# Data Science Project: Football Event Detection Milestone 3

**Alixia Birtles**, **Maxence de la Brassinne Bonardeaux**, **Jérôme Pierre**, and **Arthur Vogels**

**Abstract**

The goal of our project is to spot actions which occurs during a football game. This would help to better understand the game. We use a model that incorporate contextual information based on sequences because a previous model based on a per-frame level was generating poor results. However, we have discover an error in the latter, but still found that the contextual model outperformed the corrected version. We achieved good results compared to existing research studies which demonstrates the effectiveness of the approach used for action spotting. In order to determine whether the model could be used in real-time applications, we measured its latency. Our results show that the latency is not significant and the model can thus be deployed in real-time to build a timeline of actions.

## I. INTRODUCTION

Football is a globally renowned sport that receives significant media coverage. Journalists provide real-time commentary on games, covering all the action taking place on the pitch.

Our Data Science project aims to generate real-time events of football matches by detecting various actions that occur during the game. To achieve this goal, we have two objectives. Firstly, we aim to detect the seventeen actions of the SoccerNet action spotting task [1] by utilizing video fragments and feeding the data into a neural network. Secondly, we aim to generate a timeline event live with minimal delay. Although we have analyzed this task in depth, it has not yet been solved.

In this report, we will first provide a brief reminder of our previous results and discuss how we have managed the project. We will then detail an error we made in our last model, which resulted in unsatisfactory results. Despite this, the model still showed reasonable performance. Next, we will present a new model that combines two state-of-the-art techniques using context to detect actions, which outperforms our previous model. The new model will be compared to the previous one built for the previous milestone and fixed in this one. The results of the comparison will be analyzed in depth in order to determine the best model. Finally, the latency of the model will be measured to determine whether the model could be used in real-time.

## II. PREVIOUS WORK

So far, we have developed a model using a convolutional neural network (CNN) to identify five different actions (corner, goal, penalty, card, and kick-off) based on a single frame. The CNN was built using "EfficientNetB7" [2] to extract features, and a multilayer perceptron (MLP) is used to classify the extracted features. However, the model yielded very poor results when used to create a timeline of football events. This timeline was created using thresholds on the probabilities obtained from the CNN. While we initially attributed this to the high number of confusing images in the dataset, we later discovered a coding error that contributed to the poor performance.

Given these issues, we explored alternative models and methods to improve our results. Ultimately, we decided to pursue a model that incorporates contextual information by analyzing a sequence of frames. This approach has shown promise in early testing and we will continue to refine it as the project progresses.

## III. PROJECT MANAGEMENT

### A. Project Definition and Final Objective

The goal of our project is to develop a real-time football event recognition system using video input of a match. As said previously, the system will be able to recognize the 17 different types of events of the SoccerNet challenge. This system should be able to create a timeline of each event and we will characterize the latency of our model.

By the end of the project, we expect to have a proof-of-concept that can recognize these events. A demo will be shown for our final presentation, displaying the events like a timeline.

### B. Current Objectives

The third part of our action project comprises three main objectives.

The first objective was to investigate why our initial model, based on per-frame analysis, performed poorly.

The second objective was to improve event detection accuracy by building a model that incorporates context from several frames, based on Yahoo research. [3]

The third objective was to monitor the latency of the system and quantify the latency of model to ensure real-time performance of our demonstration.

All three objectives have been accomplished, as described in the following sections.

### C. Next objectives

For the final presentation, we intend to perform three steps:

(a) Solving one of the minor limitation of our model as it will be explained in Section VI.

(b) Deploy a demonstration of our model building a timeline. The result will be recorded on a video and shown at the final presentation

(c) Building the final report and incorporate the feedback received during our presentation.

### D. Work Distribution

During the four weeks that separated the milestone 2 from the milestone 3, here is how the work was distributed:

(a) Week 1: Arthur and Jérôme build the context model. Maxence and Alixia worked on the per-frame model to investigate more why it was failing.

(b) Week 2: Alixia analysed the latency. Arthur and Jérôme trained the neural network. Maxence build the post-processing technique to create the timeline.

(c) Week 3: Arthur and Jérôme trained the context model with a revised dataset to improve the results. Maxence analysed the results of the timeline and started the report. Alixia started to work on a demonstration of our timeline.

(d) Week 4: All of us worked on the report by building analysing the results and investigate future work.

