Automated Thumbnail Selection for Soccer Videos using Machine Learning

Andreas Husa

Thesis submitted for the degree of
Master in Programming and System Architecture
60 credits

Department of Informatics
The Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2022

Automated Thumbnail Selection for Soccer Videos using Machine Learning

Andreas Husa

Automated Thumbnail Selection for Soccer Videos using Machine Learning

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

Printed: Reprosentralen, University of Oslo

Abstract

Thumbnail selection is a very important aspect of online sport video presentation, as thumbnails capture the essence of important events, engage viewers, and make video clips attractive to watch. Traditional solutions in the soccer domain for presenting highlight clips of important events such as goals, substitutions, and cards rely on the manual or static selection of thumbnails. However, such approaches can result in the selection of sub-optimal video frames as snapshots, which degrades the overall quality of the video clip as perceived by viewers, and consequently decreases viewership, not to mention that manual processes are expensive and time consuming. In this thesis, we present an automatic thumbnail selection system for soccer videos which uses machine learning to deliver representative thumbnails with high relevance to video content and high visual quality in near real-time. Our proposed system combines a software framework which integrates logo detection, close-up detection, face detection, blur detection, and image quality analysis into a modular and customizable pipeline, and a subjective evaluation framework for the evaluation of results. We evaluate our proposed pipeline quantitatively using various soccer datasets, in terms of complexity, runtime, and adherence to a pre-defined ruleset, as well as qualitatively through a user study, in terms of the perception of output thumbnails by end-users. Our results show that an automatic end-to-end system for the selection of thumbnails based on contextual relevance and visual quality can yield attractive highlight clips, and can be used in conjunction with existing soccer broadcast pipelines which require real-time operation.

Acknowledgments

I would like to thank my supervisors Pål Halvorsen, Cise Midoglu and Michael Riegler for all the help they have given. I would also like to thank Malek Hammou, Steven Hicks, Dag Johansen and Tomas Kupka for their help as well. I also want to thank my family and friends for their support and interest in my work as well.

This project was conducted within the context of the Research Council of Norway (RCN) project AI-PRODUCER (no. 327717) [45], which aims to automate the pipeline of soccer highlight production. This work fits into the project, as this is proposing a solution on automating the thumbnail selection, which is one of the tasks in this pipeline.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Sports broadcasting and streaming are immensely popular, and the interest in viewing videos from sports events grows day by day. Today, live streaming of sports events generates most of the video traffic and is replacing live broadcasting on TV [73]. For example, 3.572 billion viewers tuned in to watch the 2018 FIFA World Cup [20], and as of 2020, soccer had a global market share of about 45% of the $500 billion sports industry [65]. However, the availability of content and the large number of games make systems for extracting highlights and providing summaries in real- or near real-time increasingly important. The generation of video summaries and highlight clips from sports games is of tremendous interest for broadcasters, as a large percent of audiences prefer to view only the main events in a game.

The state-of-the-art systems for generating soccer highlight clips consist of several manual operations. Where important events such as goals, substitutions, and cards are tagged and annotated before being published into individual clips. A typical pipeline consists of a detection phase where the video is marked with an event such as a goal or card, relevant frames are cut to generate a highlight clip, and in a second "refinement" phase, the highlight clip is further improved by customized trimming, insertion of additional textual descriptions and tags, and the selection/update of a thumbnail.

A thumbnail is an image representing a video. For soccer, thumbnails are frequently used in web pages where various highlight clips are presented in gallery form [17, 3], and serve as the first impression meant to attract people to view a highlight clip. As highlight clips summarize certain important events in a soccer game, the challenge is to find appropriate thumbnails for each clip (and not a single thumbnail for the overall game). Thumbnails need to be selected carefully to be eye-catching and should properly represent the event in the highlight clip, as unattractive thumbnails can cause low engagement (highlight clips might go unwatched due to non-appealing thumbnails) [35, 60, 32]. Manual selection can potentially yield appropriate thumbnails, but since

the selection operation is time-consuming and expensive, image quality is often not considered extensively. Static selection on the other hand, can save resources, but potentially not provide quality. In this sense, automating the thumbnail selection process has the potential to both save resources and improve quality.

