. This is to be expected for the F1 score. A higher ratio of positive samples automatically leads to higher precision and as a consequence a higher F1 score. The MCC, on the other hand, weights positive and negative samples identically and is therefore unaffected by the positive-negative ratio.

This effect may trace back to the F1 loss function used in training the classifier. If a batch contains no positive samples for an event type, the gradient of the F1 loss is equal to 0. The probability of such an invalid batch rises with the imbalance of positive and negative examples. For the extremely rare event types most

	F1			MCC			
Event Type	Pre	New	Base	Pre	New	Base	Count
Aerial Duel	03.33	**03.83**	00.66	03.24	**03.43**	00.00	2958
Clearance	**08.35**	06.81	01.24	**07.76**	05.72	00.15	4896
Completion	**24.82**	21.80	09.02	**18.39**	13.72	-00.11	42469
Contested Ball	**00.60**	00.46	00.15	**01.03**	00.47	00.01	765
Control	**05.11**	03.01	02.71	**04.17**	02.39	-00.06	12843
Deflection	**06.36**	05.11	01.60	**05.23**	03.88	00.48	5142
Goalkeeper Kick	**00.02**	00.01	00.00	**00.05**	-00.05	-00.01	42
Goalkeeper Throw	**02.27**	00.09	00.00	**05.32**	-00.28	-00.09	405
Interception	**03.16**	02.09	00.33	**04.17**	01.87	-00.15	2156
Kick-Off	00.59	**00.64**	00.00	**01.58**	01.25	-00.05	224
Pass	**27.34**	24.26	14.29	**11.75**	08.17	-00.02	67387
Recovery	**04.91**	04.44	01.55	**03.05**	02.51	00.00	7132
Set Piece	24.70	**25.63**	01.63	**25.61**	24.87	00.67	4564
Shield	01.00	**01.36**	00.50	01.20	**02.21**	00.24	1188
Shot	**07.41**	05.75	00.00	**09.19**	06.30	-00.40	1863
Tackle	03.30	**03.79**	00.97	02.46	**02.96**	00.13	3745
Take-On	**04.27**	03.72	01.23	**03.22**	02.65	00.29	4324

Table 5.7: Per frame F1 and MCC scores (%) for a model using pretrained weights (Pre), a model trained from scratch (New) and a random baseline (Base). The number of event frames per event class are given for context.

batches, therefore, do not lead to any updates of the respective output neuron. To obtain better output predictions, the batch size would need to be increased to obtain batches which are less likely to be invalid and also better approximate the true F1 loss on the entire dataset.

Another obvious effect is that transfer learning from the detection model leads to performance improvements. In most evaluation scores, the model using the pretrained weights scores considerably higher than the model trained from scratch. When the non-pretrained model has an advantage, it is usually in rare event categories where both models score poorly. Compared to a random classifier, both models can achieve significantly better scores on most event types. While the random classifier's MCC ranges between 0.47% and -0.60% for event types with more than 1000 frames, the scores for the pretrained model always lie above 1% with a maximum of 29.06%. The model is, therefore, able to learn some kind of representation for every event.

A couple of event types deserve a detailed look. For example, set pieces constitute a very interesting outlier. With only 4564 event frames, it is a fairly uncommon event but is still detected extremely well compared to other event types. This is especially

apparent when looking at the MCC score. The pretrained model has a 62% better MCC score for set-pieces than for any other event type. For the model trained from scratch, the difference is even more extreme at close to 100%. This is most likely because close to all set piece events (99%) are the first events of the video clip they appear in. Considering this, the improved set piece detection is consistent with the results of the early clip event evaluation.

Another interesting event type is the goalkeeper throw. It is an extremely rare event type, with only 405 total event frames in the video clip test dataset. However, the detection scores for the pretrained model for this event type are higher than many event types with significantly higher event frame counts. Because of the low sample size, this result has to be taken with a grain of salt. Nonetheless, the result is noteworthy because next to a throw-in, the goalkeeper throw is the only event type executed with the hands. This result may suggest the model can discriminate between which body part is used in which event.

Lastly, shots are again detected comparatively well considering the frame event counts. The unique properties separating shots from other events are on the one hand the location and on the other, the direction the ball is played. That is the location is in the opponent's box and the direction is towards the opponent's goal. As a consequence, the increased detection scores imply that the model has not only learned to detect player and ball relations, but also the position of the ball relative to the field and especially the goal.

