Evaluation of multi-modal approaches for automatic spotting and classification of events in soccer games.

Markus Stige

Thesis submitted for the degree of
Master in Programming and system architecture
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2021

Evaluation of multi-modal approaches for automatic spotting and classification of events in soccer games.

Markus Stige

Evaluation of multi-modal approaches for automatic spotting and classification of events in soccer games.

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Sports content is in significant demand in today's society. However, as the number of sports broadcasts is too high for anyone to consume, highlight reels that point to the most important events of a game are often used as a substitute for watching the entire match. To make these highlights, we must first find the events that are most interesting. Today, events from sports games are usually manually annotated by human operators, which is an expensive and time-consuming task. Recent research has shown that machine learning might be used to find these events without the need for any human intervention, potentially saving high costs of money and time. However, most approaches use only visual information to detect events, leaving out other valuable information like sound.

This thesis presents multi-modal approaches to spotting and classification of events in soccer games. We experiment with audio and visual information and explore different ways to combine these. We extract audio information through spectrograms and create audio-visual features through concatenation with pre-computed visual features. We evaluate the performance of the various approaches on the soccer-specific dataset SoccerNet [29], and compare the results to using only audio or visual information alone. We compare the results to state-of-the-art models for action spotting on SoccerNet. Furthermore, we analyze how the amount of data the models use for predictions influence the performance. For the task of spotting, we also analyze the impact of increasing the required tolerances of temporal accuracy for the predictions.

Our experiments show that multi-modal (audio and visual) approaches are beneficial for several use cases and that they show great potential for further utilization. We found that audio-visual approaches significantly improve performance for certain types of events, but that the performance depends on the configuration for other events. For classification, the audio-visual approach that uses the softmax average of an audio and a visual model increases the performance by 1.64% compared to the visual model alone. For spotting, the audio-visual approach through feature concatenation increases the performance for goals significantly, but does in some cases have a negative effect on the performance for other events.

Acknowledgements

I would like to thank my supervisors, Pål Halvorsen, Michael Riegler, and Steven Hicks for their guidance and contributions, and Olav Rongved for all the help along the way. I would also like to thank my friends and family, as well as my girlfriend, Anna Mina, for all the encouragement and patience through the process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Watching videos is a very popular activity in today's society, and there is a huge amount of video content available. You can find all kinds of videos online, and streaming videos has become easier than ever with the growing availability of internet access. On YouTube alone, there is consumed over a billion hours of video every single day [1]. A lot of this content is sports, and in recent years, there has been a growth in both searching and watch-time of sports "highlight" videos. From 2016 to 2017, the growth of watch-time was 80 percent, which shows that viewers increasingly seek the most important parts of the games [4]. 80 percent of sports viewers also say that they use an extra device to check out stats, live scores, and related videos while watching a game [4]. In other words, the fans are not only interested in the full game streams, but seek additional information and highlights as well.

The popular content is made available for the fans through annotations and tags in the videos. Usually, most of these annotations are manually tagged[1] in a costly and time-consuming operation. People have even tried to use existing metadata from media sites for offline operations [39], but this approach introduce large delays. However, with a huge amount of content, these annotation operations might not be possible to do manually. With efficient use of automatic action spotting[2], it may be possible to extract clips from the videos without the need for a human operator to manually go through all of the data. The tags can be used to extract valuable and appreciated data that otherwise would be hard or time-consuming to find. This would give value to different actors, such as the fans, the broadcasters, or even the teams themselves. This would especially help leagues and sports with fewer resources, as big leagues already have great productions of content and stats.

[1] Using for example Forzify tagging: https://forzasys.com/videos/forzify-tagging-pluss-small.mp4

[2] Spotting and detection are both used in the literature to describe the concept of predicting both the class and temporal location of an event. They are therefore used interchangeably throughout this thesis.

There are many possible approaches to create an efficient model for automatic event detection through machine learning. This is a complex task, and different approaches have been proposed [18, 29, 58, 76]. However, most approaches utilize only one modality to detect events, which means that useful information might be left out. For this thesis, we focus on the detection and classification of events in soccer videos, but we believe that our findings will generalize to other sports as well.

1.2 Problem Statement

Annotations of sports events are today manually tagged, which is an expensive and time-consuming task. With the use of well-performing automatic event spotting, the availability of popular content could increase and give a lot of value to the fans. We observe that many event detection approaches only use visual information, therefore this thesis aims to answer the question:

> *How do audio-visual approaches perform for automatic soccer event spotting and classification?*

Based in this main research question, we further define three objectives:

1. Research and develop suitable audio-visual approaches for event spotting and classification in soccer.

2. Analyze the performance of the approaches using the SoccerNet dataset [29].

3. Compare the results of the audio-visual approaches to single-modal approaches, and to the current state-of-the-art for event spotting in soccer.

1.3 Scope and Limitations

As each sport has its own set of special events, we decided to only focus on soccer for the entirety of this thesis. However, we do believe that the methods presented here are transferable to other sports as well. We limit ourselves to use the open dataset SoccerNet [29] as it is currently the largest public dataset for event spotting in soccer matches. This choice of the dataset also affects the events we predict, as it is tied to those included within SoccerNet ("goal", "substitution", and "card"). Our computational ability is limited by the hardware we have available. Therefore, for visual information, we use the pre-computed visual features supplied with the SoccerNet, and not the raw videos, as this has a significantly lower demand for computing power. We could have chosen to compute similar features ourselves, as this is a pretty straightforward approach with available off-the-shelf methods. However, it would not have made any valuable contributions to our work, and testing models with the supplied features could also be more beneficial for comparison with other approaches on the same dataset.

1.4 Research Method

There are several different methods of how to do research. For this thesis, we have based the research method on the Association for Computing Machinery's (ACM) research methodology. The fundamentals of computer science and computer engineering were reported in *Computing as a Dicipline* [21] in 1989. The report was created by a task force assigned by ACM Education Board, and it describes three paradigms; theory, abstraction, and design. We will now describe the three paradigms, and explain how this thesis is created in accordance with these.

- *Theory* The theory paradigm is related to mathematical coherent and valid theory. It includes four stages. These are (i) characterize objects of study (definition), (ii) hypothesize possible relationships among them (theorem), (iii) determine whether the relationships are true (proof), and (iv) interpret results.

- *Abstraction* The abstraction paradigm is rooted in the experimental scientific model. In the report, this includes four stages. These are (i) form a hypothesis, (ii) construct a model and make a prediction, (iii) design an experiment and collect data, and (iv) analyze results.

- *Design* The design paradigm is closely related to engineering, and it includes four stages. These are (i) State requirements, (ii) state specifications, (iii) design and implement the system, and (iv) test the system. This relates to processes for software system development or construction of physical devices.

Our work consists of researching and developing audio-visual approaches for soccer videos and analyzing the results. We prototype our approach based on a belief that audio-visual approaches would perform better than approaches using only one of the modalities. We design experiments to test this and analyze the results for different metrics. This supports the abstraction paradigm well. Furthermore, the design paradigm is supported, as the approaches we develop and test require a certain performance to be useful, and we especially work in accordance with this paradigm for the development of the audio feature extractor. We also touch upon the theory paradigm through the use of machine learning concepts and different hyperparameter optimizations for the models.

1.5 Main Contributions

As presented by the problem statement in Section 1.2, we want to assess the performance of multi-modal approaches for action spotting and classification. During the work of this thesis, we make the following main contributions:

- We research machine learning approaches for event detection and classification, and develop multi-modal approaches for these tasks. Specifically, we develop audio-visual approaches for soccer videos from the SoccerNet dataset [29], which includes the event types "goal", "card", and "substitution". We create an audio model which extracts audio features from the dataset through spectrograms and we use these to create concatenated audio-visual features. We experiment with different ways of fusing modalities and test both *late fusion* of models at decision time and *early fusion* through the concatenated features. We experiment with various hyperparameters and select optimized configurations for the models.
- We experiment with the selected models on SoccerNet and analyze the performance on the test split of the dataset. We test models for both event detection and classification, and assess the performance for several metrics. We show that the audio model extracts valuable features and that the concatenated features work well for action spotting on the dataset.
- We compare the performance of our multi-modal approaches to single-modal approaches and show that there is a great benefit, and further potential, for using more than just visual information. We present results showing that for the classification task, the best results are achieved with an audio-visual model, outperforming the best visual results from our experiments with over 1.5%. Further, we show that for action spotting, the performance with concatenated audio-visual input is superior for goals for all tested configurations. For other events, the results improve with concatenated input for some of the configurations, but it could also in some cases have a negative effect on the performance.

Our contributions are interesting in the context of the problem statement, and we present results valuable to assess the performance of audio-visual approaches. We show that the potential of multiple modalities are great, but that it might be event specific how it performs in some situations. This work gives a strong foundation for further work with multi-modal models. Furthermore, our results are presented in a paper which is under review.

1.6 Outline

Chapter 2 - Background In the background chapter, we will introduce concepts and terminology needed to understand the further work in this thesis. We will present relevant work in the field of event detection in videos, and describe the key concepts of the approaches which are now considered state-of-the-art. This chapter will provide information about the foundation of which this thesis will build upon.

Chapter 3 - Methodology In the methodology chapter, we will present the dataset that is used in the thesis. It will be described which preprocessing steps that are necessary for the dataset to fit the task, and how the dataset is being used. Further, we will describe different relevant models, and select which we want to experiment with. We present relevant hyperparameters that could influence the performance of the models and our approach to selecting these hyperparameters. We also present how different modalities can be fused together in one model, and which approaches we have experimented with for the models in this thesis.

Chapter 4 - Experiments and Results In the experiments and results chapter, the results from our experiments with the models will be presented. We will inspect the performance of the models and observe how they perform for several metrics to gain insights into how different modalities affect the models. The model's strengths and weaknesses will be highlighted, and we will try to understand why the models performs as they do. We present results for both action classification and action spotting, and assess how multi-modal approaches perform compared to single-modality approaches.

Chapter 5 - Conclusion In the conclusion chapter, the work of the thesis will be summarized, and the insights and contributions will be presented. Further, it will be discussed possible future work related to this thesis, and what we hope to see fulfilled in the future.

Chapter 2

Background

Our main purpose is to find a model that can automatically detect events in soccer games. We would like to be able to feed our model with audio and visual information from a soccer game, on which our model would identify that certain events happen and tag at which point in time the events occurred. To create this model we will use machine learning.

In this chapter, we first present the needed terminology to read and work with machine learning. Then we will introduce what is done in this field until now, and what is active areas of research, in Section 2.8. The related works will help gain insights into which approaches give the best performances, and what the main challenges in this field are today. We will present general action recognition work before we describe work done on action detection in general, and for soccer videos in particular.

2.1 Video annotation

Annotations of sports events are today manually tagged, and this might be expensive and time-consuming. Today the process of tagging requires an operator to watch the whole game, and even if the operator watches several games simultaneously, this takes a lot of time. In systems like Forzify tagging plus, the process is split into two steps. A first-level operator watches one or several games and tags events in the games as they happen. After this, a second-level operator tunes the tags with exact start/stop timestamps and relevant metadata [2]. The tagging task is expensive with a two-level approach, but if the initial tagging could be automated the amount of work could be reduced. This would make the tagging process more efficient.

2.2 Event Definition

Events can be defined in several ways, and the Cambridge dictionary defines it as "anything that happens, especially something important or unusual". But is it obvious exactly when an event occurs? Sigurdsson et al. [68] did an experiment showing that most of the time people agree on where the center of an event is in time, but that opinions on the start and ending points of the event is much more diverse. This could potentially influence the result for event detection a lot in general, since the event proposal often will be checked against some defined true timestamp for the event. For this thesis, it should not be a big problem, as the events are tagged with an anchor at a single point in time. This eliminates the problems with disagreement around starting and ending points, and keeps only the center, which most people agree on.

2.3 Action Recognition

Action recognition is the task of identifying different actions from video clips. The action may, or may not be performed throughout the whole video [31], and the goal of action recognition is to identify the actions of one or more objects from a series of observations [72]. The input is often an image or a video, and it is normal to classify for only one target action. There are also datasets containing multiple classes in the same image/video.

2.4 Action Detection

While action recognition is classifying an action that is present, action detection also has the aspect of locating when the action of interest happened in space and/or time [41]. This means that to get action detection right, you would have to both classify the right action and mark the temporal interval of which this action took place. Action detection is also referred to as action spotting, and the terms are used interchangeably in this thesis.

2.5 Machine Learning

Machine learning is a field in computer science where algorithms are used to parse data, learn from it, and create a model that should be able to predict something [19]. The algorithms find patterns in the datasets, and should be able to automatically learn from the data without being explicitly programmed how to do so [79]. We will now go through some of the core concepts of Machine Learning.

2.5.1 Supervised Learning

Supervised learning is the most common way of using Machine Learning today [80]. It is recognized by the data, where each data point is paired with a label that states the correct output for the data point. This allows the algorithms to search for patterns in the data, and try to generalize a model which predicts correct output for the given input.

2.5.2 Unsupervised Learning

Unlike supervised learning, unsupervised learning does not use data labeled with the desired output. This gives unsupervised learning different areas of use. It can be used to reduce the dimensionality of a dataset, gain insights into the structure of the data before a classifier is designed, find groups of data points with similarities (clustering), and to detect noise or outliers in the data [36, 55].

2.5.3 Classification

Classification is the task of assigning a class to a given input. This is typically done through a model trained with supervised learning. We distinguish between three types of classifications.

1. **Binary classification** is the task of predicting between two possibilities. This is typically if something is, or is not, something. An example is if a fingerprint to unlock your phone is recognized as your fingerprint or not.

2. **Multi-class classification** has a given number of classes, and is to classify which of the classes the input belongs to. It is normal to calculate some kind of probability-score for all the potential classes and choose the class with the highest value as the answer. But it can only be chosen one.

3. **Multi-label classification** is similar to Multi-class in the way that it has a given number of classes available, but is different because several classes could be predicted. The prediction of one class does not exclude the prediction of another class. An example could be face recognition in a photo featuring several people. The classifier would have to predict, and then suggest all the persons found.

2.5.4 Non-maximum Suppression

Non-maximum Suppression (NMS) is a way of reducing the number of predictions for a model. This is done to remove predictions that are close to each other, which then could be considered redundant. What is considered as "too close" is decided by the user of the algorithm, and is set through a threshold. The algorithm starts with the original predictions with confidence scores and selects the one with the highest score. Then, each of the other predictions considered to overlap according to the given threshold is removed. The one with the highest score is moved to the list of output predictions. These steps are repeated with the remaining predictions, and it continues until all predictions are removed from the original list. The list of output predictions we are left with has then no predictions closer to each other than the threshold limit.

2.5.5 Regression

Regression is also a task associated with supervised learning. In regression, the model predicts an answer to a given question, but does not predict a class. The answer is not limited by pre-given classes, and often predicts a number. An example is a model that predicts the price of a car, given some attributes that describe the car.

2.5.6 Dataset

A dataset in Machine Learning is a set of data points, stored in the same format so that a model can understand it. In the context of supervised learning, the dataset would consist of both the data points and the corresponding classes which the data points belong to. A dataset has several use cases, and it is often split into three subsets with different purposes. A training set, a validation set, and a test set. These three should be exclusive and without overlap.

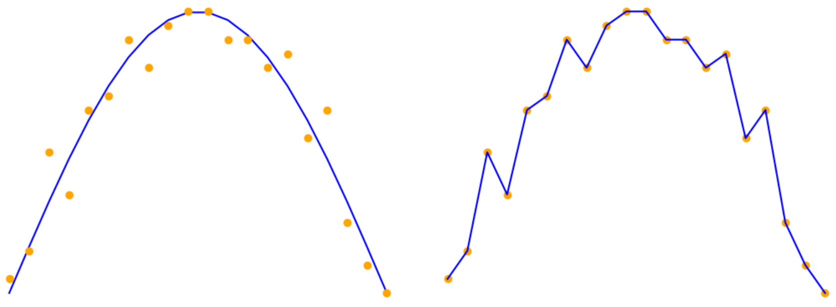
- **The Training set** is used to train the model. The data is fed into the learning algorithm, which finds patterns and learns from the data.
- **The validation set** is used to tune hyperparameters, avoid overfitting, and generally try to make the model generalize from the training data as good as possible.
- **The test set** is only used at the end when testing the performance of the model, and is not seen before that. That is because we want to see how the model performs on completely unseen data, and should be representative of how the model will perform in the real world.

2.5.7 Overfitting

Overfitting is when the model fits well on the training data, but does not generalize properly to other data. It often occurs if the model is trained for too long, so the model adapts too much to the training set. This is a problem as the goal of training a model is not to make it perform well on the training data, which we already know the class of, but to perform well on unseen data. Therefore we use the validation set to tune the hyperparameters to avoid overfitting the model. The performance on the training set should ideally not be much better than on the validation set, but it is not unusual to see a small difference in favor of the training set. Figure 2.1 shows the difference between an underfitted, a robust, and an overfitted model.

2.5.8 Principal Component Analysis

Principal Component Analysis (PCA) is a method to reduce the number of dimensions for data. The approach is a form for *feature extraction*, which means that instead of just removing features *feature elimination*, the original features are combined and reduced [8]. This way, a new set of features are



(a) Good fit

(b) Overfitted

Figure 2.1: Example of a model that fits well vs an overfitted model. This shows that it is not good to adapt to much to the training data.

generated based on a combination of the old features, and the dimensions are reduced by keeping as many of the new features as wanted.

2.5.9 Neural Networks

The idea of a neural network in machine learning is loosely based on the actual neural network of the human brain. You have many interconnected nodes, which alone only performs simple processing steps. The nodes are spread out within different layers, where the data moves through these layers while doing calculations in the nodes. The connections between two nodes in different layers are assigned with a number called a "weight" [33]. The number between all connected nodes between two layers makes the weights between the two layers. Figure 2.2 shows how the nodes in a neural network are connected with the nodes in both the previous and the next layer.

The weights between the two layers define how much the data sent through the connections is to be considered when doing calculations in the node. The node receives data from each of its incoming connections, uses the data as input of a linear function, before the result is passed to a nonlinear function called an *activation function*. This calculation gives the node a number, and if the number is considered big enough the node "fires", and sends the number to its connected nodes in the next layer. Figure 2.3 shows how a single node is working. The nonlinear function is necessary to avoid that the result just collapses to a big linear function. A popular nonlinear function is the ReLU function, which is defined in Equation 2.1.

$$ReLU(x) = max(0, x) \tag{2.1}$$

When we train a neural network we need a loss function and an optimization method. The task of a loss function is to give an evaluation of how well the algorithm models the dataset [37]. This creates a foundation for the learning in the network, as the optimization method uses the result of the loss function to optimize the model. One of the most popular optimization algorithms is the *gradient descent*. It updates the learnable

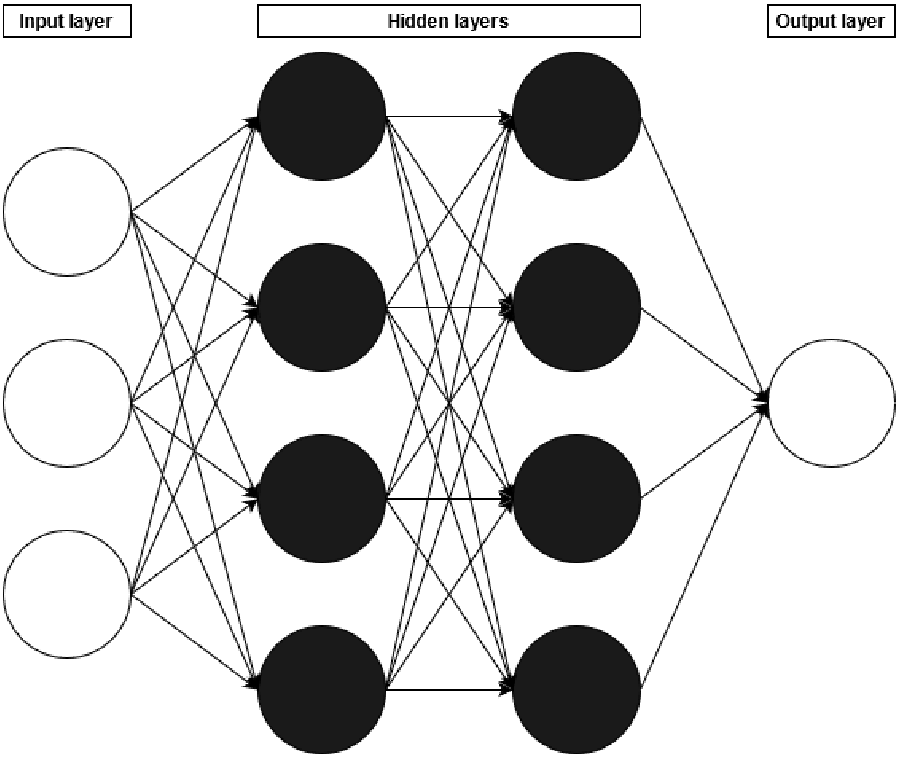


Figure 2.2: Model of a neural network with one input layer, two hidden layers and an output layer.

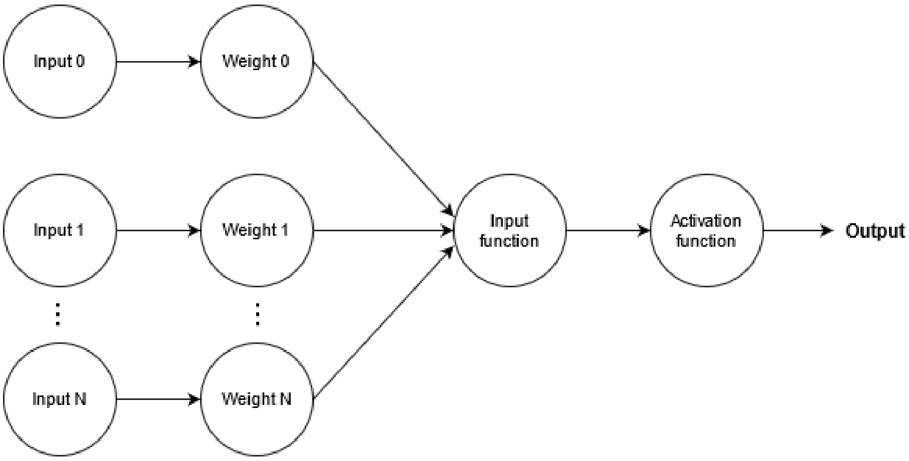


Figure 2.3: Model of the workflow for a typical single node in a neural network

weights through an iterative process where the gradients are used to move towards the minimum of the loss function. Gradient descent is formally defined in Equation 2.2.

$$w_{t+1} = w_t - \alpha \nabla_w J(w_t) \tag{2.2}$$

Where α is the learning rate, J is the loss function, and $\nabla_w J(w_t)$ is the derivative of the loss function w.r.t w.

2.5.10 Convolutional Neural Networks

A convolutional neural network takes an image as input, and is able to capture spatial and temporal information from the image through the use of relevant filters [65]. The filters are implemented as kernels that convolve over the image and assigns a value to the center pixel through calculations with the surrounding pixels. The principle is shown in Figure 2.5. This will be done for each filter you want to convolve over the image, and for each filter, the output will be a feature map, as shown in Figure 2.4. You can see that the dimensions of the output layer is smaller than in the input layer. This is because the kernel starts in the corner, as you see in Figure 2.5, and the outer pixels will then be lost since the kernel describes the central pixel of the kernel. This can be handled by zero-padding the image, so the center of the kernel begins in the corner of the original image.

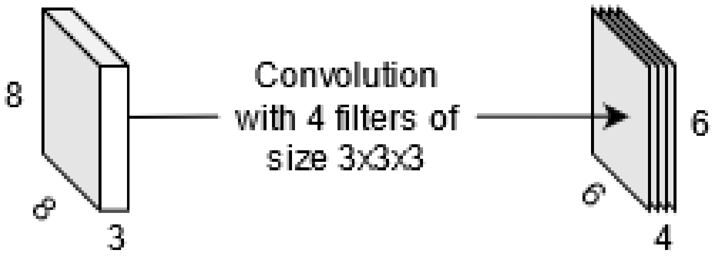
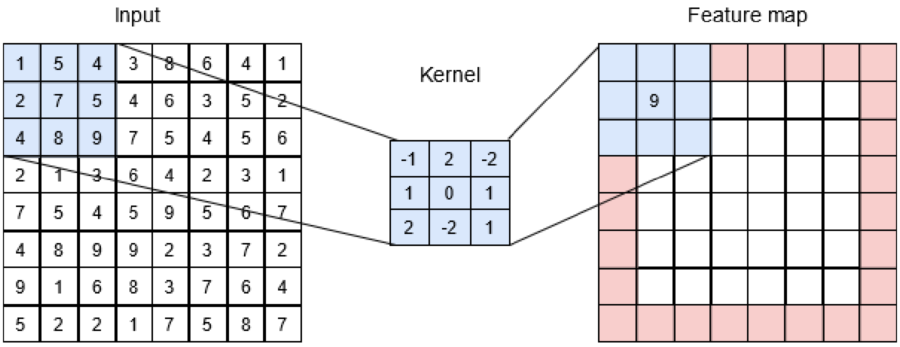


Figure 2.4: Convolution for input size 3x8x8 with 4 filters of size 3x3x3, resulting in 4 feature maps of size 6x6.

2.6 Spectrograms

Spectrograms are a way of visualizing sound by showing the amount of the different frequencies over time. The decomposition of a sounds signal frequencies is extracted through a Fourier Transform [26]. The visual representation will show both which frequencies and the amplitude of them at each moment in a timeline. An example of such a representation is shown in Figure 2.6. The Mel scale [54] is often used for the frequency scale. This is done to better match the distance in the scale to how the distance sounds to humans, since the difference between 500 Hz to 1000 Hz is much more noticeable to humans than 7500 Hz to 8000 Hz, even though the change in Hz is the same. [27]



1	5	4	3	8	6	4	1
2	7	5	4	6	3	5	2
4	8	9	7	5	4	5	6
2	1	3	6	4	2	3	1
7	5	4	5	9	5	6	7
4	8	9	9	2	3	7	2
9	1	6	8	3	7	6	4
5	2	2	1	7	5	8	7

-1	2	-2
1	0	1
2	-2	1

Figure 2.5: Illustration of a convolutional operation. The convolutional kernel shifts over the source layer, filling the pixels in the destination. The pixels interacting with the kernel, as well as the kernel itself, are marked blue. The red pixels illustrate the reduction of size from input to output when the input is not padded.

