UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

AUTOMATIC ANALYSIS OF SPORTS EVENTS IN VIDEO SEQUENCES

A Master's Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Laia Prat Ortonobas

In partial fulfilment

of the requirements for the degree of

MASTER IN ADVANCED TELECOMMUNICATIONS

Advisor: Javier Ruiz Hidalgo and Josep Ramon Morros Rubió

Barcelona, January 2023

Title of the thesis: Automatic analysis of sports events in video sequences

Author: Laia Prat Ortonobas

Advisor: Javier Ruiz Hidalgo and Josep Ramon Morros Rubió

Abstract

These last years have been an increasing need regarding the automatic sport analysis. Being soccer one of the most watched sports around the world, action spotting for soccer videos has become one of the main field studies in computer vision.

In this project, we have done a deep research and analysis of the State of the Art regarding human recognition and action recognition in soccer videos. Apart from this more theoretical part, we have also executed and compared two of the most pioneering models in sports action spotting today: CALF[3] and NetVLAD++[4] models. In addition, we have been able to make a modification in one hyperparameter, the window size, to evaluate its effect on performance.

Finally, we have concluded that we get the highest mAP performance with the NetVLAD++[4] model and that modifying the window size worsens the overall performance in both models, but the individual class performance can benefit from it as the performance is improved in some classes for different window sizes.

To my future nephew, who seems to be waiting for me to finish this project to come out.

Acknowledgments

Firstly, I would like to thank my supervisors, Ramon Morros and Javier Ruiz, for their guidance and support during the whole project. Not only they have taken care of my learning, but also they have encouraged me to do it and have given full support, emotionally and academically.

On the other hand, and not less important, I am also deeply grateful to my friends and family for their unwavering support and motivation throughout the years. Their help has been invaluable, and without it, I would not have been able to reach this point.

Revision history and approval record

Revision	Date	Purpose
0	09/10/2022	Document creation
1	12/10/2022 - 20/01/2023	Document edition
2	20/01/2023 - 23/01/2023	Document revision

Table 1: Revision history.

Written by:		Reviewed and approved by:	
Date	23/01/2023	Date	23/01/2023
Name	Laia Prat Ortonobas	Name	Javier Ruiz Hidalgo Josep Ramon Morros Rubió
Position	Project Author	Position	Project Supervisors

Table 2: Approval record.

Table of contents

List of Figures

List of Tables

1. Introduction

Action spotting research in soccer videos is a really important and emerging task because it has the potential to significantly improve the way that soccer videos are analyzed and understood. With the ability to accurately spot specific actions within a soccer video, such as goals, saves, and passes, it can be used for a large variety of applications such as video summarization, player performance analysis, and tactical analysis. Additionally, it can also be used to help automate the process of creating video highlights, making it easier for fans to access and watch the most important moments from a game and, also for creating training datasets for other soccer-related tasks such as player identification, ball tracking, and tactical analysis, which in turn can lead to more accurate and faster results.

The project aims to compare two different approaches in the same dataset, the SoccerNet-v2[1]. The first approach is Context-Aware Loss Function (CALF[3]), which aims to take into account the context of the video in order to improve the accuracy of action spotting. The second approach is Temporally-Aware Feature Pooling (NetVLAD++[4]), which aims to improve the accuracy of action spotting by pooling features from different parts of the video in a temporally-aware manner. Both approaches aim to improve the accuracy of action spotting in soccer videos, but they do so in different ways.

Apart from that, we have also been able to execute and obtain some results, we have evaluated them and compared their performances. In addition, we have performed a modification to the models in order to see the implications of window size parameter and its effects on the results.

a) Requirements and specifications

The purpose of this study has been to understand, analyze and evaluate different spotting actions in soccer video methods using Deep Learning.

In order to reach the requested goals for this project, it has been necessary to have basic theoretical knowledge of Deep Learning techniques on video object spotting.
We had to learn how an action spotting system works and also learn to manage the software environment.

To do that, we have deeply analyzed the State of the Art regarding action spotting in soccer videos and also regarding human action recognition in order to have a better understanding of the field. Once we have set the environment to be able to run the codes, we carried out several executions of training and testing to get the results and analyze them.

All in all, we are going to explain what are the two models selected that we have based on, demonstrate their effectiveness in executing them, and finally present the effect of changing the window size.

b) Statement of purpose

This project deals with action spotting techniques in football videos, and its purpose is twofold. From an educational standpoint, studying action spotting within the context of soccer videos utilizing Deep Learning techniques provides an opportunity to gain knowledge about the concept of action spotting, its implementation through various architectures, the various methodologies that are currently available, and the practical application of said methodologies. From a research perspective, the project examines the performance of two distinct systems when applied to the same dataset, providing insight into the relative efficacy of each approach.

The main requirements of the project have been the following:

- Database with videos and images to test both systems.
- Remote server to execute our scripts (provided by Image Processing Group UPC).

c) Work Plan

The different tasks of this project have been classified into 5 sections: State of the Art, Software, Databases and Modifications, and finally Documentation. In the Figure 1 you can see the time periods of each one during the project.

First, State of Art consists of the study and research about current Deep Learning techniques in action spotting and at which point the research development on this topic is. Once we had sufficient knowledge of the theory, we followed with the software and databases part. It was the moment to put the theory into practice, run different tests with the systems and the dataset chosen and compare them, after that we have evaluated the modifications proposed to each model, and finally we have written the documentation and proposed some improvements.

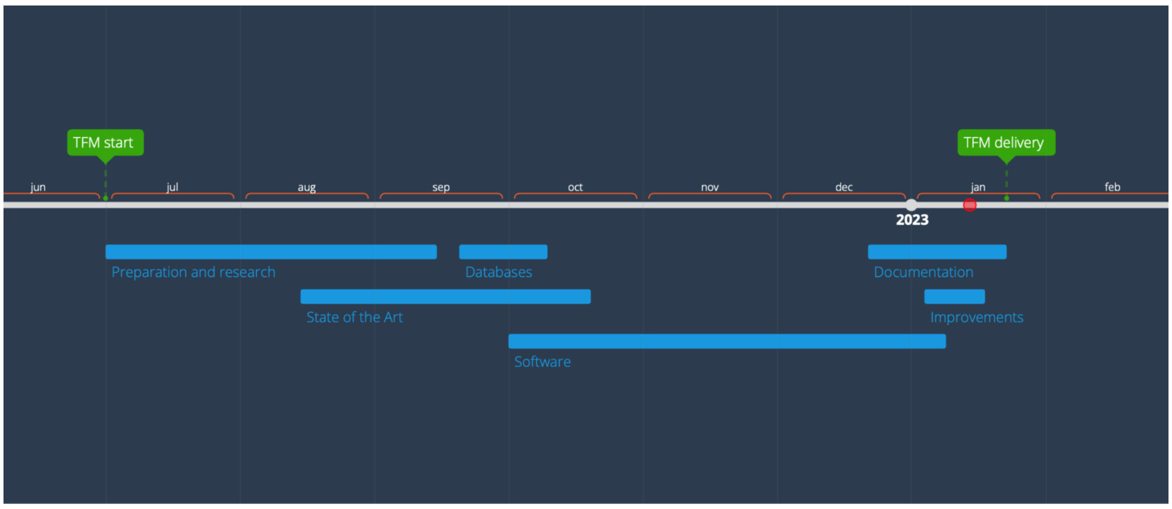


Figure 1: Time plan.

Please find attached in the appendices the work packages, tasks and milestones.

d) Deviations

Once we had started the project, we realized some facts. We thought some of the tasks would require us less time than what they have finally had, and this has led to some very busy and stressful days. Also, it is important to mention the personal situation about reconciling doing a TFM and having a full-time job.

1.1. Project structure

- State of the art of the technology used or applied in this thesis
- Methodology / project development
- Results
- Budget
- Environment Impact
- Conclusions and future development

2. State of the art of the technology used in this thesis

In this first part, we will focus on explaining the State of the Art of action recognition first, in order to introduce the State of the Art of action spotting in soccer videos later.

2.1. Human action recognition

Human action recognition is a research area that aims to identify and classify human actions in video data. It has been an area in continuous development for some years now. Especially because of the important role that it plays in the understanding of videos. It is formulated as a multi class classification problem to obtain class labels of the activity in question. Complex video scenes with different contexts, scales, and textures make action recognition a challenge.

Human action recognition using Deep Learning can have a variety of applications[29] such as surveillance and security to identify and track individuals in real-time video streams, human-computer interaction in order to recognize human actions for user interfaces, robotics to train robots for a better understanding and responding to human behavior, in health field to monitor patients' movements and detect falls or other signs of distress, in sports analysis in order to improve performance analysis and coaching, and also in virtual reality and gaming.

Action recognition can be determined using a variety of techniques without the use of Deep Learning[46], such as motion tracking. In order to solve the challenges of motion tracking, researchers began applying computer vision techniques such as optical flow, Scale-Invariant Feature Transform (SIFT)[47], Histograms of Oriented Gradients (HOG)[48], and Mean Shift[49].

During the last few years, the number of papers, models, and techniques applied in action recognition has been very large[29]. If we go back to 2013, there was a study[7] that used Hidden Markov Models (HMM) to sculpt the skeleton of a person. Objects and activities were represented by nodes, while associations between an object and its activities were represented by edges. Also using a 3D skeleton and a native occupation model, another article[8] planned a new attribute for learning human actions, thus eliminating the dependence on 3D joints.
In another paper[9] in the same year, an original extraction approach proposed fusing grayscale and depth frames to extract 3D abstraction and temporal descriptors of human activities. In 2014, a Bayesian model with skeleton features was used in a study[10] to predict human actions, by using probability weights to compensate as posterior probabilities. To anticipate human activities, also in 2014, a model proposed[11] the fusion of three machine learning methods: the K-means method to detect a person's skeleton pose, the SVM method to classify, and the Hidden Markov model (HMM) to model behavior.
Two different studies[12][13], worked on the human skeleton and key pose detection using RGB-D, which detailed actions over an extremely large area to achieve the same goal.
To improve the classification results of human activities, a particular model[14] to learn about the Spatio-temporal choices of Perennial Convolution Networks with Gated Perennial Units was

proposed. The model was based on VGG-16 associated with the ImageNet transfer learning model.

Sensors, and Long Short-Term Memory and Perennial Convolution Networks were also used by in 2017 in an article[15] to describe human motion.

In order to assist the native optical flow of the global human silhouette for human activity recognition, a new paper[16] was published relying entirely on temporal choices.

A little after, in another paper[17] it was discussed the learned spatio-temporal options in depth by discovering motion descriptors using an aggregated descriptor vector and predicting motion descriptors using SIFT geometric information.

A method[18] to derive human choices using a 3D-CNN architecture and combining the RNN with GRU hidden units was proposed in 2017.

Also in 2017, RGB-D sensors were used in another article[19] to determine human skeletal options, the K-means technique to determine posture, and sequential lower-order optimization for training knowledge.

The Recurrent Convolution Neural Network (RCNN) model for human activity recognition of the GoogleLeNet architecture was presented[20] in 2016, and also used for action recognition in another paper[21] in 2018**.** The authors used CNNs and RNNs to extract temporal and spatial options, mixing the options and measuring video classification accuracy using the GoogleLeNet design.

Later on, a technique that supported skeleton and tandem speech function extraction was designed, created and used in three different papers[22][23][24]. The RGB-D device, as well as CNN and LSTM models, were used to extract the skeleton options.

A view-guided skeleton-CNN[25] was proposed for capricious reading and human behavior recognition that continued from weak reading variations by viewing skeleton sequences while also covering a wider set of reading angles.

The DenseNet CNN model was used in another proposal[26] to classify actions, which also used an associated degree action recognition model to facilitate the transition from a skeleton to an abstraction presentation.

And it was in 2017 and in 2018 that two strategies[27] [28] focused on human cause estimation and skeletal feature extraction were presented, realizing that they provided more excellent classification rates.

In all this, the recent State of the Art methods[29] include:

- Two-Stream CNNs: two CNNs are used, one for spatio-temporal feature extraction from RGB frames, and another for extracting motion features from optical flow maps. The features from the two streams are then fused for action recognition.
- Temporal Segment Networks (TSN): the video is segmented into several parts and then processes them independently using a CNN. The final predictions are made by combining the results from all parts.
- Non-Local Neural Networks (NLNet): self-attention mechanisms are used to weight the importance of different parts of the video for action recognition.

- Slow Fast Networks: uses a slow pathway that extracts features at a lower frame rate and a fast pathway that extracts features at a higher frame rate to learn both temporal and spatial information.
- 3D CNNs: 3D CNNs are used to process the video frames directly, which capture both spatial and temporal information.

These methods achieve high performance on action recognition benchmarks. However, there are still open challenges such as actions recognition in crowded scenes, actions recognition with partial observation, and actions recognition with variations in camera viewpoints, lighting conditions, and occlusions.

2.2. <u>Action spotting in soccer videos</u>

As we have said in the beginning, it was important to start with the State of the Art regarding human action recognition to be able to explain the State of the Art about action spotting in soccer videos now.

There are several potential applications using Deep Learning to perform action spotting in soccer videos[41]. Some examples include player tracking to track individual players in a soccer match, game analysis by identifying specific actions and events, automatically generation of the highlights of a soccer match, video summarization, live commentary of the soccer match predicting the next move and providing the corresponding commentary and also automatic video editing.