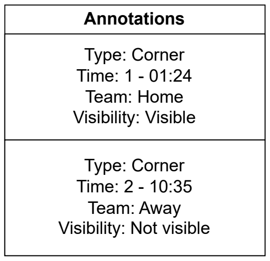
## IV. DATA

### A. Raw Data Description

Our training dataset consists of 500 videos of different European football matches,[1] each of which has been annotated for seventeen distinct types of events.[2] Each match has two halves, therefore the dataset includes 1000 MKV files of 45 minutes each. Each video has a frame rate of 25 frames per second (fps), and for each match, there is a JSON file containing annotations of the seventeen different types of events which follow the structure of the Figure 1. These annotations come from the SoccerNet dataset. [1] The choice of this dataset was motivated by the already available hand-made annotations of the different events happening on screen, which were a requirement for us to be able to perform our task.

Our first neural network requires frames as input. As extracting images from the MKV files is time-consuming,[3] we extracted images corresponding to events in the matches and stored the mapping of these images and events in a database. The details of this mapping process are explained in the following section.

For our second model, pre-extracted features of each image made available by SoccerNet. We used a special feature extractor developed by Baidu Research, [4] which provided a vector of size 8576 at a rate of 1 fps. These features were extracted using a transformer model to add context to those features.



**Annotations**

Type: Corner
Time: 1 - 01:24
Team: Home
Visibility: Visible

Type: Corner
Time: 2 - 10:35
Team: Away
Visibility: Not visible

*Figure 1: Example of how the 'Labels.json' files are organized. We have access to information such as the occurrence time of the event, which half of the match the event takes place in, whether the event is shown in the sequence and the team involved (if applicable).*

### B. Data Management

For our first model based on a CNN, the dataset is composed of single frames of each event. To augment the size of the dataset, each frame within a one-second interval after the annotated time of the event was considered as the same event. This trick allows increasing

[1] These matches come from the Premier League, Bundesliga, LaLiga, Ligue 1, Serie A, and the UEFA Champions League, spanning the period from 2014 to 2020

[2] These events include Penalty, Kick-off, Goal, Substitution, Offside, Shots on target, Shots off target, Clearance, Ball out of play, Throw-in, Foul, Indirect free-kick, Direct free-kick, Corner, Yellow card, Red card, and Yellow to red card.

[3] Around two days of one CPU on Alan Montefiore supercomputer.

the dataset size and provide patterns and features in sequences corresponding to specific actions.

As we mentioned previously, retrieving the images directly from the MKV files would be too time-consuming, so we extracted all the images from the matches and stored the path of those images in a database along with metadata such as the event it relates to, and the match it related to. The full schema is available in Appendix A. This database was essential for efficient training of the network, as using the MKV files for each event would take a lot of time. Moreover, mapping the images to an event is not trivial either, and storing this information was useful to avoid making this computation all the time. Although querying the database to retrieve the image paths added some overhead, this was done only once before training and did not affect the training time.

We used SQLite to store our database due to its simplicity and our use-case did not require a more powerful and sophisticated database management system. Physically, the database was stored on the same machine that was running the training process.

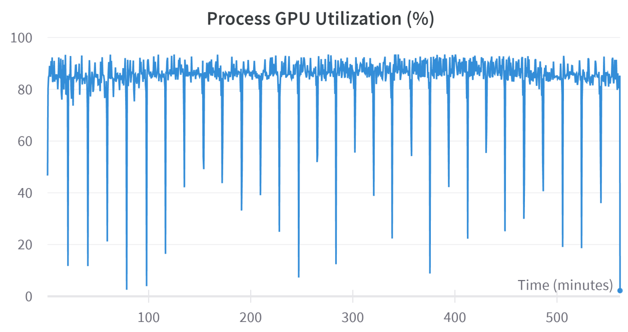


*Figure 2: Percentage of GPU utilization. Most of the time, 80 to 90% of the process utilization is used. This implies the overhead that comes from retrieving our images and the database is very small.*

The data of our second model consists of the pre-extracted features (npy files) and the JSON files. Retrieving the path of the data using the database instead of a brute force search of those files does not add any significant improvement unlike our first model. We were under the minute in each case. Therefore, managing data for this part was much easier. Indeed, in our first model, managing the data with the frames was not that easy. However, from the frames, it is quite straightforward to build a dataset for the model. In the second case, retrieving the data was easy, but the difficulty was to create a dataset for our model. We will detail this process in Section V B.

To deal with the large amount of data, we had to bring the data to the GPU when used for training and clear the memory to make space for different data. We used PyTorch, which provided some help for this. As it can be seen on Figure 2, most of the time, the GPU works at 80 to 90% of its full capacity.

## C. Data Analysis

To achieve the objective of detecting the seventeen different actions that occur during a football match, the project uses a dataset where the number of available sequences for each action class is shown in Figure 3. The dataset shows a high imbalance where the 'Ball out of play' class is dominant over the other classes. This distribution is related to the real probability of a specific action occurring during a football game. As a result, it is deduced that 'Ball out of play' is the most frequent action in a football game, while 'Penalty' or 'Red card' are less common.