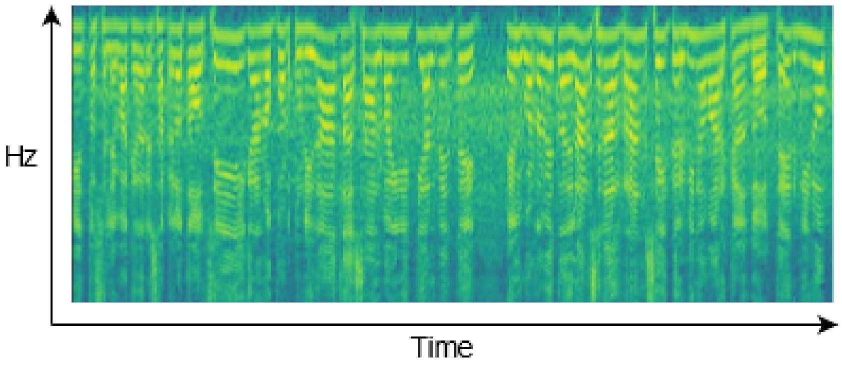


Figure 2.6: An illustration of a spectrogram. The x axis is time, and the y axis is Hz. The color represents amount of the given Hz at the given time.

2.7 Transfer Learning

Transfer learning is the concept of using a model that is trained for a task as a basis for another model with a different task [9]. This means that instead of randomly initializing the weights of a new model, you use the weights found by another model as a basis. If the task of the two models is somewhat similar this might lead to a head start in the training of the new model, as some of the insights from the other model could be relevant. In the Figure 2.7 some potential benefits are illustrated, and we can see that the starting point is higher, the model learns faster, and that it ends up converging with better performance. This is just a hypothetical comparison, but it still gives an illustration of the benefits we might get from using transfer learning.

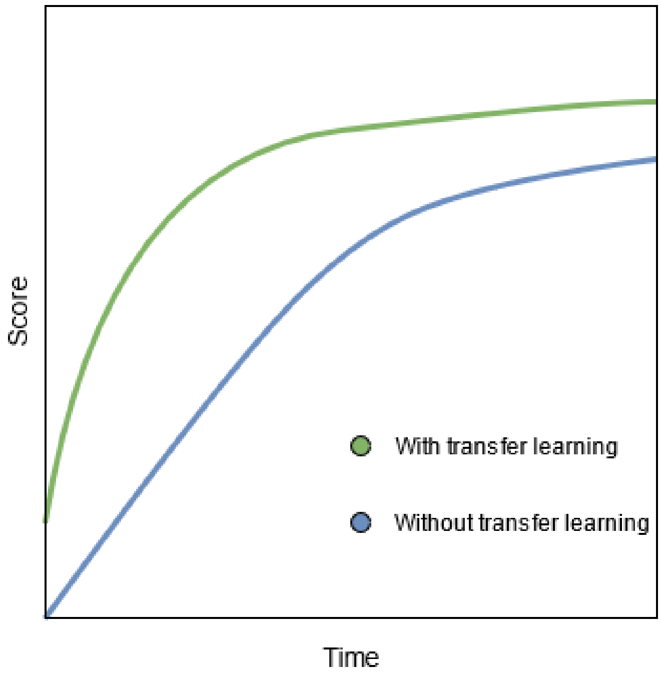


Figure 2.7: Illustration of three ways in which transfer learning might improve training. The three possible improvements are higher start, slope and asymptote.

2.8 Related Works

In this section, we will describe work already done that is related to this thesis. We will first describe some relevant datasets and the improvements in this field in recent years. Then, we will cover some important works on video and audio understanding, as well as works with multi-modality. Further, we will describe interesting works with action detection in general and action detection on soccer videos specifically. Lastly, we will mention work done with automatic video summaries.

2.8.1 Datasets

Using machine learning to perform action recognition and detection is a difficult task that usually requires large datasets with high-quality annotations. This can be expensive to create as annotations often are done manually, and the use of metadata to annotate might be inaccurate. Fortunately, there are several datasets available, which have made big contributions to the field. Some of the first datasets with the purpose of action recognition and detection was the Hollywood2 [53], UCF101 [71] and HMDB-51 [47]. State-of-the-art results on UCF101 is now at 98.69%, which shows that the field has made good progress from the baseline on 43.9% released with the dataset in 2012. More datasets have been released over the years after this, and some of the largest in terms of the number of videos are Youtube-8M [5], Sports-1M [42], and Moments in Time [56]. Youtube-8M originally had 8.2 million videos with 4800 classes when it was released in 2016, but was updated in 2018 as a smaller dataset with higher quality, containing 6.1 million videos [3]. In 2019, Youtube-8M Segments was released with segment-level annotations. This includes 230k human-verified segments with 1000 classes. Another big dataset is the Kinetics-400 [43], which was released in 2017 with over 300k videos and 400 human action classes. When it was released, it could be seen as the successor to the UCF101 and HMDB-51 for the human action video area. Later Kinetics-600 [13] and Kinetics-700 [14] have been released with more videos and classes, now containing 650k videos and 700 classes. The latest release in the Kinetics-series is the AVA-Kinetics [49], which extends the original AVA dataset[32] with videos from Kinetics-700 annotated using the AVA annotation protocol. What stands out the most with those datasets is that the actions are localized both temporally and spatially. Two other datasets are THUMOS14 [38] and ActivityNet [24], which both contributes as popular datasets.

For audio, there are several domains for the datasets, such as music, audio speech, and environmental sound classification. AudioSet [28] is a dataset from 2017, containing 632 audio event classes with over 2 million clips extracted from Youtube videos. The clips are 10 seconds long and annotated by humans. MagnaTagATune [48] is a dataset containing music from different genres. The clips are 29 seconds long, and it is a total of 25,863 clips, from 5223 songs, 445 albums, and 230 artists. Each clip contains multiple tags, annotated by humans through game-like situations where several people tag the same clips. Typical tags are for example "singer", "drums", or "jazz". A dataset for environmental sound classification is Urban8k [66]. This is a dataset containing 10 classes, such as "car horn", "dog bark" and "gunshot". There are 8732 clips, and each is no longer than 4 seconds.

A soccer-specific dataset that was released in 2018 is the SoccerNet [29]. This is a dataset containing 500 videos of soccer games from 2014 to 2017 with games from six European leagues. It has a total duration of 764 hours, and includes 6637 annotations of the event types goal, yellow/red card, and substitution. This gives a frequency of an event happening every 6.9 minutes on average, which is sparse within long videos. Recently, a new version of this dataset named SoccerNet-v2 [20] was released. In this version it was added several other event types and the total number of annotations increased to 300k. It was also introduced a new task for camera shot segmentation with boundary detection, and a replay grounding task.

2.8.2 Video understanding

Computer vision is an active area of research. One of the most important sections of this is video understanding. There are several interesting areas of video understanding, but two of the biggest problems addressed in this field are action recognition/classification and action detection. Several different approaches have been tried out for the task of action recognition. Karpathy et al. [42] provided an empirical evaluation of CNNs on large-scale video classification, encouraged by CNNs performance on image recognition problems. Multiple CNN architectures were studied to find an approach that combines information across the time domain. The CNN architecture was modified to two process streams to improve runtime performance, a context stream and a *fovea* stream. The runtime performance was increased with $2 - 4\times$, and the accuracy of classification was kept. The features learned generalized to a different smaller dataset, showing benefits of transfer learning. Simonyan and Zisserman [69] tried a two-stream convolutional network in 2014 and exceeded by a large margin the previous attempts to use deep nets for this task at the time. The network was split into one spatial stream performing action recognition on still video frames, and one temporal stream recognizing action from motion in the form of dense optical flow. Tran et al. [74] proposes using 3-dimensional convolutional networks (3D ConvNets), and found out that 3D ConvNets are more suitable for spatio-temporal feature learning than 2D ConvNets. Further, the features are compact and efficient to compute. Donahue et al. [23] used long-term recurrent convolutional networks models combining CNN as a feature extractor, and feeding the features to a special kind of RNN called long short-term memory (LSTM). This approach reached comparable results to Simonyan and Zissermans two-stream approach on UCF101. Carreira and Zisserman [12] introduces an inflated 3D ConvNet (I3D) that is expanded from the 2D ConvNet inflation. With pretraining on Kinetics, the I3D model reaches 80.9% on HMDB-51 and 98% on UCF101. Qiu et al. [62] presents Local and Global Diffusion(LGD) networks. That is a novel architecture for the learning of spatio-temporal representations capturing large-range dependencies. LGD networks have two paths, a local and a global for each spatio-temporal location. The local path describes local variation, and the global path describes holistic appearance. The LGD network outperformed several state-of-the-art models at the time on benchmarks, including UCF101, where it reaches 98.2%. Kalfaoglo et al. [40] combined 3D convolution with late temporal modeling. With the use of the Bidirectional Encoder Representations from Transformers (BERT) instead of the Temporal Global Average Pooling (TGAP), they increased performance for 3D convolution. They provide state-of-the-art results for both HMDB51 (85.10%) and UCF101 (98.69%).

When it comes to action detection the task can be divided into two parts, to generate temporal proposals and to classify the temporal proposals generated. In many cases, these two are taken apart in detection methods [15, 67, 70, 82], but they are also together as a single model in other methods [10, 50]. There are two approaches to the proposal generation task, one *top down* and one *bottom up*. The *top down* is most used in previous works [11, 35, 82], and involves pre-defined intervals and lengths. This has the problems of boundary precision and the lack of flexibility on duration. There are also methods that uses the *bottom up* approach, like TAG [82] and BSN [52]. The drawback for them is the lack of ability to generate sufficient confidence scores for retrieving proposals. Recently Lin et al. [51]

addressed this by introducing Boundary-Matching Network (BMN), which generates proposals with more precise temporal boundaries and more reliable confidence scores. Combined with an existing action classifier they report state-of-the-art performance for temporal action detection.

2.8.3 Audio understanding

As with video understanding, audio understanding is also an active area of research, and contributions to this field are regularly published. It is experimented with different datasets, such as MagnaTagATune [48] and AudioSet [28], having different sound domains. Dieleman et al. [22] tested a convolutional neural network approach for music, where they explore the possibilities for applying feature learning directly to raw audio signals. They compare this approach to an approach with a representation of sound using spectrograms and found that the approach with raw audio signals did not outperform the spectrogram approach. Choi et al. [17] presented an automatic tagging algorithm using fully convolutional neural networks with Mel-spectrograms as input. They concluded that Mel-spectrograms are effective as a time-frequency representation. Nayyar et al. [57] performed experiments on the MagnaTagAtTune [48] dataset with architectures like convolutional neural network and convolutional recurrent neural network with Mel-spectrograms as input. They showed that tagging of several types of information, such as genre, instruments, and emotions could be classified. Pons et al. [61] performed a comparison for music audio tagging between input as Log-Mel spectrograms and in waveform. They show that waveform outperformed spectrograms when it is a big amount of data available. Kong et al. [46] propose pretrained audio neural networks, trained on the AudioSet [28]. They achieve state-of-the-art performance on the AudioSet tagging, as well as for several other audio pattern recognition tasks.

2.8.4 Multi-modality

The idea of a multi-modal analysis of video content goes back at least 20 years. For example, Sadlier et al. [64] performed an audio and visual analysis of video separately, then combined the statistics of the two approaches afterward, showing the potential of multi-modal analysis. Simonyan et al. [69] propose a two-stream convolutional neural network used for human activity recognition, where the class scores for the different modalities are fused before the final predictions. Vielzeuf et al. [77] tried what they address as standard score fusions for an emotion classification challenge. The score fusions mentioned are majority voting, means and maximum of scores, and linear SVM. Kim et al. [45] propose an approach using three networks taking as input respectively image, landmark, and audio, before fusing the softmax-output from the three networks with a specialized method called emotion adaptive fusion. Arevalo et al. [7] propose an original Gated Multimodal Unit (GMU). It is meant to be used as a part of a neural network with the purpose to find an intermediate representation for data combined from different modalities. In this case, the input was textual and visual. Chen et al. [16] follow an approach of fusing text, visual, audio features through a special fusing method and use these features as input to a model. Vielzeuf et al. [78] also propose a multimodal fusion approach. It begins with two feature extraction models, one for each modality, before the features are fused and used in a classifier.

Recently and in parallel with our work, some other approaches have also been presented. Ortega et al. [59] combined audio, video, and textual features by first separately using fully connected layers, followed by concatenation. Audio-visual SlowFast [81] used video frames as input at different sample rates, combined with an audio stream that takes Log-Mel spectrograms as input, with lateral connections and a special training method to avoid overfitting, reporting state-of-the-art results on six video action classification and detection datasets. Finally, AudioVid [75] used a pretrained audio model to extract features and combine them with NetVLAD [6, 30] at different points in the model through concatenation. They found that audio generally increased the performance, i.e., a mAP of 7.43% for an action classification task and of 4.19% for an action spotting task using SoccerNet [29].

2.8.5 Action detection in soccer videos

After the release of SoccerNet in Giancola et al. [29], the works on action detection in soccer videos have accelerated. The SoccerNet was released with a baseline model reaching an Average-mAP of 49.7% for tolerances δ ranging from 5 to 60 seconds for the task of spotting, which they define as finding the temporal anchors of soccer events in a video. Giancola et al. has a sliding window approach at 0.5s stride, using C3D [74], I3D [12], and ResNet [34] as fixed feature extractors. Rongved et al. [58] use a ResNet 3D model pretrained on kinetics-400, and reports an Average-mAP of 51%, which is an increase from the baseline provided in Giancola et al [29]. Rongved et al. further showed that the model generalized to datasets containing clips from the Norwegian Eliteserien and the Swedish Allsvenskan. The results showed that in clips containing goals they could classify 87% on the samples from Allsvenskan, and 95% on the samples from Eliteserien with a threshold of 0.5. Cioppa et al. [18] introduce a novel loss function that considers the temporal context present around the actions. They address the spotting task by using the introduced contextual loss function in a temporal segmentation module, and a loss similar to YOLO [63] for an action spotting module creating the spotting predictions. This approach increased the Average-mAP for the spotting task to 62.5% and is considered the state-of-the-art for the SoccerNet spotting task. Another approach to the spotting task in SoccerNet is studied in Vats et al. [76] where they introduce a multi-tower temporal convolutional network architecture. 1D CNNs of varying kernel-sizes and receptive fields are used, and the class-probabilities are obtained by merging the information from the parallel 1D CNNs. They report an Average-mAP of 60.1%, which is a difference of 2.4% from Cioppa et al. [18] with a simpler approach using a cross-entropy loss function.

2.8.6 Automatic video summaries

Automatic video summaries and highlights generation is something that also would be interesting for the fans. This goes straight to the core of the needs of fans who want only the main events in a game. This task might be a little more forgiving when it comes to tagging too many events, as long as the recall rate is high at crucial events, such as goals. In the paper by Cioppa et al. [18], they also consider the model for automatic highlights generation and find that the segmentation scores in their approach are useful for this task. High segmentation scores that did not lead to an annotated type of event might be of interest for a game summary, even though it is not an annotated event in the system now. This could be unannotated classes of interest, like goal-scoring attempts or fouls that did not lead to a card. They manually inspected the videos with a high segmentation score for goals where there was no annotated goal as ground truth. When adjusting the threshold they found that most of the clips were considered goal opportunities. To create a highlights generator they dropped the substitution events and included cards given and goals with a segmentation score over a threshold. They found this solution to be adequate, but with room for improvement. A special weakness is the way of extracting the video clips, which they do by starting 15 seconds before spotting and ending 20 seconds after. This is not ideal, and with better video clip extraction this could give even better game summary results.

2.9 Summary

In this chapter, we have first presented useful background information and terminology needed to understand the thesis. This included machine learning-specific concepts and applications for machine learning in this thesis. We described general machine learning concepts, and specifically described neural networks and convolutional neural networks. We then presented works related to the topic of this thesis and found that automatic event annotations are an active area of research.

Several machine learning approaches have been tried for action recognition in the last decade, and for general video understanding, some impressive results have been reported. The performance of UCF101 has now exceeded 98%, with Carreira and Zisserman [12] using an inflated 3D ConvNet giving 98%, Qiu et al. [62] reporting 98.2% using a Local and Global Diffusion (LGD) networks, and Kalfaoglu et al. [40] reporting 98.69% with the use of BERT. When it comes to action detection the task is a bit harder, as you also have the temporal detection aspect. Lin et al. [51] made some promising work when introducing a Boundary-Matching Network, and it performs as state-of-the-art on both ActivityNet-1.3 and THUMOS14. We also presented some multi-modality work and described different approaches tested for multi-modality, with Audio-visual Slowfast [81] reporting state-of-the-art results on six video action classification and detection datasets.

We presented the release of SoccerNet in Giancola et al. [29] as an accelerator for work on action detection in soccer videos and described some methods that had outperformed the baseline (49.7% Average-mAP) released with the dataset. After this, we presented the most recent progress in this field with Cioppa et al. [18] and Vats et al. [76]. We presented how Cioppa et al. introduced a contextual loss function, increasing the results on the spotting task to 62.5%, and how Vats et al. performed to 60.1% with a simpler approach. We also presented recent multi-modal approaches that have been presented in parallel to our work on this task, with AudioVid [75] showing improvements compared to the baseline model. In the end, we explained how Cioppa et al. also created an automatic highlights generator by doing some small modifications to the approach.

This shows that action detection in videos in general, and soccer videos in particular, are active areas of research. There are promising works in the field, and the state-of-the-art is still improving, showing possibilities to increase the performance. Although the works are promising, the results are still not good enough to be really useful. It might be good enough for some types of events, but for important events, the sport itself requires very high accuracy. This means that more work is needed to find even better models. Observing that most approaches to this task primarily use only visual information, we find it interesting to experiment with models where both visual and audio input are combined. This gives a foundation for our work, and in the next chapter, we use the insights gained to select approaches including both audio and visual information, to test if this combination perform.

Chapter 3

Methodology

Annotations of sports events are today manually tagged, which is an expensive and time-consuming task. There is a high demand for sports video content, and with an increasing amount of content produced, it is not possible to manually annotate all content without using a vast amount of time. With the use of well-performing machine learning models, we might be able to annotate more content automatically. This requires less time-consuming work for manual operators. Through an increasing supply of datasets appropriate for training machine learning models for sports, we see that the working conditions for research in this field have been improved. This is an active area of research, and for the soccer-specific dataset SoccerNet [29], we have seen several improvements of the state-of-the-art since it was released in 2018 [18, 58, 76]. Most of the approaches have in common that they consider visual input only, and with multi-modal approaches to machine learning models showing great potential, we want to study how including the sound as input would impact the performance.

In this chapter, we describe our approach with audio-visual input for action spotting and classification on the SoccerNet dataset. We first describe the dataset in use more thoroughly before describing the models in more detail, as well as how the audio features were extracted. Then, we will introduce how the fusion of the audio and visual features has been done and how we have chosen the different configurations of the hyperparameters of the models. Finally, we discuss the evaluation metrics used to assess our experiments and interpret what they mean.

3.1 Dataset description

The dataset we used for all experiments is SoccerNet, which was presented by Giancola et al. [29] in 2018. This is a dataset whose main purpose is for action spotting in soccer games. This means that the main task for this dataset is not only to classify which events that occurs, but also when they happen in time. However, it is possible to use the annotated events for classification tasks as well. It contains 500 games from some of the biggest soccer leagues in Europe played between 2014-2017. The leagues included in the dataset are the top-level leagues in England (EPL), Spain (LaLiga), Germany (Bundesliga), France (Ligue 1), and Italy (Serie A), as well as the European Champions League. The distribution between the leagues and seasons is shown in Table 3.1.

	Season			
League	14/15	15/16	16/17	**Total**
EPL	6	49	40	95
LaLiga	18	36	63	117
Ligue 1	1	3	34	38
Bundesliga	8	18	27	53
Serie A	11	9	76	96
Champions League	37	45	19	101
Total	81	160	259	500

Table 3.1: Distribution of games per league and season in the SoccerNet dataset.

The structure of the dataset is sorted on both league and season, with one directory per league and a sub-directory for each of the seasons. Inside these, there is one folder for each game containing both the actual video in MKV format and its annotations in a JSON file. These JSON files contain some metadata about the game, as well as the name and timestamps of the annotated events in the video. In addition to this, the game folder includes pre-extracted visual features in NumPy-files. Each game is split up into two videos, one for each half of the soccer match, which results in 1000 videos in total. The dataset contains 764 hours of gameplay. The games are split into a training set, a validation set, and a test set which includes respectively 300, 100, and 100 games. The games are not saved and sorted in these splits but are divided into the correct splits by lists defining the games belonging to each set.

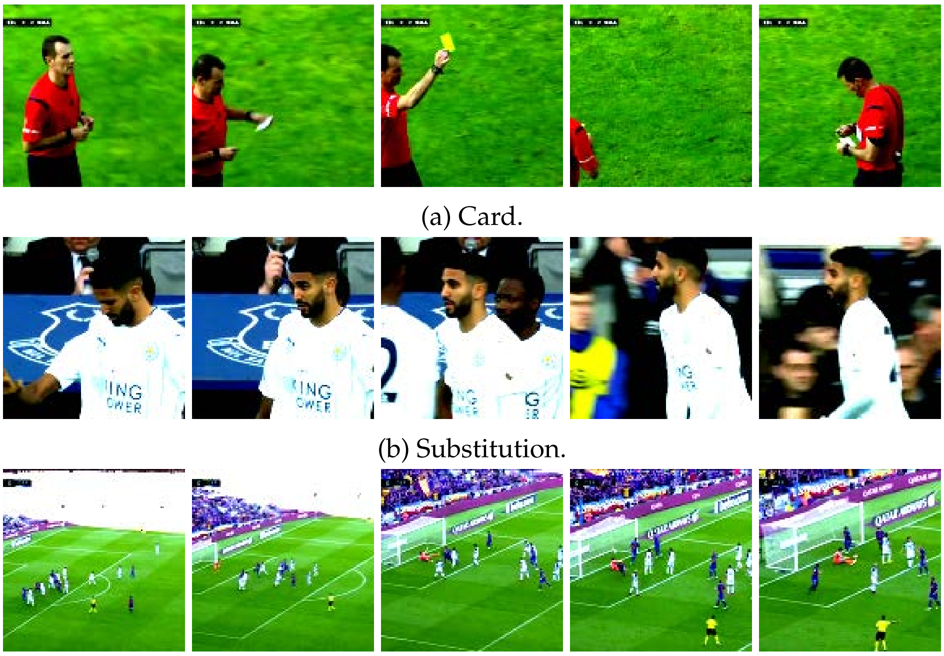
The dataset contains annotations of the 3 event types *goal*, *card*, and *substitution*. These events are defined as follows:

1. *Goal* The instant the ball crosses the goal line to end up in the net.

2. *Card* The instant the referee shows a player a yellow or a red card because of a foul or a misbehavior.

3. *Substitution* The instant a new player enters in the field.

The annotations in SoccerNet are anchored in a single frame in the video. This means that instead of marking a start and a stop time of an interval, it is annotated as a single frame at the exact moment an event is happening as defined above. This is illustrated in Figure 3.1, where we see that the middle frame is where the event is annotated for each of the examples. With this approach, we have an exact moment in time we are trying to predict. This means that we also can operate with different time-tolerances for how far away from an event a prediction is considered a true positive. There is a total number of 6,637 annotated events in the included 500 games. Knowing that there is a total of 764 hours in the dataset, this gives a frequency of approximately one event per 6.9 minutes on average. The



(a) Card.

(b) Substitution.

(c) Goal.

Figure 3.1: Sample frames from the SoccerNet dataset [29] for 3 different event types. The middle frame is at the annotated time.

distribution of events in the training, validation, and test splits is shown in Table 3.2.

Class	Training	Validation	Test
Card	1,296	396	453
Substitution	1,708	562	579
Goal	961	356	326
Total	3,965	1,314	1,358

Table 3.2: The number of samples per class in the SoccerNet dataset.

Along with the 3 classes above, we follow the example of Rongved et al. [58] and add a *background* class by sampling in between events. If the distance in time between two consecutive events is larger than 180 seconds, then a new background sample is added in the center, such that a background sample will never be within 90 seconds of another event. This adds 1,855 events to the training set, 636 events to the validation set, and 653 events to the test set. We use the dataset containing the added background class for a classification task, while only the original 3 classes are used for spotting.

SoccerNet provides three sets of visual features already extracted from the videos by C3D [74], I3D [12] and ResNet [34]. We have used the features extracted by ResNet in our experiments, and in the context of this thesis these features will be referred to as the "ResNet features". The features have been extracted from the videos by using a ResNet-152 [34] image classifier pretrained on ImageNet [**i**]. Features are extracted from single video frames at a rate of 2 times per second. Subsequently, PCA is used to reduce the feature dimension to 512. How the features are extracted is illustrated in Figure 3.2, and the outputted ResNet features are the visual features we use in our experiments.

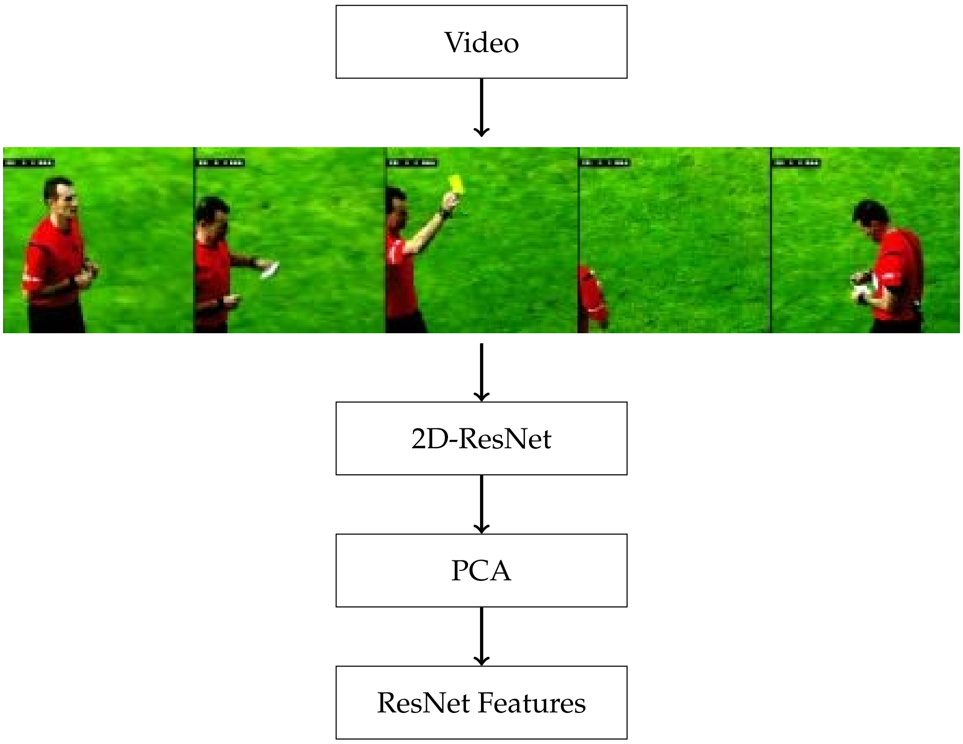


Figure 3.2: Visual representation of the pipeline which produces the ResNet features provided with SoccerNet [29]. A pre-trained ResNet is used to extract features from video, followed by PCA. The features can then be used to train a network for action detection tasks.