Several approaches were proposed in order to tackle the problem, one approach[30] was to use CNNs to extract spatio-temporal features from the video frames, and then use RNNs to model the temporal dynamics of the actions. Another one[29] proposed was to use a combination of CNNs and RNNs, such as a two-stream CNN-RNN architecture, which used a CNN to extract spatial features from individual frames and an RNN to capture temporal information. Many State of the Art methods[31][32] also propose attention-based models or Graph Convolutional Networks (GCN) to improve the performance of action spotting or also to incorporate temporal information by using Long Short-Term Memory (LSTM) networks to process the output of the CNNs over time. Another research[34] was focused on weakly-supervised action spotting, which used only video-level labels, rather than frame-level labels, to train the model. This is useful in situations where it is difficult or expensive to annotate every frame of a video.

An important aspect of action spotting in soccer videos is the use of spatio-temporal action proposals[33], which are used to identify potential regions of interest in the video, where an action may be taking place. This can be done using methods such as Sparse Spatio-Temporal cubes (SST) or using optical flow to identify regions of motion. Also, several works focus on the use of multi-modal information like audio and text from commentators to improve the performance of action spotting.

Beginning in 2018 a research paper[35] that presented a method for spotting actions in videos was proposed. They used a method to localize the temporal boundaries of those actions within the video, using a two-stream network architecture, which combined information from both the RGB and optical flow modalities of the video.

Then in 2020, there was another study[36] based on multi-modal (audio and video). They proposed to use both the audio and video streams of the footage. The algorithm used the audio stream to extract features related to crowd reactions and commentator speech, and the video stream to extract features related to the visual appearance of the soccer actions. It was in this same year, when CALF[3] method was presented, they proposed a loss function that specifically considers the temporal context naturally present around each action, rather than focusing on the single annotated frame to spot.

A two-stage paradigm[37] in 2021 was presented. They proposed fine-tuning multiple action recognition models on soccer data to extract high-level semantic features, and design a transformer-based temporal detection module to locate the target events.

In 2021, a paper[38] using a Transformer mode was presented, which allowed for capturing important features before and after action scenes, and it analyzed which time instances the model focused on when predicting an action by observing the internal weights of the transformer.

Also in 2021, NetVLAD++[4] method came out, a new feature pooling method based on NetVLAD[5], dubbed NetVLAD++[4], that embeds temporally-aware knowledge. They proposed to disentangle the context from the past and future frames and learn specific vocabularies of semantics for each subset.

Last year, in 2022, a paper[39] focused on understanding the dynamics of the game was presented, by identifying and representing the players, referees, and goalkeepers as nodes in a graph, and by modeling their temporal interactions as sequences of graphs. The idea was an unsupervised player classification algorithm that used the spatio-temporal information of the players in the video to identify the individuals and different types of soccer actions.

Last but not least, in 2022, a model[40] using a dense set of detection anchors was presented. It predicted a detection confidence and corresponding fine-grained temporal displacement for each anchor.

3. Methodology/project development

In this section of the project we are going to outline the plan for conducting the study, starting with the dataset used, as it is a very important part of any project because it provides valuable information that can be used to make better decisions, improve the accuracy of predictions, and advance research in various fields.
Sequentially, the two models chosen to carry out the comparison will be described, as well as their architectures and the procedures that have been used to collect and analyze data. We will delve into the execution of the proposed models, CALF[3], and NetVLAD++[4]. And, in the last part, we are going to discuss the modifications that were made to the models during the experimentation phase and the rationale behind them.

3.1. SoccerNet-V2

SoccerNet-v2[1] is an extension of SoccerNet[2] with new and challenging tasks including action spotting, camera shot segmentation with boundary detection, and a novel replay grounding task.

They manually annotated 300k timestamps, temporally anchored in the 764 hours of the 500 games of SoccerNet[2]. The vocabulary of the 17 classes is centered on the soccer game and soccer broadcast domains, hence it is well-defined and consistent across games. They annotated each action of the 500 games of SoccerNet[2] with a single timestamp, defined by well-established soccer rules. There is a total of 110,458 annotated actions, on average 221 actions per game, or 1 action every 25 seconds. SoccerNet-v2[1] is a significant extension of the actions of SoccerNet[2], with 16x more timestamps and 14 extra classes.

Regarding the features they provide them from ResNet [42], I3D[50] and C3D [51] computed at 2 fps for an easier experimentation, and they also provide them reduced with PCA to 512 dimensions.

SoccerNET-v2[1] is made up of 500 entire soccer games, featuring:

- Full untrimmed broadcast videos in both low and high resolution.
- Pre-computed features.
- Annotations of actions among 17 classes.
- Annotations of camera replays linked to actions.
- Annotations of camera changes and camera types for 200 games.
- 17 different actions.

The repository contains several baselines for each task:

- Action Spotting
- Camera Shot Segmentation
- Replay Grounding

Actions:

- Ball out of play: when the ball goes past one of the boundary lines of the field.
- Throw-in: moment when the player throws the ball.
- Foul: moment when the foul is committed.
- Indirect free-kick: moment when the player shoots, to resume the game after a foul, with no intention of scoring.
- Clearance (goal-kick): moment when the goalkeeper shoots.
- Shots on target: when the player attempts to score by directing the ball towards the goal frame.
- Shots off target: moment when the player shoots, with the intention to score, but the ball does not go in the direction of the goal frame.
- Corner: moment when the player shoots the corner.
- Substitution: when the substituted player exits the playing field by crossing one of the boundary lines.
- Kick-off: moment when, at the beginning of half-time or after a goal, the two players in the central circle make the first pass.
- Yellow card: moment when the referee shows the player the yellow card.
- Offside: when the assistant referee signals using their flag.
- Direct free-kick: moment when the player shoots, to resume the game after a foul, with the intention to score or if the other team forms a wall.
- Goal: moment when the ball crosses the line.
- Penalty: moment when the player shoots the penalty.
- Yellow then red card: moment when the referee shows the player the red card.
- Red card: moment when the referee shows the player the red card.

3.2. CALF

The authors of the paper[3] presents a context-aware loss function for action spotting in soccer videos. The authors propose using a combination of spatial and temporal context to improve the performance of action spotting models. The spatial context is represented by a 2D attention mechanism that takes into account the spatial layout of the players on the field, and the temporal context is represented by a 1D temporal convolutional network that captures the temporal evolution of the actions.

3.2.1. Model description

It is a system that models the temporal context information naturally present around the actions. As shown in Figure 2, what is proposed in this model is a network formed by a frame feature extractor and a temporal CNN that gives C feature vectors per frame, a temporal segmentation module that gives segmentation scores for each class from the segmentation loss, and from this loss, the network extracts features from the video up to the spotting module. Finally, the spotting module predicts the classes and the exact frames of the actions by extracting $2 + C$ values per spotting prediction, being the first value a binary indicator of the presence, the second the location of the frame annotated, and the C remaining values represent the one hot encoding of the action.

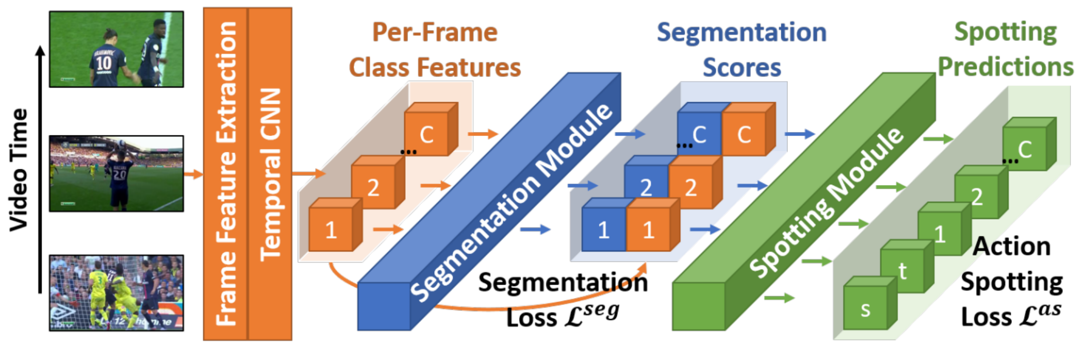


Figure 2: General scheme of CALF. Figure extracted from [3].

What they do is to define six temporal segments around each real action point, each of which induces a specific behavior in our context-aware loss function when training the network.

- 'Long before' and 'Long after' the action, their influence is negligible, so the network is trained not to predict an action.
- 'Just before' the action, there is no influence on the network, since a particular context may or may not lead to an action.
- 'Just after' the action, its contextual information is rich and unambiguous, as the action has just occurred. Therefore, the network is trained to predict an action.
- Finally, there are two 'Transition' zones, one previous to the 'Just before' segment and the other posteriors to the 'Just after' segment. In these zones the loss function is smooth, and the network is again trained not to predict an action.

For each class c, temporal segments are bounded by specific cutoff parameters and are materialized through the temporal shift encoding, which contains richer temporal context information about the action than the initial binary location annotation.

Each action is identified by a single action frame annotated as such, and each frame of the video is annotated with a One Hot encoded vector with C components for action frames or with a vector of C zeros for background frames.

The representations of the frame features, which are ResNet[42] features precomputed and provided by SoccerNet[2] dataset, pretrained on ImageNet and used as the result of frame feature extraction.
To train the network, the initial annotations are encoded in two different ways: with a Time Shift Encoding used for temporal segmentation loss, in which each raw video is time-shift encoded before training, and with a YOLO[43] encoding used for action location loss, where each new training chunk is encoded.

1st. Frame feature extractor and temporal CNN

SoccerNet provides 3 frame feature extractors with different backbone architectures, each one of them ending up with 1024, 4096, and 2048 features that are reduced with PCA to 512 features for each frame of the subsampled videos. As it can be seen in the orange part of Figure 2, the reduced features per PCA provided by the dataset are used as input to the temporal CNN. The objective of the temporal CNN is to provide Cf features for each frame, being f the number of features computed for each class and for each frame, before the segmentation module, while at the same time sweeping the temporal information between frames.

First, each frame is introduced into a two-layer MLP to reduce the dimensionality of the feature vectors of each frame, and a set of $N_F \times 32$ features called $\mathcal{F}_{MLP}$ is obtained, where N_F is the number of frames in the chunk considered.
The $\mathcal{F}_{MLP}$is introduced into a spatio-temporal pyramid, i.e. It is introduced in parallel to each of the layers of the pyramid, producing 120 features per frame, which are concatenated with the $\mathcal{F}_{MLP}$to obtain a set of $N_F \times 152$ features.

Finally, these features are introduced into a convolution layer, which produces a set of $N_F \times Cf$ features called $\mathcal{F}_{TCNN}$. These will be the features that will be introduced to the segmentation module, these features are the ones explained in the previous section SoccerNet-v2[1], which are precomputed and provided by SoccerNet [2], so they are obtained offline.

2nd. Segmentation module

From this module, which is shown in the blue part of Figure 2, we get a segmentation score per class for each frame, the $\mathcal{F}_{TCNN}$ is transformed into an output of dimension $N_F \times C$, following the steps detailed below:

- Reshape $\mathcal{F}_{TCNN}$ to have dimension $N_F \times C \times f$.
- Use batch normalization by frames.
- Activation with a sigmoid so that each frame has, for each class, a feature vector.
- For each frame and for each class, the distance d between v and the center of the unit hypercube is calculated.
- The segmentation score is obtained between [0, 1]. In this way, scores close to 1 for a class, can be interpreted as an indication that the frame is likely to belong to that class.

In this part is when the Time Shift Encoding used for the temporal segmentation loss is done, the main goal here is to encode the initial annotations to train the network. The temporal context around each action is cut into segments related to the distance to the action. The segments regroup the frames that are before, just before, just after, and long after an action, or in transition zones between segments.

Context slicing is used to perform Time Shift Encoding of each frame x in the video with a vector of length C, which contains class information about the relative location of x with respect to its nearest past or future actions.

3rd. Spotting module

This is the green part of Figure 2 and also the last part of the model. It takes as input the $\mathcal{F}_{TCNN}$ and the segmentations scores, and it outputs the spotting predictions $\widehat{\mathbf{Y}}$ of the network, $\widehat{\mathbf{Y}}$ is a matrix that regroups the spotting predictions of the network, and has dimension $N_{pred} \times (2 + C)$, being N_{pred} the number of predictions output by the network for the spotting task. The first column of $\widehat{\mathbf{Y}}$ represents the confidence scores for the spots, the second contains the predicted locations, and the other are per class classification scores.
What this module does is a one-to-one iterative matching, between the predicted locations and the ground truth locations.

It is in this second module where the YOLO[43]-like encoding is done to encode the initial annotations in a second different way, but in this case used for the action spotting loss.

Inspired by the YOLO[43] model, each real action in the video generates an action vector composed of $(2 + C)$ values, as explained previously:

- The first value is a binary indicator of the presence of the action.
- The second value is the location of the frame annotated as an action, which is calculated as the index of that frame divided by N_F.
- The remaining C values represent the One Hot encoding of the action class.

The entire video containing N_{GT} actions is encoded in a $\mathbf{Y}$ matrix of dimension $N_{GT} \times (2 + C)$, where each line represents an action vector of the video. Being N_{GT} the number of ground-truth actions in the chunk consider.

3.3. NetVLAD++

In this paper[4] the authors presents a method for action spotting in soccer videos that utilizes a temporally-aware feature pooling. The authors propose to use a temporal convolutional network (TCN) to model the temporal evolution of the actions and a pooling mechanism to aggregate the features from different frames in the video. The pooling mechanism is designed to be temporally aware, meaning that it takes into account the temporal context of the action when selecting the frames to be pooled.

3.3.1. Model description

A NetVLAD[5] based feature pooling method, called NetVLAD++[4], is proposed that incorporates temporal awareness. What it does is to partition the context before and after an action occurs and learn a semantic vocabulary, temporally aware of past and future temporal context, specific to each subset, avoiding mixing and blurring that vocabulary over time. Injecting this prior knowledge creates more informative pooling modules and more discriminative pooling features, leading to a better understanding of actions. A more efficient architecture is used for action detection, in terms of memory and computational complexity. The original temporal pooling mechanism proposed in SoccerNet[2] is improved by introducing a temporally-aware bag of words pooling modules and a learnable projection is allowed to reduce feature dimensionality.