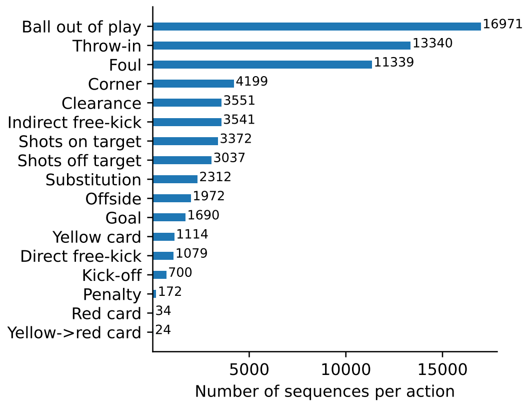


*Figure 3: Number of sequence available for each action. The action with the highest percentage of available sequence is the 'Ball out of play' which corresponds to 24.8% of all the sequences. On the other hand, 'Yellow to red card' is the least represented action with only 0.03% of available sequence. The distribution of data among the actions are not well-balanced.*

The main goal of our project is to generate a timeline that displays the different actions and their timestamps during a soccer game. To estimate when these events commonly occur within a half time period, we created a box plot (Figure 13). This plot shows that most actions can happen at any time during the match.

When creating a timeline, not only the number of actions but also the timing of the actions is important. To consider this aspect, Figure 13 was designed. The figure shows that distribution of cards tends to occur after the middle of the game. The median is above 25 minutes for each of the three different types of card. This could be due to the stress experienced by players or their fatigue, which may make them less attentive and more likely to make a foul. Another observation is that most actions have the same probability of appearing at time-step. The only exception is the 'Kick-off' which is of course far more likely at the beginning of the match.

## V. EXPERIMENTATION

### A. Models

Initially, our model used single images as input. These images were processed using an efficientNet to extract the features. Then, a prediction head consisting of a multi-layer perceptron (MLP) with dimensions of $512 \times 64 \times 6$ was added to classify the input image into one of the 6 possible classes (5 classes and one 'NoClass' class). We trained our model using images, with oversampling to balance the number of examples across different classes as it was observed in Figure 3. For the rest of this section, unless specified otherwise, we will talk about the context model. Details about this model can be found in the report of the previous milestone.

As explained above, the current model takes sequences of frames as input, which are processed by a 1D U-Net inspired by Yahoo research. The U-Net can then either be connected to a confidence head, whose goal is to identify intervals around which a specific event occurs or a temporal displacement head, whose goal is to predict where the ground truth is located in these intervals. Our model did not use raw images directly. Instead, we used a feature extractor provided by the SoccerNet community from Baidu research [4] that compresses image information into a feature vector. This feature extractor is advantageous over others, such as ResNet, because it considers context by extracting features from multiple frames. This feature extractor at 1fps has shown to improve the detection of football events compared to the classical ResNet feature extractors.

The resulting predictions are matrices of size $T \times C$, where $T$ is the number of frames and $C$ is the number of classes. In order to train the model, the target matrices are derived from annotation files provided by SoccerNet. The confidence matrix $\mathcal{M}_1$ contains 1s at positions $(t, i)$ where an event of class $i$ occurred at frame $\bar{t}$ with a radius of $r_C$, and the displacement matrix $\mathcal{M}_2$ contains the temporal displacement between frames and the nearest occurrence of class $i$ in the temporal domain $\tau = [\bar{t} - r_D, \bar{t} + r_D]$. Each model was trained separately. The reason for training the displacement model is that it is used to account for the fact that events may not occur at exactly the same frame across different video sequences. By training the model to predict the displacement between frames and the nearest occurrence of a given event, it becomes more robust to variations in timing and can make more accurate predictions in new scenarios.

The confidence model uses a generalized binary cross-entropy loss: $L(y, y') = \sum_{t=1}^{T} \sum_{c=1}^{C} BCE(y_{t,c}, y'_{t,c})$ where $y_{t,c}$ is the ground truth at time-step $t$ for class $c$ and $y'$ is the prediction of the model. This loss was chosen to accommodate multiple events appearing at the same time-step due to the radius technique. For the displacement model, the Huber loss was used for the positions within the time domain $\tau$. The Adam Optimizer with a learning rate of $1e-3$ is used, and the sharpness-aware minimization (SAM) is used for both losses to improve generalization.[5]

### B. Dataset Creation

Sending all the different sequences in a half-time to the models is not feasible. This is why we decided to select a finite amount of sequences per half-time. Our first idea for the selection was to choose the start of the sequence uniformly. However this approach leads to current events such as "Ball out of play" to be overly represented, while rarer events were overlooked.