In addition to the visual features provided by SoccerNet, we also use the audio in the dataset to produce our own audio-features directly from the media. We extract the sound from the videos in wave format, before we create spectrograms we can produce features from. How these audio-features are produced is described further in Section 3.2.2 where the audio model is presented.

While working on this thesis, the creators of SoccerNet released a second version of the dataset. This dataset is called SoccerNet-v2 [20] and expands the already released dataset. SoccerNet-v2 contains the same amount of videos but has added more events, which has increased the number of annotations to 300k. In addition to action spotting, it has been created tasks for camera shot segmentation with boundary detection, and a replay grounding task. We considered using SoccerNet-v2 instead of, or in addition to, the original dataset, but the dataset was released too late for us to do so. When it was published we had already gone through a lot of work with the original dataset, and it would not be a good idea to change dataset so late in the process. Therefore, we decided to continue using the original version of the dataset. Thus, the SoccerNet-v2 is very interesting for future work in the field.

3.2 Model Selection

As we want to test how adding sound would affect performance, we must choose which model, or models, we want to try this approach on. We also need a way to extract audio features, so we must create a model for this as well. The three models we have selected are:

- The model from Cioppa et al. [18], which is using the Context-Aware Loss Function (CALF). This model is tested for action spotting.
- An audio ResNet model with spectrograms as input. This is used as a feature extractor and is also tested for classification.
- A visual 2D-CNN model with the ResNet features supplied with the SoccerNet as input. This is tested for classification.

In this section, we will describe the models we have tested, and discuss why they were chosen.

3.2.1 CALF model

Reasons for selection

To effectively observe how the models perform for action spotting on soccer videos when we add audio information, we think it is a good idea to use a model we know works well for this task with visual input alone already. This way we know the model will perform with visual input for the task, and can effectively compare how adding audio information affects the performance. Therefore, we look at the models that have been tested for action spotting on SoccerNet [29] before. Some of the models considered are the baseline models supplied with the SoccerNet paper and the top-performing models for the task. The two best performing models (at time we made our choice) are the model described in Vats et al. [76] and the model described in Cioppa et al. [18], which is using the Context-Aware Loss Function (CALF).

The CALF model is the best performing model based on the average-mAP metric (see Section 3.5 for an explanation of metrics), and is considered the current state-of-the-art model architecture for this task. We also see that the code for CALF is supplied, with code for training and running the model. This is also supplied with the baseline model, but we could not find this for Vats et al. When considering if any of the three stands out as an intuitively better fit to include audio information we did not find that one of them stands out as easier.

In total, we see that the CALF model seems like the best option in our case because it is the best performing model and has openly available code to give us a great foundation to work on. With available code for training and running the model, we get more time to create code to extract more analysis data and dive deeper down into the results of the performance when adding audio input. It also allows us to test several configurations of the model, as the hyperparameters that fit best for video input do not necessarily fit best when adding audio.

Description

Cioppa et al. [18] introduced a loss function that considers the temporal context present around the actions, called Context-Aware Loss Function (CALF). Their model consists of a base convolution, a segmentation module that uses their novel loss function, and a spotting module. The overall structure is illustrated in Figure 3.3. In the context of this work, we refer to this model as the “CALF model". The inputs to the model are the ResNet [34] features provided with the SoccerNet dataset [29] (see Section 3.1 for details about the ResNet features).

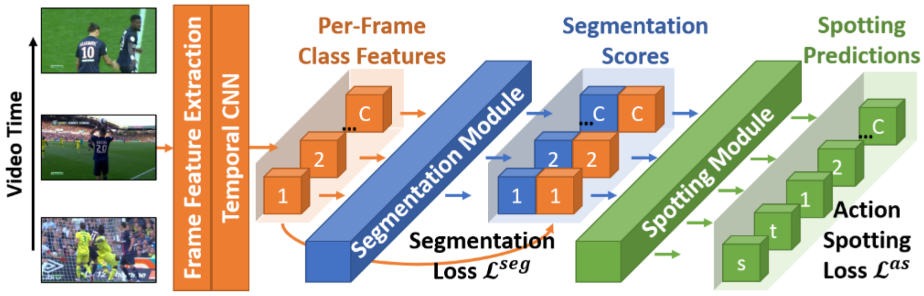


Figure 3.3: An overview of the CALF model. Reprinted from Cioppa et al. [18]

The concept of the Context-Aware Loss Function (CALF) is to not only penalize the model if a prediction is wrong but also include some information on how far away it was from a correct prediction. The loss function differentiates how big the loss is based on the distance between the annotations and the predictions. This way the model tries to learn which type of information should have the most impact on the prediction. This is logical, as the frames far before an event do not give away as much information as the frames right after an event occurred. To do this, there are six "temporal segments" around each event, where the loss is weighted differently based on which segment the frame belongs to. The segments are illustrated in Figure 3.4. The segments *far before* (definitely no action) and *far after* (definitely no action) is not desirable to have an impact on the predictions. Thus, the frames in these segments are trained to not predict anything. This is done by heavily penalizing predictions of events, and rewarding predictions of no actions. The segment *just before* (possible action) is not wanted to influence the decision, as the lead up to an event occurring could potentially lead up to either of an event happening or not. Since you can not tell from the frames just before if the event will happen or not, the network is not influenced by these frames. For the segment *just after* (action occurred) there is a lot of information about the event happening or not, so these frames are trained to heavily influence to predict an event. A low score is therefore heavily penalized, and high scores are rewarded. Finally, the two last segments are *transition zones* between the two segments close to the event, and the two segments far away. This is to create a gradual change in the influence of the frames.

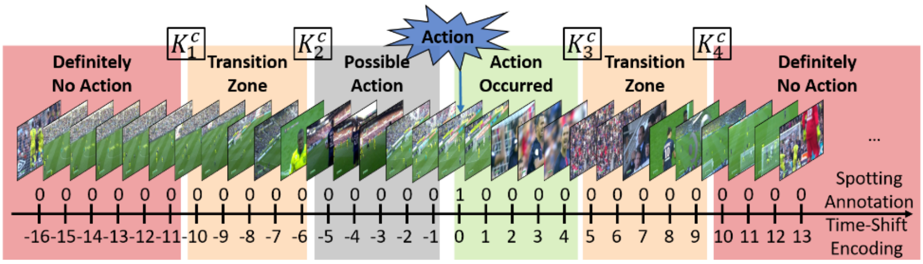


Figure 3.4: Illustration of the different temporal segments in the CALF model. Reprinted from Cioppa et al. [18]

The model takes as input chunks of frames, and the chosen chunk size in the original CALF model covers 120 seconds of gameplay. The receptive field (the temporal dimension of the convolutional kernel) in the original model is 40. The chunk of input frames is first sent through a multi-layer perceptron (MLP) consisting of two layers, which reduces the features per frame to 32. These output features are used in a temporal pyramid with four different lengths. The temporal pyramid output is concatenated with the output features from the two-layer MLP, which produces 152 features per frame. This is fed to a convolution layer which creates the input to the segmentation module with 48 features per frame. The segmentation module reshapes this to be 16 features per class per frame, and these features are used to create a segmentation score for each class per frame.

The spotting module takes as input both the output and the ReLU activated input of the segmentation module. The two inputs of the spotting module are concatenated. The dimensions are reduced with temporal max-pooling three times, with ReLU activated convolutions in between. The last temporal max-pooling is flattened out and fed to two fully connected layers: one with sigmoid activation to obtain the confidence scores and timestamps, and one with a softmax activation to obtain the class spotting. These results are concatenated to create the action spotting in the same format as the ground truth actions. Each prediction then contains 5 values; a confidence score, a timestamp, and one softmax activated score for each of the three classes.

The model outputs both the segmentation score and the spotting predictions. There is one loss function for each of the two outputs, and they are combined to calculate the total error of the model. The segmentation loss uses information about where a frame is located in relation to ground truth events to decide how big of an influence the score for that frame should have on the total loss. For each frame, the loss is computed for each class based on the position in relation to the given class, as described earlier. This is done based on a Time Shift Encoding each frame has for each class. The Time Shift Encoding represents how far away from an event the frame is, and this decides which segment it belongs to, and thereby how the loss should be calculated. This is summed together to create a total segmentation loss.

For the spotting loss, an iterative one-to-one matching is used to match up each ground truth event with a prediction. For each of the ground truth events, the weighted squared error loss is calculated for each of the attributes (confidence score, timestamp and one for each class), which is summed together and added to the loss of predictions matched up with no ground truth events. These predictions calculate a weighted loss based on only the attribute of whether an event is present or not. The spotting loss is added to a weighted segmentation loss to produce the total loss for the model.

Implementation

For the CALF model, we used the code supplied with the original paper [18] for training. For testing we used a combination of supplied and new code, to extract in-depth results. We used an Nvidia DGX-2, which consists of 16 Nvidia Tesla V100 GPUs and has a total memory capacity of 512 Gigabytes.

We experiment with ResNet features, audio features, and concatenated features that combine the ResNet features and audio features. The audio features are the features extracted with the audio model in Section 3.2.2 for window size 8. We vary the chunk size and temporal receptive field, as it might affect the results for lower tolerances and when audio features are used. The configurations we test on the test set have chunk sizes 120 and 60, with receptive fields 40, 20, and 5. These three configurations are referred to as *CALF-120-40*, *CALF-60-20* and *CALF-60-5*. See Section 3.4 for details of the selection process.

We use a learning rate of 0.002 for all variations of the CALF model. For our tests with only visual ResNet features, we train CALF for 300 epochs, validating every 20 epochs. Due to instabilities when training with audio and concatenated features, we train for 30 epochs on audio features and 50 epochs on concatenated features and validate every other epoch. We use the weights that achieve the best Average-mAP on the validation set during training for each of the models. When testing, we calculate the best threshold for the models on each of the tolerances on the validation set and use these when calculating the performance on the test set. Non-maximum suppression (NMS) is also applied to the predictions, with an overlap threshold corresponding to the tolerance for the predictions.

3.2.2 Audio ResNet model

Reasons for selection

In addition to the CALF model that we want to use for the action spotting task, we need a model to extract audio features. We need these features both as audio features alone and together with the visual ResNet features supplied with the SoccerNet to create concatenated audio-visual features.

We draw inspiration from the Audiovisual SlowFast Networks [81], which use spectrograms and a ResNet-based network. We do not do the same modifications to the ResNet but look at ResNet [34] as a good option to work as the CNN in our model. We use ResNet for several reasons.

- It is available as an "off the shelf" model in PyTorch.
- It is not of the heaviest neural networks to run but is still a network we have confidence in that will do a good job for our purpose.
- It can be used as a feature extractor if we remove the final layer of the model.
- The features fit well to concatenate together with visual features to use with the CALF model. (See section 3.3).

Description

To create the audio features we need to have access to the audio from the dataset. We do this by changing the video format of the videos in the dataset to *avi* and use FFmpeg [25] to extract the sound in wave-form from the original media. When we have access to the audio, we can extract the features with a model. Convolutional Neural Networks are a good way to generate features, and we can utilize this if we first get the audio in a compatible input format for a CNN. To do this we generate spectrograms to create a visual representation of the sound, and then use these spectrograms to extract features through the neural network.

To generate the audio features, we train a ResNet model on spectrograms. We use the SoccerNet dataset with the additional background events to train the model for a classification task. With this approach, we most importantly get an audio feature extractor by removing the last layer in the ResNet model. But in addition to this, we also get a classifier we can test on the dataset events. By testing the classifier we can confirm that the features extracted in the model contains useful information, and gain more insight into how a model using sound performs. We can also create a comparable 2D-CNN network for visual input and compare the models to each other, as well as to a combined approach. This way we can gain even more insights into how visual, audio and combined models perform.

Summarized, we first extract audio from video in wave-form, which again is used to generate Log-Mel spectrograms. We then use a CNN as a classifier, i.e., an 18-layered 2D-ResNet [34], but remove the last layer to generate features. Figure 3.5 shows the pipeline for the audio model.

We also test with different window sizes (denoted W) over which the spectrograms are generated, representing the temporal extent of the input used.

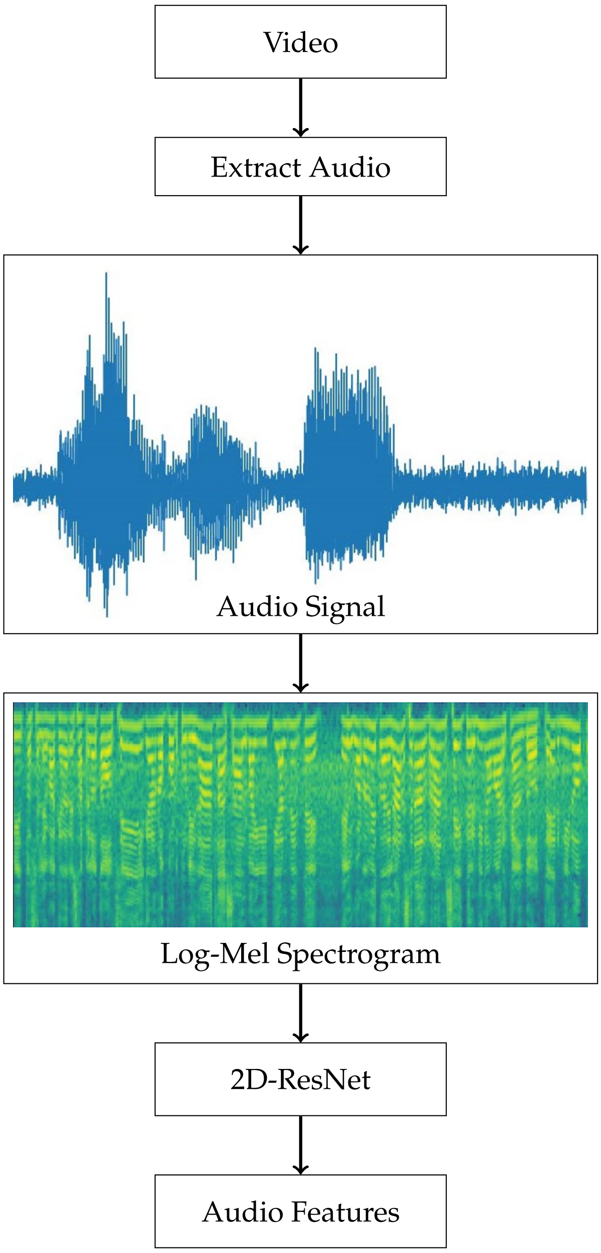


Figure 3.5: Visual representation of the pipeline used by the audio feature extractor. First, audio is extracted from the video. The audio is used to compute Log-Mel spectrograms, which are then used as inputs to a 2D-ResNet that creates the features.

Implementation

We implemented the audio model in PyTorch [60] and trained on an Nvidia DGX-2, which consists of 16 Nvidia Tesla V100 GPUs and has a total memory capacity of 512 Gigabytes. We use a minibatch size of 32, an initial learning rate of 0.001, and a momentum of 0.9. We use a scheduler that reduces the learning rate by a multiplicative factor of 0.1 every 10 epochs, and we train for 25 epochs. During training, we save the model that had the best accuracy on the validation set and evaluate its performance on the test set. We first generate Log-Mel spectrograms, after which we train on a 2D-ResNet as described in Section 3.2.2. We train and test models with the window sizes 2, 4, 8, 16 and 32.

3.2.3 2D-CNN model

Reasons for selection

We want to test a classification model with a comparable approach to the audio model, but with visual and combined input. This way we can observe how the audio model itself performs compared to these, and create a better foundation to learn more from our experiments. This will give additional insights into how sound affects performance, not just through the extracted features as input to the CALF model, but through a comparison to a similar visual model as well. We choose a 2D-CNN network, comparable to the audio model approach, but with visual input instead. This is chosen for several reasons. It is easily accessible, which is convenient to save some time in the process. It has similarities to the audio model approach, making them comparable for the classification task. Furthermore, we know that 2D-CNN approaches perform well for classification tasks for visual input. All together we think this is a good model selection for having a model to compare the audio model to, which we are confident will perform well, while still being time-efficient to create.

Description

We use a 2D-CNN model that uses the pre-extracted ResNet features provided by SoccerNet. The model takes in $(2 * W) \times F$ features for each sample, where W is the window size used for sampling in seconds, and $F = 512$. Inspired by the approach in SoccerNet [29], we first use a 2D convolution with a kernel $1 \times F$. This is followed by batch-normalization and another 2D convolution that has a kernel of $\frac{W}{2} \times 32$, such that it has a temporal receptive field of $\frac{W}{2}$. Finally, we have two fully connected layers and an output layer. The model is trained and tested on SoccerNet with the added background class as a classification model. A detailed workflow of the model is illustrated in Figure 3.6.

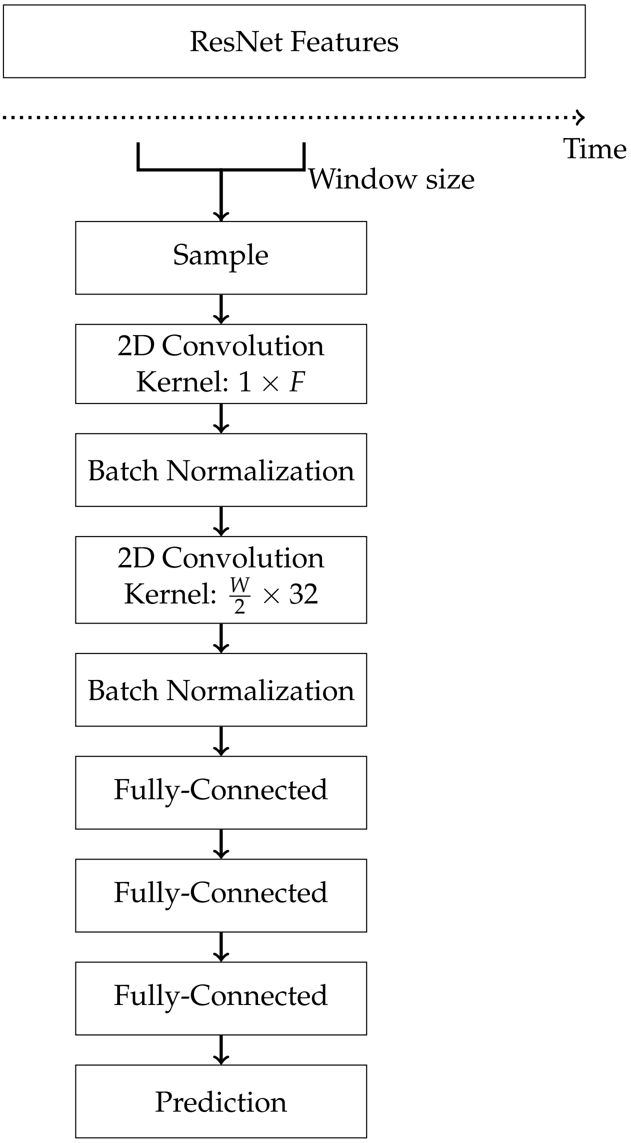


Figure 3.6: Detailed workflow for the 2D-CNN model. This model uses a pre-computed set of visual features as input.

Implementation

The implementation details for the visual 2D-CNN model are mainly the same as for the audio model described in Section 3.2.2. The only difference is that for the visual 2D-CNN model we use the supplied ResNet features from SoccerNet as input, instead of the spectrograms we use with the audio model.

3.3 Modality Fusion

To assess the effects of using multiple modalities, results from different models must be combined. In this respect, there are several ways of fusing different modalities. The two opposite concepts when it comes to modality fusion are *early fusion* and *late fusion*. We have tested both, with *early fusion* for concatenating the input for the CALF model, and *late fusion* by combining the audio model with the visual 2D-CNN model at the prediction level.

3.3.1 Early fusion

Early fusion, also referred to as *data-level fusion* or *input-level fusion* is a traditional way of fusing data *before* conducting an analysis [44]. Within the context of our work, this approach translates to generating concatenated audio-visual features (generation of audio features and concatenate them to existing visual ResNet features), which can be used as input to a model. Of the multi-modal approaches mentioned in the background chapter (Section 2.8.4, Arevalo et al. [7] and Chen et al. [16] propose approaches considered as *early fusion*. Vielzeuf et al. [78] propose a multi-modal fusion approach, where they argue that their approach is neither a pure early or late fusion approach, but we find it more similar to *early fusion* than *late fusion*.

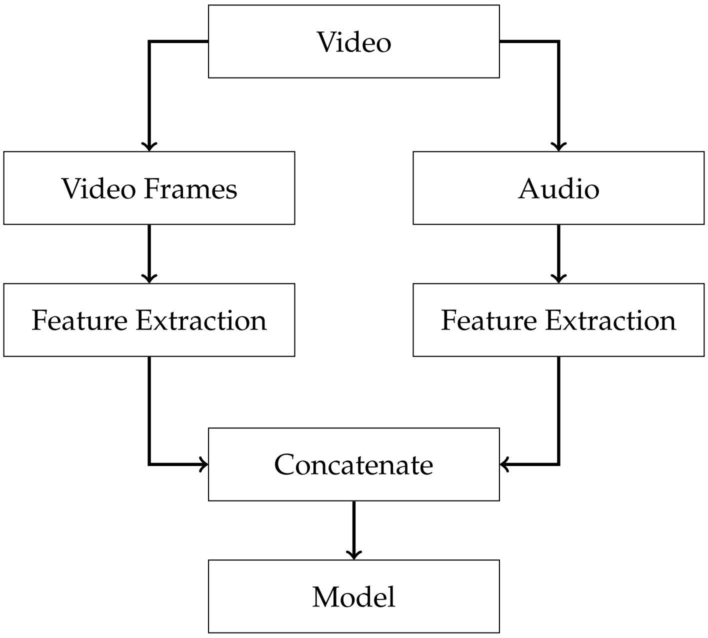


Figure 3.7: *Early fusion.* Visualization of how audio-visual features can be created. A ResNet is used to compute visual features based on single frames. For the audio, a Log-Mel spectrogram is used to train a 2D-ResNet, and further used as a feature extractor by removing the output layer. These features are then concatenated.

We choose to do this approach for the testing of the CALF model, and the reasoning behind this is explained in Section 3.3.3. We use the ResNet features supplied in SoccerNet [29] as our visual features. For our audio features, we generate these using the audio model described in

Section 3.2.2. We first train an audio model for classification on SoccerNet using Log-Mel spectrograms on window size 8 (See Section 3.4.1 for details of why this window size was chosen). Next, we remove the last layer, resulting in a 512-dimensional feature vector. These are the same dimensions as the ResNet features provided with SoccerNet. This means that if we extract them at the same frequency we can concatenate them together over time. The audio features are therefore also calculated two times per second, and we concatenate them such that we align them over time, resulting in a 1024-dimensional feature vector two times per second. This way we have created feature vectors containing information from both audio and video, which suits our multi-modal approach well. This is used as input to the CALF model, so we can compare the performance for fused features to visual-only features, as well as audio features. We illustrate the process of concatenated feature generation in Figure 3.7. The concatenation itself, when the features are extracted, is a relatively straightforward concatenation. It might be more complex ways to concatenate features, but showing that performance can be increased with a relatively simple approach could imply a great potential for feature concatenation when it comes to action spotting.

3.3.2 Late fusion

In *late fusion*, also referred to as *decision-level fusion*, data sources are used independently until fusion at a decision-making stage. This method might be simpler than *early fusion*, particularly when the data sources are significantly varied from each other in terms of sampling rate, data dimensionality, and unit of measurement [44]. Of the multi-modal work mentioned in the background chapter (Section 2.8.4, Simonyan et al. [69], Vielzeuf et al. [77], and Kim et al. [45] propose approaches considered as *late fusion*.

Within the context of our work, this approach translates to fusing independently built visual and audio models by taking the softmax average, or softmax max, of their predictions at test time. The approach is illustrated in Figure 3.8. The intuition behind this is that, in cases where the visual model might make a strong prediction that an event has occurred, the audio model might have weaker and more uniform predictions or vice versa.

For *softmax average* fusion, we use the audio and visual models, which have been trained on their respective inputs. For each sample, we take the average softmax prediction between the two models. For *softmax max* fusion, we calculate it similarly to softmax average, except that instead of average, we use the maximum softmax prediction value between the two models. We choose to use the *late fusion* approach with the audio model with spectrograms as input, and the 2D-CNN model with ResNet features as input as the visual model. The reasons for testing them with this fusion approach are explained in the following chapter below.

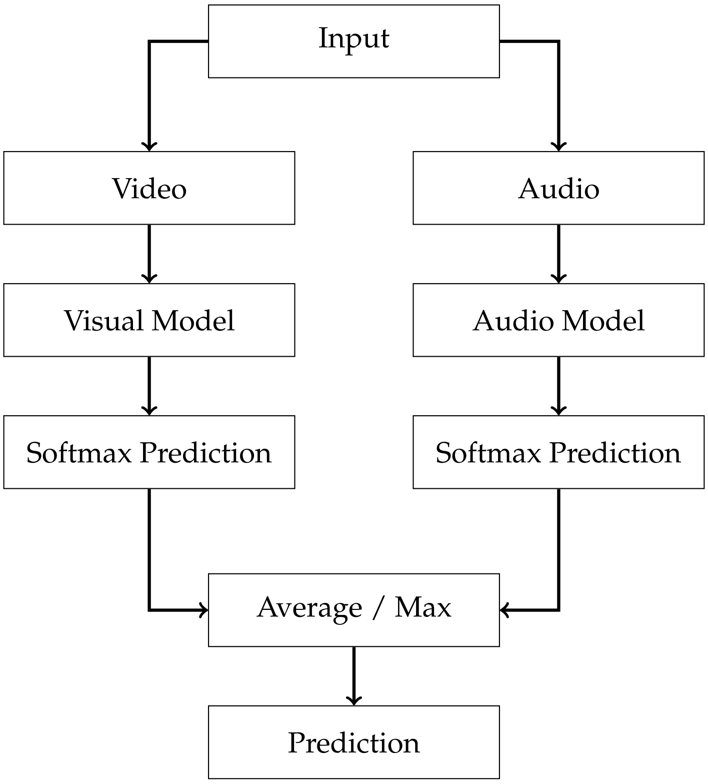


Figure 3.8: *Late fusion.* Visualization of how two seperate models can be fused through softmax average.

3.3.3 Fusion approach selection

We wanted to try out both *early fusion* and *late fusion* approaches, so we had to select which approach we wanted to test for which models. When looking at the two fusion approaches and how they would be implemented in the models, we chose to test *early fusion* for the CALF model and *late fusion* for the audio model and 2D-CNN. This has a couple of reasons. The first is that we want to compare the audio model to the 2D-CNN model for the classification task on SoccerNet. Since the audio features are generated with the same audio model, we think that it might be unfortunate to compare the audio model to the 2D-CNN model with concatenated audio-visual features as input. This because the audio features used would have been generated through the same model we compare the 2D-CNN to. The second reason is that it is more convenient to adjust the models we have implemented to fuse together at the stage of the softmax output than to re-implement and modify the supplied code for the CALF model to do this. It would be easier for the CALF model to test with generated audiovisual input. We also know that we extract features with the same dimensions as the visual ResNet features that are used as input to the CALF model. With this in mind, we know it would be possible to concatenate the features and align them over time, which gives us a good way to experiment with an *early fusion* approach for the CALF model.