6 Discussion & Future Outlook

In summary, the evaluations of the different models give rise to several conclusions and also further questions. The progressive model development yielded some insightful results. In detail, the pre-processing methods showed that it was best to leave image feature extraction to the convolutional neural network. It is exceedingly difficult to create handcrafted methods that succeed at removing all irrelevant and keeping all relevant information. Before important information is lost, it is better to pass the entire image to the network and let it learn to extract the relevant features itself.

The data imbalance sensitive loss functions showed that major improvements can be achieved by attending to imbalanced data correctly. Both the F1 and MCC loss were able to improve on the commonly used wBCE, making them both viable alternatives for imbalanced data. However, the criticism voiced against the F1 score also translates to the F1 loss. It has an inherent bias for the positive class, which reflects itself in high positive classification ratios when a classifier is uninformed. For tasks in which correct negative classifications are also of importance, the MCC loss provides a more balanced metric. A downside of both loss functions is the intra-batch dependency. When the data imbalance becomes too extreme or batch sizes are too small, the intra-batch dependency can lead to invalid batches. If a batch contains no positive or negative samples, the F1 and MCC losses are undefined because of a division by 0. This may be the cause of the low detection rate for rare event types in the event classification. Nonetheless, both loss functions show great promise in providing an out the box solution for many tasks with imbalanced data.

The final progressive experiment, which examined the effect of different additions to the neural network architecture, resulted in more open questions than originally posed. The attention augmentation was unable to learn to produce any meaningful features. Several reasons may be the actual cause of this issue. First and foremost, because the attention augmentation convolution is relatively new, no out of the box implementation for the pytorch framework could be found. The original publication does provide a comprehensive overview of the algorithm for the tensorflow framework, potential errors when converting the code into pytorch cannot be ruled out though.

Otherwise, the necessity to train the attention augmented block from scratch may also have lead to the performance loss. As the classification experiments showed, pretraining a model on a related task can provide large performance improvements.

However, the minor loss in performance shows the half of the channels without attention augmentation did manage to learn adequate weights. It is therefore deemed unlikely that this is the cause for the near constant output of the attention augmented channels.

Even though also unlikely, applying the attention augmentation only to the final block of the ResNet may also have hindered the network in learning a good attentional representation. Despite the layers being independent of one another, it may be necessary to generate attention based features in early layers for the later attention augmented layers to work at all. Finally, it may also be the case that the specific subtask of detecting player-ball interaction events is not suited for attention augmentation.

In the end, any, all or even none of the aforementioned reasons may be the true cause of why the attention augmentation was unable to provide any additional value. The attention augmentation implementation should have certainly been verified beforehand. The pragmatic approach would have been to reproduce the results from the original paper, thereby ruling out any issues with the implementation. The pretrained weights could then have been applied to the task in this thesis. Nonetheless, the non-result was included for completeness and as motivation for further research.

The temporalization of the spatial features also deserves a closer look. As already mentioned in Section 4.1.3, alternative architectures exist which have shown better performance in video action classification tasks. However, these architectures are unsuitable for long video clips. Instead, a split spatial and temporal processing via a ResNet and LSTM network was used. Even though adding the recurrence did improve performance, the improvement was smaller than expected. First of all, a recurrent model should be able to better take advantage of the event frame window structure. A positively tagged frame is always neighbored by at least one other positive frame. A non-recurrent model is unaware of this structure as it is unaware of neighboring frames. The recurrent model, on the other hand, has information on preceding frames and can factor classifications from preceding frames into its decision process.

Additionally, the temporalization of spatial features should allow for the detection of movement. Admittedly, to detect player-ball relations, predominantly, good spatial features are required. Nonetheless, being able to determine the direction and velocity of player extremities and the ball should aid in predicting the onset and end of an event frame window.

Most likely, the 2048 dimensional LSTM layer is far from complex enough to obtain a robust representation of detailed movement within a video though. A more sophisticated architecture working on a larger range of features may be necessary. The main challenge is how to efficiently extract continuous movement features, without having to do expensive redundant computations. Continuous in the sense that frames are iteratively given as input, such that the system is not constrained by

video length. Avoiding redundant computations means every frame should ideally be processed only once. This would exclude approaches like applying 3D convolutional networks to iteratively shifted frame stacks. Developing such an architecture remains an open research question.