1.2 Problem Statement

Addressing the challenges of manual and static selection of thumbnails on video clips above, we want to research a method for selecting compelling thumbnails for soccer goal video clips automatically in real-time. The overall research question we aim to answer is:

> 'How can we select good thumbnails from soccer goal event clips automatically?'

To do this, we aim to implement a pipeline with several ML-based components that are added with the intention to assist the pipeline to select thumbnails that are following our ruleset. To create a ruleset, we need to look at related work and get subjective answers from several people. The ruleset we define, is reassessed during the work, as we do not get the subjective answers until later in the work. To tell if the thumbnails the pipeline selects are compelling, we want to get user validation through user study, where the automatic selected thumbnails can be compared to static and manual selected thumbnails. We want to analyze the system performance to evaluate its capabilities to perform in real-time. The performance of our framework will also be compared to other state-of-the-art automatic thumbnail selection frameworks. Then, to answer the overall research question, we narrow the tasks down to smaller parts, where we have defined 5 objectives that each will bring us closer to a final conclusion:

Objective 1 Define a ruleset of what a thumbnail should contain, by identifying the properties of a good thumbnail through related work and user study.

Objective 2 Implement end-to-end, automated, modular, configurable pipeline for selecting thumbnails, following our given ruleset in first objective.

Objective 3 Get user validation for the selected thumbnails of our framework through user study.

Objective 4 Analyze system performance and evaluate its real-time performance.

Objective 5 Compare our framework with other state-of-the-art automatic thumbnail selection frameworks, in terms of end-to-end duration and output.

1.3 Scope

This thesis focus on goal events specifically in the sport of soccer. We mainly use data from Scandinavian and English leagues for training and testing. It is also with a focus on real-time operation in mind. The thumbnail selection is only supposed to select frames within a given video and not modify anything other than scale. To make the thumbnail selection relevant, it is dependent on that the video clip it is given, is directly related to the event or if it is a longer video, the event is given with an annotation mark.

1.4 Research Methods

The overall goal for this thesis is to implement and test a framework for automatic thumbnail selection on soccer video clips. To do this, we have based our research method on Association for Computing Machinery (ACM)'s methodology [15]. This work describes three paradigms:

- **Theory paradigm** consists of four steps and is rooted in mathematics. Characterize objects of study (definition), hypothesize possible relationships among them (theorem), determine whether the relationships are true (proof), and interpret results.
- **Abstraction paradigm** consists of four stages and is rooted in experimental scientific method. Form a hypothesis, construct a model and make a prediction, design an experiment and collect data, and analyze results.
- **Design paradigm** consists of four steps and is rooted in engineering. State requirements, state specifications, design and implement the system, and test the system.

This thesis mostly applies the design paradigm. We state requirements, design and implement the system. We finally test the final system to determine the practical use for automatic thumbnail selection in soccer videos.

1.5 Main Contributions

In this work, our goal is to replicate the potential performance of manual thumbnail selection with an automated system, which is much faster (applicable in real-time), cheaper, and quantitative (documentable and reproducible). This will not only save resources for the top soccer leagues which already rely on manual selection, but also enable similar services for less resource-capable leagues where automation is the only alternative. We present HOST-ATS [26, 25], a holistic system for the automatic selection and evaluation of soccer video thumbnails. HOST-ATS is composed of: (1) a dashboard-controlled Machine Learning (ML) pipeline, and (2) a dynamic

user survey. The pipeline uses ML to automatically select thumbnails for soccer videos in near real-time, and can be configured via a Graphical User Interface (GUI). It combines logo detection, close-up shot detection, face detection, image quality prediction, and blur detection. HOST-ATS also includes a dynamic user survey for evaluating the results of the pipeline qualitatively (through user studies). This web-based survey is fully configurable and easy to update via Continuous Integration (CI), allowing for the dynamic aggregation of participant responses to different sets of multimedia assets.

- We propose a modular and customizable automatic thumbnail selection pipeline, which integrates pre-processing (options for trimming, down-scaling, and down-sampling), logo detection, close-up detection, face detection, and image quality analysis (image quality prediction and blur detection). The modular and lightweight implementation allows for the agile integration and benchmarking of various ML methods from literature in each module. This pipeline can be controlled via a dashboard with a user-friendly GUI.

- We evaluate our proposed pipeline quantitatively using various soccer datasets, in terms of system performance (complexity and runtime) as well as adherence to a pre-defined ruleset, under different configurations for components.