3.4 Hyperparameter selection

Hyperparameter selection is an important part of any machine learning process. Without the correct set of hyperparameters, we risk that the model does not work as well as it could, or in some cases, fails to learn at all. In this section, we discuss our selection of hyperparameters and explain why they were chosen.

3.4.1 Audio model and 2D-CNN

We want to compare the performance of the audio model to the 2D-CNN model, but we also want to compare them to the multi-modal approach for the models, where we combine the softmax output by taking either the average or max value. We want to experiment with different window sizes, both to analyze how window sizes affect the performance of the models, but also to use in the selection of which window size to use in the audio model we choose for our audio feature extractor.

We test 2, 4, 8, 16, and 32 seconds as the window sizes as this gives a nice variety of how much audio the model gets as input. We train them for 25 epochs and run the tests on the validation set. The results for the tests are presented in Section 4.1.1, as a preliminary experiment. We use these results to select a meaningful window size for the audio feature extractor and to select the combined model we want to use.

When we have to choose the window size to use for the audio feature extractor, there are two things we want to take into account. First, we must select a model that performs well on the validation set, as this implies that it can generalize to unseen data. The second is that we must consider that bigger window sizes also mean more temporal information to the single features. For the first point, we use the results we present in Section 4.1.1, and we see that the performance has increased for the audio model when we increase the window size. This means that to get the best performing model, we should use bigger window sizes. However, we do not want the features to have a too big temporal extent since they are going to represent moments in time. When we consider both of these aspects, we see that it becomes a trade-off as they pull in different directions when it comes to window size. Therefore, we choose something in the middle for our window size. We choose window size 8 for our audio feature extractor, as this gives us features from a model with an accuracy that exceeds 70%, while still having a relatively small temporal window, at 4 seconds at each side of the center of the window.

We want to compare how the combined approaches to each other, to choose which of them we want to test further. From the results presented in Section 4.1.1, we observe that the models perform very similar, but that the one using softmax average consistently outperforms the one which uses softmax max. Therefore, we choose to let the combined model using softmax average represent the combined approach in our experiments on the test set.

3.4.2 CALF

We use the CALF model to test how the concatenation of features affects spotting performance, but several hyperparameters could influence how well this approach works. Two of the hyperparameters we expected would affect this the most were the chunk size and the receptive field. The chunk size is the size of the chunks of consecutive video frames that are sent to the network in seconds, and the receptive field is the temporal receptive field (in seconds) of what influences the results for a frame. This means the temporal dimension of the convolutional kernel. Furthermore, the number of epochs used to train the model will affect the results, but we suspect that the number of epochs could be reduced by a fair amount from 1000 in the original paper, while still keeping good performance. Reducing the number of epochs would give us the opportunity to train more models with different configurations of chunk sizes and receptive fields. This way we could analyze how it affects the performance, and use more models when testing the concatenated features on the test set later.

Therefore, we decide to use an iterative approach, where we start with a low number of epochs and a relatively high number of different configurations before we increase the number of epochs for the configurations that we find interesting. This way, we sift out configurations we do not want to test further and increase the performance of the interesting ones, while still maintaining a relatively low cost. At the same time, we can observe both the average-mAP on the validation set and how many validation cycles it has been since the model last improved under training. We can compare the average-mAP of the *CALF-120-40* to a pretrained set of weights supplied with the CALF-code, to observe if the model has approximately the same average-mAP on the validation set. The supplied weights are produced by using the same configurations as the *CALF-120-40* model we are testing, but with 1000 epochs of training. The supplied weights achieve an average-mAP on the validation set at 0.6410. This way we have two metrics to assess if we consider the number of epochs sufficient to proceed with the models to our experiments with different features. The approach has been performed for the visual features only, as this was the first step in our process where we wanted to find which configurations we wanted to test the multi-modal approach on. In the first iteration, we compared the results with and without NMS applied to the results. Based on these results, we continued using NMS further in the experiments. The results with and without NMS are presented in Section 4.1.4.

Other approaches, such as random search, grid search, and Bayesian optimization could have been helpful or interesting additions to our manual approach. Bayesian optimization was already performed by Cioppa et al. for the CALF model, so we assume this approach would again have pointed at a chunk size of 120 and a receptive field of 40 as the best configuration. Our approach also differs slightly from regular hyperparameter optimization, as we try to consider not only how the model works with the current visual input but also takes into account that they will be tested with different input as well. The visual input will still be present in the concatenated features, so observing the performance and selecting configurations that give interesting results with visual-only input is still valuable. However, the best performance overall is not the only factor that influences the selection. We also want to try to understand which could have potential in combination with sound. Therefore we reward models with small receptive fields that show potential, as we suspect that audio information changes more rapidly, and therefore could be interesting in a shorter amount of time. We also reward models with higher performance on lower tolerances, as these have better predictions closest to the actual event.

The plan was originally to train and test the models with audio and concatenated input for the same number of epochs (300) and validate on the validation set with the same frequency (20). When training with the audio and concatenated features we observed that this had to be modified, as the training was very unstable for these approaches. This is described further in Section 3.4.2.

Visual features

The configuration for the CALF model was selected by using the hyperparameters in the original CALf model as a starting point. This is because we know this configuration has been chosen based on a Bayesian optimization. This configuration has a chunk size of 120 seconds, and a receptive field of 40. We name this configuration of the model *CALF-120-40*, and use the same naming approach for the other configurations tested, with the first number representing chunk size, and the second number representing the receptive field. Since we have a starting point we know works well, we experiment with both a higher and a lower number for the chunk size. We test with chunk sizes 60 and 180, as well as the original 120 with a combination of several receptive fields. Since we believe that audio information in the soccer domain might rapidly change during the game, we primarily want to test configurations with receptive fields at the original size and lower. The original ratio between the chunk size and receptive field is a receptive field at 1/3 of the chunk size, so for all the configurations we use this as the biggest receptive field, and test with other lower numbers as well. We decide that 5 is going to be our smallest value for the receptive fields we are testing. Since the biggest value we are going to test is 180/3, we end up with values ranging from 5 to 60 for the receptive field. We choose our values to be 5, 20, 40, and 60 and combine all the three chunk sizes with the receptive fields that do not exceed the upper boundary we have set at 1/3 of the given chunk sizes. This gives us a total of 9 configurations tested, and they are listed in Table 3.3.

In the first iteration, we train the models for 150 epochs, validating every 20th. We test the models on the validation set and extract results for all the tolerances from 5 to 60 with an increasing step of 5 between each tolerance. The results for the configurations we tested with 150 epochs are presented in Section 4.1.2. Based on these results, we selected three configurations we wanted to use in our further experiments with multi-

Chunk size	Receptive field
60	5
60	20
120	5
120	20
120	40
180	5
180	20
180	40
180	60

Table 3.3: The 9 different combinations of chunk sizes and receptive fields we investigated with the CALF model.

modal input. The selected configurations are *CALF-120-40*, *CALF-60-5* and *CALF-60-20*. We train these for 300 epochs, as we found it necessary with an increase in epochs based on the results from the models trained for 150 epochs. The results on the validation set for the selected models are presented in Section 4.1.2. In that section, we also present results to back up why we think 300 training epochs is sufficient to perform further experiments with the models.

Concatenated and audio features

As mentioned earlier, the initial plan was to train and test the CALF model with audio and concatenated features in the same way as we did with the visual features. This shown to be an issue in the training phase. The number of training epochs at 300, combined with validation testing at every 20 epochs had major problems. When we evaluated the results, we saw that the results were very poor and unstable. Especially for sound were the results useless, with mAP and average-mAP scores below 1%. This could mean that the models needed more training time, but given that the results were as low as they were, we rather thought it would be in the other end, and that validation every 20 epochs were too sparse to evaluate a working model before overfitting. Xiao et al. [81] also report results indicating that audio is more prone to overfitting earlier than video, so we investigate further with this in mind.

We experiment with different learning rates with little response. When training, it is the validations along the way that is the most time-consuming. This means that if we want to increase the frequency of validations, it would take a very long time to complete the training of each model if we continue with 300 training epochs. Therefore, we reduce the number of training epochs and experiment with increased frequencies for the validations. We present the findings from these experiments in Section 4.1.3, and based on what we learned we decided to keep a lower number of training epochs with a higher frequency of validations. For the model with concatenated features, we set the number of training epochs to 50, with validation every other epoch. For the model with only audio features, we set the number of epochs to 30, with validation every other epoch.

3.5 Evaluation metrics

When we want to assess the performance of a model, we need metrics to present objective measures. This gives us concrete numbers to work with and creates a foundation for our analysis of the performance. It could be tempting to use only one metric to create a simple way to compare models to each other, but results are rarely that trivial. A single metric could be misleading, and will often only cover a fraction of what we actually want to understand about the performance of the system. For example, a model could find 100 percent of every event, giving a recall of 100 only by predicting everything all the time. Even though this gives a perfect recall, it gives a horrible precision performance. The other way around, you can achieve great precision by only predicting an event when nearly 100 percent sure, but then achieve a terrible recall. This shows us that optimizing for one metric could affect others in a bad way, and by only looking at one metric one could miss important aspects of the performance of a system. By introducing several metrics we can assess multiple parts of a model's performance, and we can also optimize the system based on what metrics we find most important. With multiple metrics, you gain a deeper understanding, and could also to a higher extent find out where the system has the most potential for improvement.

To measure the performance of the tested models we have used standard metrics like accuracy, precision, recall, and F1-score in our experiments. A prediction is considered to be a *True Positive (TP)* when the model predicts the correct class, a *False Positive (FP)* when a class is incorrectly predicted, a *True Negative (TN)* when a class is correctly rejected, and a *False Negative (FN)* when a class is incorrectly rejected. This is defined in Table 3.4 .

	Positive ground truth	**Negative ground truth**
Positive prediction	True positive (TP)	False positive (FP)
Negative prediction	False negative (FN)	True negative (TN)

Table 3.4: Definition of true and false predictions

Accuracy is defined as the number of correct predictions over the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

Precision is a metric that describes how many of the predicted events that were correct predictions. It is formally defined in Equation 3.2

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

Recall is a metric that describes how many of the ground truth events were found by the model. This is done by dividing the correctly predicted events by the total number of events. It is formally defined in Equation 3.3

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

F1 is a metric that combines the precision and recall values through their harmonic mean. It is formally defined in Equation 3.4

$$F1 = \frac{2(Precision \cdot Recall)}{Precision + Recall} \tag{3.4}$$

We regard each class separately as a one-vs-all binary problem and consider a positive prediction as a possible true positive if it is within a tolerance δ of the ground truth event with a confidence equal or higher than our threshold. Formally, we use the condition in Equation 3.5:

$$|gt_{spot} - p_{spot}| < \frac{\delta}{2} \tag{3.5}$$

where gt_{spot} is a ground truth spot, and p_{spot} a predicted spot in seconds. We take predictions that match the criteria in Equation 3.5 and create unique pairs of predicted spots and ground truth spots. These are matched in a greedy fashion, where each ground truth spot is matched with the closest prediction. Predicted spots that have no match are considered a *False Positive (FP)*. For a given gt_{spot}, when no predictions are made where this condition holds, we consider it a *False Negative (FN)*. We use the condition in Equation 3.5 to calculate the Average Precision for each class:

$$AP = \sum_{n}(R_n - R_{n-1})P_n \tag{3.6}$$

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

$$mAP = \frac{\sum_{i=1}^{C} AP_i}{C} \tag{3.7}$$

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

3.6 Summary

In this chapter, we have described our approach to comparing how using both audio and visual information for event spotting and classification compare to approaches using only audio or visual information on the soccer-specific dataset SoccerNet. We started with introducing the SoccerNet dataset, describing how it is build up, and how we have used it. Further, we have explained that we have used three models for the task.

The first model is the state-of-the-art model on SoccerNet called the CALF model, and this is tested on the spotting task of SoccerNet. This model is tested with both audio, visual, and concatenated audio-visual features. The second model is an audio model that converts the sound from the original videos into spectrograms, and train a ResNet classifier with these as input. This model is also used as an audio feature extractor, by removing the last layer of the model. The third model is a 2D-CNN model used with visual input, to compare to the performance of the audio model. The 2D-CNN model is also used to create an audio-visual approach through *late fusion* with the audio model, by fusing the softmax-output from both models. Further, we have described that we wanted to test several configurations of the CALF model, and how we chose the three configurations we call *CALF-120-40*, *CALF-60-20*, and *CALF-60-5*. We have also presented which window sizes we have tested for the audio and 2D-CNN models, and which were used for the audio feature extractor. Finally, we presented the evaluation metrics that we have used to analyze the performance of the models.

Chapter 4

Experiments and Results

In Chapter 3, we found that we wanted to test two different multi-modal approaches on the SoccerNet, one for the task of action spotting and one for the task of classification. We decided to test *early fusion* for the spotting task with the CALF model by extracting audio features from the dataset and concatenating them with the supplied visual ResNet features. We decided to try this approach for three configurations of the model, which we named *CALF-120-40*, *CALF-60-20*, and *CALF-60-5*. For the classification task, we created an audio model, which takes generated spectrograms as input. This was combined with a visual 2D-CNN model through *late fusion*, by taking the softmax average scores of the two models. This approach is tested with five different window sizes varying from 2 to 32 seconds.

We will start by presenting the results from the preliminary experiments performed to select and configure our approaches. For the preliminary experiments, we used the validation set to obtain the presented results. In the further experiments for action spotting and classification, the testing is performed on the test set. After the preliminary experiments, we will present the results for the action spotting approach, before we will continue with the results for the classification models. For both tasks we will start by describing the overall results before we will analyze the results class-wise to observe the differences between the event types. Further, we compare the results we have observed in this thesis to other approaches tested on the SoccerNet, and discuss how the results stand in a bigger perspective when it comes to usefulness in a real-life scenario. We also discuss some extra factors that possibly could affect the results on this dataset.

4.1 Preliminary experimental results

In the process of selecting hyperparameters for our models, we have performed several preliminary experiments to help us make a good choice. In this section, we present the results from these experiments, which formed a foundation for our hyperparameter selection in Section 3.4.

4.1.1 Window size for classification models

For the classification models, we tested the audio model, the visual 2D-CNN model, and the combined models with softmax average/max on the validation set to obtain valuable insights to use in the hyperparameter selection. We tested models with 2, 4, 8, 16, and 32 seconds as window sizes since this gives a nice variety how much audio the model gets as input. We trained all the models for 25 epochs.

From Table 4.1, we observe that all models show a gradual increase in accuracy as the window sizes get bigger. The visual 2D-CNN model achieves the highest accuracy for the four smallest window sizes with 81.08% for window size 2, 84.26% for window size 4, 87.08% for window size 8, and 89.74% for window size 16. The model achieving the closest results for these four window sizes is the combined model which uses a softmax average. This model achieves an accuracy of 78.72%, 82.31%, 86.21%, and 89.38% for these four window sizes in increasing order, so it is not very far behind the visual model. The combined model which is combined though softmax max achieves very similar results to the softmax average model, with 0.26% lower accuracy for window size 2, equal accuracy for window size 4, 0.42% lower for window size 8, and 0.35% lower for window size 16. For the biggest window size, the two combined models outperform both of the single modality models, with a max accuracy score for the softmax average model at 92.05%. This is followed by the softmax max model at 91.59% and the visual 2D-CNN at 90.67%.

	Accuracy (%)			
Window size	**Audio**	**Visual**	**Softmax average**	**Softmax max**
2	63.54	**81.08**	78.72	78.46
4	68.26	**84.26**	82.31	82.31
8	72.15	**87.08**	86.21	85.79
16	73.90	**89.74**	89.38	89.03
32	75.49	90.67	**92.05**	91.59

Table 4.1: Comparison of the accuracy (%) of classification on the validation set, for different models and fusion alternatives. The audio model is described in Section 3.2.2 and the visual 2D-CNN model with pre-extracted ResNet features is described in Section 3.2.3. The fusion of the audio and video models is performed using *late fusion* (either softmax average or softmax max), as described in section 3.3.2.

For all the window sizes, we observe that the audio model is the model that achieves the lowest accuracy. It starts with 63.54% at window size 2 and increases for all window sizes until it reaches 75.49 at window size 32. This means that the audio model stays double digits behind the visual model for all window sizes, with the closest window size being 8, where the accuracy is 14.93% lower.

A trend we observe is that the model that is combined through softmax average outperforms the model that is combined through softmax max for all window sizes. The difference is not very big, but, consistently, the softmax average always performs a little bit better. As these two approaches are fairly similar, but with the softmax average consistently achieving a better accuracy, we choose to continue with the softmax average model representing the combined model approach in the experiments on the test set.

4.1.2 Chunk size and receptive field for visual input for CALF

We performed experiments for the CALF model with visual input to select which configurations we wanted to test with multi-modal input, as described in Section 3.4.2. In this section we present the results from the experiments with different chunk sizes and receptive fields for both 150 and 300 epochs. We also present the results used to back up why we found 300 epochs sufficient for our further experiments.

We test the models on the validation set and extract results for all the tolerances from 5 to 60 with an increasing step of 5 between each tolerance. In the tables where results are shown, we present the results for the tolerances 5, 20, 40, and 60, as presenting all would decrease the readability of the tables with an overflow of information. With these four tolerances, in addition to the Average-mAP which is produced using all the tolerances, we can still show the essence of the performance for different areas of tolerances.

150 epochs

We observe from Table 4.2 that for average-mAP, it is the *CALF-120-40* that is the top-performer. This is as expected, as this is the configuration used in the original paper [18]. This configuration achieves an average-mAP of 63.80%, with the configuration closest being CALF-120-20 at 61.42%. *CALF-120-40* also achieves the best results for all the three tolerances from 20-60, while the *CALF-60-20* achieves the best score for the smallest tolerance at 5. For the tolerance 5, we observe that it is the two models with chunk size 60 that performs best, with the one with a receptive field of 20 clearly ahead, with a score of 38.02% against a score at 32.90% for the one with a receptive field of 5. This is a pretty low score but shows that these models perform significantly better than the competitors when we demand a very precise prediction, as the closest performance for tolerance 5 was for the CALF-120-20 with a mAP score at 28.88%. We observe that it is clearly divided between chunk sizes for this tolerance, where lower chunk sizes are preferable over bigger ones. When assessing the impact of the receptive field for tolerance 5, we observe that it generally seems to favor increasing receptive fields when comparing models with the same chunk size. The exception is for the receptive field of 20 and 40, where both chunk size 120 and 180 show a very small mAP-increase in favor of the receptive field of 20.

We observe a gradual increase in mAP scores when we increase the tolerance for how far away from the ground truth event it can be a prediction and still be considered a true positive. The biggest change in mAP is between the tolerances of 5 and 20. For the tolerance of 20, several

CALF model		**mAP**				
Chunk size	**Receptive field**	**tolerance = 5**	**tolerance = 20**	**tolerance = 40**	**tolerance = 60**	**Average-mAP**
60	5	0.3290	0.5232	0.5740	0.5849	0.5375
60	20	**0.3803**	0.5639	0.6168	0.6381	0.5831
120	5	0.2421	0.5379	0.6196	0.6431	0.5610
120	20	0.2888	0.5915	0.6681	0.6815	0.6142
120	40	0.2885	**0.6191**	**0.6902**	**0.7137**	**0.6380**
180	5	0.1081	0.4649	0.5875	0.6413	0.5066
180	20	0.2138	0.5592	0.6568	0.6720	0.5861
180	40	0.2118	0.5866	0.6715	0.6949	0.6054
180	60	0.2207	0.6048	0.6738	0.7111	0.6092

Table 4.2: Results on the validation set for 9 different configurations of the CALF model. Trained on 150 epochs.

of the configurations have exceeded the *CALF-60-20* model, with the best performing being the *CALF-120-40* at 61.91%. With this tolerance, we observe a general preference for bigger receptive fields, now also between the receptive fields of 20 and 40. This time we also observe that the chunk size of 120 is outperforming the other two chunk sizes when looking at models with equal receptive fields. When comparing this way we also observe that chunk size 60 is achieving higher scores than chunk size 180 for equal receptive fields, but that CALF-180-40 and CALF-180-60 is still performing better than the best configuration for chunk size 60. When moving up to a tolerance of 40 we observe that 120-40 still performs best, and the same preference for bigger receptive fields. Comparing the same receptive fields we observe that chunk size 120 is still the best, but this time 180 achieves higher scores than 60. When we look at the results for the final tolerance of 60 seconds, we observe the same results in terms of which configurations perform best, but with a small increase in the scores for all models.

When we consider the number of epochs trained, we first notice that the performance of the *CALF-120-40* model performs pretty similar to the model with the supplied trained weights. The scores only differ with 0.3%, and to a certain extent, this confirms our belief that reducing epochs does not damage the performance too much. When we take a look at the training data, we also extract information about how many epochs the models had trained without improving performance. This information is listed in Table 4.3. We test this on the validation set every 20th epoch, so we know that the number of epochs will move in steps of 20. We observe that even though we see a very similar result as the pretrained weights we compare to, the average number of epochs since the best weights were found is only approximately 22. This could imply that it is still some noticeable movement in the training. Therefore we want to choose the models we want to test further and train them for a longer period of time.

300 epochs

To select some of the models, we evaluate the performance of the models and try to assess which configurations we believe could be of interest to combine with audio features. Primarily, we want to continue with the

CALF model		
Chunk size	Receptive field	Epochs since best model found
60	5	0
60	20	40
120	5	20
120	20	20
120	40	20
180	5	60
180	20	20
180	40	20
180	60	0
Average		22.2222

Table 4.3: Number of epochs since it was last found a new best model for each configuration when the training of 150 epochs was finished

CALF-120-40 model, as this achieves the best results for all tolerances from 20 to 60, in addition to the best average-mAP. This is also the configuration chosen in the original CALF paper, and we find this configuration strong. Further, we want to test a model with a small receptive field, as we suspect the sound could benefit from small receptive fields in soccer. 5 is our smallest receptive field, so we assess the three models with this. We think *CALF-60-5* could be of interest, as this has the best score among the models with the same receptive field for the most accurate tolerance and a pretty similar score as 120-5 for a tolerance of 20. The final configuration we proceed with is the *CALF-60-20*, as this has the significantly best performance on the lowest tolerance. It also has a receptive field of size between the two other chosen models, so it fits well for our approach further, as it could help provide some insights into how the receptive field affects the performance of the model when combined with audio.

We double the number of epochs we train the models to 300 epochs. When looking at the number of epochs since the models last improved this time, we observe that the average is significantly higher than the last time, calculating to over 113. This is mainly because of the big number from the 60-5 model, but we see that for both of the other models, we also have numbers bigger than the average from the previous training. The number are shown in Table 4.4. We also observe that the average-mAP

CALF model		
Chunk size	Receptive field	Epochs since best model found
60	5	240
60	20	40
120	40	60
Average		113.3333

Table 4.4: Number of epochs since it was last found a new best model for each configuration when the training of 300 epochs was finished

has increased by 0.09% for the CALF 120-40 model when we double the epochs, while the *CALF-60-5* have increased with under 1%, and *CALF-60-20* decreasing the average-mAP by a negligible 0.05%. The *CALF-120-40* model trained for 300 epochs now has a difference from the pretrained weights of about 0.2% on the validation set. This could substantiate that to extensively train the models further would not gain too much performance. When we observe the results for the models, we see that the performances are pretty much as expected, with *CALF-120-40* performing best for average-mAP and tolerances from 20-60, while *CALF-60-20* still performs significantly better at the tolerance of 5. The biggest surprise is that the *CALF-60-5* has decreased the results for the tolerance of 5, which means that it has increased for the higher tolerances since the average-mAP has increased. These results are shown in Table 4.5

CALF model		**mAP**				
Chunksize	**Receptive field**	**tolerance = 5**	**tolerance = 20**	**tolerance = 40**	**tolerance = 60**	**Average-mAP**
60	5	0.2745	0.5494	0.5907	0.6021	0.5470
60	20	**0.3769**	0.5685	0.6110	0.6326	0.5826
120	40	0.2940	**0.6333**	**0.6738**	**0.7124**	**0.6389**

Table 4.5: Results on the validation set for 3 different configurations of the CALF model. Trained on 300 epochs.

4.1.3 Epochs and validation frequency for audio and concatenated input for CALF

Initially, we wanted to train the CALF model for the same number of epochs with the same validation frequency for visual, audio, and concatenated input. This was not possible as the audio and concatenated input experienced troubles during training for these epochs and validation frequency. In this section, we present the findings of the experimenting we did to select the hyperparameters for these two input types in Section 3.4.2. We suspected that overfitting could be the problem, so we tried different approaches to deal with the problem. We experimented with different learning rates with little response. We experimented further with lower validation frequencies, and combined this with a lower number of training epochs, as it would take a long time to complete the training with a combination of many epochs and frequent validation.

When we did this, we observed that the performance increased early, before suddenly dropping drastically. This keeps us thinking that it could be some kind of overfitting and that the model is vulnerable to this when trained with our audio features. For the concatenated features, this approach seems to get the performance up to more normal levels, and we observe performance more as expected. For the audio features, the training is still very unstable, but at least we see that for some configurations the model is capable of learning from the audio features, even if the performance is still low.

Since we observe that we get results more as expected on the validation set for concatenated features, we decide to stay with this approach but to validate more frequently with a lower number of total training epochs. To avoid the overfitting problem we set the number of epochs between each validation to 2, and the number of epochs to 50. This effectively deals with the unstable training problem for the concatenated features, at least on the validation set that we used to validate the model.

For the audio features, we choose even fewer epochs of training, as the performance drops after very few epochs of training. We choose 30 epochs and have validation every other epoch. We still have big issues with unstable training and bad performance. The results for the validations when training with the audio features were peaking even earlier than for the concatenated features, and for some of the configurations, it is not obtained useful performance at all. As we can not exclude the possibility that it is because it is just how audio features perform with this model, even though this is unlikely given the results of other models with audio, we decide to keep the results and show them together with the visual and concatenated features. We discuss some perspectives on why the CALF model with our audio features performs as it does in Section 4.4.1.

4.1.4 Non-maximum Suppression (NMS)

For our testing with the CALF models on the action spotting task, we chose to apply Non-maximum Suppression (NMS) on the results. This was done based on experiments on the results on the validation set with visual input. The effect for the different configurations we tested are presented in Table 4.6 As we observe from this table, all the configurations benefit from NMS. The configurations showing most increase in performance with NMS are the ones with the lowest chunk size. It seems like the the difference between the results with and without NMS decreases on average as we move towards bigger chunk sizes. We performed this experiment for all the 9 configurations of CALF that we trained for 150 epochs, and based on these number we chose to continue using NMS for the CALF configurations.