The previous pooling methods, VLAD[6] and NetVLAD[5] were permutation invariant, so they did not consider the order of the frames, but only aggregated the features as a set. NetVLAD[5] was a differentiable pooling technique inspired by VLAD[6]. This last one learned clusters of features descriptors and defined an aggregation of feature as the average displacement of each features with respect to the center of its closer cluster. NetVLAD[5] generalized VLAD[6] by softening the assignment for full differentiable capability, and disentangling the definition of the cluster and the assignment of the samples.

The task of action localization, i.e., locating well-defined actions in time, anchored with unique timestamps throughout a video, is usually solved by observing a temporal context around a given timestamp and regressing the action in time. That is why this new time-aware pooling module, called NetVLAD++[4], is used. The past and future context are defined as the frame feature with a temporal offset, then the class-aware action is reduced to a singularly anchored point using non-maximum suppression techniques.

Taking into account the Figure 6, we can see the whole architecture where the proposed pooling module is integrated and which is based on:

- Pre-trained frame feature encoder.
- Dimensionality reduction.
- The proposed pooling module from a temporarily sliding window, NetVLAD++[4].
- Per-frame classifier representing a class-aware actionness.

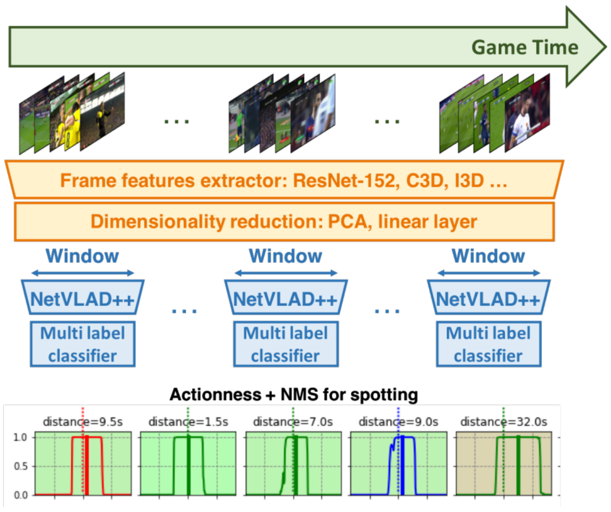

Game Time
Frame features extractor: ResNet-152, C3D, I3D ...
Dimensionality reduction: PCA, linear layer
Window
NetVLAD++
Multi label classifier
Actionness + NMS for spotting
distance=9.5s
distance=1.5s
distance=7.0s
distance=9.0s
distance=32.0s
1.0
0.5
0.0

Figure 3: Action spotting architecture in NetVLAD++. Figure extracted from [4].

In the last part of Figure 6 it can be seen how the action spotting is then performed using a non-maximum suppression (NMS).

1st. Frame feature extraction and dimensionality reduction

As it can be seen in Figure 6, the method starts with a pre-trained frame feature encoder followed by a dimensionality reduction. The reduction of the features goes from 2048 to 512, instead of using a PCA they use a linear layer. They tested that a linear layer learned a better linear combination of frame features by removing the orthogonality constraint introduced by the PCA, the one used in the CALF[3] model. Furthermore, learning a PCA reduction is only possible offline, making it impractical for online training, as it requires prior feature extraction.

2nd. Temporally-aware pooling

Chunks of time window T are considered throughout the video. The set of temporally contiguous features is divided into equal parts before and after the center of the window and clustered accordingly. The features are normalized along the feature dimension, and NetVLAD++[4] for each feature subset is applied. The output NetVLAD++[4] features are concatenated along the feature dimension, resulting in a feature of dimension $(K_b + K_a) \times D$, being K_b clusters from the past context, K_a clusters from the future and D the features dimension.

3rd. Video Chunk Classification

A non-overlapping window chunks are considered in training, with a sliding window of stride T. A classifier is built on the top of pooled feature, composed of a single neural layer with sigmoid activation and dropout. Since multiple actions can occur in the same temporal window, a multi-label classification approach is considered. A video chunk is labeled with all classes appearing in the chunk with a multi-label one hot encoding, and it is optimized for a multi-label binary cross-entropy loss.

3.4. Modifications

The modifications we wanted to carry out have been mainly related to hyperparameter tuning. Tuning the hyperparameters of a model is a challenging task, as there are many hyperparameters that can be optimized. This includes the learning rate, batch size, window size, train/test split ratio, number of epochs, number of layers… among many others.

For our project, we decided to modify the temporal window size, which in CALF[3] is name chunk size. In the context of action spotting, this hyperparameter refers to a fixed-length video segment that is used as input to the model.

The window size can affect the performance of the model, as it determines the temporal context that is used to represent the action. Larger window size will include more frames and therefore provide more temporal context, but it may also include irrelevant information that can reduce the performance of the model. A smaller window size will include fewer frames and provide less temporal context, but it may be more efficient and reduce the risk of overfitting. The optimal window size can vary depending on the dataset, the action classes, and the model architecture. It's important to experiment with different window sizes to find the best value that works well with the model and the dataset.

What we have done as a modification on both models has been increasing and decreasing the value of these hyperparameters to see the effect of them in the performance and their importance in the models.

3.4.1. CALF modification

In this model[3], the network is trained in batches of chunks, where a chunk is a segment of the video defined by the number of seconds.
Each batch contains chunks extracted from a single raw video and a chunk, around each ground-truth action, is extracted, such that the action is randomly located within the chunk. Then, to balance the batch, N_{GT}/C chunks composed of background frames only are randomly extracted, being N_{GT} the number of ground truth frames and C the number of classes.

3.4.2. NetVLAD ++ modification

The window size is used in the feature pooling step of the model[4], where the features from a sequence of frames are combined to produce a compact and robust representation of the action. A window of time T seconds along the video is considered, and the temporally contiguous set of features are split equally before and after the center of the window and pooled accordingly.

4. Results

In the results section of this project, we are going to present the findings of the study, including a clear and concise summary of the data collected and analyzed, as well as the values obtained in the trainings and tests that were performed, these two sets of data have been the same in both methods[3][4] to be able to make a correct comparison of the results between the two models.

First we will present the results of each model and then their comparison. After this part, there will be the results of the modifications made and finally a qualitative analysis.

4.1. CALF

The CALF[3] model was trained using the Adam optimizer with a learning rate of 0.0001. The batch size was set to 32, and it was trained for 18000 chunks per epoch, with 1000 as the maximum number of epochs.

A context-aware loss function was used to guide the training process. This loss function took into account the temporal context of the action instances, which allows the model to learn more robust representations of the actions. There are two different losses:

- Context-aware loss, used in the temporal segmentation module
- YOLO[43]-like loss, used in the action spotting module

The losses of these modules are calculated based on the re-encodings. It is necessary to retrain the initial annotations in two different encodings in order to train the network, the Time Shift Encoding used for the temporal segmentation loss and the YOLO[43]-like encoding used for the action spotting loss.

The results provided by the authors[3] demonstrate that the context-aware loss function improves the model's ability to spot actions in soccer videos, and it was found that especially in cases where the actions are temporally close to each other, as their work goes one step further, by directly designing a temporal context-aware segmentation loss.

Analyzing the values provided in Table 7 and Table 8 we could see that we obtain very similar results as the ones given by the authors, which demonstrates the generalization capability, simply using an enhanced context-aware spotting loss function. A loss function for temporal action segmentation, further used for the task of action spotting, which is parameterized by the time shifts of the frames from the ground-truth actions.
There are also a per class results provided, find the tables attached in the appendixes (in the CALF[3] training and test parts). For a given class, a fix a tolerance is set around each annotated action to determine positive predictions and these results are aggregated in a confusion matrix. An action is considered spotted when its confidence score exceeds some threshold optimized for the F1 score on the validation set. From the confusion matrix, it can be computed the precision, recall and F1 score for that class and for that tolerance.

In this case the best spotted action have been ‘Foul’ for both execution's run and test, which is quite understandable as it is an action in which the players are still for a few seconds in the same position, and it is also a bit determined. While the worst detected action have been ‘Yellow then red card’ action for training and ‘Red card’ action for testing, having had both a very low result in both executions as they are quite similar and really difficult to distinguish because there could be a lot of different situations which lead to them.

4.2. NetVLAD++

This model[4] was also trained using the Adam optimizer with a learning rate of 0.0001, like the CALF[3]. However, in this case, the batch size was set to 256 and the model has been trained with 1000 maximum number of epochs.
A classifier, such as a fully connected neural network, is trained on the pooled features to predict the action labels, and also it is trained using a supervised learning algorithm, such as backpropagation.

ExtractResNET_TF2 has been used as a model architecture for extracting features from videos. It is based on the ResNet[42] architecture, and it is designed to take in a video as input and extract features from it that can be used for action spotting. We noted this because it was a difficult task to face as we have to adapt different parts of the code to be able to run the model, because we had to extract the features first.

If we take a look at Table 7 and Table 8 again, it can be demonstrated that the authors of the model[4] have implemented an efficient architecture for action spotting, leading to the State of the Art performances for action spotting in SoccerNet-v2[1].

In relation to the class results, whose tables can be found attached in the appendices too (in NetVLAD++[4] training and test parts), most classes appears to provide optimal results. Yet, ‘Clearances’, ‘Kick-Off’, ‘Yellow card’ and ‘Red car’ appear not to benefit from the temporal-awareness. Nevertheless, the improvement is consistent regardless of the action visibility regarding the temporal-awareness. The assumption made by the authors[4] is that those actions are already discriminative enough without it. However, the action giving the best mAP value is ‘Goal’, which is comprehensible as it is really benefit from the temporal-awareness, mainly because of the presence of cheering previously. On the other hand, the worse one is again ‘Yellow then red card’, as in CALF[3] model.

In general, according to the authors[4] the results of running NetVLAD++[4] in soccer video datasets show that it is effective in learning compact and discriminative video representations, which improves the performance of action recognition tasks.

4.3. Comparison

As it has been mentioned, CALF[3] and NetVLAD++[4] are both models used for action spotting in soccer videos. However, they have different architectures and are used for different purposes.

The CALF[3] is a loss function that is specifically designed to improve the performance of action spotting in videos. Context information can include information such as the position and size of the action within the frame, and the temporal relations between the actions.
On the other hand, NetVLAD++[4] is trained to learn compact video representation, it combines a CNN and a VLAD[6] pooling layer to learn the video representations.

- **Training**

METRIC	CALF	Temporally-Aware
Average mAP visibility all	0.418	**0.613**
Average mAP visibility visible	0.447	**0.654**
Average mAP visibility unshown	0.334	**0.435**
Execution time	27452.31 seconds	**2384.36 seconds**

Table 4: Performance comparison between values obtained by us of CALF[3] and NetVLAD++[4] in training set.

- **Test**

METRIC	CALF	Temporally-Aware
Average mAP visibility all	0.380	**0.526**
Average mAP visibility visible	0.398	**0.587**
Average mAP visibility unshown	0.307	**0.342**
Execution time	560.62 seconds	**426.11 seconds**

Table 5: Performance comparison between values obtained by us of CALF[3] and NetVLAD++[4] in test set.

In terms of action spotting, it is said that CALF[3] is more specialized for this task, whereas NetVLAD++[4] is more general and is designed for video retrieval tasks too. However, in terms of performance, it depends on the dataset and the type of action spotting task being considered. In general, both models have been shown to achieve good performance on soccer video datasets, but NetVLAD++[4] has given better performance in training and testing.

More concretely, as it is displayed in Table 7 and Table 8, on the test evaluation NetVLAD++[4] overcomes CALF[3] model by 27.76% on 'Average mAP visibility all' metric, by 32.2% on 'Average mAP visibility visible', and by 10.23% on 'Average mAP visibility unshown'. In addition, the execution time is shorter, NetVLAD++[4] takes 134.51 seconds less than CALF[3].

The NetVLAD++[4] improvement comes from 3 main differences w.r.t the other methods:

- An optimized NMS head to extract spotting results.
- A linear layer for the frame features instead of using a PCA reduction.
- A temporally-aware pooling module.

Both approaches, NetVLAD++[4] and CALF[3], consider temporal semantic regions around the action to spot. CALF[3] defines a high-level semantic context from different temporal regions far distant, just before and just after an action occurs, and a hand-crafted loss function is introduced that weights the contextual information. Still, the same features are leverage for the context before and after the actions occurs. In contrast, in NetVLAD++[4] the far distant semantic context is dropped and specific features from different vocabulary for the past and future temporal context are learned.

4.4. Modifications

In this part, we refer to the process of adjusting some hyperparameters of the model, to improve its performance on the task at hand. It has been a process of trial and error, in which we have tested different values to understand the reason of the ones set by the authors of both models[3][4] and to see the affectation of them on the performance. As explained before, we have modified the window size for both models, being named chunk size in case of CALF[3].

The fact is, reducing the window size in action spotting models can have a negative impact on the performance of the model. The window size refers to the temporal duration of the video segments that the model processes at a time. A larger window size allows the model to consider more context and temporal information when making predictions, which can improve the performance of the model. On the other hand, a smaller window size limits the amount of context and temporal information that the model has access to, which can lead to a decrease in performance. This is because, in a video, the context, such as the background, the position of the player... is important for action spotting and understanding, reducing the window size, the model can lose some of this context, making it harder to identify the action. Also, the temporal information is important in order to understand the dynamics of the action, reducing the window size, can make the model miss some of the frames that compose the action.