To tackle the problem, the idea consisted in tracking the number of occurrences of each class and dynamically inferring a distribution that would favor the less frequent classes during training. In brief, when choosing the sequences to consider for the current half-time, we refer to the number of occurrences of each class, normalize it to create a distribution, take the complement of the probabilities and create a new distribution from these.

This new distribution can be seen as a multinomial from which we will choose which class needs to be present in our video sequence. Then, we randomly select an event of this class and select uniformly the start of a clip containing it. The second method was implemented and lead to an increase in the average-mAP of 12%. It is worth noting that this technique does not guarantee that all actions will appear as often in the training sequences, but it does ensure that rarer actions are given more attention during the training.

### C. Metric used: Average-mAP

The metric used to evaluate and compare different models on the SoccerNet dataset is called the average-mean average precision (Average-mAP). Its use as a benchmark for the action-spotting task was presented in the SoccerNet introductory paper [6].To use it, we first define a tolerance $\delta$ around each true event. A prediction is considered as a true positive if the predicted event occurs within a certain time window centered around the true event, where the size of the window is determined by a tolerance parameter. A false positive is a prediction of an event outside of the time window for a true event of the same class. A false negative occurs when no predicted event falls within the time window for a true event.

The classical mAP can be computed base on the previous definitions. The metric is computed for several tolerance thresholds $\delta$ (5-60 seconds). The resulting mAP values for each tolerance are then averaged to obtain the average-mAP.

### D. Training

Most of the training has been inspired from Yahoo research [3] except our dataset creation. The radius for the confidence model was 3 seconds and 6 seconds for the displacement model. The learning rate was $1e-3$. The initial time window was 112 seconds. Both models have been trained for almost 24 hours with a batch size of 200. The resulting losses are respectively displayed in Figure 4 and Figure 5.

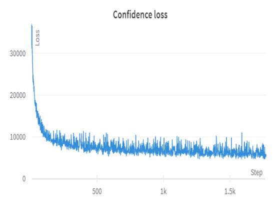


*Figure 4: Evolution of the loss for the confidence model. The loss converges after 1500 steps.*

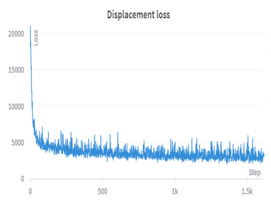


*Figure 5: Evolution of the loss for the displacement model. The loss converges after 1000 steps.*

### E. Post-processing

After creating the two models, we computed the confidence and displacement for each action at each time-step. Then, we shifted the confidence from their displacements and kept the maximum probability when several probabilities appear at the same time-step. To reduce false positives, we used Non-Maximum Suppression (NMS) technique with a time window of 20 seconds, where probabilities within the window and next to the highest probabilities are set to 0. With this technique, we achieved an Average-mAP of 78%, reproducing the best known result of 2022.

The NMS was also applied to the per-frame model once a prediction was computed for each frame of the match.

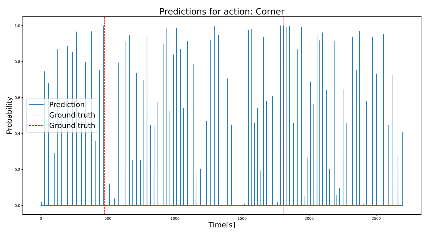
## VI. EVALUATION

As our intention is to create a live timeline, decreasing the time window will reduce latency but this must be done without a significant impact on performance. We tested three different sizes: 32, 64 and 112 seconds. We denote M@T the context model of a window of T seconds and M@F the model using the CNN. We also report the best result of 2022 on the SoccerNet action spotting challenge. The results are shown in table I. As it can be seen, the model using a time window of 112 seconds is the best, most of the time by a significant margin except for 'Yellow cards'. However, the model using a time window of 64 seconds still performs quite well (Average-mAP of 78% vs 76%) but when the time window decreases to 32 seconds, performance drops significantly (Average-mAP of 66%). The four worst classes (Average-AP under 70% for M@112) are 'Shots on target', 'Shots off target', 'Red card' and 'Yellow to red card'. The performance of last two classes can be explained by Figure 3 because we do not have many sequences of that type. However, thanks to oversampling in our dataset, we managed to detect them sometimes. The first two classes are most probably due to confusion between images. Indeed, those two classes are quite similar and can be even confused with goals. For the rest of this section, we will use M@112 unless specified otherwise.