CALF model		Average-mAP	
Chunk size	**Receptive field**	**With NMS**	**Without NMS**
60	5	0.5375	0.4504
60	20	0.5831	0.4712
120	5	0.5610	0.5150
120	20	0.6142	0.5302
120	40	0.6380	0.5644
180	5	0.5066	0.4747
180	20	0.5861	0.5583
180	40	0.6054	0.5852
180	60	0.6092	0.5730

Table 4.6: Results on the validation set with and without NMS for the CALF models we trained for 150 epochs. This shows a significant advantage of using NMS.

4.1.5 Summary of preliminary experiments

In this section, we have presented the results from several preliminary experiments. These were performed on the validation set and formed a foundation for our selection and configuration of models to test. For the classification models, we test with the window sizes 2, 4, 8, 16, and 32. We selected the combined model that uses softmax average to be the combined model we test further. To extract the audio features, we selected the model with a window size of 8 seconds. These features are concatenated with the supplied ResNet features from SoccerNet and used as input to the CALF model. For the CALF model, we experimented with a variety of chunk sizes and receptive fields, and selected the three configurations we call *CALF-120-40*, *CALF-60-20*, and *CALF-60-5*. For visual, audio, and concatenated input, the models are trained for respectively 300, 30, and 50 epochs, and NMS is applied to the results.

In the further sections, we present the results from the experiments performed with the selected models, and the results presented are results from the test set.

4.2 Action spotting results

For the action spotting task, we have tested three different configurations of the CALF model, and all three are tested with visual, audio, and concatenated audiovisual features as input. We first analyze the overall results for the action spotting task for all three classes combined and compare the performance for the different input features. Then, we look into the class-wise performance for the different inputs.

4.2.1 Overall results

		Tolerance = 5			Tolerance = 20			Tolerance = 40			Tolerance = 60			
CALF Model	**Input type**	**Precision**	**Recall**	**mAP**	**Precision**	**Recall**	**mAP**	**Precision**	**Recall**	**mAP**	**Precision**	**Recall**	**mAP**	**Average-mAP**
60-5	Audio	NaN	0.0735	0.0010	NaN	0.1913	0.0075	NaN	0.3721	0.0187	NaN	0.3501	0.0293	0.0145
60-5	ResNet	0.1551	0.2947	**0.2545**	0.4385	0.5507	0.5113	0.5488	0.6110	0.5495	0.5766	0.6147	0.5634	0.5092
60-5	Concat	0.1425	0.1729	0.2123	0.4330	0.4780	**0.5209**	0.5453	0.5547	**0.5998**	0.5889	0.6135	**0.6156**	**0.5408**
60-20	Audio	0.0695	0.0704	0.0771	0.2416	0.1505	0.1940	0.3146	0.2084	0.2321	0.3729	0.2117	0.2475	0.2069
60-20	ResNet	0.3259	0.3069	**0.3655**	0.6401	0.5117	**0.5493**	0.6882	0.5327	**0.5871**	0.6145	0.5790	0.5971	**0.5574**
60-20	Concat	0.2419	0.2303	0.2769	0.5290	0.4659	0.4915	0.5984	0.5016	0.5683	0.6176	0.5398	**0.6136**	0.5231
120-40	Audio	NaN	0.0374	0.0007	NaN	0.1519	0.0045	NaN	0.1585	0.0161	NaN	0.2454	0.0264	0.0123
120-40	ResNet	0.2195	0.2535	**0.2869**	0.6383	0.5819	**0.6067**	0.7199	0.6438	**0.6425**	0.7446	0.6506	**0.6530**	**0.6007**
120-40	Concat	0.1681	0.1855	0.2139	0.5749	0.4674	0.5579	0.6254	0.5836	0.6106	0.6602	0.5931	0.6345	0.5629

Table 4.7: Performance of the model with different configurations, tested on the test set. The numbers describing the CALF model are the chunk size (first number) and receptive field (second number). The mAP values indicate the Average Precision values averaged over all event types. The NaN values means "Not a Number", and occurs when it is tried to divide by zero in the calculations for precision. This means that it was made 0 predictions for one of the event types.

For the overall results from Table 4.7, we observe that the best performance for all tolerances combined (based on average-mAP) are *CALF-120-40* with only visual ResNet features as input with a score of 60.07%. It is followed by the *CALF-120-40* model with concatenated input at 56.29% and *CALF-60-20* with visual input at 55.74%. This shows us that also when using concatenated audio-visual input, the configuration using chunk size 120 with a receptive field of 40 performs best among our selected configurations when it comes to average-mAP. For the average-mAP, we observe that the models using only visual input outperform the concatenated input for two of the three configurations, with *CALF-60-5* being the only model where concatenated input achieves the best results (54.08%). This is an over 3% increase from the same model with only visual input. This could imply that concatenated features are more beneficial compared to an equal model with visual features when the receptive field is low. This is because it is only observed for *CALF-60-5*, and not *CALF-60-20*, where the only difference is the receptive field. The *CALF-60-20* also has a lower average-mAP than *CALF-60-5* for the concatenated input. This could mean that the concatenated features also in total benefits from smaller receptive fields, and not only when compared to an equal configuration with different input. Further, we also observe in all cases that the audio input alone performs worst, with maximal average-mAP at 20.69% for *CALF-60-20*. The two other configurations with audio input have an average-mAP of 1.23%, and 1.45%, which imply that the models have learned close to nothing. In total for the average-mAP, we see the tendencies of the model performing better with bigger chunk sizes, and that the best results are gained through visual input alone.

Results per tolerance

We do not want to only look at the average-mAP, as this metric only gives some insights into how well the models perform across all tolerances for the mAP-scores. When observing the results for the lowest tolerance, which is 5, we see that *CALF-60-20* with visual input is winning by quite a bit, having a 7% margin down to the closest score. The best score is at 36.55%, while the runner-up is the *CALF-120-40* with visual input at 28.69%, followed by *CALF-60-20* with concatenated input a little over 1% further back. For this tolerance, we observe that visual features are the dominating input type, as all three configurations perform best with this input here. Further, we observe that the recall values are better for visual input, with a significant increase from concatenated input to visual input. The scores are still at 30.69% at best, and therefore still not very useful. The precision values are a bit less different on average, but still clearly in favor of visual input for tolerance 5.

Not surprisingly, after observing the average-mAP results for audio-only, we see that the models using only audio are performing poorly, with two of them at around 0.1%, and *CALF-60-20* at 7.71%. For the two worst models we also observe some Not a Number (NaN) values for the precision metric. If we again look at Equation 3.2 we see that the only possibility for this to happen is if we divide by zero. This could only happen if both true positive, and false-positive predictions are zero, meaning that no predictions were made. As the recall score is not zero, we know that it is not the case for the model in total, which means that this problem comes from one of the classes not predicting any events. When one class has a NaN precision, this will affect the calculations when combining this metric for all three classes, giving a total NaN score, since combining with NaN is not possible. This problem continues through all tolerances for audio input for *CALF-60-5* and *CALF-120-40*.

When moving up to a tolerance of 20, we observe that this is the first tolerance where the concatenated input achieves better mAP than the visual input for *CALF-60-5*. From tolerance 20 and out, this is the case for this model. It is still not the best-performing model among all. With a mAP score of 52.09%, it is beaten by both of the other models for a tolerance of 20. *CALF-120-40* is still the winner with a mAP score of 60.67% for visual input. We observe that in addition to the increased mAP scores, the recall and precision scores have increased by a lot when moving the tolerance from 5 to 20. The recall values for the best performing model (*CALF-120-40*) is for this tolerance at 58.19%. This means that we have found over half of the events in the test set by a window of 10 seconds of each side of the center frame where the event is annotated. We see that even though the audio model still has the NaN values for precision, the recall values here have increased to between 15-19% for the various configurations. This could mean that the audio models are at least learning something.

For a tolerance of 40, we observe that the concatenated input for *CALF-60-5* achieves the highest mAP score among the input types for this model (59.98%), as well as outperforming the best mAP score for *CALF-60-20*. It is still outperformed by the *CALF-120-40* with both visual (64.25%) and concatenated (61.06%) input. This means that *CALF-120-40* still performs best both for visual and concatenated input. When it comes to precision and recall we observe that for the *CALF-120-40* with visual input, the precision value has surpassed 70% by almost two percent. With a recall value at 64.38%, we see that it is a significantly better combination than any of the other models with a tolerance of 40.

Finally, we observe the highest tolerance at 60 seconds, with still the visual *CALF-120-40* as the best performing model at an mAP score at 65.30%. What is particularly interesting at this tolerance is that both of the models with chunk size 60 have their best performance with concatenated input, with *CALF-60-5* barely outperforming *CALF-60-20* with 61.56% against 61.36%. This is a pretty close race but still favors the smaller receptive field over the bigger, when the chunk size is equal. When we increase the chunk size on the other hand, we observe that it is still *CALF-120-40* that performs the very best in total for concatenated features (63.45%). This could mean that the concatenated features benefit from bigger chunk sizes, even though it seems to prefer smaller receptive fields, and that chunk size is of even bigger importance.

For the overall results of the action spotting task, we observe that for most of the models the visual features alone were to prefer, as this performs best when it comes to mAP scores. The exceptions were the *CALF-60-5* model, having the best scores for the concatenated input for most tolerances, and the *CALF-60-20* which had the best score for concatenated input at a tolerance of 60. Still, none of these were able to outperform the best visual model. We also found that for equal chunk size the model seems to prefer smaller receptive fields for the concatenated input, but that chunk size affects the results even more, as *CALF-120-40* still performed better than the other two also for concatenated input.

4.2.2 Class-wise results

Until now, we have presented the results combined over all classes, as this says something about how the models perform as a whole for the events in the dataset. But, this is not the only information of interest when we evaluate the performance of the models. The three events are different, and with different actions, it could also be that it differs from event to event how much the modalities could help to identify the event. One could imagine the audio input can make a bigger impact on certain events with bigger audible cues, for example, the loud celebration after a goal has been scored. Therefore, we are curious to evaluate how the models perform when we observe only one class at a time. We will observe the results for the different input types, and evaluate their performance for the individual events. We will also compare the strengths and weaknesses of the different input types to each other. The results for the models on the different types of events are shown in Table 4.8.

4.2.3 Class-wise results with visual input

The first class-wise results we are presenting are for the visual ResNet features. This uses the same input to the model as in the original CALF model[18].

Average Precision results

We start by getting an overview of the performance over the different tolerances by looking at the average across the Average Precisions (AP). We observe that for all three configurations that goals are the easiest to predict, with around 9% and 10% margin to the closest event type. The *CALF-120-40* is not surprisingly the best performer, as this is the configuration chosen in the original paper which uses visual input, and also performs best when we look at all classes combined. For goals, it has an average of 69.12% Average Precision, with *CALF-60-20* being the closest competitor at 66.53% and *CALF-60-5* achieving a score of 61.11%. Further, we observe that the second easiest event to predict on average is substitutions, again with a clear margin to the last event, being cards. The order of how the configurations perform for substitutions from best to worst is the same as for goals, with *CALF-120-40* performing best at 60.06%. When compared to the best score for cards, which is *CALF-120-40* with 47.75%, we observe that it is an over 12% difference in performance between the best models for substitutions and cards. With the performance for goals at over 9% better than substitutions again, we see that it is a significant difference in performance between the different event types.

			Tolerance = 5			Tolerance = 20			Tolerance = 40			Tolerance = 60			
CALF Model	**Class**	**Input type**	**Precision**	**Recall**	**AP**	**Precision**	**Recall**	**AP**	**Precision**	**Recall**	**AP**	**Precision**	**Recall**	**AP**	**Average-AP**
60-5	Card	Audio	0.0030	0.0667	0.0009	0.0096	0.2133	0.0089	0.0197	0.4378	0.0239	0.0254	0.5644	0.0398	0.0183
60-5	Card	ResNet	0.1271	0.2733	**0.2031**	0.3489	0.4978	**0.3748**	0.4348	0.5556	0.4067	0.4486	0.5622	0.4184	0.3701
60-5	Card	Concat	0.1455	0.0689	0.1242	0.4673	0.2222	0.3716	0.5779	0.2556	**0.4218**	0.6109	0.3000	**0.4427**	**0.3728**
60-5	Sub	Audio	0.0022	0.1537	0.0021	0.0102	0.3604	0.0134	0.0192	0.6784	0.0320	0.0275	0.4859	0.0479	0.0254
60-5	Sub	ResNet	0.1464	0.2827	**0.2148**	0.4353	0.5530	**0.5323**	0.5668	0.6148	0.5868	0.6185	0.6131	0.6007	0.5213
60-5	Sub	Concat	0.0941	0.2014	0.1421	0.2912	0.5583	0.5003	0.4183	0.6784	**0.6311**	0.4732	0.7491	**0.6528**	**0.5299**
60-5	Goal	Audio	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	0
60-5	Goal	ResNet	0.1918	0.3282	0.3456	0.5312	0.6012	0.6269	0.6448	0.6626	0.6551	0.6626	0.6687	0.6709	0.6111
60-5	Goal	Concat	0.1879	0.2485	**0.3705**	0.5406	0.6534	**0.6909**	0.6398	0.7301	**0.7465**	0.6825	0.7914	**0.7514**	**0.6879**
60-20	Card	Audio	0.0229	0.0178	0.0101	0.0609	0.0156	0.0580	0.0879	0.0356	0.0761	0.1069	0.0311	0.0885	0.0618
60-20	Card	ResNet	0.2574	0.2511	**0.2973**	0.5012	0.4511	**0.4144**	0.5293	0.4622	**0.4495**	0.4395	0.5089	0.4670	**0.4238**
60-20	Card	Concat	0.2243	0.1889	0.1891	0.4330	0.3444	0.3647	0.5044	0.3800	0.4391	0.5221	0.4200	**0.4676**	0.3865
60-20	Sub	Audio	0.0251	0.1413	0.1193	0.1104	0.3410	0.2367	0.1623	0.4576	0.3111	0.2194	0.4753	0.3439	0.2695
60-20	Sub	ResNet	0.2832	0.2862	**0.3244**	0.6297	0.5318	**0.5594**	0.7048	0.5654	**0.6177**	0.6161	0.6237	**0.6231**	**0.5640**
60-20	Sub	Concat	0.1063	0.2014	0.1779	0.3040	0.5318	0.3896	0.3974	0.5848	0.4708	0.4320	0.6290	0.5550	0.4250
60-20	Goal	Audio	0.1604	0.0521	0.1019	0.5536	0.0951	0.2872	0.6935	0.1319	0.3090	0.7925	0.1288	0.3101	0.2782
60-20	Goal	ResNet	0.4371	0.3834	**0.4746**	0.7895	0.5521	0.6741	0.8304	0.5706	0.6941	0.7880	0.6043	0.7012	0.6653
60-20	Goal	Concat	0.3952	0.3006	0.4637	0.8500	0.5215	**0.7201**	0.8934	0.5399	**0.7951**	0.8986	0.5706	**0.8182**	**0.7383**
120-40	Card	Audio	0.0015	0.0222	0.0004	0.0068	0.1022	0.0042	0.0155	0.2333	0.0184	0.0261	0.3933	0.0343	0.0140
120-40	Card	ResNet	0.2103	0.2444	**0.2394**	0.5624	0.5311	**0.4802**	0.6268	0.5822	**0.5239**	0.6522	0.6000	**0.5399**	**0.4775**
120-40	Card	Concat	0.1505	0.1378	0.1587	0.5236	0.2956	0.4043	0.5230	0.4044	0.4453	0.5503	0.4133	0.4607	0.4012
120-40	Sub	Audio	0.0038	0.0901	0.0017	0.0099	0.3534	0.0091	0.0203	0.2420	0.0299	0.0287	0.3428	0.0449	0.0231
120-40	Sub	ResNet	0.1745	0.2155	**0.2341**	0.5685	0.5795	**0.6269**	0.6933	0.6590	**0.6660**	0.7218	0.6555	**0.6735**	**0.6006**
120-40	Sub	Concat	0.1047	0.1979	0.1562	0.3844	0.5053	0.5102	0.4682	0.6378	0.5629	0.5384	0.6572	0.5988	0.5021
120-40	Goal	Audio	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	0
120-40	Goal	ResNet	0.2737	0.3006	**0.3872**	0.7841	0.6350	0.7131	0.8396	0.6902	0.7375	0.8598	0.6963	0.7455	0.6912
120-40	Goal	Concat	0.2491	0.2209	0.3269	0.8167	0.6012	**0.7592**	0.8851	0.7086	**0.8237**	0.8919	0.7086	**0.8441**	**0.7507**

Table 4.8: Comparison of precision, recall and Average Precision per class (event type), for the CALF model with different configurations. The numbers describing the CALF model are the chunk size (first number) and receptive field (second number). The NaN values means "Not a Number", and occurs when it is tried to divide by zero in the calculations for precision. This means that it was made 0 predictions for the event type. Highlighted cells indicate the best Average Precision score in each individual experiment.

When we observe the results for the different tolerances in Table 4.8, we see that the tendency of goals being the easiest event to predict begins already at the lowest tolerance. The best model for Average Precision with tolerance 5 for goals is the *CALF-60-20*, with a score of 47.46%, with the closest model being *CALF-120-40* at 38.72%. *CALF-60-20* also achieves the highest score for both of the other event types with 32.44% for substitution and 29.73% for cards. This follows up on the earlier findings of *CALF-60-20* performing the best among the configurations for the lowest tolerance.

The order of which configurations performing best for tolerance 5 is the same for all three events, with *CALF-120-40* being second and *CALF-60-5* being last. Since the *CALF-60-20* performs best for this tolerance, we compare the performance of *CALF-60-20* for the three events. We observe that the order of event-difficulty is the same as for the average across tolerances, with goals at the top and cards at the bottom. However, we observe that the difference between substitutions and cards is smaller than for bigger tolerances. A reason for this could be that the event of giving a card happens faster, and does not necessarily keep giving hints over a longer period of time. This is different from both goals and substitutions, where the cues of the event actually happen for a longer period of time, even though it is temporally annotated to one single frame. This means that for cards you typically have a shorter window of time with interesting information, and if it is not predicted within a close tolerance of the event, it is less likely to predict it further away than for the other two events. The two other events could be predicted further away from the actual annotation, but still be predicted with actions related to the event on the screen. This can be backed up by looking at the increase of the performance when changing the tolerance from 5 to 20. When looking at these results for all the configurations we observe an average increase for goals at 26.89 and substitutions at 31.51. With cards on the other hand we observe an increase of only 17.65, showing that it increases less than the two other event types when moving up to a higher tolerance.

Another interesting takeaway for tolerance 5 is that for all events and all configurations except one, the best performing input is visual alone. The exception is for goals for *CALF-60-5*, where the concatenated input achieves the highest score. When we move up to a tolerance of 20 however, the visual input is not as dominating anymore, as the concatenated input takes over as the input type performing best for goals for all models. For the other two event types, the visual input still wins for tolerance 20. When observing the scores for this tolerance, we observe that the goals and substitutions have increased more in performance compared to tolerance 5 than the cards, as we mentioned earlier. We see that unlike for tolerance 5, that *CALF-120-40* is the configuration achieving the best Average Precision scores for tolerance 20. This applies to the rest of the tolerances as well, meaning that it is only for tolerance 5 that *CALF-120-40* does not achieve the highest scores for the three events. The best Average Precision score for tolerance 20 with visual input is 71.31% for goals, 62.69% for substitutions, and 48.02% for cards.

For the tolerances 40 and 60, we see pretty similar results to each other, with only a few percent improvements for a tolerance of 60. Something interesting for these is that for *CALF-60-5* the visual input is no longer the top-performing input for any of the event types. The input type performing the best for these tolerances for *CALF-60-5* is the concatenated features. In addition to the goals, which are won by concatenated input for all the configurations for these tolerances, we observe that for the highest tolerance the concatenated features win marginally for cards for *CALF-60-20* as well. For the other events the visual input performs best, and it is *CALF-120-40* that achieves the best Average Precision scores.

Confusion matrices

In addition to evaluating the Average Precision score, we create confusion matrices so we can understand further what the models actually predict. This way we see the numbers behind the precision and recall values and can assess some strengths and weaknesses of the different configurations. The confusion matrices describe which events have been predicted for the ground truth actions, and show what went wrong for failed or missing predictions. We create confusion matrices for each configuration for tolerance 40, as 20 seconds in each direction could be considered a reasonable distance for a true positive.

We begin with looking at the confusion matrix for *CALF-60-5* in Table 4.9. We observe that while it is a few predictions of wrong classes, it is a very small number compared to the number of events where no predictions were made, or where a prediction was made with no annotated event nearby. This means that it is a very small amount of confusion between classes. We observe that the confusion matrix reflects that goals are the easiest to predict with both the lowest number of false negatives and false positives. Even though the number of events is not perfectly balanced, the difference in performance is clear, giving goals both the best precision and recall. Even though the precision and recall for goals are better than for cards and substitutions, it is still not very good, and the *CALF-60-5* still has a lot of wrong predictions. For cards, we observe a more distinct reason for failure, as the number of false positives is much higher than missed ground truths. This shows through the precision and recall scores giving a lower precision than recall. The case is similar for substitutions, but the difference is smaller, resulting in a smaller difference between the two metrics.

		Ground truth			
		Goal	**Card**	**Sub**	**None**
Prediction	**Goal**	216	0	0	119
	Card	2	250	10	313
	Sub	5	5	348	256
	None	103	195	208	-

Table 4.9: Confusion matrix for *CALF-60-5* with visual ResNet input evaluated at a tolerance of 40. For ground truth, "none" means the absence of an event at a position where it has been predicted one. For predictions, "none" means that no class was predicted at a place where a ground truth event was present.

When moving on to the confusion matrix for *CALF-60-20* in Table 4.10, we observe some significant differences from *CALF-60-5*. Overall we see that it has made a much lower number of predictions. This shows for the number of predictions without a belonging ground truth event, which is a good thing, but also for the number of correctly predicted events. This means that the number of ground truth events that were not found by the model has increased, giving a worse recall than *CALF-60-5*. But in return, fewer predictions of non-existing events create a higher precision score. By reducing the number of predictions the *CALF-60-20* achieved a better result in total, as the precision of the model is so much better than for *CALF-60-5*, giving it a better Average Precision in total.

		Ground truth			
		Goal	**Card**	**Sub**	**None**
Prediction	**Goal**	186	0	0	38
	Card	2	208	3	180
	Sub	3	1	320	130
	None	135	241	243	-

Table 4.10: Confusion matrix for *CALF-60-20* with visual ResNet input evaluated at a tolerance of 40. For ground truth, "none" means the absence of an event at a position where it has been predicted one. For predictions, "none" means that no class was predicted at a place where a ground truth event was present.

When observing the confusion matrix for *CALF-120-40* in Table 4.11, we see that the number of correct predictions is higher than the other two for all the event types. We observe that the number of false positives is closer to *CALF-60-20*, but that it includes more true positives as well. With the fewest of false negatives among the configurations as well, this achieves a total score that outperforms the two other configurations. We observe that the particularly strong side of the model is the precision for goals, where the number of true positives is much bigger than for the false positives. This creates a precision score of almost 84% for goals, which is even better than the *CALF-60-20*, which was closest of the visual input models with approximately 83%. *CALF-120-40* also achieves a better recall score for goals, giving a combined result that is stronger. When comparing for substitutions, we observe that *CALF-60-20* achieves a higher precision score at just above 70%, while for cards the precision is better for *CALF-120-40* again. Combined with a greater recall score for all classes, this makes the *CALF-120-40* achieve the highest score in total.

		Ground truth			
		Goal	**Card**	**Sub**	**None**
Prediction	**Goal**	225	1	0	42
	Card	2	262	1	153
	Sub	2	3	373	160
	None	97	184	192	-

Table 4.11: Confusion matrix for *CALF-120-40* with visual ResNet input evaluated at a tolerance of 40. For ground truth, "none" means the absence of an event at a position where it has been predicted one. For predictions, "none" means that no class was predicted at a place where a ground truth event was present.

4.2.4 Class-wise results with audio input

The CALF model with audio features alone as input has in our experiments not performed well. As already mentioned in Section 3.4.1, it had major troubles under training, so when we observe the results on the test set it is not surprising that it achieves weak results. As we described in Section 4.2.1 the model with audio input had results with a maximum of a few percent for *CALF-60-5* and *CALF-120-40*, while *CALF-60-20* had somewhat more meaningful results at about 20%. But, we do not only want to observe the results combined over all classes, we also want to observe if there are differences between classes. This way we can gain some insights into why the models behave as they do.

When we observe the average-AP scores over all the tolerances for the classes, we immediately notice that the two weak configurations both get a score of 0 for goals. They show 0 for Average Precision and recall and a Not a Number (NaN) value for precision. This NaN value means that it is tried to divide by 0 in the calculation. Since the equation divides by the sum of the true positives and the false positives it means that the models have predicted zero events. This also means that we have found the reason for the NaN values in Table 4.7, as the precision tries to combine the precision score for all three classes. Further, we observe that the two weak models actually provide some "normal" values for Recall and even have a recall score at 67.84% for the substitution event for *CALF-60-5* with tolerance 40. But, we also see that the precision score is at 1.92%, showing that the reason for the high recall is a huge number of predictions. This is confirmed by the confusion matrix for *CALF-60-5* in Table 4.12 where we see a huge number of predictions, which is the reason for the high recall score. We see the same concept, but with smaller numbers for *CALF-120-40* in Table 4.13. For these two confusion matrices, we also observe a much higher amount of predictions of the wrong class and not just false positives at spots without an event present. This could be due to the very high number of predictions, which means that some predictions would happen inside the tolerance of other events.

		Ground truth			
		Goal	**Card**	**Sub**	**None**
Prediction	**Goal**	0	0	0	0
	Card	63	197	120	9644
	Sub	233	215	384	19163
	None	30	38	62	-

Table 4.12: Confusion matrix for *CALF-60-5* with our extracted audio features as input, evaluated at a tolerance of 40. For ground truth, "none" means the absence of an event at a position where it has been predicted one. For predictions, "none" means that no class was predicted at a place where a ground truth event was present.

Moving on to the *CALF-60-20* model, which was the only of the three with some meaningful results, we observe that for the card event the results are significantly worse than for the two other events. It has an average AP of 6.18%, so compared to goals at 27.82% and substitutions at 26.95%, this is pretty low. This could mean that cards are just much harder to predict with sound alone, or that the model just failed to learn this class because of the earlier described problems for the audio input. It could of course be a combination, since the two other event types give much higher scores,

		Ground truth			
		Goal	**Card**	**Sub**	**None**
Prediction	**Goal**	0	0	0	0
	Card	53	105	117	6495
	Sub	111	149	137	6352
	None	162	196	312	-

Table 4.13: Confusion matrix for *CALF-120-40* with our extracted audio features as input, evaluated at a tolerance of 40. For ground truth, "none" means the absence of an event at a position where it has been predicted one. For predictions, "none" means that no class was predicted at a place where a ground truth event was present.

and the result is still several times better than for the two other models. This could indicate that cards could be harder to spot by sound. It might be generally more and louder noise from a celebration after a goal, or applause during substitutions. This way it makes sense that cards are harder to spot by sound.