4.4.1. CALF - Chunk size

In this case, we have increased and reduced the chunk size in order to see how this affected the performance. As it was expected in both executions, training and testing, we are able to see that the proposed baseline by the authors[3], chunk size = 120 s, is the correct one, as in overall it is giving the highest performance.

- **Training**

METRIC	Chunk size: 60s	Chunk size: 90 s	Baseline - C. size: 120 s	Chunk size: 180 s	Chunk size: 240 s
Average mAP visibility all	0.387	0.402	**0.418**	0.293	0.207
Average mAP visibility visible	0.403	0.419	**0.447**	0.304	0.226
Average mAP visibility unshown	0.254	0.293	**0.334**	0.213	0.159
Execution time	58633.26 s	41269.35 s	**27452.31 s**	66767.63 s	85134.41 s

Table 5: Performance of chunk size modification obtained by us from executing the training set in CALF[3].

- **Test**

METRIC	Chunk size: 60s	Chunk size: 90 s	Baseline - C. size: 120 s	Chunk size: 180 s	Chunk size: 240 s
Average mAP visibility all	0.309	0.321	**0.380**	0.235	0.187
Average mAP visibility visible	0.350	0.367	**0.398**	0.272	0.206
Average mAP visibility unshown	0.213	0.247	**0.307**	0.199	0.143
Execution time	1087.45 s	832.21 s	**560.62 s**	904.15 s	1254.58 s

Table 6: Performance of chunk size modification obtained by us from executing the test set in CALF[3].

What we wanted to note, as it was quite surprising, increasing the chunk size gives worse performance than decreasing it. We believe that the reason for this is the model is generalizing better than increasing the size, as a smaller window size can reduce the risk of overfitting, because the model will not be able to rely on irrelevant information from outside the window. Also, a smaller window size can help the model adapt to changes in the underlying distribution of the data.

4.4.2. NetVLAD++ - Window size

As it was expected, the most optimal value for the window has been T = 15, even though specific action classes could benefit from different window's size. The authors[4] consider that taking different temporal window per class could lead to optimal results, yet impractical in their architecture that consider a single window to pool features from.

- **Training**

METRIC	Window size: 05 s	Window size: 10 s	Baseline - W. size: 15 s	Window size: 20 s	Window size: 25 s	Window size: 30 s
Average mAP visibility all	0.502	0.583	**0.620**	0.608	0.578	0.547
Average mAP visibility visible	0.593	0.607	0.651	**0.667**	0.622	0.584
Average mAP visibility unshown	0.359	**0.392**	0.376	0.335	0.319	0.307
Execution time	3390.34 s	2782.66 s	2384.36 s	2025.76 s	1832.21 s	**1543.89 s**

Table 7: Performance of window size modification obtained by us from executing the training set in NetVLAD++[4].

- **Test**

METRIC	Window size: 05 s	Window size: 10 s	Baseline - W. size: 15 s	Window size: 20 s	Window size: 25 s	Window size: 30 s
Average mAP visibility all	0.467	0.515	**0.526**	0.519	0.503	0.493
Average mAP visibility visible	0.538	0.577	0.587	**0.591**	0.549	0.520
Average mAP visibility unshown	0.291	**0.352**	0.342	0.321	0.306	0.297
Execution time	**213.77 s**	311.46 s	426.11 s	579.82 s	764.96 s	988.43 s

Table 8: Performance of window size modification obtained by us from executing the test set in NetVLAD++[4].

Analyzing the values, we found that there are several points quite interesting to point. First, see that the best performance for 'Average mAP visibility visible' is obtained with 'window size = 20 s', while the 'Average mAP visibility unshown' is with 'window size = 10 s' . However, looking at the overall performance and taking into account also the execution time the best is obtained with a 'window size = 15 s', as the authors[4] proposed. Related to the execution time, it is worth to mention that the bigger the window size, the bigger the execution time.

Finally, comment that, increasing the window gives a better performance than decreasing it. As explained before, this is because larger window size allows the model to consider more context and temporal information when making predictions, and in some cases it can improve the performance.

4.5. Qualitative evaluation

Analyzing the results obtained, we realized that it was very interesting to add a qualitative evaluation to accompany the performance numbers.

What we have done is to compare the JSON files produced by the models with the detections with their respective ground truth JSON files. These kinds of files are typically used to store the output of the models in a structured format. The main advantage of using JSON (JavaScript Object Notation) format is that it can be easily read and understood by both humans and machines.

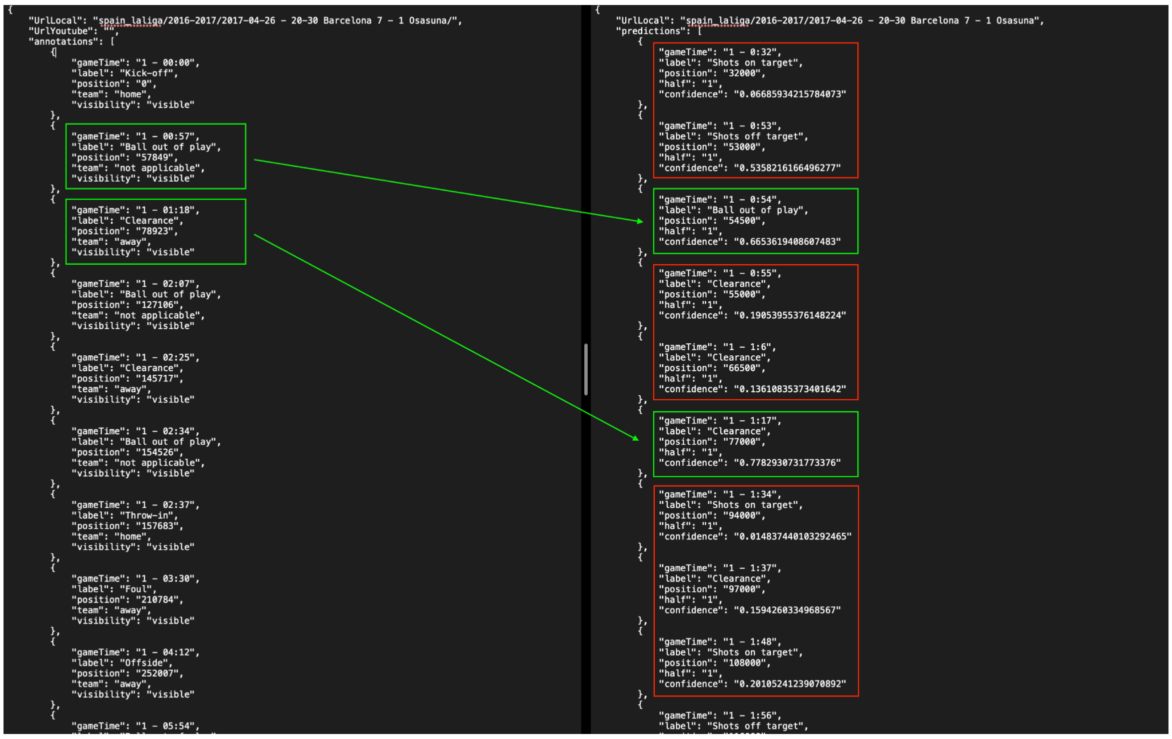
As the models are used for action spotting in soccer videos, the JSON files produced by them with the detections contain information about the action time and half in the match, the label action, the position, along with a confidence score for each detection. While, the annotations file also contains the team and visibility, and, as expected, does not contain the confidence score.

In order to easily exemplify the qualitative evaluation made with different matches, in this part we are going to focus on only one match: "Barcelona 7 - 1 Osasuna", from "Spain La Liga", on the 26th of April 2017 at 20:30h.

- **CALF**

In this model[3], the annotations (ground truth) and also the detections are shown in the time order of their occurrence in the match. So as you go through the document, you can follow what happened in the match. The left file is the annotations file, while the right is the one containing the detections.

The green squares are the correct detections (TP) with its annotation, as you can see, in Figure 4, for all of them the confidence is quite high. On the contrary, the red ones are the wrong detections (FP) which used to have really low confidence, also shown in Figure 4.

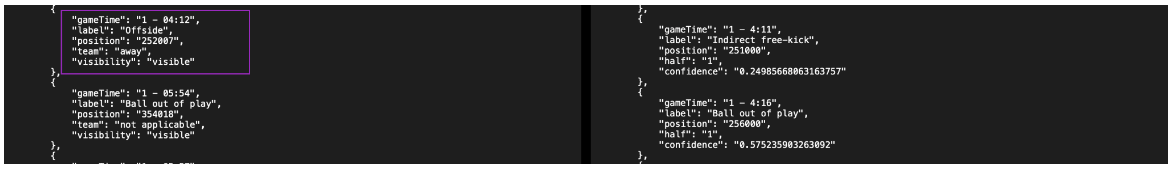


```
{
    "UrlLocal": "spain_laliga/2016-2017/2017-04-26 - 20-30 Barcelona 7 - 1 Osasuna/",
    "UrlYoutube": "",
    "annotations": [
        {
            "gameTime": "1 - 00:00",
            "label": "Kick-off",
            "position": "0",
            "team": "home",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 00:57",
            "label": "Ball out of play",
            "position": "57849",
            "team": "not applicable",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 01:18",
            "label": "Clearance",
            "position": "78923",
            "team": "away",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 02:07",
            "label": "Ball out of play",
            "position": "127106",
            "team": "not applicable",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 02:25",
            "label": "Clearance",
            "position": "145717",
            "team": "away",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 02:34",
            "label": "Ball out of play",
            "position": "154526",
            "team": "not applicable",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 02:37",
            "label": "Throw-in",
            "position": "157683",
            "team": "home",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 03:30",
            "label": "Foul",
            "position": "210784",
            "team": "away",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 04:12",
            "label": "Offside",
            "position": "252007",
            "team": "away",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 05:54",
```

```
{
    "UrlLocal": "spain_laliga/2016-2017/2017-04-26 - 20-30 Barcelona 7 - 1 Osasuna",
    "predictions": [
        {
            "gameTime": "1 - 0:32",
            "label": "Shots on target",
            "position": "32000",
            "half": "1",
            "confidence": "0.06685934215784073"
        },
        {
            "gameTime": "1 - 0:53",
            "label": "Shots off target",
            "position": "53000",
            "half": "1",
            "confidence": "0.5358216166496277"
        },
        {
            "gameTime": "1 - 0:54",
            "label": "Ball out of play",
            "position": "54500",
            "half": "1",
            "confidence": "0.6653619408607483"
        },
        {
            "gameTime": "1 - 0:55",
            "label": "Clearance",
            "position": "55000",
            "half": "1",
            "confidence": "0.19053955376148224"
        },
        {
            "gameTime": "1 - 1:6",
            "label": "Clearance",
            "position": "66500",
            "half": "1",
            "confidence": "0.13610835373401642"
        },
        {
            "gameTime": "1 - 1:17",
            "label": "Clearance",
            "position": "77000",
            "half": "1",
            "confidence": "0.7782930731773376"
        },
        {
            "gameTime": "1 - 1:34",
            "label": "Shots on target",
            "position": "94000",
            "half": "1",
            "confidence": "0.014837440103292465"
        },
        {
            "gameTime": "1 - 1:37",
            "label": "Clearance",
            "position": "97000",
            "half": "1",
            "confidence": "0.1594260334968567"
        },
        {
            "gameTime": "1 - 1:48",
            "label": "Shots on target",
            "position": "108000",
            "half": "1",
            "confidence": "0.20105241239070892"
        },
        {
            "gameTime": "1 - 1:56",
            "label": "Shots off target",
```

Figure 4: Annotations (left) and CALF[3] correct and wrong detections (right) of "Barcelona 7 - 1 Osasuna" match.

In the Figure 5, there is a purple square which is showing a missed detection, 'Offside. Compared to the image of the match below, Figure 6, we understand that the reason is the model is being confused with another label, 'Free-kick', but with low confidence, the confusion here quite is understandable as the ball is at the penalty spot.



```
        {
            "gameTime": "1 - 04:12",
            "label": "Offside",
            "position": "252007",
            "team": "away",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 05:54",
            "label": "Ball out of play",
            "position": "354018",
            "team": "not applicable",
            "visibility": "visible"
        },
        {
```

```
        },
        {
            "gameTime": "1 - 4:11",
            "label": "Indirect free-kick",
            "position": "251000",
            "half": "1",
            "confidence": "0.24985668063163757"
        },
        {
            "gameTime": "1 - 4:16",
            "label": "Ball out of play",
            "position": "256000",
            "half": "1",
            "confidence": "0.575235903263092"
        },
```

Figure 5: Missed annotation - Offside - (left) and CALF[3] detections (right) of "Barcelona 7 - 1 Osasuna" match.