- We run a subjective evaluation (user study) involving 43 participants, to evaluate the performance of the proposed pipeline qualitatively. The subjective evaluation campaign is conducted using a novel survey framework for crowdsourced feedback collection on multimedia assets, which is a plug-and-play system component for any future studies as well[1].

- Additionally, we run a small benchmark study with a state-of-the art thumbnail selector which is generic (non soccer-specific).

- We provide a discussion of the generalizability of our approach, along with the limitations and pitfalls of our pipeline, and suggest potential improvements and future work topics.

Overall, the novelty of our work includes: the combination of various independently applicable approaches in an end-to-end automatic thumbnail selection pipeline, with an overarching goal and a generalized ruleset; corresponding quantitative performance and complexity analyses; a novel user study framework for the qualitative evaluation of different thumbnail candidates; as well as the insights gained from the study and subsequent discussions. Our automatic thumbnail selection system is able to reduce production costs by automating a traditionally complex and labor-intensive task, accompanied by end-user validation.

[1]Live deployment of the HOST-ATS subjective evaluation (user study) framework: https://host-ats.herokuapp.com

1.6 Outline

Chapter 2 Background and Related Work In this chapter, we introduce work related to soccer highlight production. Also, in this chapter, we describe key concepts and terminology related to ML. At the end of this chapter, we present related work to different ML modules that can be relevant for the implementation of a thumbnail selection pipeline.

Chapter 3 Methodology In this chapter, we describe our proposed thumbnail selection pipeline in detail. We also present our user study framework.

Chapter 4 Experiments and Results (First Iteration) In this chapter, we present the results from our quantitative analysis of the temporary pipeline in a first iteration. The performance of the pipeline is assessed by a user study.

Chapter 5 Experiments and Results (Second Iteration) In this chapter, we present the results from our quantitative analysis of the final pipeline in a second iteration. We compare the performance of first and second iteration pipeline. We also compare our framework to other frameworks.

Chapter 6 Discussion In this chapter, we discuss and elaborate our decisions made in previous chapters.

Chapter 7 Conclusion In this chapter, we summarize the work done in this thesis and suggest ideas for future work.

Chapter 2

Background and Related Work

Based on the challenges described in the previous chapter, we aim to implement a framework that can automatically select thumbnails from soccer goal videos. To understand the problem and the solution, it is necessary to understand the concepts on which it builds. This chapter goes through relevant terminology, the automation of soccer highlight production, an introduction to machine learning and relevant models.

2.1 Terminology

Terminology used throughout this thesis:

- **Soccer:** Also called association football[1], played in accordance with a codified set of rules known as the Laws of the Game (LOTG)[2] by the International Football Association Board.
- **Broadcast:** A transmit of television.
- **Streaming:** Different than broadcasting, Over-the-Top (OTT) delivery of content.
- **Soccer game/match:** Soccer game is any game including unofficial.
- **Football club:** 'An organization of players, managers, owners or members associated with a specific football team.' [13]
- **Event:** (in the context of multimedia content, not to be confused with the entire soccer game) Also called "highlight". According to the Cambridge Dictionary an event is defined as: 'anything that happens, especially something important or unusual' [18]. In this thesis it will be focused on something important or unusual in a soccer match. For this occasion an event is distinct and the list below includes types of events that are relevant in a soccer match.
  - Goal: When the ball passes the goal line.

[1] https://en.wikipedia.org/wiki/Names_for_association_football
[2] https://www.fifplay.com/downloads/documents/laws-of-the-game-2021-2022.pdf

  - Card: When the referee hands a yellow or red card to a player.
  - Substitution: When one player is substituted off and another player goes on the pitch.

- **Highlight clip:** Video clip displaying a particular event from a soccer game.
- **Tagging/annotation:** Setting timestamps on events in soccer match, adding metadata, etc..
- **Tagging center:** A typical tagging center in live operation is shown in Figure 2.1.
- **Thumbnail:** An image used for representing a video.
- **Shot:** A sequence of frames captured by a single camera.
- **Shot/scene boundary:** A transition between two successive shots.
- **Close-up:** An image or shot that shows subject up close and detailed. Typically making the subject cover most of the frame.