A further reason why the recurrence only produced marginal performance improvements may be the actual task definition. The model is tasked to predict the eventedness for each frame, without any knowledge of succeeding frames. Because of the event frame window, multiple frames before the actual event are already tagged as containing an event. In consequence, a model has to predict the eventedness of a frame before the event has occurred. In essence, for frames before an event, the detection task turns into a prediction task. When viewing examples of misclassified events, a considerable number of examples were found where the model detected the event onset a couple of frames too late. In many of these examples, the ball moved extremely fast are was unexpectedly deflected. Hence, the prediction of the event onset was especially difficult.

As a solution, the event frame window could be shortened to only include frames following the event. This would not account for an occasionally misplaced timestamp though. Instead, allowing the model to have access to several preceding frames is a more elegant solution. In other words, the model should detect the eventedness of a frame several frames in the past. Thereby, the model would have the full sequence of event frames at its disposal.

In conclusion, the proposed model shows that the detection of detailed soccer events is possible. A model working solely on spatial features already of frames creates a solid baseline. Expanding these to spatiotemporal features leads to conclusive performance improvements. However, if adding visual attention helps in detecting events in videos could not be answered in the scope of this thesis.

7 Conclusion

To conclude, the goal of this thesis was to create a neural network architecture that can detect soccer events continuously solely from a soccer broadcast video. A variety of preprocessing methods, loss functions and neural network architectures were tested. In the end, a recurrent model extracting spatiotemporal features achieves the best results. Frame and event-based evaluations show that the proposed model can identify a significant portion of events from the test dataset. Even though the model struggles in scenes in which the ball is occluded or difficult to detect, it is able to show the potential of automatic continuous event detection systems.

Several directions for future research exists. For the proposed model specifically, increasing the video resolution and granting the model access to future frames may already lead to performance improvements. For continuous video processing using artificial neural networks in general, a closer investigation into the effects of visual attention is sensible. Furthermore, a richer spatiotemporal feature space is desirable. Instead of temporalizing high-level spatial features, spatiotemporal features need to be extracted at much finer detail to improve their informational value. How to do this efficiently and effectively, however, remains an open question.

After an extensive search, this work seems to be the first in continuous frame-based event detection using neural networks. Most previous work on neural network aided video processing focused solely on video-based labels. Only a small portion implemented models to localize events. However, none could be found that operated on arbitrary length videos and number of events. Lightweight but capable video processing pipelines are especially useful for real-time detection and should therefore not be neglected.

On top of this, this thesis is considered to make a couple of other noteworthy contributions. First of all, the MCC loss is shown to be a viable loss function for training neural networks on imbalanced data. It fills the role of an unbiased alternative to the previously proposed F1 loss function. Next, visual attention is applied in a neural network video processing framework. However, none of the evaluation measures were able to produce conclusive evidence that visual attention has a positive effect on video event detection.

Finally, it has shown that soccer event detection poses a challenging and diverse subtask in the class of video event detection. Applying relatively straightforward pretrained neural network models leads to adequate benchmark results. Improving on this benchmark is far from trivial though, requiring the integration of spatiotem-

poral features into a unified framework. At the same time, the clearly defined events and visual consistency of soccer videos make it easy to diagnose the behavior of neural network models. Soccer event detection could hence act as a superb test bed for advancing research in neural networks for video processing.

Bibliography

[1] Sami Abu-El-Haija et al. "Youtube-8m: A large-scale video classification benchmark". In: *arXiv preprint arXiv:1609.08675* (2016).

[2] Yuri Sousa Aurelio et al. "Learning from imbalanced data sets with weighted cross-entropy function". In: *Neural Processing Letters* 50.2 (2019), pp. 1937–1949.

[3] Moez Baccouche et al. "Action classification in soccer videos with long short-term memory recurrent neural networks". In: *International Conference on Artificial Neural Networks*. Springer. 2010, pp. 154–159.

[4] Nicolas Ballas et al. "Delving deeper into convolutional networks for learning video representations". In: *arXiv preprint arXiv:1511.06432* (2015).

[5] Michael Beetz et al. "Visually tracking football games based on TV broadcasts". In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2007.

[6] Irwan Bello et al. "Attention augmented convolutional networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 3286–3295.

[7] Joao Carreira and Andrew Zisserman. "Quo vadis, action recognition? a new model and the kinetics dataset". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[8] Liang-Chieh Chen et al. "Encoder-decoder with atrous separable convolution for semantic image segmentation". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 801–818.