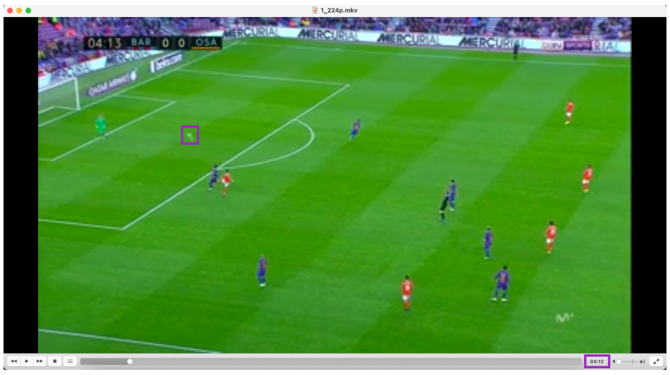
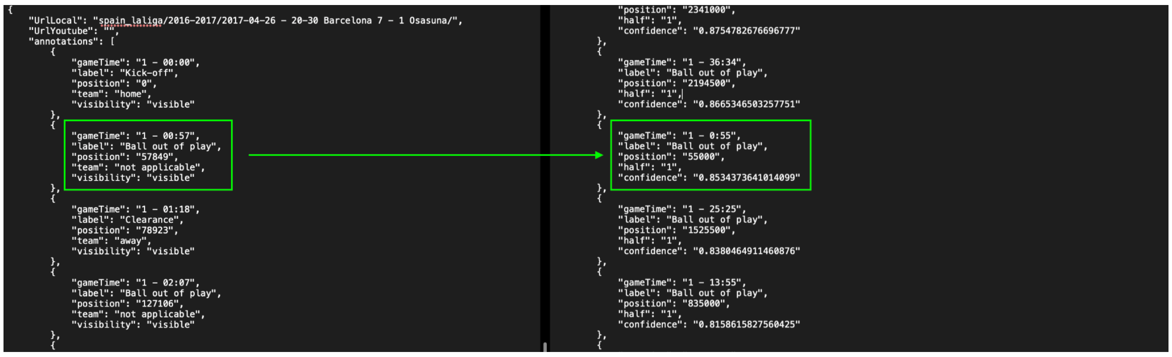


Figure 6: "Barcelona 7 - 1 Osasuna" match at 'gameTime = 04:12'.

- **NetVLAD++**

The main difference in these files has been that the detections follow a different order. Instead of following the match actions order, as in CALF[3], it follows a class order but is divided into two parts of the match. That is, it shows all the actions of a class, and then all the actions of the next class, and so on until all the classes have been shown.

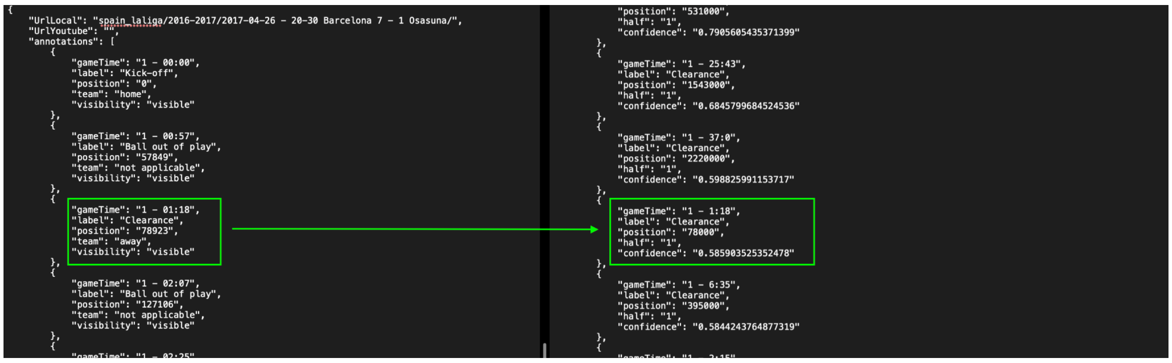
Regarding the color squares and the position of the files in the screenshot, we have followed the same format. Green for the correct detections, Figure 7 and Figure 8, red for the wrong ones, Figure 9 and Figure 10, and purple, Figure 11, for the missed, and also in the left the annotations file and in the right the detections file.



```
{
    "UrlLocal": "spain_laliga/2016-2017/2017-04-26 - 20-30 Barcelona 7 - 1 Osasuna/",
    "UrlYoutube": "",
    "annotations": [
        {
            "gameTime": "1 - 00:00",
            "label": "Kick-off",
            "position": "0",
            "team": "home",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 00:57",
            "label": "Ball out of play",
            "position": "57849",
            "team": "not applicable",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 01:18",
            "label": "Clearance",
            "position": "78923",
            "team": "away",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 02:07",
            "label": "Ball out of play",
            "position": "127106",
            "team": "not applicable",
            "visibility": "visible"
        },
        {
```

```
            "position": "2341000",
            "half": "1",
            "confidence": "0.8754782676696777"
        },
        {
            "gameTime": "1 - 36:34",
            "label": "Ball out of play",
            "position": "2194500",
            "half": "1",
            "confidence": "0.8665346503257751"
        },
        {
            "gameTime": "1 - 0:55",
            "label": "Ball out of play",
            "position": "55000",
            "half": "1",
            "confidence": "0.8534373641014099"
        },
        {
            "gameTime": "1 - 25:25",
            "label": "Ball out of play",
            "position": "1525500",
            "half": "1",
            "confidence": "0.8380464911460876"
        },
        {
            "gameTime": "1 - 13:55",
            "label": "Ball out of play",
            "position": "835000",
            "half": "1",
            "confidence": "0.8158615827560425"
        },
        {
```

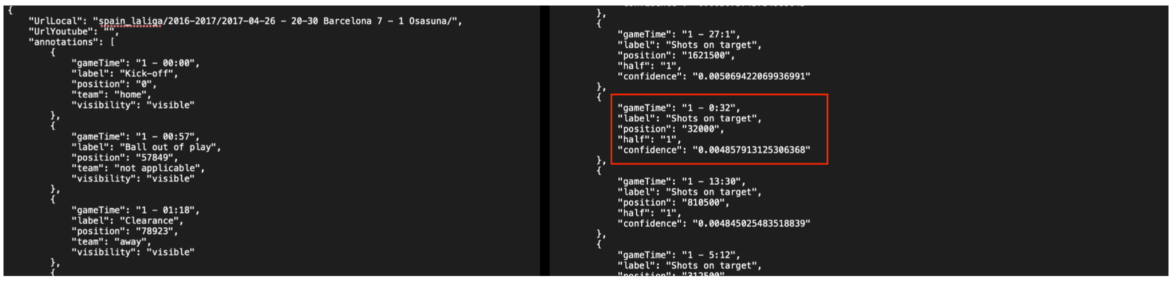
Figure 7: Annotations (left) and NetVLAD+ [4] correct detection - 'Ball out of play' - (right) of "Barcelona 7 - 1 Osasuna" match.



```
{
    "UrlLocal": "spain_laliga/2016-2017/2017-04-26 - 20-30 Barcelona 7 - 1 Osasuna/",
    "UrlYoutube": "",
    "annotations": [
        {
            "gameTime": "1 - 00:00",
            "label": "Kick-off",
            "position": "0",
            "team": "home",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 00:57",
            "label": "Ball out of play",
            "position": "57849",
            "team": "not applicable",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 01:18",
            "label": "Clearance",
            "position": "78923",
            "team": "away",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 02:07",
            "label": "Ball out of play",
            "position": "127106",
            "team": "not applicable",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 02:25"
```

```
            "position": "531000",
            "half": "1",
            "confidence": "0.7905605435371399"
        },
        {
            "gameTime": "1 - 25:43",
            "label": "Clearance",
            "position": "1543000",
            "half": "1",
            "confidence": "0.6845799684524536"
        },
        {
            "gameTime": "1 - 37:0",
            "label": "Clearance",
            "position": "2220000",
            "half": "1",
            "confidence": "0.598825991153717"
        },
        {
            "gameTime": "1 - 1:18",
            "label": "Clearance",
            "position": "78000",
            "half": "1",
            "confidence": "0.585903525352478"
        },
        {
            "gameTime": "1 - 6:35",
            "label": "Clearance",
            "position": "395000",
            "half": "1",
            "confidence": "0.5844243764877319"
        },
        {
            "gameTime": "1 - 2:15"
```

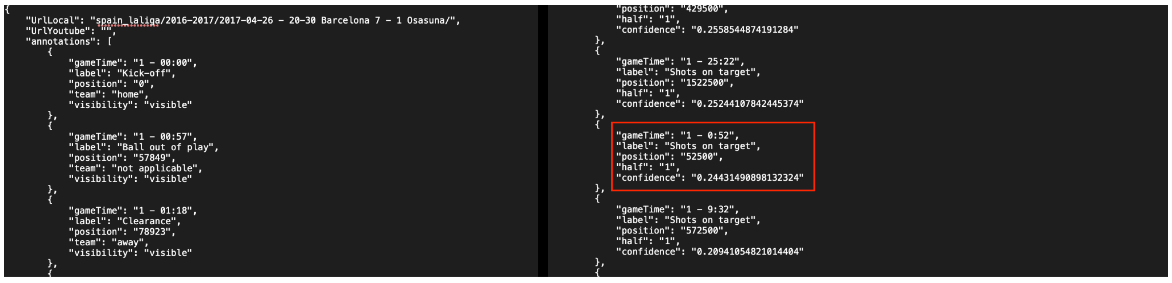
Figure 8: Annotations (left) and NetVLAD++[4] correct detection - 'Clearance' - (right) of "Barcelona 7 - 1 Osasuna" match.



```
{
    "UrlLocal": "spain_laliga/2016-2017/2017-04-26 - 20-30 Barcelona 7 - 1 Osasuna/",
    "UrlYoutube": "",
    "annotations": [
        {
            "gameTime": "1 - 00:00",
            "label": "Kick-off",
            "position": "0",
            "team": "home",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 00:57",
            "label": "Ball out of play",
            "position": "57849",
            "team": "not applicable",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 01:18",
            "label": "Clearance",
            "position": "78923",
            "team": "away",
            "visibility": "visible"
        },
        {
```

```
        },
        {
            "gameTime": "1 - 27:1",
            "label": "Shots on target",
            "position": "1621500",
            "half": "1",
            "confidence": "0.005069422069936991"
        },
        {
            "gameTime": "1 - 0:32",
            "label": "Shots on target",
            "position": "32000",
            "half": "1",
            "confidence": "0.004857913125306368"
        },
        {
            "gameTime": "1 - 13:30",
            "label": "Shots on target",
            "position": "810500",
            "half": "1",
            "confidence": "0.004845025483518839"
        },
        {
            "gameTime": "1 - 5:12",
            "label": "Shots on target",
            "position": "312500"
```

Figure 9: Annotations (left) and NetVLAD++[4] wrong detection - 'Shots on target' - (right) of "Barcelona 7 - 1 Osasuna" match.

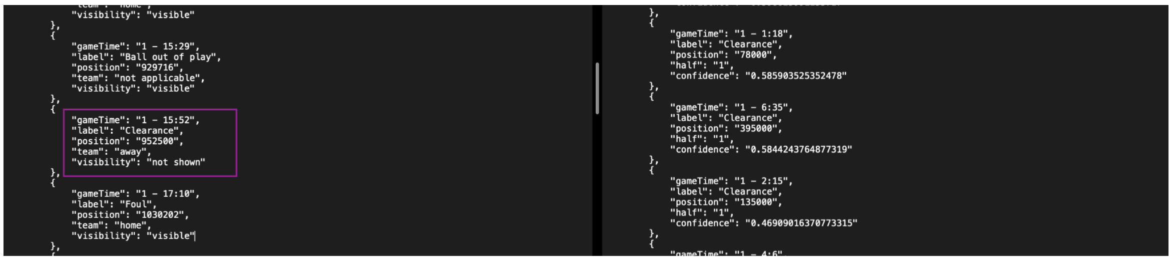


```
{
    "UrlLocal": "spain_laliga/2016-2017/2017-04-26 - 20-30 Barcelona 7 - 1 Osasuna/",
    "UrlYoutube": "",
    "annotations": [
        {
            "gameTime": "1 - 00:00",
            "label": "Kick-off",
            "position": "0",
            "team": "home",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 00:57",
            "label": "Ball out of play",
            "position": "57849",
            "team": "not applicable",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 01:18",
            "label": "Clearance",
            "position": "78923",
            "team": "away",
            "visibility": "visible"
        },
        {
```

```
            "position": "429500",
            "half": "1",
            "confidence": "0.2558544874191284"
        },
        {
            "gameTime": "1 - 25:22",
            "label": "Shots on target",
            "position": "1522500",
            "half": "1",
            "confidence": "0.25244107842445374"
        },
        {
            "gameTime": "1 - 0:52",
            "label": "Shots on target",
            "position": "52500",
            "half": "1",
            "confidence": "0.24431490898132324"
        },
        {
            "gameTime": "1 - 9:32",
            "label": "Shots on target",
            "position": "572500",
            "half": "1",
            "confidence": "0.20941054821014404"
        },
        {
```

Figure 10: Annotations (left) and NetVLAD++[4] wrong detection - 'Shots on target' - (right) of "Barcelona 7 - 1 Osasuna" match.