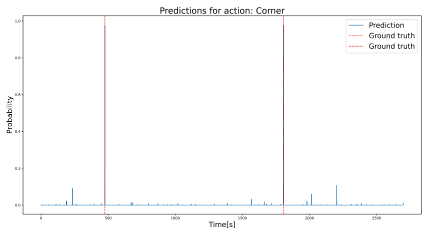
| Average-AP (%) | M@32 | M@64 | M@112 | M@F | Best(2022) |
|---|---|---|---|---|---|
| Penalty | 88 | 93 | **100** | 0 | NA |
| Kick-off | 52 | **80** | **80** | 14 | NA |
| Goal | 87 | 86 | **87** | 11 | NA |
| Substitution | 76 | 80 | **82** | NA | NA |
| Offside | 68 | 68 | **79** | NA | NA |
| Shots on target | 60 | 61 | **64** | NA | NA |
| Shots off target | 57 | 57 | **68** | NA | NA |
| Clearance | 72 | 71 | **79** | NA | NA |
| Ball out of play | **85** | 80 | 83 | NA | NA |
| Throw-in | **82** | 80 | 81 | NA | NA |
| Foul | 79 | 80 | **82** | NA | NA |
| Indirect free-kick | 66 | 65 | **73** | NA | NA |
| Direct free-kick | 83 | 77 | **88** | NA | NA |
| Corner | 88 | 86 | **88** | 34 | NA |
| Yellow card | 81 | 87 | **89** | 9 | NA |
| Red card | 0.4 | **100** | 67 | NA | NA |
| Yellow to Red card | 1 | 58 | **59** | NA | NA |
| mAP | 66 | 77 | **78** | 14 | **78** |

*Table I: Average average-Precision on each class. (NA: is for Not available) The best model is nearly always the model using a time window of 112 seconds, but the model using a time window of 64 seconds also performs very well compared to it. We managed to reproduce the results obtained by state of the art techniques.*

In our previous milestone, we discovered that our first model, which was based on a per-frame level, had a high rate of false positives due to the lack of contextual information. Since many images can appear similar even though they represent different events, this can cause confusion for the model. When we compared the performance of our first model with a new one that used context to solve this issue, as shown in Figure 6, we found that the latter had almost no false positives. Although there were occasional spikes in the probability, they were minor and did not affect the overall accuracy of the model.



(a) Per-frame model.



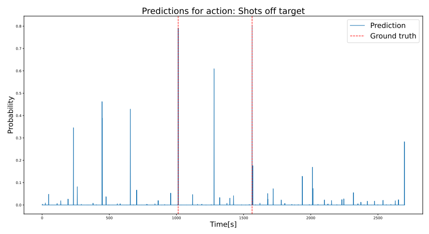
(b) Context model on a time window of 112 seconds.

*Figure 6: Comparison of the predictions of both models on one half of a match. The context model is very reliable and has very few false positives.*

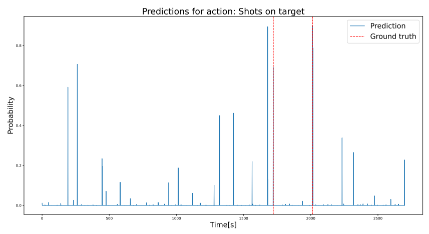
Although this is very encouraging, some care should be taken. Indeed, the 'Corner' class was one of our best result. For classes with less performance like 'Shots on target' and 'shots off target', false positives arrive more often as it can be seen in Figure 7.

Finally, we show that a possible explanation for the small jumps that can be observed in Figure 6 is the beginning of the window. Indeed, the predictions of a whole sequence is shown in Figure 8 versus the ground truth. It can be observed false positives appear at the beginning of the window for the classes 'Foul' and 'Shots on target'. We believe our model performs worse at the beginning of a sequence because it has no contextual information before the action occurs and that would be an explanation for the false positives observed in Figure 6.

In conclusion, our model demonstrated high performance in detecting most of the actions with reliability. However, certain classes were more challenging to detect due to limited examples in the dataset or similarity between sequences. Additionally, the model exhibited higher false positive rates at the beginning of a sequence, possibly due to the lack of contextual information. To address these limitations, we propose potential solutions in Section VIII.



(a) Shots off target.



(b) Shots on target.

*Figure 7: Comparison of the predictions of M@112 on a full half of a match. Several times during the game, high probabilities occur for those events although nothing happened.*

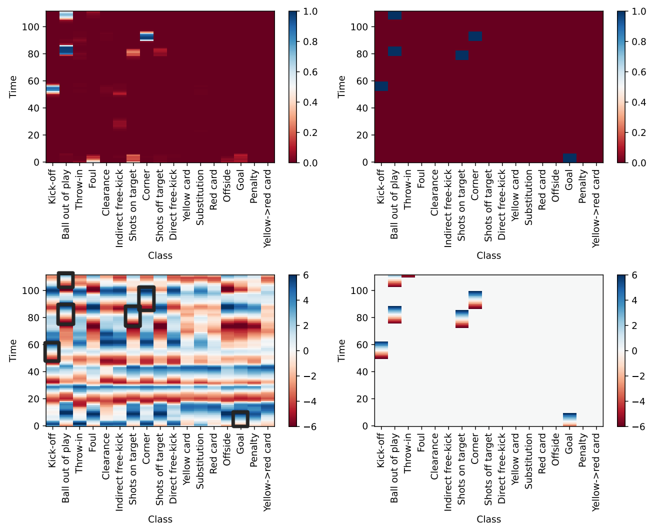


*Figure 8: Predictions of the classes for the confidence and the displacement compared to the ground truth. The predictions seem to be very similar to the ground truth except at the beginning where false positives arrive.*