[9] Min Chen, Chengcui Zhang, and Shu-Ching Chen. "Semantic event extraction using neural network ensembles". In: *International Conference on Semantic Computing (ICSC 2007)*. IEEE. 2007, pp. 575–580.

[10] Rewon Child et al. "Generating long sequences with sparse transformers". In: *arXiv preprint arXiv:1904.10509* (2019).

[11] Anthony Cioppa, Adrien Deliège, and Marc Van Droogenbroeck. "A bottom-up approach based on semantics for the interpretation of the main camera stream in soccer games". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 1765–1774.

[12] Anthony Cioppa et al. "A Context-Aware Loss Function for Action Spotting in Soccer Videos". In: *arXiv preprint arXiv:1912.01326* (2019).

[13] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[14] Shaza M Abd Elrahman and Ajith Abraham. "A review of class imbalance problem". In: *Journal of Network and Innovative Computing* 1.2013 (2013), pp. 332–340.

[15] Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. "A multiple resampling method for learning from imbalanced datasets". In: *Computational intelligence* 20.1 (2004), pp. 18–36.

[16] Bernard Ghanem Fabian Caba Heilbron Victor Escorcia and Juan Carlos Niebles. "ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 961–970.

[17] William Falcon et al. *PyTorch Lightning*. 2019. URL: `https://github.com/PytorchLightning/pytorch-lightning`.

[18] Christoph Feichtenhofer et al. "Slowfast networks for video recognition". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 6202–6211.

[19] Silvio Giancola et al. "Soccernet: A scalable dataset for action spotting in soccer videos". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 1711–1721.

[20] matchmetrics GmbH. 2020. URL: `https://www.matchmetrics.com`.

[21] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[22] Kaiming He et al. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[23] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[24] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-excitation networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7132–7141.

[25] Haroon Idrees et al. "The THUMOS challenge on action recognition for videos "in the wild"". In: *Computer Vision and Image Understanding* 155 (2017), pp. 1–23.

[26] Tomoki Imai et al. "Play recognition using soccer tracking data based on machine learning". In: *International Conference on Network-Based Information Systems*. Springer. 2018, pp. 875–884.

[27] Xu Jia et al. "Dynamic filter networks". In: *Advances in Neural Information Processing Systems*. 2016, pp. 667–675.

[28] Rafal Kapela et al. "Real-time event detection in field sport videos". In: *Computer vision in Sports*. Springer, 2014, pp. 293–316.

[29] Andrej Karpathy et al. "Large-scale video classification with convolutional neural networks". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014, pp. 1725–1732.

[30] Will Kay et al. "The kinetics human action video dataset". In: *arXiv preprint arXiv:1705.06950* (2017).

[31] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[33] Hildegard Kuehne et al. "HMDB: a large video database for human motion recognition". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 2556–2563.

[34] Xiaokun Li and Fatih Murat Porikli. "A hidden Markov model framework for traffic event detection using video features". In: *2004 International Conference on Image Processing, 2004. ICIP'04*. Vol. 5. IEEE. 2004, pp. 2901–2904.

[35] Tianwei Lin et al. "BMN: Boundary-matching network for temporal action proposal generation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 3889–3898.

[36] Tianwei Lin et al. "BSN: Boundary sensitive network for temporal action proposal generation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 3–19.

[37] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[38] M Manafifard, Hamid Ebadi, and H Abrishami Moghaddam. "A survey on player tracking in soccer videos". In: *Computer Vision and Image Understanding* 159 (2017), pp. 19–46.

[39] Brian W Matthews. "Comparison of the predicted and observed secondary structure of T4 phage lysozyme". In: *Biochimica et Biophysica Acta (BBA)-Protein Structure* 405.2 (1975), pp. 442–451.

[40] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[41] Joan Pastor-Pellicer et al. "F-measure as the error function to train neural networks". In: *International Work-Conference on Artificial Neural Networks*. Springer. 2013, pp. 376–384.

[42] Adam Paszke et al. "PyTorch: An imperative style, high-performance deep learning library". In: *Advances in Neural Information Processing Systems*. 2019, pp. 8024–8035.

[43] David Martin Powers. "Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation". In: *Journal of Machine Learning Technologies* 2.1 (2011), pp. 37–63.

[44] K Pradeep. "Significant event detection in sports video using audio cues". In: *International Journal of Innovations in Engineering and Technology (IJIET)* 3.1 (2013), pp. 144–151.