Here in the Figure 11 you can see one action missed for the NetVLAD++[4] model, it is a 'Clearance'. We have looked through all the 'Clearance' actions in the JSON annotations document and there were not any 'Clearance' about this time.



```
            "team": "home",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 15:29",
            "label": "Ball out of play",
            "position": "929716",
            "team": "not applicable",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 15:52",
            "label": "Clearance",
            "position": "952500",
            "team": "away",
            "visibility": "not shown"
        },
        {
            "gameTime": "1 - 17:10",
            "label": "Foul",
            "position": "1030202",
            "team": "home",
            "visibility": "visible"
        },
```

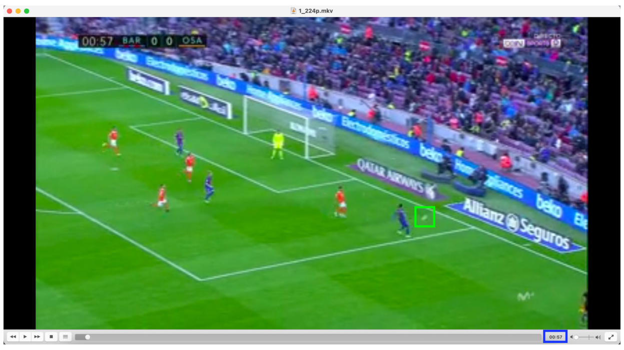
```
        },
        {
            "gameTime": "1 - 1:18",
            "label": "Clearance",
            "position": "78000",
            "half": "1",
            "confidence": "0.585903525352478"
        },
        {
            "gameTime": "1 - 6:35",
            "label": "Clearance",
            "position": "395000",
            "half": "1",
            "confidence": "0.5844243764877319"
        },
        {
            "gameTime": "1 - 2:15",
            "label": "Clearance",
            "position": "135000",
            "half": "1",
            "confidence": "0.46909016370773315"
        },
        {
            "gameTime": "1 - 4:6",
```

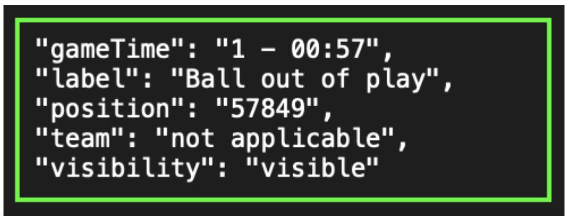
Figure 11: Missed annotation - Clearance - (left) and NetVLAD++[4] detections (right) of "Barcelona 7 - 1 Osasuna" match

For NetVLAD++[4] we can also determine that correct detections have high confidence, while incorrect detections have low confidence. However, it is important to note that the wrong detections have a lower accuracy and the correct ones fit in time better than the correct ones in the CALF[3] model.

- **Comparisons with images**

In the Figure 12 we can see the match at 'gameTime = 00:57', when we can see the ball out of the play as the annotation in Figure 13 says.



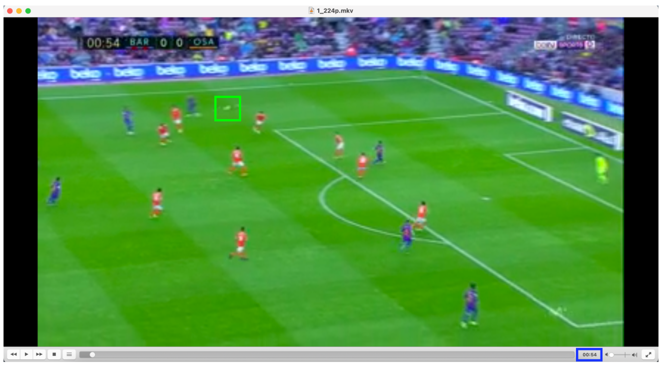


```
"gameTime": "1 - 00:57",
"label": "Ball out of play",
"position": "57849",
"team": "not applicable",
"visibility": "visible"
```

Figure 13: Correct detection - Ball out of play - in CALF[3].

Figure 12: "Barcelona 7 - 1 Osasuna" match at 'gameTime = 00:57'.

However, the annotations, in Figure 15, for CALF[3] found this 'Ball out of play' at a different time, concretely at 'gameTime = 00:54', when the match is not at that point yet, as we can see in the Figure 14.



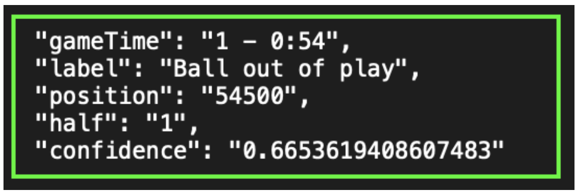
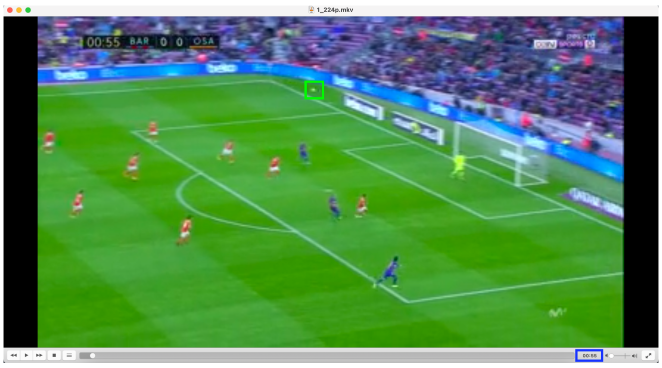

```
"gameTime": "1 - 0:54",
"label": "Ball out of play",
"position": "54500",
"half": "1",
"confidence": "0.6653619408607483"
```

Figure 15: Correct detection - Ball out of play - in CALF[3].

Figure 14: "Barcelona 7 - 1 Osasuna" match at 'gameTime = 00:54'.

While in the NetVLAD++[4] detects it at 'gameTime = 00:55', Figure 17, when the image is almost the same as in CALF[3], Figure 16.



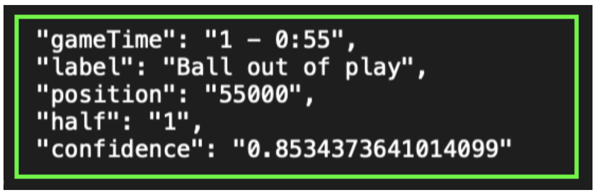
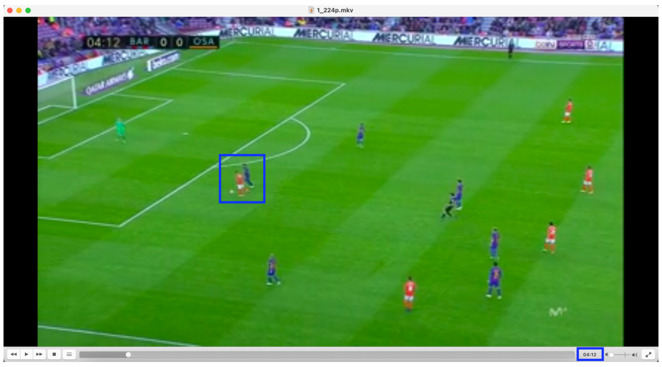
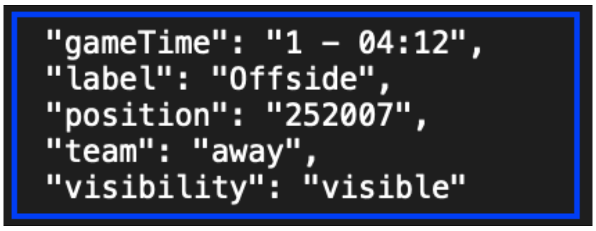

```
"gameTime": "1 - 0:55",
"label": "Ball out of play",
"position": "55000",
"half": "1",
"confidence": "0.8534373641014099"
```

Figure 17: Correct detection - Ball out of play - in NetVLAD++ [4].

Figure 16: "Barcelona 7 - 1 Osasuna" match at 'gameTime = 00:55'.

In 'gameTime = 04:12' there is an 'Offside', Figure 19, as it can be seen in the annotations file. Looking at Figure 18, it can be determined that it is a really difficult action to detect, and it coincides with the one missed for the CALF[3] model, explained in a previous point, Figure 5 and Figure 6.



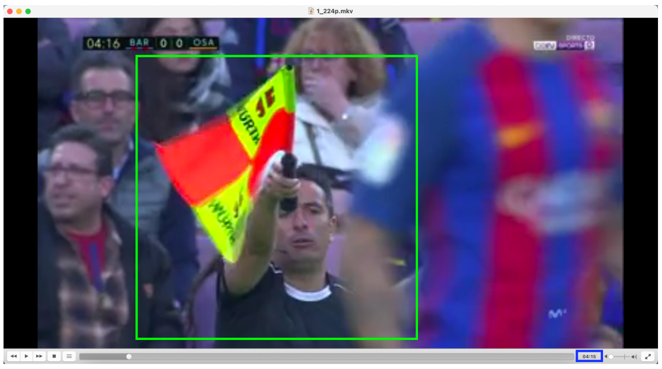


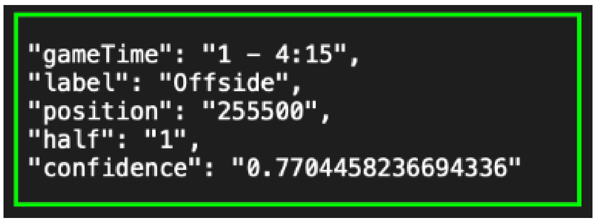
```
"gameTime": "1 – 04:12",
"label": "Offside",
"position": "252007",
"team": "away",
"visibility": "visible"
```

Figure 19: Missed detection - Offside - in CALF[3]

Figure 18: "Barcelona 7 - 1 Osasuna" match at 'gameTime = 04:12'.

However, this same action is detected by the NetVLAD++[4]model with a quite high confidence, Figure 21. Being detected at 'gameTime = 04:15', when the match is as seen in Figure 20.





```
"gameTime": "1 – 4:15",
"label": "Offside",
"position": "255500",
"half": "1",
"confidence": "0.7704458236694336"
```

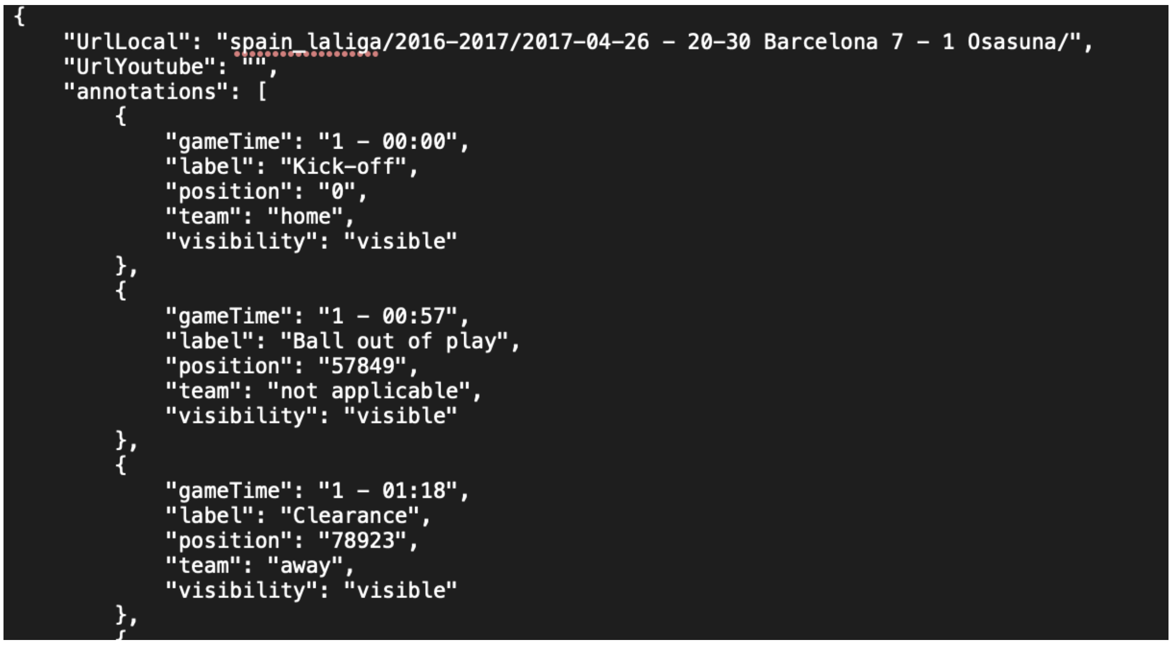
Figure 21: Correct detection - Offside - in NetVLAD++ [4].

Figure 20: "Barcelona 7 - 1 Osasuna" match at 'gameTime = 04:15'.

The detection is in a different moment as the model benefits from the temporally-aware feature pooling for detecting it because of the posterior action, which is the referee raising the flag. This is a clear example of why NetVLAD++[4] overcomes CALF[3].

Finally, in the following figures, will find some examples where the models fail and detect wrong actions.

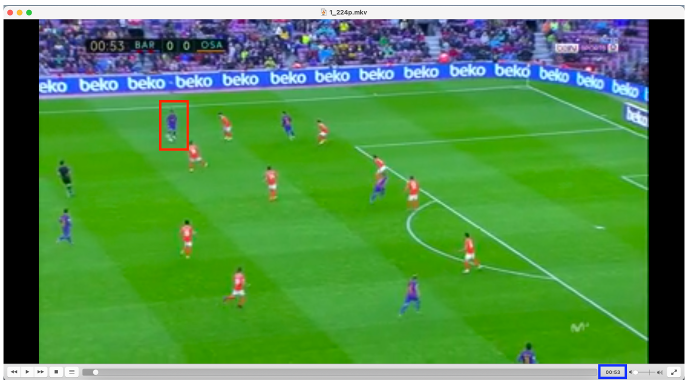
In Figure 22 you can see the annotations of the match and that there is no 'Shots on target' or 'Shots off target' between 'gameTime = 00:00' and 'gameTime = 01:18'.

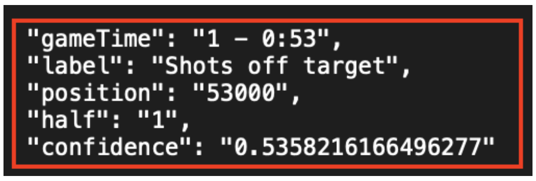


```
{
    "UrlLocal": "spain_laliga/2016-2017/2017-04-26 - 20-30 Barcelona 7 - 1 Osasuna/",
    "UrlYoutube": "",
    "annotations": [
        {
            "gameTime": "1 - 00:00",
            "label": "Kick-off",
            "position": "0",
            "team": "home",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 00:57",
            "label": "Ball out of play",
            "position": "57849",
            "team": "not applicable",
            "visibility": "visible"
        },
        {
            "gameTime": "1 - 01:18",
            "label": "Clearance",
            "position": "78923",
            "team": "away",
            "visibility": "visible"
        },
```

Figure 22: Annotations' file of time between 'gameTime = 00:00' and 'gameTime = 01:18'.