2.2 AI-PRODUCER

AI-PRODUCER [45] is a framework that aims to automate the entire event and highlight production in sports. This can reduce production costs by automating a traditionally complex and labor-intensive task. The entire pipeline consists of a event detection [42, 52, 53] phase where the video is marked with an event such as a goal or card (Figure 2.1b), relevant frames are cut to generate a highlight clip, and in a second "refinement" phase, the highlight clip is further improved by customized trimming [70, 69], insertion of additional textual descriptions and tags, and the selection/update of a thumbnail (Figure 2.1d).

To achieve an automated framework like the AI-producer, it is possible to automate each phase individually. This work focuses on automating the last phase, thumbnail selection.

The pipeline of event and highlight production in soccer is illustrated in Figure 2.2. The order of the tasks in the pipeline is from left to right and the black lines between the blue boxes, indicates a dependency for the task on the right side of the line. A task is only depending on the earliest task on the left side [3]. Since a task is not depending on all its previous tasks to be finished, to be started on, some bottlenecks are avoided. All the tasks, except 'Game summarization', are tasks that are provided during the soccer match. 'Game summarization' is made after the match is finished. In a

[3] In the pipeline of event and highlight production in soccer, the tasks with dependencies (black lines), are only depending on the first occurring task on the left side of the black line. For instance, 'Thumbnail selection' is only depending on 'Event detection and detailed annotation'. The 'Event clipping' is just complementary for the task. The same occurs for the 'Text description'. 'Game summarization' is first of all dependending on 'Event clipping', where 'Text description' is complementary.

manual production, the time usage on each typically takes several seconds. From the actual event occurrence, to the highlight video clip is ready, could take about a minute. This highly depends on resources and efficiency of the available tools in the production. In an automatic pipeline, with state-of-the-art components on each task, the total execution time could result in a few seconds.

For the thumbnail selection, it is possible to select a relevant thumbnail with only having the annotation mark. The event clipping can help for the thumbnail selection to know the cut points the thumbnail should be inside. If the thumbnail selection is only dependent on the event annotation, the thumbnail selection can avoid being a potential bottleneck, if the event clipping is spending more time than the thumbnail selection.

(a) One person following multiple games. **(b)** Detection and classification.

(c) Metadata and fine-granular clipping. **(d)** Thumbnail selection.

Figure 2.1: A manual live tagging operation by people in a cumbersome, error-prone, and tedious manual process [70]

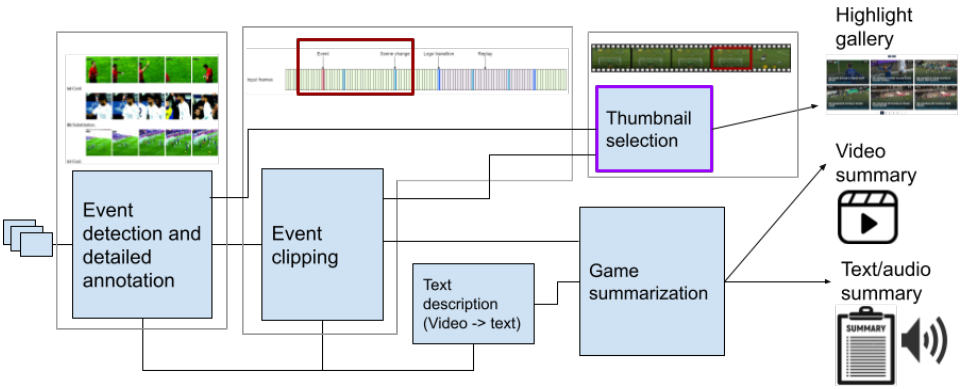


Figure 2.2: Illustration of the pipeline of event and highlight production in soccer presented in [45]. The graphics above the blue boxes is illustrating the specific task it is above. The purple square around 'Thumbnail selection', highlights what task this thesis is focusing on.

2.2.1 Event Detection

Event detection, also called *action detection* or *action spotting*, has lately received a lot of attention. For example, multiple variants of two-stream convolutional neural networks (CNNs) have been applied to the problem [29, 59], and extended to include 3D convolution and pooling [19, 6]. Wang et al. [58, 75] proposed temporal segment networks (TSN), and C3D [66] explored 3D convolution learning spatio-temporal features. Tran et al. [67] used (2+1)D convolutions to allow the network to learn spatial and temporal features separately. Further approaches aim at finding temporal points in the timeline [27, 51, 21, 76, 36, 37, 53], but even though many of these works present interesting approaches and promising results, such technologies are not yet ready to be used in real-life deployments. The reason is that most of the proposed models are computationally expensive and relatively inaccurate. In deployments where action/event annotation results are used in an official context (e.g., live sports broadcasts), these must be 100% accurate, i.e., no false alarms or missed events are allowed, so manual operations are still needed.