## VII. DEPLOYMENT: ANALYSIS OF THE LATENCY

The timeline system consists of three steps: extracting images from the video, transforming each image into a feature vector, and creating a timeline using our model. To extract frames from the MKV video files and create features, we tested different libraries and found the two fastest methods are shown in Table II. The important point is that retrieving frames from a 45-minute video takes less than 45 minutes. Therefore, it can be done in real-time, making the latency negligible. Indeed, frames can be extracted in real-time of the match without any delay.

| Library | 'OpenCV' | 'MediaPy' |
|---|---|---|
| Average time (sec) | 489 | 530.8 |

*Table II: Mean time (in seconds) required to transform a mkv file into a sequence of frames for different libraries, averaged over five videos. The "OpenCV" library is the fastest.*

The second step involves generating features from the extracted frames. However, in our project, we used Baidu features [4] instead of implementing this step ourselves. Despite this, we still included this subpart in the latency analysis for real-world conditions. We assumed that the time required to generate features from the frames and convert them into tensors would be relatively short and comparable to the time our model takes to transform a window size of features into predictions.

The last step of the model consists of sending features, window by window, into the neural network to obtain output and apply post-processing techniques. We measured the time required to generate predictions per window, based on a half-time of a football game, and the results are presented in Table III.

| Window size (sec) | 32 | 64 | 112 |
|---|---|---|---|
| Time over a window (msec) | 76.1 | 95.5 | 117.5 |

*Table III: Time required to compute the output of the model depending on the size of the window of the model. For all three methods, the model takes around less than one second per window. Therefore, there is very little latency.*

Taking into account the entire system, we have shown that it should have very low latency. While the main latency is due to extracting the frames, it can be done in real-time, so it should not be a problem. The main latency will be determined by the window size used in the model, as the entire window is required to make a prediction. Therefore, a model using a window size of T seconds will have a minimum latency of T seconds before any prediction can be made.

Based on the average-mAP in the Table I and the latency of the model in the Table III to generate predictions from a window of frames, we decided to take a window of 64 seconds for our final model. Indeed, the decrease in performance is minimal and the latency coming from the size of the window is reduced by nearly two. (112 seconds versus 64 seconds)

## VIII. IMPROVEMENTS AND NEXT STEPS

We believe the performance of our model can be improved by fine-tuning hyper-parameters or exploring other training techniques. However, since our model is based on existing research, we do not expect a significant improvement. To enhance the model, we can explore three directions. The first direction involves adding sound to the model, which could make it more complex. The second and third directions aim to address the limitations of our model.

(a) The next step to improve the model is to incorporate sound, as research has shown that combining images and sound can significantly improve the Average-mAP. [7] The architecture of the paper for incorporating sound is similar to the existing architecture, but an audio feature extractor is added to extract features from the spectrogram of the sound at 1 fps. The extracted features will then be combined with image features using VGG [8], a deep convolutional network architecture, and merged before passing through the 1D U-net. This approach is motivated by the belief that detecting patterns using both predictions is better than detecting them separately, and by the fact that early merging of methods in the original paper led to better results.

The architecture is shown in Figure 9.

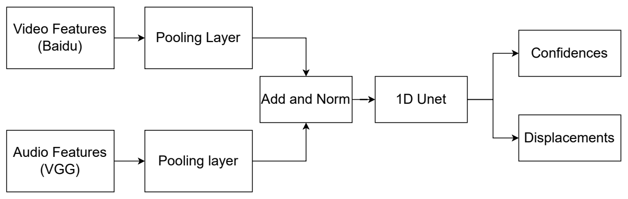


*Figure 9: Design of the architecture merging audio features and images features. The merging part appears before the 1D U-net.*

(b) The next step to improve the model is to merge similar events, which are rare in the dataset, such as red cards. A two-stage detection approach was shown to be effective in previous research [9]. The approach involves combining similar events, such as yellow and red cards, and then using a second neural network to distinguish between them.

(c) The proposed solution for the problem of false positives at the beginning of a sequence is to create overlap between the sequences and discard the predictions of the beginning of the overlapping sequence. This way, when creating the timeline, predictions from each time-step can still be obtained while discarding the less reliable predictions from the beginning of the sequence.