[45] Zhuwei Qin et al. "How convolutional neural network see the world-A survey of convolutional neural network visualization methods". In: *arXiv preprint arXiv:1804.11191* (2018).

[46] Houari Sabirin, Hiroshi Sankoh, and Sei Naito. "Automatic Soccer Player Tracking in Single Camera with Robust Occlusion Handling Using Attribute Matching". In: *IEICE Transactions on Information and Systems* 98.8 (2015), pp. 1580–1588.

[47] Saikat Sarkar, Amlan Chakrabarti, and Dipti Prasad Mukherjee. "Generation of Ball Possession Statistics in Soccer using Minimum-Cost Flow Network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019.

[48] Yongduek Seo et al. "Where are the ball and players? Soccer game analysis with color-based tracking and image mosaick". In: *International Conference on Image Analysis and Processing*. Springer. 1997, pp. 196–203.

[49] Gunnar A Sigurdsson et al. "Hollywood in homes: Crowdsourcing data collection for activity understanding". In: *European Conference on Computer Vision*. Springer. 2016, pp. 510–526.

[50] Karen Simonyan and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos". In: *Advances in neural information processing systems*. 2014, pp. 568–576.

[51] Bharat Singh et al. "A multi-stream bi-directional recurrent neural network for fine-grained action detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1961–1970.

[52] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. "UCF101: A dataset of 101 human actions classes from videos in the wild". In: *arXiv preprint arXiv:1212.0402* (2012).

[53] Manuel Stein et al. "From Movement to Events: Improving Soccer Match Annotations". In: *International Conference on Multimedia Modeling*. Springer. 2019, pp. 130–142.

[54] Ying-li Tian et al. "Event detection, query, and retrieval for video surveillance". In: *Artificial intelligence for maximizing content based image retrieval*. IGI Global, 2009, pp. 342–370.

[55] Du Tran et al. "Learning spatiotemporal features with 3d convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4489–4497.

[56] Grigorios Tsagkatakis, Mustafa Jaber, and Panagiotis Tsakalides. "Goal!! event detection in sports video". In: *Electronic Imaging* 2017.16 (2017), pp. 15–20.

[57] Takamasa Tsunoda et al. "Football action recognition using hierarchical LSTM". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017, pp. 99–107.

[58] Noor Ul Huda et al. "Estimating the number of soccer players using simulation-based occlusion handling". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 1824–1833.

[59] J Leon V. (https://tex.stackexchange.com/users/154390/j-leon v). *How do I draw an LSTM cell in Tikz?* Tex Stack Exchange. URL: `https://tex.stackexchange.com/q/432312`.

[60] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[61] Limin Wang et al. "Temporal segment networks for action recognition in videos". In: *IEEE transactions on pattern analysis and machine intelligence* 41.11 (2018), pp. 2740–2755.

[62] Shoujin Wang et al. "Training deep neural networks on imbalanced datasets". In: *2016 international joint conference on neural networks (IJCNN)*. IEEE. 2016, pp. 4368–4374.

[63] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[64] Qizhe Xie et al. "Self-training with Noisy Student improves ImageNet classification". In: *arXiv preprint arXiv:1911.04252* (2019).

[65] Yuanjun Xiong et al. "A pursuit of temporal accuracy in general activity detection". In: *arXiv preprint arXiv:1703.02716* (2017).

[66] Huijuan Xu et al. "Joint event detection and description in continuous video streams". In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, pp. 396–405.

[67] Ho-Sub Yoon, Young-lae J Bae, and Young-kyu Yang. "A soccer image sequence mosaicking and analysis method using line and advertisement board detection". In: *ETRI journal* 24.6 (2002), pp. 443–454.

[68] Guangyu Zhu et al. "Automatic multi-player detection and tracking in broadcast sports video using support vector machine and particle filter". In: *2006 IEEE International Conference on Multimedia and Expo*. IEEE. 2006, pp. 1629–1632.

Eigentsändigkeitserklärung

Ferdinand Schlatt

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit selbstständig verfasst und gelieferte Datensätze, Zeichnungen, Skizzen und graphische Darstellungen selbstständig erstellt habe. Ich habe keine anderen Quellen als die angegebenen benutzt und habe die Stellen der Arbeit, die anderen Werken entnommen sind - einschließlich verwendeter Tabellen und Abbildungen - in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht.

Bielefeld, den 11. Mai 2021

Unterschrift