Another relevant work is made by Rongved [42, 52]. His purpose was to detect events in soccer videos with 3D CNN. The algorithm made, used a sliding window approach to scan over a given video to detect events. The events events were classified as goal, card, substitution and background. Background class is indicating there are no event like goal, card or substitution occurring. The model could detect events with high recall, low latency and accurate time estimation. It had lower precision than the model made by Cioppa [11], which was referred to as 'current state-of-the-art'. Cioppa's model compared to Rongved's model, had higher latency and could perform better if a less accurate time estimation was accepted.

2.2.2 Event Clipping

In the area of *event clipping*, the amount of existing work is limited. Koumaras et al. [34] presented a shot detection algorithm, and Zawbaa et al. [80] implemented a more tailored algorithm to handle cuts that transitioned gradually over several frames. Zawbaa et al. [79] classified soccer video scenes as long, medium, close-up, and audience/out of field, and several papers presented good results regarding scene classification [77, 79, 47]. Video clips can also contain replays after an event, and replay detection can help filter out irrelevant replays. Ren et al. [50] introduced the class labels play, focus, replay, and breaks. Detecting replays in soccer games using a logo-based approach was shown to be effective using a support vector machine (SVM) algorithm, but not so effective using an artificial neural network (ANN) [79, 80]. Furthermore, it was shown that audio may be an important modality for finding good clipping points. Raventos et al. [48] used audio features to give an importance score to video highlights, and Tjondronegoro et al. [64] used audio for a summarization method, detecting whistle sounds based on the frequency and pitch of the audio. Finally, some work focused on learning spatio-temporal features using various ML approaches [59, 6, 66], and Chen et al. [9] used entropy-based motion approach towards the problem of video segmentation in sports events.

Valand and Kadragic [70, 69] made a model for event clipping that were tested on soccer videos from Eliteserien and Premier League. Through a survey, the event clipping their models made were more preferred than the already existing event clipping for Eliteserien. The already existing one had a static clipping that included a constant number of frames before the event and after the event. This method could for instance result in clipping in the middle of a replay.Valand and Kadragic made two solutions consisting of logo detection, scene boundary detection and a production protocol. The first one included some frames before the goal and everything shown on the screen until the replays was done. The other one cut out most of the celebration scenes after the goal and skipped right to replay videos.

2.3 Thumbnail Selection

As a representative snapshot, thumbnails capture the essence of a video and provide the first impression to the viewers. It helps in recognizing what video it is representing in a visual way and is broadly used in video streaming services [78, 68, 72]. For soccer, thumbnails are frequently used in web pages where various highlight clips are presented in gallery form [17, 3], shown in Figure 2.5, and serve as the first impression meant to attract people to view a highlight clip.

2.3.1 What is a good thumbnail?

Related work indicates that a good thumbnail is relevant to the corresponding video, and appears interesting and attractive in terms of content and

image quality [35, 60, 32]. These related works, also states that a good thumbnail is crucial to catch the viewers interest and if a person views a video clip, it can be dependent on the thumbnail. Though this work interests in good thumbnails for soccer specifically, we have not found any work related to this topic.

The article by Knott [32], points out 4 ways to make thumbnails interesting. Color use, close-ups of faces, use of bold clear text, and consistency. These are characteristics he recognizes on most of the thumbnails he inspects on popular videos on Youtube [78]. The thumbnails in Figure 2.3 is also mentioned as good examples with these characteristics. To clarify what Knott mean by color use, it is to use color in a way that it attracts the eyes of the viewer and there are many methods to do so. Close-ups of faces, should convey emotions to make the thumbnail interesting and informative. Bold and clear text is to make it easy for the viewer to know what the video is about. Consistency, is about making the thumbnails from the same production similar in terms of themes and colors. This way, the viewer can possibly know what production is presenting the video by recognizing their pattern. The article by Law [35], also approves that color use, bold and clear text, and consistency as methods to make a thumbnail compelling.