When observing the results for goals and substitutions we see that they are pretty comparable, and which performed best switched throughout the tolerances. For the tolerance of 40, the goals have an Average Precision of 30.90%, while substitutions have 31.11%. We observe that for goals the precision is pretty high at 69.35%, as it is a low number of predictions for this event, with many of them correctly predicted. But, with finding few of the total number of goals it ends up with a recall at 13.19%. Opposite of the goal event, the substitution achieves high recall, with a score at 45.76%, but has a high number of predictions, giving a low precision at 16.23%. What is also interesting is that when the model predicts a goal wrong, it is rare that it is another event, but is more likely to not be nearby any other event. When it is a ground truth goal on the other hand, it is almost as likely to predict one of the other events as a goal, if an event is predicted. It is interesting that these mispredictions only go in one direction, as it might have been natural to think that if two events are similar in the sound they should both predict each other. Again, substitutions are opposite to goals, as the probability of predicting any other event when the ground truth is substitution is very low. There are also some predictions of substitutions that had another class as ground truth, but this might also be due to the high number of predictions for substitution.

		Ground truth			
		Goal	**Card**	**Sub**	**None**
Prediction	**Goal**	43	2	0	17
	Card	33	16	4	130
	Sub	9	22	259	1305
	None	241	410	303	-

Table 4.14: Confusion matrix for *CALF-60-20* with our extracted audio features as input, evaluated at a tolerance of 40. For ground truth, "none" means the absence of an event at a position where it has been predicted one. For predictions, "none" means that no class was predicted at a place where a ground truth event was present.

4.2.5 Class-wise results with concatenated input

Finally, we observe the results for the CALF models with concatenated audio-visual input. This is maybe the most interesting input, as this is the *early fusion* approach we wanted to compare to visual-only input.

Average Precision results

When we observe the results we see that for all models this input increases the average AP scores for goals compared to visual input. For *CALF-60-5*, we also observe that the average AP has increased for all three event types. This is not the case for the two other models, as these show a reduction in average-AP for card and substitution. Since we observe an increase for all events for *CALF-60-5*, but not for *CALF-60-20*, it could imply that when compared to visual-only input for the same configuration, concatenated input works better for smaller receptive fields. Even though the scores for *CALF-60-5* have increased compared to visual input for *CALF-60-5*, the actual score is still better for both goal and card for *CALF-60-20*. This is why the total average-mAP score for *CALF-60-20* is better, even with *CALF-60-5* achieving better results in total for substitution. *CALF-120-40* achieves even higher scores for goals and cards, thus being the best performer for these events. But, for substitutions the *CALF-60-5* also outperforms *CALF-120-40*, meaning that it performs best among the models with concatenated input for this event.

When we observe the best performing average scores for the three event types with concatenated input, we find that the order of difficulty is the same as for visual input. This means that goals is the easiest with its 75.07%, followed by substitutions at 52.99%, with cards last with 40.12%. These scores show that even though the order of events is the same as for visual input, it is bigger differences between goals and the two other events for concatenated input. This is because concatenated features have shown a positive effect for goals and a negative effect for cards and substitutions for the best performing models.

We also observe the Average Precision for the different tolerances and start with tolerance 5. What is particularly interesting for this tolerance is that the reason why *CALF-60-20* is the best performer for mAP is that it achieves a high score for goals compared to the other configurations. It achieves an Average Precision of 46.37%, which is over 9% better than *CALF-60-5*, which is the closest score for goals with concatenated input. Interestingly, *CALF-120-40* is performing worst among the models for this tolerance, even though it achieves the best average-AP score for goals. *CALF-60-20* also performs best for both of the other two events, but with considerably lower scores. For this tolerance, both *CALF-120-40* and *CALF-60-20* also achieve higher scores for cards than substitutions, but the scores are low, and the differences small, so it might not be of much interest.

When moving to the tolerance of 20 we observe the same tendency as for visual input, with goals and substitutions increasing the scores more than for cards. We still believe that a possible reason for this could be that the event of showing a card often has a shorter interval of valuable information around the annotated frame than substitution and goals, which are events with cues over a somewhat longer period of time. This means that if the card given is not spotted by the model by the time of the lowest tolerance, it is a lower chance for it to predict it with further distance than for the other two events, as more of the cues are gone. We do not think that introducing sound to the input features should change this thought notably in one way or another. For tolerance 20 and above we observe that the concatenated input performs better than visual for goals for all models. For tolerance 20 we observe that among the models with concatenated input, the *CALF-120-40* has taken over as the best performing model for all three classes. For goals, the Average Precision has increased to 75.92%, which is considerably higher than substitutions at 51.02% and cards at 40.43%. Among *CALF-60-20* and *CALF-60-5* it is *CALF-60-20* that achieves the highest score for goals, while *CALF-60-5* wins for both cards and substitutions.

With the tolerances 40 and 60, we observe that *CALF-60-5* achieves best scores with concatenated input for all events. For the other two models, the best Average Precision score for goals is achieved with concatenated input. With tolerance 60, *CALF-60-20* also achieves best score for cards with concatenated input. Further, we observe that for these two tolerances (40 and 60) the *CALF-120-40* with audio-visual input does not perform best in total for all three classes anymore. *CALF-60-5* achieves better scores for substitutions for both tolerances, while *CALF-60-20* performs best for cards at tolerance 60. The highest Average Precision scores among the classes are still goals for all models. We again find that the smallest increase in Average Precision is found when moving from tolerance 40 to 60.

Confusion matrices

		Ground truth			
		Goal	**Card**	**Sub**	**None**
Prediction	**Goal**	238	6	4	124
	Card	0	115	1	83
	Sub	3	15	384	516
	None	85	314	177	-

Table 4.15: Confusion matrix for *CALF-60-5* with concatenated audio-visual input evaluated at a tolerance of 40. For ground truth, "none" means the absence of an event at a position where it has been predicted one. For predictions, "none" means that no class was predicted at a place where a ground truth event was present.

We also want to look into confusion matrices for concatenated input, so we create confusion matrices for a tolerance of 40 again. We start with the confusion matrix for *CALF-60-5* in Table 4.15. The first thing we notice when observing the confusion matrix is that the model has a higher number of total predictions for substitutions than for the two other events combined. This follows the pattern of how the models have behaved on visual and audio input from before. With this, we mean that it generally has been predicted more substitutions and that audio features had a much higher number of predictions for substitutions than for the other two events. That the number of predictions for substitutions is generally higher is not so weird as this event also has the highest number of annotations. What is more interesting is that it seems like using the audio features still affects the results in the direction of predicting more substitutions. However, the eagerness to do so has been drastically modified by combining the input with visual features. Interestingly, the highest number of predictions for substitutions is still for *CALF-60-5*, as it also is for audio input alone. With the high number of predictions, we also get a pretty high recall score at 67.84% for substitutions. On the other hand, it also creates a lot of false positives, achieving a precision score of 41.83. Further, substitutions are also predicted 18 times when the ground truth event is either goal or card. Again, this could be due to a high number of predictions, so it is not necessarily because the model struggles with separating the classes.

For cards, in particular, we observe a low number of predictions, but still enough false positives to not really achieve a great precision either. This results in the model performing weakest on cards, with an especially low recall at 25.56%, even without a great precision, which is at 57.79%. Compared to this the model performs much better for goals, with a much higher recall at 73.01%, while still keeping a better precision at 63.98%.

		Ground truth			
		Goal	**Card**	**Sub**	**None**
Prediction	**Goal**	176	1	0	20
	Card	0	171	0	168
	Sub	3	10	331	489
	None	147	268	235	-

Table 4.16: Confusion matrix for *CALF-60-20* with concatenated audio-visual input evaluated at a tolerance of 40. For ground truth, "none" means the absence of an event at a position where it has been predicted one. For predictions, "none" means that no class was predicted at a place where a ground truth event was present.

Moving on to the confusion matrix for *CALF-60-20* in Table 4.16, the first thing we notice is that the number of false positives for goal predictions are significantly lower than it was for *CALF-60-5*. It is for *CALF-60-20* down at 20, while it was at 124 for *CALF-60-5*, when considering only the ones where no other ground truth event was present. This does of course impact the precision score a lot, as it is actually very few predictions for goals that were wrong. The model achieves a precision score for goals at 89.34, which is the highest precision score at this tolerance for any event and model. But, this score is not so useful alone, and the model finds barely over half of the ground truth goals. That is not so strong and emphasizes how hard the trade-off between precision and recall could be.

We observe that substitutions are still predicted at a higher rate than the other two, but at a slightly lower rate than for *CALF-60-5*. What is surprising with this is that even though the model predicts fewer substitutions, the precision is actually lower than for *CALF-60-5*. This means that *CALF-60-20* predicts worse for the substitutions than *CALF-60-5*. This is also shown by the recall score, as this is also worse. For cards, the number of true positives found by *CALF-60-20* has increased from *CALF-60-5*, but it does not help very much as the recall rate is still down at 38%. The precision has also fallen, after introducing a big increase of false positives.

		Ground truth			
		Goal	**Card**	**Sub**	**None**
Prediction	**Goal**	231	0	0	30
	Card	0	182	1	165
	Sub	0	10	361	400
	None	95	258	204	-

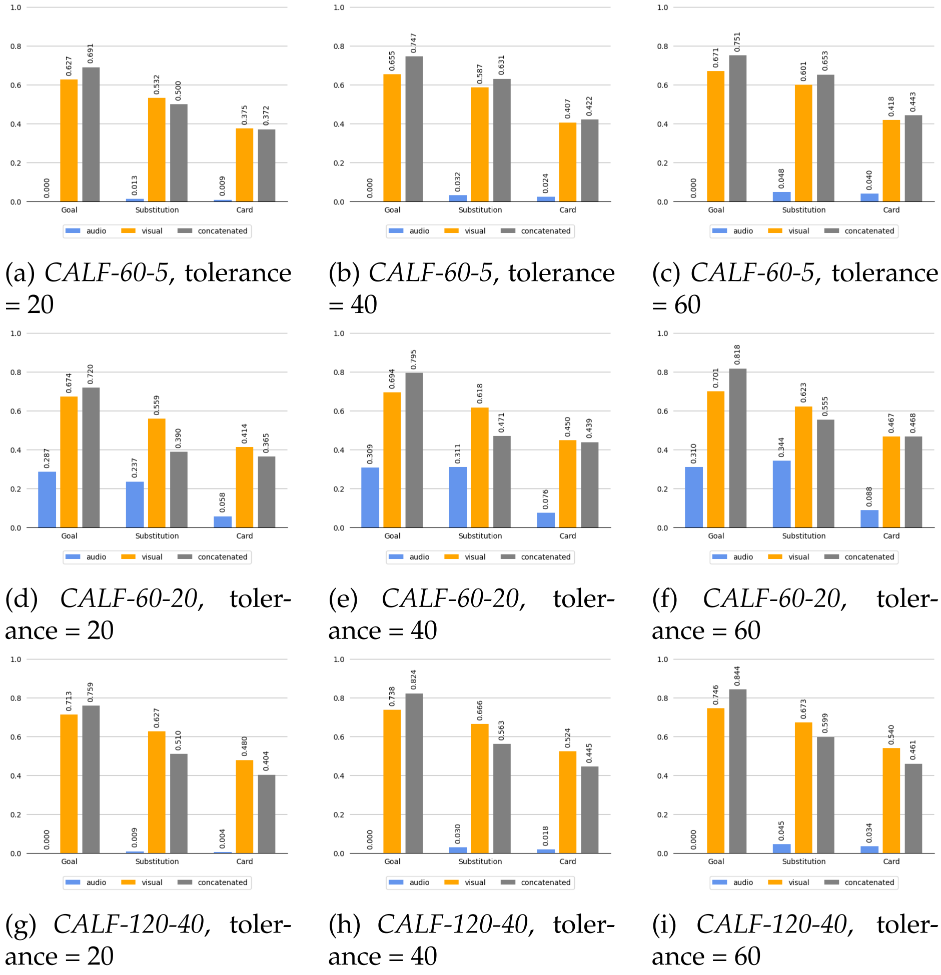
Table 4.17: Confusion matrix for *CALF-120-40* with concatenated audio-visual input evaluated at a tolerance of 40. For ground truth, "none" means the absence of an event at a position where it has been predicted one. For predictions, "none" means that no class was predicted at a place where a ground truth event was present.

Finally, we observe the confusion matrix for *CALF-120-40* in Table 4.17. We observe that it has a similarly low number for false-positive goal predictions as *CALF-60-20*, but that the number of false-negative goals is more close to *CALF-60-5*. This means that *CALF-120-40* has numbers similar to the good parts of each of the other models, which means that it is the best in total. It has a recall of over 70%, while still keeping a precision at a strong 88.51%. This means that this model performs pretty well for goals. If we also compare these goal results to the performance for visual-only input, we can see that the concatenated input performs better for all metrics we observe. By comparing this with the confusion matrix for visual input in Table 4.11, we can also see that with concatenated input we have removed the few confusions with other classes. This results in a really clean look for the goal event, with 0 confusion with other classes, both for goals not being predicted when it is a different event present and for other events not being predicted when the ground truth is a goal.

For cards with *CALF-120-40*, we observe a slight improvement compared to *CALF-60-20*, with both recall and precision at around 2% better. With a recall at 40.44% and precision at 52.30%, this is also a much more even trade-off between the two metrics than for *CALF-60-5*, even though *CALF-60-5* has better precision. When we compare to the visual input for the same model, we observe that the visual input performs better for both recall and precision, resulting in a better total score.

For substitutions, we observe that the number of false positives has decreased compared to the two other models with concatenated input. This gives better precision than both of them, but it is still clearly behind the same model with visual input. When compared to visual input we observe that the number of predictions is much lower for visual input alone, but still, the visual input has a higher number of true positives. Concatenated input also has more confusion among the three classes, and not just more false positives away from a ground truth event.

4.2.6 Summary of action spotting results



(a) *CALF-60-5*, tolerance = 20

(b) *CALF-60-5*, tolerance = 40

(c) *CALF-60-5*, tolerance = 60

(d) *CALF-60-20*, tolerance = 20

(e) *CALF-60-20*, tolerance = 40

(f) *CALF-60-20*, tolerance = 60

(g) *CALF-120-40*, tolerance = 20

(h) *CALF-120-40*, tolerance = 40

(i) *CALF-120-40*, tolerance = 60

Figure 4.1: Spotting performance in terms of Average Precision per event type, for the *CALF-120-40*, *CALF-60-5*, and *CALF-60-20* models over the tolerances 20, 40, and 60. In general, we can observe that for goals, adding audio information almost always improves performance. For other events, it depends on the configuration.

To summarize the action spotting results, we found that the effect of using the audio-visual features as input depends on the configuration and event type. For goals, the audio-visual approach is superior to audio or visual input alone and achieves significantly better maximum results for all configurations. For the other events, it varies based on the model configuration and prediction tolerance. On average, the *CALF-60-5* model benefits from audio-visual input for all event types, but for lower tolerances, cards and substitutions perform better with visual input alone. The *CALF-60-20* and *CALF-120-40* achieves better average-mAP scores with visual input alone for cards and substitutions, but the *CALF-60-20* achieves a higher AP for cards for the highest tolerance with audio-visual input. The highest mAP and average-mAP scores in total are achieved by *CALF-120-40*. The results for the event types with the different input are visualized for three tolerances in Figure 4.1.

4.3 Classification results

For the classification task, we have tested three different models. One visual model that takes the supplied ResNet features from SoccerNet [29] as input, one audio model that takes Log-Mel spectrograms as input, and a combined model which combine the output of the first two models through their softmax average. All the models are trained and tested with window sizes varying from 2 to 32 seconds. The classification task is different from the spotting task in that it uses the modified dataset which includes a background event. The models are trained and tested on clips where one of the events is present, and the task is to classify which it is. This way it does not take into consideration the temporal annotation aspect, as the spotting task does. We will first present the overall results for the models before we present the class-wise results for the tested models.

4.3.1 Overall results

For the overall results we first look at the results in Table 4.18. This presents the accuracy results on the test set for all the models for the 5 window sizes. In general, we observe that all of the models increase their accuracy when the window size is increased. For all the models the biggest increase is gained when moving from window size 2 to 4, with a more gradual increase for each window size after this. We observe a gradual change for which model achieves the best accuracy results when increasing the window size. With a small window size (2 to 8), the visual model performs best, but the combined model gradually comes closer when the window is increased. When we increase the window size to 16, the combined model outperforms the audio and visual models. The combined model stays in front also for window size 32 with an accuracy of 90.85%. The audio model achieves the lowest accuracy for all window sizes, with a gap of approximately 15% up to the nearest model for most of them. The accuracy for the audio model is ranging from 61.11% with window size 2, to 73.89% for window size 32.

It is interesting that even though the audio model consistently performs worse than the visual model, a higher score is achieved when they are combined. It is also interesting that the boundary for when the combined model starts to outperform visual-only is so clear, with gradually closing in from window size 2 to 8, before the combined model outperforms visual-only from window sizes bigger than this. This is easy to spot by looking at Figure 4.2.

We observe from Table 4.19 that the results F1 scores reflects the same

	Accuracy (%)		
Window size	Audio	Visual	Combined
2	61.11	**79.81**	76.93
4	67.28	**84.39**	82.40
8	71.11	**87.22**	85.98
16	72.55	87.87	**89.21**
32	73.89	89.21	**90.85**

Table 4.18: Comparison of the accuracy (%) of classification on the test set. The audio model is described in Section 3.2.2 and the visual 2D-CNN model with pre-extracted ResNet features is described in Section 3.2.3. The fusion of the audio and visual models is performed using *late fusion* through softmax average, as described in section 3.3.2.

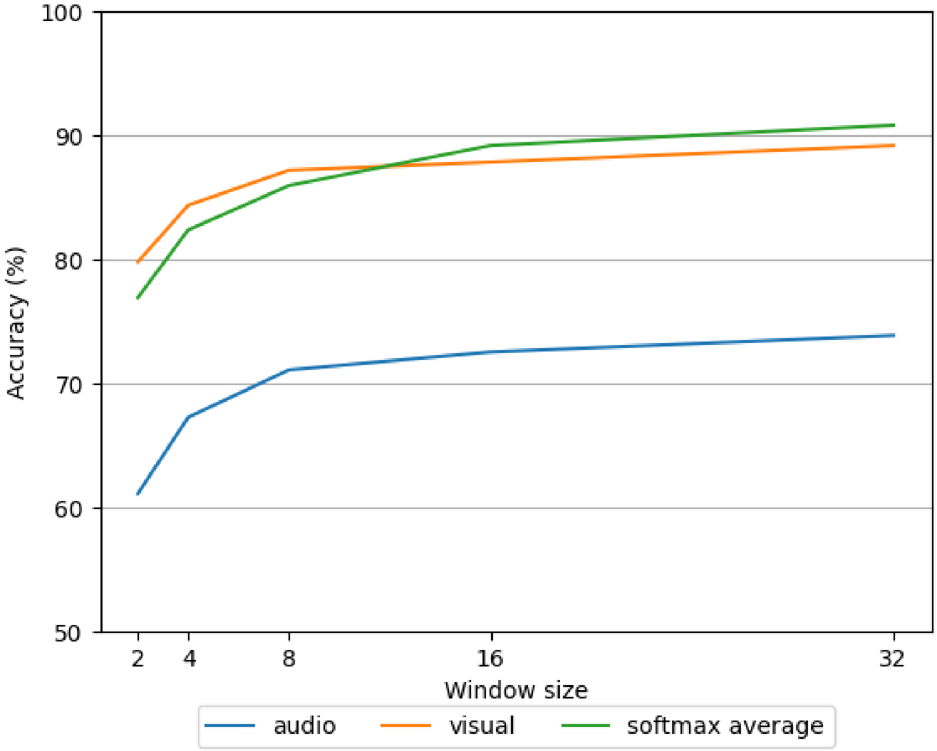


Figure 4.2: A graph displaying the accuracy for the classification models for the different window sizes on the test set. The classification models tested are the audio model, the visual 2D-CNN model and the combined model through softmax average.

order as for the accuracy. For the first three windows sizes, the visual model achieves the highest scores, before the combined model outperforms the other for the two biggest window sizes. The highest scores are achieved with bigger window sizes, meaning that it is the combined model with window size 32 that achieves the overall highest F1 score, which is 91.3%. The audio model achieves the lowest score for all window sizes for precision, recall and F1.

We do not only want to compare the overall results for the models,

	W = 2			W = 4			W = 8			W = 16			W = 32		
Model	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Audio	0.641	0.632	0.636	0.698	0.697	0.697	0.736	0.730	0.733	0.750	0.741	0.744	0.752	0.746	0.749
Visual	0.803	0.812	**0.806**	0.850	0.852	**0.851**	0.876	0.879	**0.878**	0.889	0.882	0.885	0.902	0.898	0.899
Combined	0.786	0.787	0.786	0.835	0.838	0.836	0.870	0.870	0.870	0.902	0.898	**0.900**	0.913	0.912	**0.913**

Table 4.19: Comparison of precision, recall, and F1-score for the audio model, visual 2D-CNN, and combined model on the test set. W is the window size used for the input. The results for the combined model are obtained using *late fusion* with softmax average.

but also how they perform class-wise. For this, we compare the audio model, the visual 2D-CNN, and the combined model. We still compare the results for the five different window sizes, and we mainly focus on the F1, precision, and recall scores. As the F1 score is the harmonic mean between precision and recall, is this a metric that gives a little more insight into the overall performance for the class. Therefore, we highlight the best F1 scores for each event type in Table 4.20.

4.3.2 Class-wise results for the visual model

For the visual model, we observe from Table 4.20 that the window size 32 performs best for all event types with respect to F1 scores. The performance is not very much better than smaller window sizes, and particularly substitutions have a very similar performance from window size 8 to 32. For substitutions, it is even a tiny decrease of performance in the F1 score between the window sizes 8 and 16. We further observe that the best F1 scores for the visual model are achieved for the goal event, at a maximum of 95%. The second easiest event for the visual model is substitutions at 90.1%, followed by cards with an 88.2% F1 score. This leaves the generated background class as the hardest class to predict, with an F1 score at 86.4%.

We generally observe that the visual model outperforms the audio model and the combined model for three of the four classes for the window sizes 2, 4, and 8. The only event type that is different is the goals, which achieves the best F1 scores with the combined model for all window sizes. Even though the visual model is outperformed for the goal events, we observe that as we increase the window size, it gradually becomes a closer battle between the two, with only a 0.4% difference for window size 32. This could imply that it is the visual model in particular that benefits from the increased window size, but that the combined model also benefits from this, as the visual model is a part of the combined model as well. This could be the reason for these results, as the combined model even decreases the results for goals from window size 16 to 32. These thoughts are confirmed by observing the results for the audio model, where we find that the F1 score increases until window size 8, but from there and out the performance decreases significantly. This is visualized in Figure 4.3. We also observe a somewhat similar tendency for the results for cards, which is visualized in

		W = 2			W = 4			W = 8			W = 16			W = 32		
Class	**Model**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Card	Audio	0.564	0.552	0.558	0.601	0.618	0.609	0.650	0.647	0.648	0.672	0.704	0.688	0.632	0.664	0.648
Card	Visual	0.811	0.757	**0.783**	0.836	0.808	**0.822**	0.869	0.848	**0.858**	0.873	0.850	0.861	0.873	0.892	**0.882**
Card	Mixed	0.776	0.751	0.763	0.796	0.790	0.793	0.865	0.832	0.848	0.870	0.874	**0.872**	0.867	0.892	0.879
Sub	Audio	0.625	0.573	0.598	0.651	0.672	0.661	0.694	0.718	0.706	0.777	0.674	0.722	0.817	0.770	0.793
Sub	Visual	0.866	0.794	**0.829**	0.885	0.874	**0.879**	0.909	0.883	**0.896**	0.923	0.869	0.895	0.933	0.870	0.901
Sub	Combined	0.803	0.741	0.771	0.818	0.841	0.830	0.857	0.872	0.865	0.922	0.893	**0.907**	0.945	0.914	**0.929**
Goal	Audio	0.851	0.825	0.838	0.919	0.902	0.910	0.942	0.902	0.922	0.908	0.874	0.891	0.867	0.837	0.852
Goal	Visual	0.803	0.936	0.864	0.885	0.923	0.904	0.898	0.942	0.919	0.941	0.923	0.932	0.959	0.942	0.950
Goal	Combined	0.878	0.929	**0.903**	0.937	0.951	**0.944**	0.940	0.957	**0.948**	0.969	0.948	**0.958**	0.960	0.948	**0.954**
Back	Audio	0.524	0.579	0.550	0.622	0.597	0.609	0.658	0.654	0.656	0.646	0.712	0.677	0.691	0.714	0.702
Back	Visual	0.734	0.761	**0.747**	0.793	0.802	**0.798**	0.830	0.845	**0.838**	0.820	0.885	0.851	0.842	0.887	0.864
Back	Combined	0.684	0.727	0.705	0.791	0.769	0.780	0.818	0.819	0.819	0.846	0.876	**0.861**	0.882	0.896	**0.889**

Table 4.20: Class-wise comparison of precision, recall, and F1-score per class (event type) for the audio model, visual 2D-CNN, and combined model on the test set. W is the window size used for the input. The results for the combined input types are obtained using *late fusion* with softmax average. "Back" means the background event, and "sub" means the substitution event.

Figure 4.4. Here, the visual model again ends up behind the combined model along the way, but after the audio model starts to decrease at a certain point, the visual model starts to get closer to the combined model again. In this case, it even surpasses the combined model at the end, but the results are really similar for window size 32, and the visual model is only 0.3% better.

For goals, we also find that the visual model generally achieves a higher recall than precision for the three smallest window sizes. For the final two window sizes this is reversed, so the precision is higher than the recall. This could imply that the model understands better the difference between goals and goal opportunities, which could occur in the background class, when more context is added through a bigger window size. Since the recall values do not fall notably, the model still predicts approximately the same amount of the goals but has fewer wrong predictions for the class.

For cards and substitutions, we observe an opposite balance for precision and recall than for goals for the lower window sizes. They start with a higher precision number but have a lower recall. This indicates that the model might be predicting fewer of these actions and that they are wrongly predicted as goals instead since goals have such a high recall with lower precision. Both cards and substitutions keep a higher precision than recall score for all of the window sizes except for 32, where cards have a higher recall. The background class is the only one having higher recall than precision for all window sizes for the visual model.