Our plan for the final presentation is to implement the improvement of discarding the predictions at the beginning of a sequence to increase the reliability of the timeline. However, we will not be able to explore the first two improvements due to time constraints. Additionally, we aim to present a live demonstration where a user can select any match from our test set and observe the timeline generated by our model.

## IX. CONCLUSION

The model that we have finally built to detect the actions in a football game incorporate contextual information. This results in a network that learns on a sequence of frames around action, instead on only one frame. This network uses the sequences of frames expressed as features, where each sequence corresponds to an action. However, we encountered an issue with the imbalanced distribution of actions in the dataset, with some actions being less frequent than others. To address this issue, we over sampled the less frequent classes during training.

To evaluate the models and to compare it to existing research studies, the metric Average-mAP has been used. Our results show that the contextual model we have implemented achieve comparable performance to the references for certain window size. Our model outperformed the previous model we fixed, which had been generating poor results.

In the final stage, we will focus on the deployment of our model in real-time to detect actions during football games. Our goal is to show a video of a game and the actions which are detected at the same time. To achieve this, we had to measure the latency of the model and found that the results are not significant, which indicates that a real-conditions deployment is possible.

The contextual model takes as input a window size of frames which has been decided to be of the size of 64 seconds based on the trade-off between the results of the model, the latency, and the window size.

Overall, our project proves the effectiveness of the contextual approach for action spotting in football games. In the future, we plan to deploy our model in real-conditions with the aim of providing valuable insights for football analysts or coaches.

## X. ACKNOWLEDGEMENTS

We extend our sincere thanks to Anthony Cioppa for his invaluable advice on building the context-aware model. Without his help, we would not have been able to develop this complex model. We utilized ChatGPT to assist us in checking our grammar and clarity throughout the report. Specifically, we wrote our own text and then fed it into ChatGPT to obtain reformulated versions that were more clear and error-free. This technique was applied to most of the report.

## Appendix A: Database schema

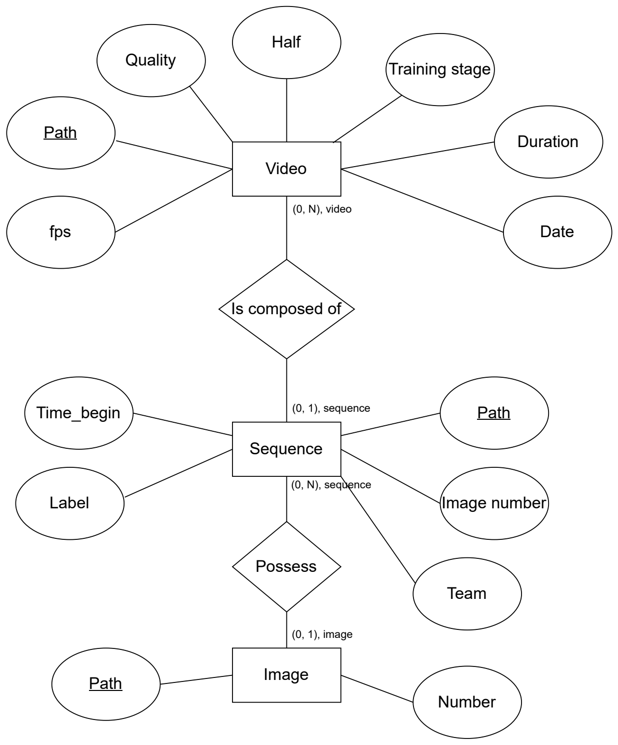


*Figure 10: Entity-relationship schema of our database storing relevant metadata for our data analysis.*

## Appendix B: Model Architecture

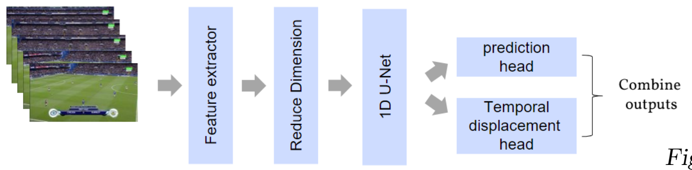


*Figure 11: Architecture of the model inspired by João et al [3]. Features are first extracted by the Baidu feature extractor [4]. They are then reduced using 1x1 convolutional kernels and fed to a 1D-Unet. Two different prediction heads are implemented in order to increase the precision on the tight average-MAP benchmark. [3]*

### 1. Feature extractor

Along the project, different feature extractors were considered. Following Zou et al [4], the Baidu features, available on SoccerNet, were chosen and led to satisfactory results. These features, of dimension 8576, are then fed to a dimension reducer. The usefulness of this module is twofold: at first, it reduces the complexity of the U-Net by decreasing the input space. Besides, it allows an implementation that is independent of the features dimension by always outputting 64-dimensional features.