While CALF[3] model at 'gameTime = 00:53', Figure 24, detects a 'Shots off target' with a fairly high confidence of 0.53, which we see in Figure 23, it is a clear wrong detection of the model.





```
"gameTime": "1 - 0:53",
"label": "Shots off target",
"position": "53000",
"half": "1",
"confidence": "0.535821616496277"
```

Figure 24: Wrong detection - Shots off target - in CALF[3].

Figure 23: "Barcelona 7 - 1 Osasuna" match at 'gameTime = 00:53'.

But it is not the only one that is wrong, as the NetVLAD++[4] model also detects an incorrect action at this time. Specifically, it detects 'Shots on target' at 'gameTime = 00:52', see the match at that moment in Figure 25, and with a confidence of 0.24, see Figure 26, which is lower than CALF[3] but not low enough.

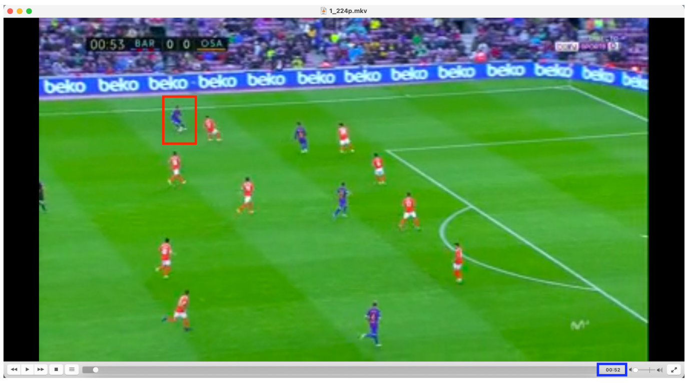
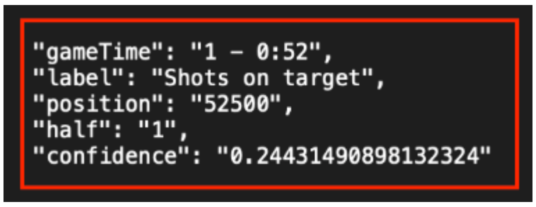


Figure 25: “Barcelona 7 - 1 Osasuna” match at ‘gameTime = 00:52’.



```
"gameTime": "1 – 0:52",
"label": "Shots on target",
"position": "52500",
"half": "1",
"confidence": "0.24431490898132324"
```

Figure 26: Wrong detection - Shots on target - in NetVLAD++[4].

The reason for these errors is that the match is in a moment where there are no very decisive actions and shots, passes or kicks off that can easily be confused and lead to mistakes.

5. Budget

This project is about software learning with no cost on any hardware or physical components but the computer where we have worked on. We have used, at no cost, the resources available at the Image Processing Group (IPG) at UPC. But in order to approximately know the computational cost of using it we have based on a conducted test with DNNs, PyTorch, and 1 GPU, on the common CIFAR10 Benchmark[44].

We have extrapolated the numbers computed for the CIFAR10 Benchmark[44] to this project case. In order to calculate the pricing of GPU usage, we have taken as a reference the Amazon EC2 on-spot instance pricing. Using the instance with one GPU costs €0.85 per hour. If we assume that we have run the executions 22 times, one for each model and modification, and also for training and testing. Having a duration of 85h 38 m in total, being 83.65 h, then its cost is:

*83.65 h * €0.85 = €71.10*

In addition, we have calculated the price of renting a coworking space, which includes the indirect costs of light, water and heat. We have taken into account a price of €4/h[1] for 192h:

*192 h * €4 = €768*

Finally, we must include the working hours we have spent during the semester. They have been approximately 1 hour per week in the case of the supervisor and 8 hours per week in the case of the junior engineer, during 24 weeks. If we consider a salary of €60/h for the supervisor and €20/h for a junior engineer, we can conclude the cost of the hours spent has been:

*2 supervisors * 24 hours * €60/h = €2880*

*1 junior engineer * 192 hours * €20/h = €3840*

GPU cost:	€71.10
Renting a coworking space:	€768
Supervisor Salary:	€2880
Engineer Salary:	€3840
TOTAL (€) :	**€7559.10**

Table 9: Breakdown of the project costs.

[1] Based on the prices got from: https://wegalavant.com/2021/12/17/8-best-coworking-cafes-in-barcelona/

6. Environment Impact

This project has no direct environmental impact, neither positive nor negative. However, We believe it is important to mention what implies environmentally and socially the realization of this project.

On the environmental part, we want to reference the GPU consumption. As in the budget part, we have based on the same conducted test. With the same extrapolation done before of the numbers computed for the CIFAR10 Benchmark[44] to this project case, we have concluded that training and testing the system would take 85h 38 m in total.

Regarding energy efficiency, we want to mention that training and automatically configuring DNNs can be an extremely consuming task in terms of GPU computation power. In our case, as we said before, we have used the GPU offered by the Image Processing Group, which is a P100 GPU. This single GPU consumes 300 Watts, which means a total consumption of 76800 Watts per hour, according to the CIFAR10 Benchmark[44] sequences parameters. In this study, we have consumed 6424.32 kW.

From a social point of view, we strongly believe it is every time more important and necessary to pay attention and be aware of the impact that a project causes. To make this report, we have based ourselves on the sustainable development goals determined by the UN[45] that this project achieves.

Firstly, quality education. This project shows the importance of access to quality public education, to be able to reach a good level of knowledge, have the possibility to obtain a job, and escape from poverty.

Another very important goal is to achieve gender equality and empower all women and girls. Gender equality is necessary for a prosperous, equal, and sustainable world. It is true that every time there are more women serving in leadership positions, but it is very important to continue this fight for our rights. We think that by doing this technological project by me, a woman, is a demonstration of every time we are more, and we are everywhere with no discrimination by our gender.

And last but not least, build resilient infrastructure, promote sustainable industrialization, and foster innovation. Making technological projects about innovative methods promotes interest in this field and can unleash competitive and economic forces in the future.

7. Conclusions and future development

For now, we can claim the achievement of the main objectives proposed, specifically the research, execution, and evaluation of both models proposed and also the evaluation of modifications to them.

We have verified the results of each model, and we have compared them to be able to determine that the NetVLAD++[4] is a better model for action spotting in soccer videos, giving a 27.76% better performance than CALF[3] model in 'Average mAP visible all' metric. In addition, after a deep analysis, we have made one modification to each model, and we have concluded that for both models[3][4] changing this hyperparameter worsens the performance and as it was expected the best value for it is the one proposed by the authors. But the most interesting thing has been that in CALF[3], increasing the chunk size gives a worse performance than decreasing it, and on the other hand, in NetVLAD++[4], it is better to increase it before decrease it.
Apart from that quantitative analysis we have also done a qualitative analysis which has helped us to check that NetVLAD++[4] model makes better detections, having low accuracies for the wrong detections, high for the correct ones and getting closer to the correct annotation time.

Finally, considering future implementations, what we would like to do was to make some other modifications, apart from changing the window size, such as changing the number of features or looking only to previous context or only to the posterior to see if one is more valuable than the other. However, the most interesting improvement would be to train and test both models with a different dataset to see how they perform with a dataset different from the one they are specialized in, for example a dataset with basketball matches instead of soccer.

Bibliography

[1] Deliege, A., Cioppa, A., Giancola, S., Seikavandi, M. J., Dueholm, J. V., Nasrollahi, K., Ghanem, B., Moeslund, T. B., & Van Droogenbroeck, M. (2021). SoccerNet-v2: A Dataset and Benchmarks for Holistic Understanding of Broadcast Soccer Videos. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

[2] Giancola, S., Amine, M., Dghaily, T., & Ghanem, B. (2018). SoccerNet: A Scalable Dataset for Action Spotting in Soccer Videos. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

[3] Cioppa, A., Deliege, A., Giancola, S., Ghanem, B., Van Droogenbroeck, M., Gade, R., & Moeslund, T. B. (2020). A Context-Aware Loss Function for Action Spotting in Soccer Videos. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[4] Giancola, S., & Ghanem, B. (2021). Temporally-Aware Feature Pooling for Action Spotting in Soccer Broadcasts. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

[5] Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., & Sivic, J. (2016a). NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[6] Arandjelovic, R., & Zisserman, A. (2013). All About VLAD. *2013 IEEE Conference on Computer Vision and Pattern Recognition*.

[7] Koppula, H. S., Gupta, R., & Saxena, A. (2013). Learning human activities and object affordances from RGB-D videos. *The International Journal of Robotics Research*, *32*(8), 951–970.

[8] Wang, J., Liu, Z., & Wu, Y. (2014). *Human Action Recognition with Depth Cameras*. Springer Publishing.

[9] Bingbing Ni, Yong Pei, Moulin, P., & Shuicheng Yan. (2013). Multilevel Depth and Image Fusion for Human Activity Detection. *IEEE Transactions on Cybernetics*, *43*(5), 1383–1394.

[10] Faria, Diego & Premebida, Cristiano & Nunes, Urbano. (2014). A Probalistic Approach for Human Everyday Activities Recognition using Body Motion from RGB-D Images. Proceedings - IEEE International Workshop on Robot and Human Interactive Communication. 2014.

[11] Gaglio, S., Re, G. L., & Morana, M. (2015). Human Activity Recognition Process Using 3-D Posture Data. *IEEE Transactions on Human-Machine Systems*, *45*(5), 586–597.

[12] Shan, J., & Akella, S. (2014). 3D human action segmentation and recognition using pose kinetic energy. *2014 IEEE International Workshop on Advanced Robotics and Its Social Impacts*.

[13] Cippitelli, E., Gasparrini, S., Gambi, E., & Spinsante, S. (2016). A Human Activity Recognition System Using Skeleton Data from RGBD Sensors. *Computational Intelligence and Neuroscience*, *2016*, 1–14.

[14] Ballas, N., Yao, L., Pal, C., & Courville, A. (2016). Delving Deeper into Convolutional Networks for Learning Video Representations. *International Conference on Learning Representations*.

[15] Murad, A., & Pyun, J. Y. (2017). Deep Recurrent Neural Networks for Human Activity Recognition. *Sensors*, *17*(11), 2556.

[16] Zhang, J., Cao, Y., & Wu, Q. (2021). Vector of Locally and Adaptively Aggregated Descriptors for Image Feature Representation. *Pattern Recognition*, *116*, 107952.

[17] Zhang, N., Hu, Z., Lee, S., & Lee, E. (2018). Human Action Recognition Based on Global Silhouette and Local Optical Flow. *Proceedings of the International Symposium on Mechanical Engineering and Material Science (ISMEMS 2017)*.

[18] Zhao, R., Ali, H., & van der Smagt, P. (2017). Two-stream RNN/CNN for action recognition in 3D videos. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[19] Manzi, A., Dario, P., & Cavallo, F. (2017). A Human Activity Recognition System Based on Dynamic Clustering of Skeleton Data. *Sensors*, *17*(5), 1100.

[20] Xu, Z., Hu, J., & Deng, W. (2016). Recurrent convolutional neural network for video classification. *2016 IEEE International Conference on Multimedia and Expo (ICME)*.

[21] Baldominos, A., Saez, Y., & Isasi, P. (2018). Evolutionary Design of Convolutional Neural Networks for Human Activity Recognition in Sensor-Rich Environments. *Sensors*, *18*(4), 1288.

[22] Das, S., Koperski, M., Brémond, F., & Francesca, G. (2018). A Fusion of Appearance based CNNs and Temporal evolution of Skeleton with LSTM for Daily Living Action Recognition.

[23] Cruz-Silva, J. E., Montiel-Pérez, J. Y., & Sossa-Azuela, H. (2019). 3-D Human Body Posture Reconstruction by Computer Vision. *Advances in Soft Computing*, 579–588.

[24] Khaire, P., Kumar, P., & Imran, J. (2018). Combining CNN streams of RGB-D and skeletal data for human activity recognition. *Pattern Recognition Letters*, *115*, 107–116.

[25] Ji, Y., Xu, F., Yang, Y., Shen, F., Shen, H. T., & Zheng, W. S. (2018). A Large-scale RGB-D Database for Arbitrary-view Human Action Recognition. *Proceedings of the 26th ACM International Conference on Multimedia*.

[26] Wang, P., Li, W., Ogunbona, P., Wan, J., & Escalera, S. (2018). RGB-D-based human motion recognition with deep learning - A survey. *Comput. Vis. Image Underst.*, *171*, 118-139.

[27] Singh, S., Arora, C., & Jawahar, C. (2017). Trajectory aligned features for first person action recognition. *Pattern Recognition*, *62*, 45–55.

[28] Sigurdsson, G. A., Gupta, A., Schmid, C., Farhadi, A., & Alahari, K. (2018). Charades-Ego: A Large-Scale Dataset of Paired Third and First Person Videos. *ArXiv: Computer Vision and Pattern Recognition*.

[29] Muhamada, A. W., & Mohammed, A. A. (2022). Review on recent Computer Vision Methods for Human Action Recognition. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, *10*(4), 361–379.

[30] Sakaino, H. (2019). Spatio-Temporal Feature Extraction/Recognition in Videos Based on Energy Optimization. *IEEE Transactions on Image Processing*, *28*(7), 3395–3407.

[31] Hao, H., Wang, Y., Xia, Y., Zhao, J., & Shen, F. (2020). Temporal Convolutional Attention-based Network For Sequence Modeling. *Cornell University - ArXiv*.

[32] Mo, Y., Wu, Y., Yang, X., Liu, F., & Liao, Y. (2022). Review the state-of-the-art technologies of semantic segmentation based on deep learning. *Neurocomputing*, *493*, 626–646.