4.3.3 Class-wise results for the audio model

Unlike the audio approach for the spotting task, the audio model achieves meaningful results for the classification task. The audio model still

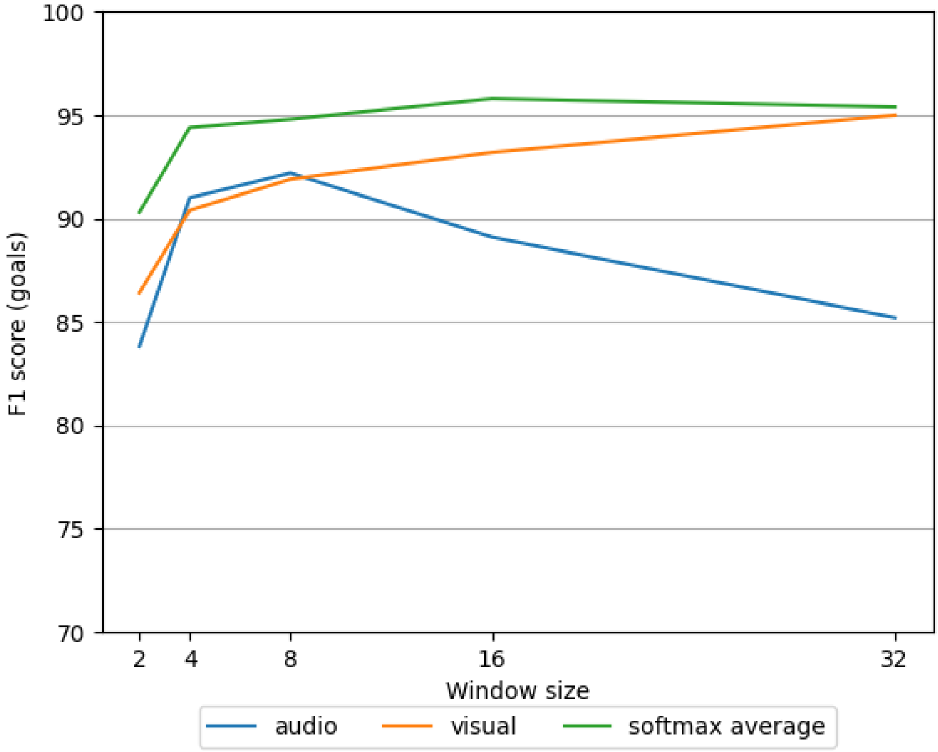


Figure 4.3: A graph displaying the F1 scores for the classification models for the event type goal for different window sizes. This shows that all models increase to a certain windows size, before audio and combined eventually decreases.

performs worst when we look at only the top-performing window sizes for each class, but the difference from the other two is not that big. The audio model even achieves better results for some combinations of window size and event type. The audio model has almost the same order of which event it predicts best as the visual model, and it is the same as for the combined model. This means that it finds goals as the easiest to predict, with the best F1 score at 92.2%. Even though this is not better than the other two models, it shows that the audio model has learned a lot and that audio input is valuable information. In second place we find substitutions at 79.3% F1, but unlike the visual model, the audio model has a better F1 score for background than for card. The best background score is 70.2%, while the best for cards is 68.8%.

The audio model differs from the visual model when it comes to how it reacts to increasing window sizes. Where the visual model generally improved performance with increased window size, the audio model seems to reach a threshold at some point for goals and cards, after which the performance starts decreasing. This can be spotted by looking at Figure 4.3 and Figure 4.4, where it is a point in the plot where the audio model reaches its top, before it starts to decrease in performance. This is not the case for the substitution and background events, as they keep

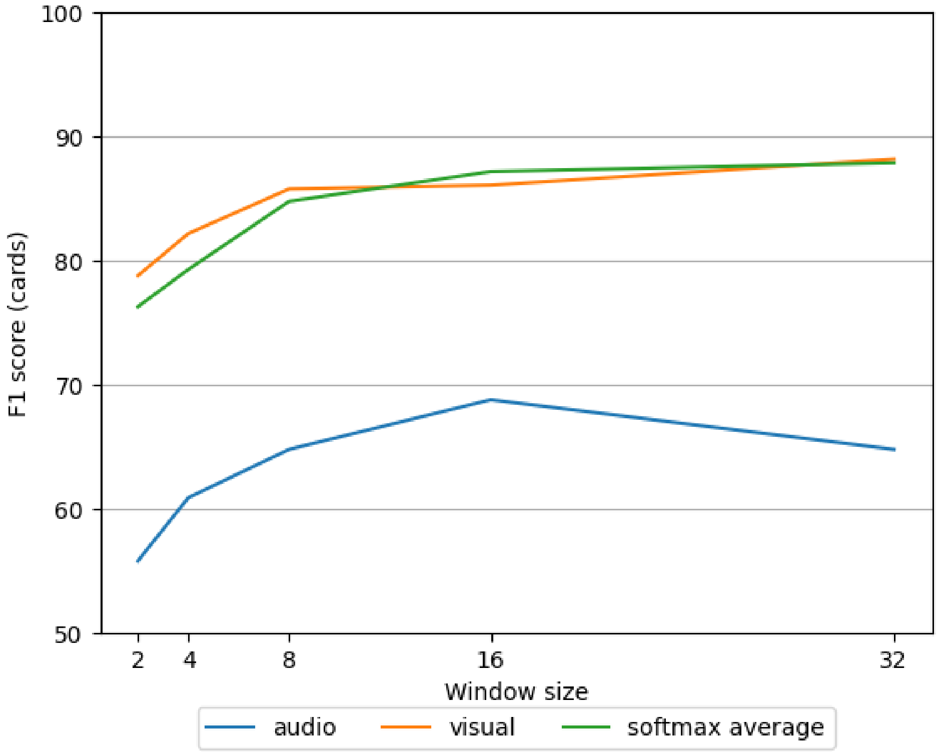


Figure 4.4: A graph displaying the F1 scores for the classification models for the event type card for different window sizes. This shows that the visual and combined models increase to the biggest window size, while the audio model reaches a top for window size 16.

increasing until the biggest window size we tested. This can be observed in Figure 4.5 and Figure 4.6, where the plots for the audio model does not have the same maximum point with a subsequent drop. Instead of a drop, we actually observe a boost in performance for substitutions and background when moving from 16 to 32 in window size. This increase for substitutions is mainly because of an almost 10% increase in recall, but the precision was improved as well. The case is not the same for the background event, which also increased, but had a negligible increase in recall. However, the precision increased, and is the reason for the improvement of the F1 score.

The results for goals with the audio model are particularly interesting, as it for this event type performs much better compared to the other events. Where the other events start far behind the other models already from window size 2, the audio model begins much stronger for goals. The results are already comparable to the visual model for this window size, and when the window size is increased to 4 and 8, the audio model outperforms the visual model for goals. After this, the audio model only decreases in performance and ends up far behind for the rest of the window sizes. For the other events, the results from the audio model are not as comparable to the visual model and the combined model's results. As we observe from the Figures 4.6, Figures 4.4 and Figures 4.5, the blue line

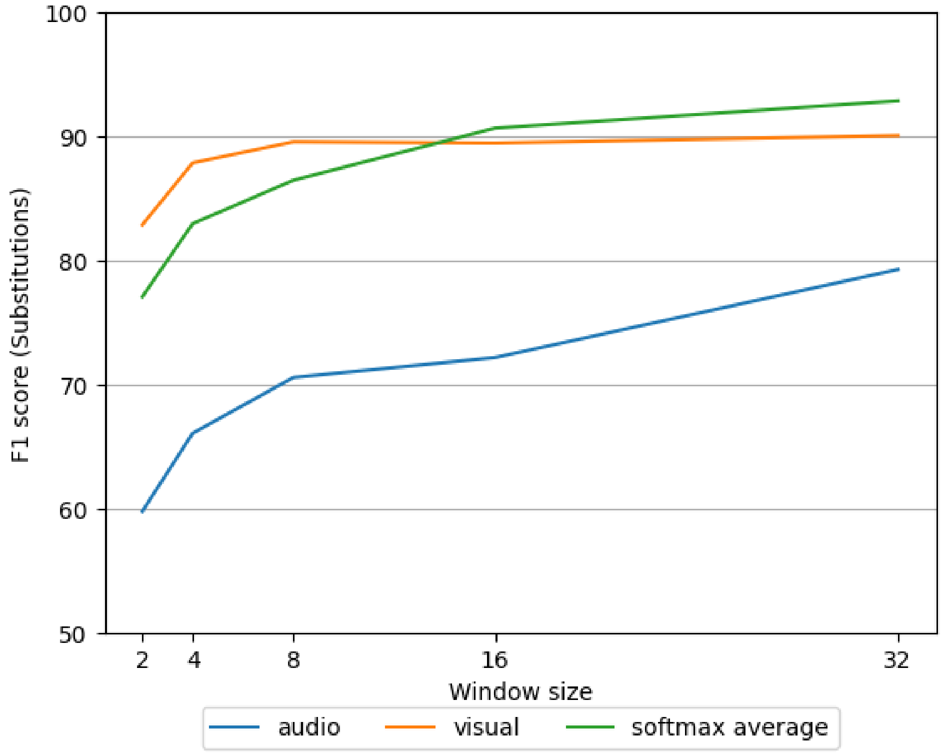


Figure 4.5: A graph displaying the F1 scores for the classification models for the event type substitution for different window sizes. This shows that all models increase to the biggest window size.

representing the performances of the audio model is relatively far below the other two models, showing that the audio model alone is nowhere near outperforming them for these event types.

This shows that the events differ in how much audio information is relevant for the event type. This is as we suspected from before, with goals and substitutions having the most to gain from audio, while cards are harder to predict based on sound. The additional background task is hard to know exactly what would sound like, as it is generated just based on the absence of the three original classes in the dataset. In the actual game, there are several other events that are not a part of the dataset we have used, and all of these could potentially be present for the background task. This makes it hard to intuitively think of how a general background event sounds like.

4.3.4 Class-wise results for the combined model

The first observation from Table 4.20 for the combined model is that it actually is the model that achieves the highest F1 score for three out of the four classes. This is when we look at only the highest achieved score for each event type across all window sizes. The only event type that achieves a higher maximum score with another model is cards, where the visual

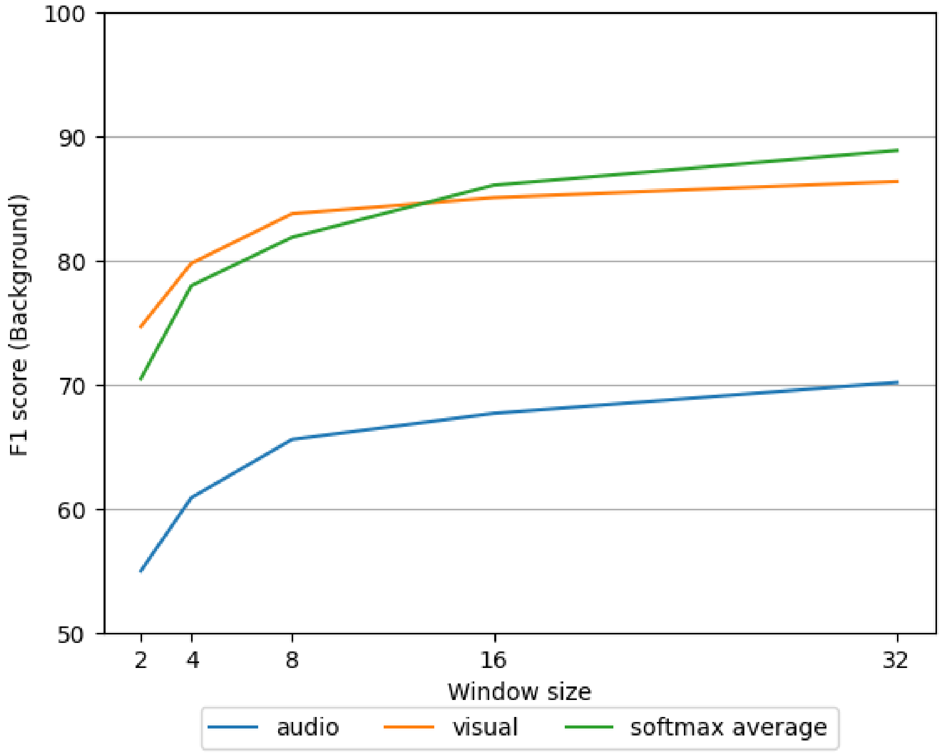


Figure 4.6: A graph displaying the F1 scores for the classification models for the event type background for different window sizes. This shows that all models increase to the biggest window size.

model achieved the best F1 score. The difference is very small, with the visual model at 88.2% against the combined model at 87.9%. These are very similar results, and they even have the same recall, with only a small difference in precision.

For the other events, the difference is bigger, with the combined model outperforming both the visual and the audio model for all three. The closest results are for goals, where the best-combined model (window size 16) achieves a 0.8% better F1 score than the best visual model (window size 32), and a 3.6% better score than the best audio model (window size 8). For all other events than goals, the combined model outperforms the audio model by double digits. For the visual model on the other hand the results are closer. The combined model achieves a 2.8% better F1 score for substitutions and 2.5% better F1 score for the background event for window size 32, which achieves the best results for both models for these events.

Starting with the event type goal, we observe that this is the only event type where the combined model achieves the best F1 results for all the tested window sizes. This is interesting as the other event types do not show better results compared to the other two models before the window size has reached 8. This could be related to that the audio model performs well already at window size 2 for goals, proving to contribute with valuable information already at that point. For the other events, the audio model shows poor results for the smallest window size, which could mean that it is harder for the combined model to perform well, as the audio model is a part of the combined model. It is interesting to observe that even though the audio and visual models have pretty similar results for many of the window sizes for goals, the results of combining them improves the performance to a higher score than any of them reaches alone. This could mean that the intuition behind the combination of the models makes sense and that two predictions based on different modalities could help each other to create a better result.

We observe from Figure 4.3 that for goals, the combined model reaches its best score at window size 16 before the results decrease a bit for window size 32. By looking at the two other models in the graph as well, we find this development to be as expected. This is because the combined model is based on the two other models, and as we observe, the audio model has had a big decrease in the performance for the bigger window sizes. The visual model keeps increasing the score, but the decrease of the audio model is bigger. It is therefore as expected that the performance of the combined model should drop as a result. This could mean that even though the combined model outperforms goals for the models we have tested, it is no guarantee that this would have been the case if we had tested with even bigger window sizes for the video model. The combined model seems to have reached a maximum level, while the visual model potentially could have been even better, and challenged the combined model's top results even more for this event. On the other hand, we do not know that the maximum results for the combined model are at the correct spot either. It is several window sizes to explore between the window size 8, where the audio model start dropping, and 16 where the max score is found for the combined model. This means that the audio and the combined model potentially could have higher values between these two window sizes, but as they are not tested, the graph displays it as a line between the two values it has available. This way, even though the graph goes upwards from window size 8 to 16 for the combined model, it could be that it is hidden better results in between before the results start to decrease. For the same reason, it is no guarantee that the visual model actually still improved at the point of window size 32. The only thing we know from the graph is that it is better than window size 16. So even though it could seem from the graph that if we increased the window size even further it would perform better, it is no certainty that it would really be the case.

For the event type card, we observe in Table 4.4 a kind of similar development in the results as for the goals. The results achieved for this event are not as good as for goals, but the development of the results compared to each other is similar. The similarity is that halfway through the window sizes, the combined model's performance flattens out, at the same time the audio model results start to decrease. This time the decrease of the audio model starts from window size 16, but we do not observe a decrease in the results of the combined model. Instead, we observe a lower increase than before, allowing the visual model to barely jump ahead to be the model achieving the best result for the biggest window size. This way we see that the falling performance of the audio model still affects the combined model enough to end up as weaker than the visual model in the end, even though the combined model does not actually decrease. The development of the model's performances for cards still has a difference from the results for goals. We observe that where the combined model outperforms the visual model already from the smallest window size for goals, we find that this is not the case for cards. Actually, the combined model only performs better than the visual model for one window size for cards, with the visual model performing best both before and after window size 16. 16 is the window size where the audio model achieves its maximum score, so this makes sense.

The development of the results are fairly similar to cards for the substitution and background events, but for these two, the combined model stays ahead for both window size 16 and 32, while the results for the visual model is the one that seems to flatten out. As we see for the lowest window sizes, the visual model starts as the strongest performer for both classes, before the combined model gets closer, and eventually takes over as the top performer for the final two biggest windows sizes. The development of the graph for the bigger window sizes is different from the goals and cards, where the results seemed to flatten out or decrease for the combined model. For substitution and background, the development is rather opposite, with the results still improving between window sizes 16 to 32. This seems to have a direct correlation with that the audio model keep increasing the results for the bigger window sizes as well, unlike for the other two events. This way the audio model keeps giving better results to the combined model, while the visual model seems to increase less. This makes the combined model increase its lead over the visual model for these events for the bigger window sizes.

4.3.5 Summary of classification results

In this section, we have presented the classification results from our experiments. To summarize the presented results, we found that the combined approach through softmax-average showed good results compared to the audio model and the visual model. The highest overall results are achieved with this combined model, reaching an accuracy of 90.85% on the test set. The visual model performed best for lower window sizes, but for the bigger window sizes the combined model was better, and it was with these window sizes the best results were achieved. For the different event types, the combined approach also performs best for three out of four of them. The only event type where the visual model achieves a higher maximum score is for cards, and the difference is at only 0.3%. All in all, we found that the audio-visual approach was beneficial for the classification task.

4.4 Discussion

In the previous sections, we have presented the results from our experiments for action spotting and classification with our audio-visual approaches. With these results presented, there are some topics we want to discuss further, and that is done in this section.

4.4.1 Classification vs spotting results

We have tested two different multi-modal approaches, one for the spotting task, and one for the classification task. We have presented results for both, and compared the multi-modal approach to visual-only and audio-only approaches for their respective tasks. When we observed these results, we find that it is a difference in how the multi-modal approaches performed compared to the single-modality approaches for classification and spotting.

For the spotting task, we primarily observed improved results for the event type goal but did at the same time often observe a drop in performance for the other two classes for two of the configurations. We did also observe improvement for all classes for *CALF-60-5*, but this configuration is not the best-performing model among all the tested ones. When looking at the results from only the best performing models for each of the different classes, we find that the goals are the only events with an increase for our tested approach.

For the classification task, on the other hand, we find a more general improvement for the audio-visual approach. Even though the visual-only model outperforms the combined approach for smaller window sizes, the overall tendency of the performance is that the combined model performs better when the window sizes are increased. These models are also the ones with the highest performance in total and are therefore the ones that are interesting to observe the results from. This means that for the models with the best results, the models benefit from an approach combining the audio and visual model through softmax average. This approach achieves the best F1 scores for three out of four classes (including the background event), with cards being the only event that performs better with the visual-only model. And even for this event, the difference is at only 0.3% in the F1 score.

Looking at these results we find that our tested approaches have in general worked better for the classification task than for the spotting task. This could be caused by several reasons, and we can not say for certain why this is. But, we have some alternatives to what could have influenced the results in this direction.

Classification format

The first is that sound is a more effective modality to include when we are facing a classification task than a spotting task. It could be that the restrictions given by a classification format, with the certainty of some kind of event being present, could be beneficial to sound. This because with this premise, it could be easier to just distinguish between the given events, rather than also having to understand what is not an event. A drawback for the likeliness of this reason is that the background task is present in the dataset used for classification. This means that even though we just distinguish between given events, the background event still contains events where no event is supposed to happen, as this is the nature of this background class. This way the model to some extent still has to distinguish between a "real" event being present or not.

Late vs early fusion

The second possible factor is that the late fusion approach could be more effective for our domain. We experimented with two different ways of fusing modalities, with *early fusion* for the spotting approach and *late fusion* for the classification approach. It could be that the *late fusion* approach is either more effective for this domain in general, or that we just have tested a better version of this than we managed to do for our *early fusion* approach. The *late fusion* approach is sometimes considered easier to perform, so it could be that this approach just is better implemented, as it is easier to do. It could also be that the selected model for the *early fusion* approach does not fit well for concatenated audio-visual input.

Extraction of audio features

This leads us to the last factor that we will discuss. This is that it is possible that our extracted audio features, or our concatenation of features, do not provide good enough information as input to the model. It is possible that either our extraction model for the audio features, or the chosen window size for this extraction, could have been better for some of the events. If we look back at the results for the classification models, we find that for the chosen window size for the audio feature extractor, the visual model still outperforms the combined and audio model for three out of four classes. This is the final tested window size before the combined approach takes over as the general best-performer. The audio model and combined model performs better than the visual model for goals at this window size, but it does not do so for the other events. As the combined approach starts outperforming the other approaches after this window size, it is possible that a bigger window size for the audio feature extractor could have benefited more than just the goal event for the spotting task. A drawback for this possible solution is that the audio model, which extracts the audio features, starts to decrease in results after window size 8 for some of the events. This means that the audio features might have been worse rather than better if the window size were also increased for the audio feature extractor. It is also possible that the audio features would have benefited more from having a smaller rather than bigger window size. It could be that given the nature of the CALF model, where the temporal distances plays an important part, it would have been better to use a smaller window size than we did, as this would contain information from a smaller period of time for each of the features.

4.4.2 Selective input for different events

We find that the spotting approach with the CALF model and audio-visual input achieves different results for the event types. This could mean that it differs for event types how valuable the concatenated information is when we are trying to perform action spotting for the events. This could be a general observation for spotting these events, or it could be due to some of the reasons mentioned in the previous section (Section 4.4.1). Regardless, the results for the CALF models we tested show this differentiation between events, and a possible approach to max the performance for this task could be to distinguish between which input modalities are used to predict each event. With this approach we could benefit from the best possible information source for the different events, and stop the performance decrease for the events that do not benefit from the same input.

We have not created a full model for this approach, but still want to discuss how this approach could boost the total performance of the tested CALF model. By extracting the results from the best input type for each of the events, we could put the results together and calculate the results in the same way we did for our other models. This way we would get a hypothetical combined result and could see the tendency of how big of an impact this would have, compared to the approaches with the same input type for all events. Therefore, we revisit the results for the CALF models per event in Table 4.8 and find that the best average-mAP results are achieved by the *CALF-120-40* model for all events. For the card and substitution events, the best results are achieved with visual-only input. The best results are for goals achieved with concatenated input. Therefore, we combine the Average Precision results for *CALF-120-40* with visual input for cards and substitutions with the results with concatenated input for goals. With these results, we create the mAP scores for each tolerance and then calculate the average-mAP score.

The calculated results show an average-mAP at 62.17%. This means that it has increased by over 2% compared to the average-mAP for *CALF-120-40* with visual input, which was the best performing model for this metric in our spotting experiment. Further, we also find that for both of the two biggest tolerances, the mAP score has increased by 2.87% and 3.28% compared to the same model. This shows us that an approach where the input is optimized for each of the given events, we could increase the performance of the models compared to just using one single type of input and hope that this is the best for all.

4.4.3 Spotting in practice

When looking at the results, we should keep in mind the practical consequences for a system performing this task in real life. In this context, the spotting results are more of interest than the classification results, as the spotting task is more what we actually want to use this kind of system for. We want to be able to use the system in real life so that we can spot different events in real game scenarios, and not just in a classification task. Therefore, we want to discuss some of the key factors and trade-offs that we have to consider when we want to assess how the models would perform in practice.

Importance of metrics

The first factor is which of the metrics would matter more and less in such a context. A metric that we have used to compare the different configurations of the CALF model is the average-mAP. This is a metric we have used to gain some insights into how the models perform over all the tolerances. This works fine for this purpose, so we get a number to compare general performance over several tolerances, but in a practical perspective, we are not interested in how the models perform over different tolerances. In a practical context, we have to choose one model for a given tolerance which we consider to be within what we want to consider a prediction or not. What this tolerance is must be chosen given what we consider is "close enough" to be a true positive, and it is not a general correct answer for this. This will be a trade-off between how many true positives we could add by increasing this tolerance, against how precise the predictions are temporally. Since we know that the average-mAP is not a relevant metric for one given model and tolerance, we could look at the mAP scores instead. These apply as a mean of the Average Precisions of the event types and could give valuable information about the performance of the model. But, with the performance differing between the event types it is not certain that this metric gives exactly the information that we need. As the performance is different, it could be of interest to look at a trade-off between what event types we actually want to spot, and how important it is to spot all of them. Goals are the main purpose of a soccer game, and it is what actually means something for the competitive aspect of the sport. In this view, one could argue that goals are more important to spot correctly and that substitutions and cards are more of an addition that could be of less interest than goals. Even though they are of interest, and that it is better with great performance for all event that exists, we view goals as being at a different level of importance than the rest. This is relevant as it becomes a trade-off between how many event types we want to include, and how many false predictions we want to avoid.

When assessing the performance of the different events in a real-life system, two of the most relevant metrics would be precision and recall. With these two metrics, we get numbers saying exactly how many of the events were spotted by the model, and how many of the predicted events were wrong. These are maybe the two metrics that are easiest to wrap your head around, and the meaning of them is straightforward to understand. In a perfect system, we would of course have 100% for both recall of precision, but this is of course not the case in a real system. Therefore, we get a trade-off between precision and recall, since they pull in different directions when it comes to how high the threshold for a positive prediction should be. Since we have to make a trade-off between precision and recall, we would argue that for a real-life system with action spotting in soccer as its application, the recall rate is of bigger importance than precision. This is because until systems with close to perfect spotting exist, we would still need some kind of manual monitoring of the results. With a manual operator looking at the results from the system, the most important aspect is that as many of the true positive events as possible are annotated by the model. This way the operator could have shorter clips to evaluate if are a true positive or not and the false positive are filtered out. This is unlike having a manual operator watching and annotating whole games manually, as this would take more time than just looking at the possible events and sifting out the wrong ones. The manual operator could then also monitor several games at once, as it is a lower workload for each of the games. In a process like this, recall is more important than precision, but a high recall would mean lower precision. Lower precision is still a problem, as the lower the precision, the higher the number of false positives annotated by the system. If the number of false positives is too high, the benefit of this approach would be gone, and it would still be a lot of manual work for the annotations. Precision is therefore still an important metric, but just not as important as recall.

When we consider our thought of high recall being important for a spotting task in practice, we observe from our spotting results that even though our recall scores are relatively good, they are still not high enough to be really useful. For a reasonable tolerance of 40 (20 in each direction of the annotated event), we find that the highest reported recall value in Table 4.8 is at 73.01% for goals, which is too low to be of any value in real life. Even when making a small adjustment and optimizing the results for recall, we do not achieve a combination of recall over 90% with precision over 50% at the same time. This means that even though we would have optimized the performance to fit an annotation approach that catches all events, with subsequent human filtering of false positives, the performance would still not be good enough. We would not catch all events without creating a big amount of false positives, making the subsequent filtering time-consuming.