### 2. 1D U-Net

The use of a 1D U-Net was motivated by the need of incorporating context to the predictions. The 2D U-Net architecture was first introduced in Ronneberger et al [10] and was shown to perform well on localization tasks, over the two dimensions of an image. Considering a 1D U-Net allows the transition from a spatial localization task to a temporal localization task. The model is composed of four contracting layers, each of them leading to a division by two of the temporal dimension; 4 expanding layers, to restore the initial temporal dimension; long-skip connections from the left part to the right part of the network, ensuring the taking into account of the global context. The choice of architecture for the convolutional blocks was inspired by He et al [11, 12] and consist in pre-activated bottleneck residual blocks. For further information on the architecture, please refer to the report of the second milestone.

### 3. Prediction heads

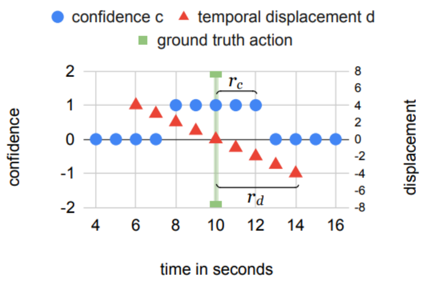
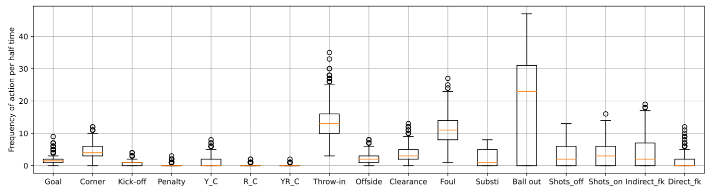


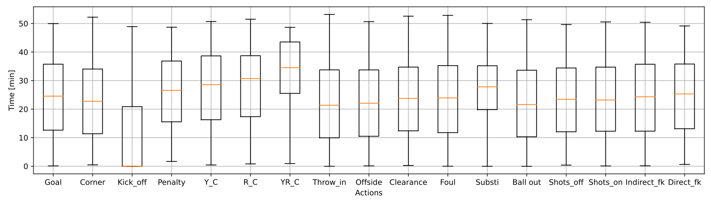
*Figure 12: Representation of the confidence and temporal displacement for a given class around an event actually taking place. [3]*

The model will be trained twice to output two $T \times C$ matrices containing an approximation of the confidence and temporal displacement measures. In practice, we train the model with the confidence head before the model with the temporal displacement head. The usefulness of the latter is particularly highlighted when one considers small tolerances $\delta$.

## Appendix C: Data analysis



(a) Number of actions per half-time.



(b) Distribution of the action occurrence time.

*Figure 13: (a) Distribution of the number of actions for a half-time game of football. In a football game, the most common action is the 'Ball out of play' followed by the 'Throw-in'. This seems obvious as every time the ball goes out of the pitch, it has to be throw-in. On the other side, the actions such as the 'Cards' (red or yellow) and the 'Penalty' are much less frequent, with a mean around 0. (b) Distribution of the moment, during a half-time game, at which each action among the seventeen occurs over the entire database. Most of the time, action occur during all the game and have a median around the middle of it. Some action show a median higher or smaller than this half of the half-time, like the 'Yellow-red card' and the 'Kick-off'.*

[1] Adrien Deliège, Anthony Cioppa, Silvio Giancola, Meisam Jamshidi, Jacob Dueholm, Kamal Nasrollahi, Bernard Ghanem, Thomas Moeslund, and Marc Droogenbroeck. Soccernet-v2: A dataset and benchmarks for holistic understanding of broadcast soccer videos. pages 4503–4514, 06 2021.

[2] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[3] João V. B. Soares, Avijit Shah, and Topojoy Biswas. Temporally precise action spotting in soccer videos using dense detection anchors, 2022.

[4] Xin Zhou, Le Kang, Zhiyu Cheng, Bo He, and Jingyu Xin. Feature combination meets attention: Baidu soccer embeddings and transformer based temporal detection, 2021.

[5] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization, 2021.

[6] Silvio Giancola, Mohieddine Amine, Tarek Dghaily, and Bernard Ghanem. SoccerNet: A scalable dataset for action spotting in soccer videos. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, jun 2018.

[7] Bastien Vanderplaetse and Stéphane Dupont. Improved soccer action spotting using both audio and video streams. pages 3921–3931, 06 2020.

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[9] Ali Karimi, Ramin Toosi, and Mohammad Ali Akhaee. Soccer event detection using deep learning. *arXiv preprint arXiv:2102.04331*, 2021.

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks, 2016.