[33] H. Minoura et al., "Action Spotting and Temporal Attention Analysis in Soccer Videos," 2021 17th International Conference on Machine Vision and Applications (MVA), Aichi, Japan, 2021, pp. 1-6.

[34] Nguyen, P., Ramanan, D., & Fowlkes, C. (2019). Weakly-Supervised Action Localization With Background Modeling. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*.

[35] Alwassel, H., Caba Heilbron, F., & Ghanem, B. (2018). Action Search: Spotting Actions in Videos and Its Application to Temporal Action Localization. *Computer Vision – ECCV 2018*, 253–269.

[36] Vanderplaetse, B., & Dupont, S. (2020). Improved Soccer Action Spotting using both Audio and Video Streams. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

[37] Zhou, X., Kang, L., Cheng, Z., He, B., & Xin, J. (2021). Feature Combination Meets Attention: Baidu Soccer Embeddings and Transformer based Temporal Detection. *ArXiv: Computer Vision and Pattern Recognition*.

[38] H. Minoura et al., "Action Spotting and Temporal Attention Analysis in Soccer Videos," 2021 17th International Conference on Machine Vision and Applications (MVA), Aichi, Japan, 2021, pp. 1-6.

[39] Cartas, A., Ballester, C., & Haro, G. (2022). A Graph-Based Method for Soccer Action Spotting Using Unsupervised Player Classification. *Proceedings of the 5th International ACM Workshop on Multimedia Content Analysis in Sports*.

[40] Soares, J. V. B., Shah, A., & Biswas, T. (2022). Temporally Precise Action Spotting in Soccer Videos Using Dense Detection Anchors. *2022 IEEE International Conference on Image Processing (ICIP)*.

[41] F. Wu et al., "A Survey on Video Action Recognition in Sports: Datasets, Methods and Applications," in IEEE Transactions on Multimedia.

[42] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[43] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[44] *Stanford DAWN Deep Learning Benchmark (DAWNBench) · CIFAR10 Training*. (n.d.).

[45] United Nations. (n.d.). *United Nations | Peace, dignity and equality on a healthy planet*.

[46] Sharma, S. (2021, December 14). *Deep Learning Architectures for Action Recognition - Towards Data Science*. Medium.

[47] Lindeberg, T. (2012). Scale Invariant Feature Transform. *Scholarpedia*, *7*(5), 10491.

[48] Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*.

[49] Wen, Z. Q., & Cai, Z. X. (2006). Mean Shift Algorithm and its Application in Tracking of Objects. *2006 International Conference on Machine Learning and Cybernetics*.

[50] Carreira, J., & Zisserman, A. (2017). Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[51] Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning Spatiotemporal Features with 3D Convolutional Networks. *2015 IEEE International Conference on Computer Vision (ICCV)*.

Appendices

- **CALF - Training**

METRIC: Average mAP visibility all per class	**VALUE**
Class 1	0.236
Class 2	0.398
Class 3	0.722
Class 4	0.529
Class 5	0.295
Class 6	0.242
Class 7	0.283
Class 8	0.560
Class 9	0.662
Class 10	0.585
Class 11	0.528
Class 12	0.406
Class 13	0.418
Class 14	0.734
Class 15	0.420
Class 16	0.007
Class 17	0.080

Table 10: Average mAP visibility all per class performance of CALF[3] in training set obtained by us.

METRIC: Average mAP visibility visible per class	**VALUE**
Class 1	0.244
Class 2	0.341
Class 3	0.730
Class 4	0.623
Class 5	0.325
Class 6	0.244
Class 7	0.283
Class 8	0.629
Class 9	0.664
Class 10	0.616
Class 11	0.535
Class 12	0.402
Class 13	0.512
Class 14	0.744
Class 15	0.464
Class 16	0.139
Class 17	0.097

Table 11: Average mAP visibility visible per class performance of CALF[3] in training set obtained by us.

METRIC: Average mAP visibility unshown per class	**VALUE**
Class 1	0.0
Class 2	0.423
Class 3	0.0

Class 4	0.164
Class 5	0.110
Class 6	0.0
Class 7	0.545
Class 8	0.479
Class 9	0.618
Class 10	0.514
Class 11	0.274
Class 12	0.449
Class 13	0.046
Class 14	0.642
Class 15	0.084
Class 16	0.0
Class 17	0.0

Table 12: Average mAP visibility unshown per class performance of CALF[3] in training set obtained by us.

- **CALF - Test**

METRIC: Average mAP visibility all per class	**VALUE**
Class 1	0.310
Class 2	0.365
Class 3	0.654
Class 4	0.463
Class 5	0.223
Class 6	0.220
Class 7	0.213

Class 8	0.507
Class 9	0.652
Class 10	0.563
Class 11	0.489
Class 12	0.391
Class 13	0.389
Class 14	0.636
Class 15	0.380
Class 16	0.003
Class 17	0.0

Table 13: Average mAP visibility all per class performance of CALF[3] in test set obtained by us.

METRIC: Average mAP visibility visible per class	**VALUE**
Class 1	0.316
Class 2	0.279
Class 3	0.663
Class 4	0.573
Class 5	0.276
Class 6	0.221
Class 7	0.213
Class 8	0.568
Class 9	0.654
Class 10	0.583
Class 11	0.490

Class 12	0.373
Class 13	0.431
Class 14	0.696
Class 15	0.414
Class 16	0.019
Class 17	0.0

Table 14: Average mAP visibility visible all per class performance of CALF[3] in test set obtained by us.

METRIC: Average mAP unshown per class	**VALUE**
Class 1	0.0
Class 2	0.395
Class 3	0.0
Class 4	0.101
Class 5	0.044
Class 6	0.0
Class 7	0.545
Class 8	0.452
Class 9	0.612
Class 10	0.485
Class 11	0.320
Class 12	0.420
Class 13	0.087
Class 14	0.454
Class 15	0.077

Class 16	0.0
Class 17	0.0

Table 15: Average mAP visibility unshown per class performance of CALF[3] in test set obtained by us.

- **TEMPORALLY AWARE - Training**

METRIC: Average mAP visibility all per class	**VALUE**
Class 1	0.741
Class 2	0.581
Class 3	0.732
Class 4	0.690
Class 5	0.357
Class 6	0.386
Class 7	0.398
Class 8	0.571
Class 9	0.700
Class 10	0.687
Class 11	0.642
Class 12	0.438
Class 13	0.585
Class 14	0.796
Class 15	0.547
Class 16	0.041
Class 17	0.052

Table 16: Average mAP visibility all per class performance of NetVLAD++[4] in training set obtained by us.

METRIC: Average mAP visibility visible per class	**VALUE**
Class 1	0.819
Class 2	0.718
Class 3	0.740
Class 4	0.756
Class 5	0.390
Class 6	0.388
Class 7	0.401
Class 8	0.683
Class 9	0.704
Class 10	0.720
Class 11	0.648
Class 12	0.401
Class 13	0.660
Class 14	0.803
Class 15	0.592
Class 16	0.307
Class 17	0.252

Table 17: Average mAP visibility visible per class performance of NetVLAD++[4] in training set obtained by us.

METRIC: Average mAP visibility unshown per class	**VALUE**
Class 1	0.0
Class 2	0.513
Class 3	0.0

Class 4	0.321
Class 5	0.069
Class 6	0.043
Class 7	0.002
Class 8	0.458
Class 9	0.642
Class 10	0.606
Class 11	0.390
Class 12	0.503
Class 13	0.031
Class 14	0.739
Class 15	0.131
Class 16	0.0
Class 17	0.0

Table 18: Average mAP visibility unshown per class performance of NetVLAD++[4] in training set obtained by us.

- **NetVLAD++ - Test**

METRIC: Average mAP visibility all per class	**VALUE**
Class 1	0.741
Class 2	0.581
Class 3	0.732
Class 4	0.690
Class 5	0.357
Class 6	0.386
Class 7	0.398

Class 8	0.571
Class 9	0.700
Class 10	0.687
Class 11	0.642
Class 12	0.438
Class 13	0.585
Class 14	0.796
Class 15	0.547
Class 16	0.041
Class 17	0.052

Table 19: Average mAP visibility all per class performance of NetVLAD++[4] in test set obtained by us.

METRIC: Average mAP visibility visible per class	**VALUE**
Class 1	0.819
Class 2	0.718
Class 3	0.740
Class 4	0.756
Class 5	0.390
Class 6	0.388
Class 7	0.401
Class 8	0.683
Class 9	0.704
Class 10	0.720
Class 11	0.648

Class 12	0.401
Class 13	0.660
Class 14	0.803
Class 15	0.592
Class 16	0.307
Class 17	0.252

Table 20: Average mAP visibility visible per class performance of NetVLAD++[4] in test set obtained by us.

METRIC: Average mAP visibility unshown per class	**VALUE**
Class 1	0.0
Class 2	0.513
Class 3	0.0
Class 4	0.321
Class 5	0.069
Class 6	0.043
Class 7	0.002
Class 8	0.458
Class 9	0.642
Class 10	0.606
Class 11	0.390
Class 12	0.503
Class 13	0.031
Class 14	0.739
Class 15	0.131

Class 16	0.0
Class 17	0.0

Table 21: Average mAP visibility unshown per class performance of NetVLAD++[4] in test set obtained by us.

- **Work Packages, Tasks and Milestones**

Project: Preparation and research	**WP ref:** WP1
Major constituent: Theory	Sheet 1 of 7
Short description: Study and learn about Human Action Recognition and Action Spotting in soccer videos.	Start event: 01/07/2022 End event: 15/09/2022
Internal task T1: Do some labs of DLAI subject. Internal task T2: Watch some video lectures on Deep Learning theory proposed by my advisors. Internal task T3: Establish the goal of the project.	

Table 22: Work package 1.

Project: State of the Art	**WP ref:** WP2
Major constituent: Theory	Sheet 2 of 7
Short description: Research of the State of the Art about Human Action Recognition and Action Spotting in soccer videos.	Start event: 15/08/2022 End event: 20/10/2022
Internal task T1: Do some research about the matter. Internal task T2: Set up the main ideas. Internal task T3: Present to my advisors.	

Table 23: Work package 2.

Project: Databases	**WP ref:** WP3
Major constituent: Data	Sheet 3 of 7

Short description: Find the proper dataset, download and use it.	Start event: 7/04/2020 End event: 28/05/2020
Internal task T1: Research about the different datasets that there are in the market nowadays. Internal task T2: Request the download to my advisors. Internal task T3: Use the dataset to run some experiments.	

Table 24: Work package 3.

Project: Software	**WP ref:** WP4
Major constituent: Practical work	Sheet 4 of 7
Short description: Learn to use, run and modify the models.	Start event: 01/10/2022 End event: 09/01/2023
Internal task T1: Download and learn programming tools. Internal task T2: Run different models, training and tests. Internal task T3: Make some modifications and run again.	

Table 25: Work package 4.

Project: Improvements	**WP ref:** WP5
Major constituent: Improvements and tests	Sheet 5 of 7
Short description: Make different tests with other datasets and challenges. If there is time available, investigate on improvements for the tracker.	Start event: 08/06/2020 End event: 24/06/2020
Internal task T1: Make some test with different challenges. Internal task T2: Try to improve the system.	

Table 26: Work package 5.

Project: Final Report	**WP ref:** WP6
Major constituent: Documentation	Sheet 6 of 7

Short description: Write the Final Report.	Start event: End event:	
Internal task T1: Make a conclusion with all the done work and check if the project goals were accomplished. Internal task T2: Write the final report document.	Deliverable: *Final Report*	Date: 23/01/2023

Table 27: Work package 6.

Project: Presentation	**WP ref:** WP7	
Major constituent: Documentation	Sheet 7 of 7	
Short description: Prepare the final presentation.	Start event: 23/05/2020 End event: TBC	
Internal task T1: Prepare the final presentation.	Deliverables: *Presentation*	Dates: TBC

Table 28: Work package 7.

Milestones and tasks

WP#	**Task#**	**Short title**	**Milestone / deliverable**	**Date**
1	1 & 2	Previous knowledge	Have the exercises done	15/09/2022
1	3	Establish the project	Set up the idea	15/09/2022
3	4	Software	Have the executions done	09/01/2023
7	2	Final Report	Final Report	08/06/2023
8	1	Final Project	Final Presentation	TBC

Table 29: Milestones and tasks.

Glossary

UPC: Universitat Politècnia de Catalunya

TFM: Treball Final de Màster

CALF: Context-Aware Loss Function

NetVLAD: Vector of Locally Aggregated Descriptors Network

VLAD: Vector of Locally Aggregated Descriptors

IPG: Image Processing Group

HOG: Histograms of Oriented Gradients

SIFT: Scale-Invariant Feature Transform

HMM: Hidden Markov Models

RGB: Red Green and Blue

RGB - D: Red Green Blue and Depth

VGG-16: Visual Geometry Group with 16 convolutional layers

SVM: Support Vector Machine

CNN: Convolutional Neural Network

RNN: Recurrent Neural Network

LSTM: Long Short Term Memory

TSN: Temporal Segment Networks

NLNet: Non-Local Neural Networks

VR: Virtual Reality

GCN: Graph Convolutional Networks

SST: Sparse Spatio-Temporal

ResNet: Residual Networks

YOLO: You Only Look Once

PCA: Principal Component Analysis

NMS: Non-Maximum Suppression

mAP: mean Average Precision

JSON: JavaScript Object Notation

TP: True Positives

FP: False Positives

CIFAR10: Canadian Institute For Advance Research - 10 different classes

Amazon EC2: Amazon Elastic Compute Cloud

DNN: Deep Neural Network

GPU: Graphic Processing Unit