Delay and performance

Another aspect of spotting in practice is if the spotting is being performed live, or if it is being done afterward for some kind of highlight extraction or summary. For the live approach, there are more aspects to take into consideration, as the spotting would need to be performed faster with less delay. With live spotting, we would also need to extract both the visual and audio features as the game proceeds, before we would have additional delay based on the number of frames which the model takes as input. To ensure low delay for the spotting it would therefore be beneficial to operate with smaller chunk sizes. This would again be a trade-off between how good spotting results we are getting and how fast we need them, as our experiments have shown that bigger chunk sizes generally perform better. For the other situation where we want to create a sort of summary or highlight extractor, we do not have the same restrictions as for live spotting. For this task, we could focus on maximum performance, and could therefore use the model we find having the best results rather than the fastest one. It is also lower demand for a very high precision score, as in a summary it could be fine to also include events such as goal scoring attempts that ultimately did not lead to a goal. This shows that it is not a general approach that fits all tasks, but that how to configure the model in use depends on the application of the model.

4.4.4 Comparison of results with related work

We want to compare the results we have found not only to the other models from our experiments but also to other work on the SoccerNet. There are several other approaches [29, 58, 76] tested on the SoccerNet, as well as the original CALF approach from Cioppa et al. [18] that we also want to compare our results to. In addition to the original CALF results we choose to compare our results to Rongved et al. [58], Vats et al. [76], and the baseline results supplied with the dataset [29].

Cioppa et al. report an average-mAP of 62.5, and we notice that this is better than our approach with the CALF model with visual input, which achieves an average-mAP of 60.07. This could imply that even though the average-mAP achieved for the *CALF-120-40* model on the validation set is almost equal to what was achieved with the supplied weights for the code, the model seems to have benefited from the extra training epochs when tested on the test set. This is worth noticing for the CALF results with visual input, as this means it could be achieved better results for this. The impact of using concatenated features is still clear though, as the changes in performance for the different event types are so significant. This even with concatenated features we believe have the potential of being optimized further. It is also worth noting that this metric provides some insights into how the model performs combined over all tolerances, and not so much about the performance in a real-life scenario. Further, we can also compare the original CALF score against our score with optimized input, which we discussed in Section 4.4.2. Our approach with selective input for different events achieves an average-mAP of 62.17, which is much closer. This even with the usage of the visual results from our weaker visual model (compared to the original CALF model) for 2 out of 3 events.

The comparison of the CALF model with visual input to other approaches is already performed when the model was introduced by Cioppa et al. [18]. The comparison of our findings for the CALF model with visual input to the other results is therefore not so interesting. What is interesting is comparing the other approaches to the best results for the concatenated input, and the results for optimized input. From Table 4.21 we observe that the concatenated input approach with *CALF-120-40* achieves higher average-mAP (56.29%) than both the baseline (49.7%) and Rongved et al. [58] (51%). It does not outperform Vats et al. [76] (62.1%) or the original CALF model (62.5%). The optimized input results with concatenated input for goals and visual-only input for cards and substitutions achieve 62.17% average-mAP, meaning that among the models that were introduced when we started this work, it is only the original CALF model that achieved a better result for average-mAP.

Model	**Average-mAP**
Tomei et al. [73]	65.5%
Cioppa et al. [18]	62.5%
Selective input (CALF)	62.17%
Vats et al. [76]	62.1%
Concatenated input (CALF)	56.29
Rongved et al. [58]	51%
Giancola et al. [29] (*baseline*)	49.7%

Table 4.21: Comparison of average-mAP scores for different approaches. The Selective input approach is the CALF model where we used concatenated input for goals and visual-only input for substitutions and cards. The concatenated input approach is the *CALF-120-40* with concatenated audio-visual input.

Late in the process of this thesis, it was released a new model tested on the SoccerNet, which outperformed the previous state-of-the-art models. This is presented in Tomei et al. [73] and report an average-mAP of 65.5% with the same visual ResNet features. They also report an additional boost of 9.6% when finetuning their own ResNet features to the task. They emphasize that this shows that the pre-computed features are a good starting point for research and comparison purposes, but that better performance can be gained through end-to-end training. This means that the supplied features fit our purpose of comparing the performance of visual-only vs audio-visual approaches, but that even better total performance can be achieved by finetuning the features. Furthermore, it is interesting that from the class-wise results they report in the paper, our approach with concatenated audio-visual features still achieves a better Average Precision for goals for most of the tolerances.

4.4.5 Deviation of events from dataset

When we were looking at the numbers in the confusion matrices for the spotting task, we observed that the total number of events in the test set deviated from the numbers given in the SoccerNet paper [29]. This made us look into how big this difference was, and if it also applied to the training and validation splits. When looking at the numbers for all the splits of the dataset we found some deviations for all the splits. The deviations were always in the negative direction, meaning that it was fewer events than stated by the SoccerNet.

As we observe from Table 4.22, where the deviations are presented, the differences were found only for substitutions and cards, and not for goals. Most of the differences were substitutions, but some also cards. The numbers are relatively small, compared to the total number of events which is 6637. We are only lacking 16 of the events in the test set, so we conclude

Class	Training	Validation	Test	Difference
Card	9	1	3	13
Substitution	29	10	13	52
Goal	0	0	0	0
Difference	38	11	16	65

Table 4.22: The number of samples per class that deviated from the given number in the SoccerNet dataset, when tested with the CALF model.

that the final results would not be affected to any degree worth considering. After investigating how this difference has been introduced, we believe that it is related to the way the CALF code extracts the events, as the events are not supplied directly in the format used for testing in the CALF model. This also means that it only applies to the spotting task, as the classification models do not utilize the event extraction from the CALF code.

4.4.6 Impact of half-time substitutions

When we look at how the events have been annotated, it caught our attention that the annotations of the substitutions done at half-time were annotated at the first frame of the second half. This seemed like a hard event to predict, as the beginning of each of the videos is stated in the SoccerNet paper [29] to be at the start of each of the halves. We also watched the beginning of several second halves and found that they do begin almost exactly at kick-off. Substitutions done at half-time are in reality performed before the kick-off of the second half, and would therefore not be shown at all in the videos. This makes these substitutions practically impossible to spot, and we wanted to observe how this affected the results. To do this we removed the annotations of substitutions at the first frame of the videos and tested the best performing model (*CALF-120-40* with visual input). The average-AP score for substitutions was increased by 2.2% (from 60.06% to 62.26%), while the average-mAP increased with 0.76% (from 60.07% to 60.83%). This is in line with the findings in Cioppa et al. [18], which also address this issue. They state that this is an issue for 28% of the games, and applies for merely 5% of the total substitutions on the test set.

4.4.7 Retrospect of process

Through the work of this thesis, there have been several occasions where we have learned after a part of the process was done that we could have solved it in a different way. When looking at the project in retrospect it is therefore things we could have done differently, and we would like to address some of them here.

When we experimented with different chunk sizes and receptive fields for the CALF model, it might have been better to use a higher amount of models with audio and concatenated features as input and optimize on the validation set using one of the well-known hyperparameter approaches. This is unlike what we did with first experimenting with the hyperparameters on visual input to find which configurations we wanted to test with concatenated input. By evaluating with the validation set first, we would not have had to increase the number of models we ended up testing on the test set, even though it would have increased the number of models tested on the validation set. This could potentially have had the possibility to find an audio or concatenated feature model with even better overall results. However, the main goal was to show how adding audio to the input would impact the performance, and this goal is still well preserved in our approach. This is because we test and compare performance for different input types on the same chosen configurations. It would also have been more computationally heavy to train and test the same number of models also for models with audio and concatenated features as input.

A smaller error found when looking over the results, is that we chose the number of 150 epochs for the first iteration of hyperparameter testing. When we did this we did not consider that the validation cycles were at 20, which does not add up to 150. This means that the final 10 epochs of training were not taken into account when we kept the best-performing weights from the validations.

When we compared our results for the CALF model with visual input, we found that the original paper reports better results than what we found. This is described in Section 4.4.4 where we compare results with related works. We trained the model for fewer epochs than the original paper because the results on the validation set were already similar to the supplied CALF-weights. The number of epochs could be the reason for the weaker results, so in retrospect, it might have been better to use a higher number of training epochs to maybe come closer to the performance of the original paper on the test set.

When we calculated the overall precision and recall scores, we did so by taking the average across the precision and recall scores for the individual events. As the number of events for each of the event types is not completely even, this metric becomes more an average across the classes and not the precision and recall as a whole for all events observed as one. By calculating precision and recall over all classes at the same time we could have avoided some problems for the overall scores for these two metrics, such as the NaN values we encountered when trying to calculate the average over the three event types when one of them included a NaN value. It does not change any of the assessments or conclusions, as we mainly assessed the results class-wise, and the results across classes were to a greater extent driven by other metrics. The results for these metrics still provide value, but they might be more precise with all event types together, and since we also would avoid some problems along the way we could have benefited from doing so.

4.5 Summary

In this chapter, we have presented our results for both the action spotting and classification task on the SoccerNet dataset [29]. We started with presenting the results for some preliminary experiments that were done during the hyperparameter selection in Section 3.4. Then, we presented the results for our action spotting approach, where we used the CALF model with concatenated audio-visual input. We showed that the approach with visual input achieved better scores in total, but that the concatenated input achieved better results for goals. Further, we presented the results for the classification task, where we described that the best results were achieved through a combined model through the softmax average of the visual model and the audio model. Finally, we discussed several aspects of the presented results and the process leading up to them. We discussed the differences between the classification and spotting results, how differentiating what input to use for each event type could increase performance, how our results compared to other approaches to the SoccerNet dataset, and how spotting in a real-life scenario would differ from this academic approach. In the end, we discussed some aspects that possibly could influence the results, before we mentioned some steps of the process we could have done differently.

Chapter 5

Conclusion

Today, sports games are usually manually annotated by human operators, which is an expensive and time-consuming task. Recent research has shown that machine learning might be used to automatically find exciting events without any human intervention, potentially saving high costs of money and time. However, most approaches only use visual data to detect events, leaving out other valuable information like sound. In this thesis, we experimented with audio information and tested different approaches for combining visual and audio data for spotting and classification.

Our results showed that audio-visual approaches were beneficial for several use cases, but that the advantage could differ depending on the configuration and event type. For goals, the performance significantly increased with the audio-visual approaches compared to using a single modality. For the top-performing spotting model, the average AP for goals was nearly 6% better with concatenated audio-visual input than with visual input alone. Furthermore, the difference in Average Precision for the highest tolerance was nearly 10% for the same models. However, the performance decreased for cards and substitutions for the best performing model but did improve with other configurations. For classification, the audio-visual approach increased the total accuracy by 1.64% compared to the visual-only model. The best results for both goals and substitutions were achieved with the combined model. For cards, the performance was similar for the visual and the combined model, with only a 0.3% difference in favor of the visual model. In total, we found that the audio-visual approaches have shown great potential, which we believe can be utilized further in the future.

5.1 Main Contributions

We wanted to to assess the performance of audio-visual approaches for action spotting and classification, as described in Section 1.2. During our work, we have made the following contributions:

- We have researched machine learning approaches for event detection and classification, and developed multi-modal approaches for these tasks. Specifically, we have developed audio-visual approaches for soccer videos from the SoccerNet dataset [29], which includes the event types "goal", "card", and "substitution". We created an audio model which extracted audio features from the dataset through spectrograms and these were used to create concatenated audio-visual features. We experimented with different ways of fusing modalities and tested both *late fusion* of models at decision time and *early fusion* through the concatenated features. We have experimented with various hyperparameters and selected optimized configurations for the models.

- We have experimented with the selected models on the test split of the SoccerNet dataset and analyzed the performance. We tested models for both event detection and classification, and assessed the performance for several metrics. We have showed that the audio model extracts valuable features and that the concatenated features worked well for action spotting on the dataset.
- We have compared the performance of our multi-modal approaches to single-modal approaches and shown that there was a great benefit of, and further potential for, using more than just visual information. We presented results showing that for the classification task, the best results were achieved with an audio-visual model, outperforming the best visual results from our experiments with over 1.5%. Further, we have shown that for action spotting, the performance with concatenated audio-visual input was superior for goals for all tested configurations. For other events, the results improved with concatenated input for some of the configurations, but could also in some cases have a negative effect on the performance.

Our contributions are interesting in the context of the problem statement, and we have presented results valuable to assess the performance of audio-visual approaches. We have shown the potential of multiple modalities, but that it might be event-specific how it performs in some situations. This work has given a strong foundation for further work with multi-modal models. Furthermore, our results have been presented in a paper which is under review.

5.2 Future Work

To continue the work with audio-visual approaches for event detection in sports there are several steps that could be taken. First of all, it would be interesting to observe how the tested models would perform on other datasets than SoccerNet [29]. This would give insights into how the approaches generalize to other data. In particular, the SoccerNet-v2 [20] would be a good first step, as this would show how the models perform for other events in the same format, with the same features available. Further, it would be interesting to experiment with audio-visual approaches for other sports as well.

Testing the audio and concatenated features as input to other models than the ones tested in this thesis would also be a related task. It could be valuable to experiment with extracting own visual features, as well as other methods for concatenation of audio-visual features. This would be relevant both for the models we have tested and for other potential models. Furthermore, experimenting with different window sizes for the audio feature extractor would be valuable. This would be interesting for the CALF model in particular, as there are results to compare to, and the model emphasizes temporal context. It would therefore be interesting to observe how the model reacts to audio features of shorter temporal size.

Bibliography

[1] URL: https://www.youtube.com/about/press/ (visited on 12/05/2020).

[2] URL: https://forzasys.com/forzify.html (visited on 18/06/2020).

[3] URL: https://research.google.com/youtube8m/ (visited on 22/05/2020).

[4] *3 ways online video is changing what it means to be a sports fan*. Jan. 2018. URL: https://www.thinkwithgoogle.com/consumer-insights/sports-fans-video-insights/ (visited on 12/05/2020).

[5] Sami Abu-El-Haija et al. 'YouTube-8M: A Large-Scale Video Classification Benchmark'. In: (Sept. 2016).

[6] Relja Arandjelović et al. *NetVLAD: CNN architecture for weakly supervised place recognition*. 2016. arXiv: 1511.07247 [cs.CV].

[7] John Arevalo et al. *Gated Multimodal Units for Information Fusion*. 2017. arXiv: 1702.01992 [stat.ML].

[8] Matt Brems. *A One-Stop Shop for Principal Component Analysis*. 17th Apr. 2017. URL: https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c (visited on 24/04/2021).

[9] Jason Brownlee. *A Gentle Introduction to Transfer Learning for Deep Learning*. 16th Sept. 2019. URL: https://machinelearningmastery.com/transfer-learning-for-deep-learning/ (visited on 15/05/2020).

[10] Shyamal Buch et al. 'End-to-End, Single-Stream Temporal Action Detection in Untrimmed Videos'. In: *BMVC*. 2017.

[11] Shyamal Buch et al. 'SST: Single-Stream Temporal Action Proposals'. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 6373–6382.

[12] Joao Carreira and Andrew Zisserman. *Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset*. 2017. arXiv: 1705.07750 [cs.CV].

[13] Joao Carreira et al. *A Short Note about Kinetics-600*. 2018. arXiv: 1808.01340 [cs.CV].

[14] Joao Carreira et al. *A Short Note on the Kinetics-700 Human Action Dataset*. 2019. arXiv: 1907.06987 [cs.CV].

[15] Yu-Wei Chao et al. *Rethinking the Faster R-CNN Architecture for Temporal Action Localization*. 2018. arXiv: 1804.07667 [cs.CV].

[16] Minghai Chen et al. *Multimodal Sentiment Analysis with Word-Level Fusion and Reinforcement Learning*. 2018. arXiv: 1802.00924 [cs.LG].

[17] Keunwoo Choi, George Fazekas and Mark Sandler. *Automatic tagging using deep convolutional neural networks*. 2016. arXiv: 1606.00298 [cs.SD].

[18] Anthony Cioppa et al. *A Context-Aware Loss Function for Action Spotting in Soccer Videos*. 2019. arXiv: 1912.01326 [cs.CV].

[19] Michael Copeland. 29th July 2016. URL: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/ (visited on 13/05/2020).

[20] Adrien Deliège et al. *SoccerNet-v2: A Dataset and Benchmarks for Holistic Understanding of Broadcast Soccer Videos*. 2021. arXiv: 2011.13367 [cs.CV].

[21] Peter Denning et al. 'Computing as a discipline'. In: *Computer* 22 (Mar. 1989), pp. 63–70. DOI: 10.1109/2.19833.

[22] S. Dieleman and B. Schrauwen. 'End-to-end learning for music audio'. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014, pp. 6964–6968. DOI: 10.1109/ICASSP.2014.6854950.

[23] Jeff Donahue et al. *Long-term Recurrent Convolutional Networks for Visual Recognition and Description*. 2014. arXiv: 1411.4389 [cs.CV].

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

[25] *FFmpeg (4.3)*. 2020. URL: ffmpeg.org.

[26] *Fourier Transform*. URL: https://en.wikipedia.org/wiki/Fourier_transform (visited on 26/03/2021).

[27] Dalya Gartzman. *Getting to Know the Mel Spectrogram*. Aug. 2019. URL: https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0 (visited on 26/03/2021).

[28] Jort F. Gemmeke et al. 'Audio Set: An ontology and human-labeled dataset for audio events'. In: *Proc. IEEE ICASSP 2017*. New Orleans, LA, 2017.

[29] Silvio Giancola et al. 'SoccerNet: A Scalable Dataset for Action Spotting in Soccer Videos'. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (June 2018). DOI: 10.1109/cvprw.2018.00223. URL: http://dx.doi.org/10.1109/CVPRW.2018.00223.

[30] Rohit Girdhar et al. *ActionVLAD: Learning spatio-temporal aggregation for action classification*. 2017. arXiv: 1704.02895 [cs.CV].

[31] Rohit Gosh. *Deep Learning for Videos: A 2018 Guide to Action Recognition*. URL: http://blog.qure.ai/notes/deep-learning-for-videos-action-recognition-review (visited on 13/05/2020).

[32] Chunhui Gu et al. *AVA: A Video Dataset of Spatio-temporally Localized Atomic Visual Actions*. 2017. arXiv: 1705.08421 [cs.CV].

[33] Larry Hardesty. *Explained: Neural networks*. 14th Apr. 2017. URL: http://news.mit.edu/2017/explained-neural-networks-deep-learning-0414 (visited on 15/05/2020).

[34] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[35] Fabian Heilbron, Juan Carlos Niebles and Bernard Ghanem. 'Fast Temporal Activity Proposals for Efficient Detection of Human Actions in Untrimmed Videos'. In: June 2016. DOI: 10.1109/CVPR.2016.211.

[36] Martin Heller. *Unsupervised learning explained*. 7th Aug. 2019. URL: https://www.infoworld.com/article/3429017/unsupervised-learning-explained.html (visited on 13/05/2020).

[37] *Introduction to Loss Functions*. 30th Apr. 2018. URL: https://algorithmia.com/blog/introduction-to-loss-functions (visited on 15/05/2020).

[38] Y.-G. Jiang et al. *THUMOS Challenge: Action Recognition with a Large Number of Classes*. http://crcv.ucf.edu/THUMOS14/. 2014.

[39] Dag Johansen et al. 'Search-based composition, streaming and playback of video archive content'. In: *Multimedia Tools and Applications* (1st Nov. 2012). DOI: 10.1007/s11042-011-0847-5.

[40] M. Esat Kalfaoglu, Sinan Kalkan and A. Aydin Alatan. *Late Temporal Modeling in 3D CNN Architectures with BERT for Action Recognition*. 2020. arXiv: 2008.01232 [cs.CV].

[41] Soo-Min Kang and Richard P. Wildes. 'Review of Action Recognition and Detection Methods'. In: *CoRR* abs/1610.06906 (2016). arXiv: 1610.06906. URL: http://arxiv.org/abs/1610.06906.

[42] Andrej Karpathy et al. 'Large-scale Video Classification with Convolutional Neural Networks'. In: *CVPR*. 2014.

[43] Will Kay et al. *The Kinetics Human Action Video Dataset*. 2017. arXiv: 1705.06950 [cs.CV].

[44] Bahador Khaleghi et al. 'Multisensor Data Fusion: A Review of the State-of-the-Art'. In: *Inf. Fusion* 14.1 (Jan. 2013), pp. 28–44. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2011.08.001. URL: https://doi.org/10.1016/j.inffus.2011.08.001.

[45] D. Kim et al. 'Multi-modal emotion recognition using semi-supervised learning and multiple neural networks in the wild'. In: *Proceedings of the 19th ACM International Conference on Multimodal Interaction* (2017).

[46] Qiuqiang Kong et al. *PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition*. 2020. arXiv: 1912.10211 [cs.SD].

[47] Hilde Kuehne et al. 'HMDB51: A Large Video Database for Human Motion Recognition'. In: Nov. 2011, pp. 2556–2563. DOI: 10.1109/ICCV.2011.6126543.

[48] Edith Law et al. 'Evaluation of Algorithms Using Games: The Case of Music Tagging.' In: Jan. 2009, pp. 387–392.

[49] Ang Li et al. *The AVA-Kinetics Localized Human Actions Video Dataset*. 2020. arXiv: 2005.00214 [cs.CV].

[50] Tianwei Lin, Xu Zhao and Zheng Shou. 'Single Shot Temporal Action Detection'. In: *Proceedings of the 2017 ACM on Multimedia Conference - MM '17* (2017). DOI: 10.1145/3123266.3123343. URL: http://dx.doi.org/10.1145/3123266.3123343.

[51] Tianwei Lin et al. *BMN: Boundary-Matching Network for Temporal Action Proposal Generation*. 2019. arXiv: 1907.09702 [cs.CV].

[52] Tianwei Lin et al. *BSN: Boundary Sensitive Network for Temporal Action Proposal Generation*. 2018. arXiv: 1806.02964 [cs.CV].

[53] Marcin Marszałek, Ivan Laptev and Cordelia Schmid. 'Actions in Context'. In: *IEEE Conference on Computer Vision & Pattern Recognition*. 2009.

[54] *Mel Scale*. URL: https://en.wikipedia.org/wiki/Mel_scale (visited on 26/03/2021).

[55] Sanatan Mishra. *Unsupervised Learning and Data Clustering*. 19th May 2017. URL: https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a (visited on 13/05/2020).

[56] Mathew Monfort et al. 'Moments in Time Dataset: one million videos for event understanding'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019), pp. 1–8. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2019.2901464.

[57] R. K. Nayyar et al. 'Content-based auto-tagging of audios using deep learning'. In: *2017 International Conference on Big Data, IoT and Data Science (BID)*. 2017, pp. 30–36. DOI: 10.1109/BID.2017.8336569.

[58] Olav A. Norgård Rongved et al. 'Real-Time Detection of Events in Soccer Videos using 3D Convolutional Neural Networks'. In: *Proc. of IEEE International Symposium on Multimedia (ISM)*. 2020, pp. 135–144. DOI: 10.1109/ISM.2020.00030.

[59] Juan Ortega et al. *Multimodal Fusion with Deep Neural Networks for Audio-Video Emotion Recognition*. July 2019.

[60] Adam Paszke et al. 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'. In: *Proc. of NIPS*. 2019, pp. 8024–8035.

[61] Jordi Pons et al. *End-to-end learning for music audio tagging at scale*. 2018. arXiv: 1711.02520 [cs.SD].

[62] Zhaofan Qiu et al. *Learning Spatio-Temporal Representation with Local and Global Diffusion*. 2019. arXiv: 1906.05571 [cs.CV].

[63] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2015. arXiv: 1506.02640 [cs.CV].

[64] D. A. Sadlier et al. 'A combined audio-visual contribution to event detection in field sports broadcast video. Case study: Gaelic football'. In: *Proc. of IEEE ISSPIT*. 2003, pp. 552–555. DOI: 10.1109/ISSPIT.2003.1341180.

[65] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. 15th Dec. 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (visited on 15/05/2020).

[66] Justin Salamon, Christopher Jacoby and Juan Bello. 'A Dataset and Taxonomy for Urban Sound Research'. In: Nov. 2014. DOI: 10.1145/2647868.2655045.

[67] Zheng Shou, Dongang Wang and Shih-Fu Chang. *Temporal Action Localization in Untrimmed Videos via Multi-stage CNNs*. 2016. arXiv: 1601.02129 [cs.CV].

[68] Gunnar A. Sigurdsson, Olga Russakovsky and Abhinav Gupta. *What Actions are Needed for Understanding Human Actions in Videos?* 2017. arXiv: 1708.02696 [cs.CV].

[69] Karen Simonyan and Andrew Zisserman. *Two-Stream Convolutional Networks for Action Recognition in Videos*. 2014. arXiv: 1406.2199 [cs.CV].

[70] Gurkirt Singh and Fabio Cuzzolin. *Untrimmed Video Classification for Activity Detection: submission to ActivityNet Challenge*. 2016. arXiv: 1607.01979 [cs.CV].

[71] Khurram Soomro, Amir Roshan Zamir and Mubarak Shah. *UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild*. 2012. arXiv: 1212.0402 [cs.CV].

[72] Subetha Thankaraj and Chitrakala Gopalan. 'A survey on human activity recognition from videos'. In: Feb. 2016, pp. 1–7. DOI: 10.1109/ICICES.2016.7518920.

[73] Matteo Tomei et al. *RMS-Net: Regression and Masking for Soccer Event Spotting*. 2021. arXiv: 2102.07624 [cs.CV].

[74] Du Tran et al. *Learning Spatiotemporal Features with 3D Convolutional Networks*. 2014. arXiv: 1412.0767 [cs.CV].

[75] Bastien Vanderplaetse and Stephane Dupont. 'Improved Soccer Action Spotting Using Both Audio and Video Streams'. In: *Proc. of IEEE CVPR Workshops*. June 2020.

[76] Kanav Vats et al. *Event detection in coarsely annotated sports videos via parallel multi receptive field 1D convolutions*. 2020. arXiv: 2004.06172 [cs.CV].

[77] Valentin Vielzeuf, Stéphane Pateux and Frédéric Jurie. *Temporal Multimodal Fusion for Video Emotion Classification in the Wild*. 2017. arXiv: 1709.07200 [cs.CV].

[78] Valentin Vielzeuf et al. *CentralNet: a Multilayer Approach for Multimodal Fusion*. 2018. arXiv: 1808.07275 [cs.AI].

[79] *What is Machine Learning? A definition*. 6th May 2020. URL: https://expertsystem.com/machine-learning-definition/ (visited on 13/05/2020).

[80] Aidan Wilson. *A Brief Introduction to Supervised Learning*. 29th Sept. 2019. URL: https://towardsdatascience.com/a-brief-introduction-to-supervised-learning-54a3e3932590 (visited on 13/05/2020).

[81] Fanyi Xiao et al. *Audiovisual SlowFast Networks for Video Recognition*. 2020. arXiv: 2001.08740 [cs.CV].

[82] Yue Zhao et al. *Temporal Action Detection with Structured Segment Networks*. 2017. arXiv: 1704.06228 [cs.CV].