The article by Song [60] focuses on two main criteria for the thumbnail, relevance and attractiveness. To fulfill the relevance criteria, the thumbnail image has to be from the video itself and be representative for the video content. To fulfill the attractiveness criteria, the visual attractiveness of the thumbnail is measured using computational aesthetics.

Given all these criteria from different sources, it does not seem that the thumbnails in Figure 2.5a follow many of these criteria.

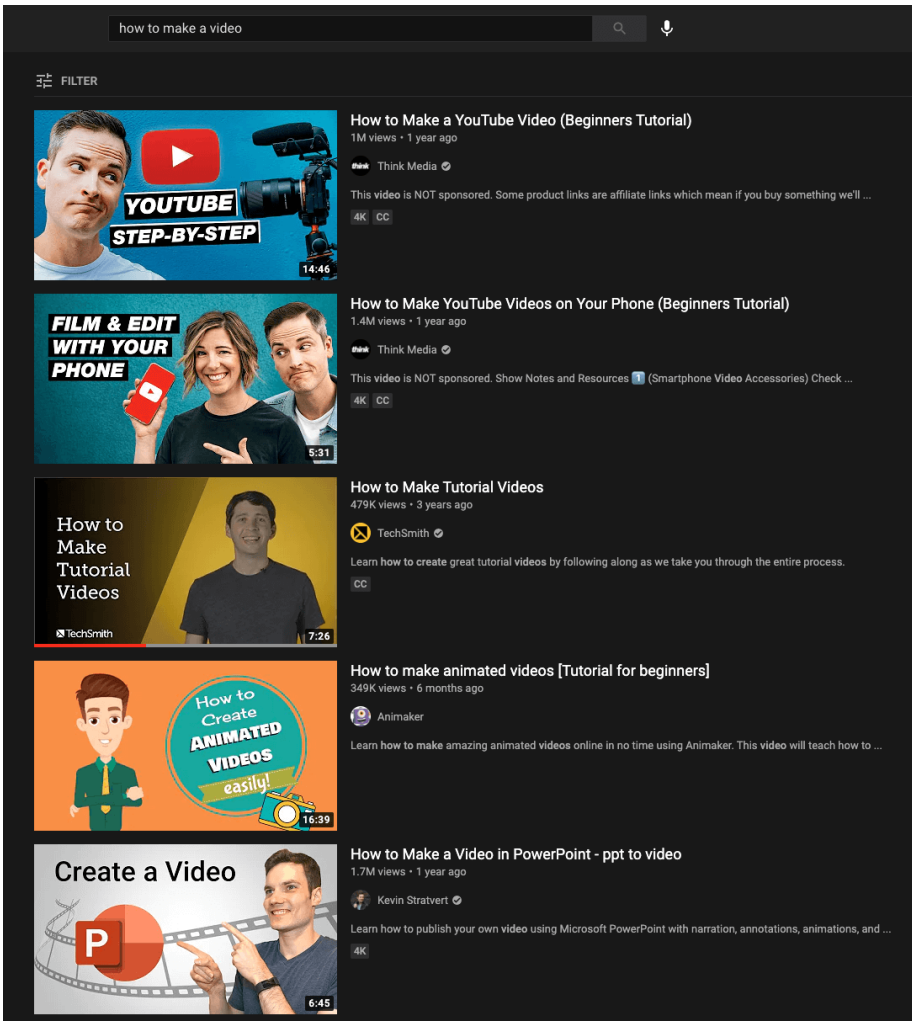


Figure 2.3: Thumbnails from Youtube [78], which Knott [32] mentions as good thumbnail examples.

2.3.2 What is a good thumbnail for soccer?

When selecting a thumbnail for soccer, the criteria for a good thumbnail could differ from the general thumbnail, as it is a more specific field. In video production of soccer matches, there are typical types of content appearing frequently on the screen throughout the stream. Relevant types of images the video production usually consists of are shown in Figure 2.4. Given that a thumbnail only can be directly from the video, the selector has to decide which types of content is more favorable than other. There is not much related work to thumbnail selection for soccer or sports in general. However, Don [16] wrote an article giving advice on how to create compelling thumbnails for soccer highlight videos. Even though this thesis is focusing on specifically soccer goal events, it could be relevant information. Don provide a list of official soccer highlight video thumbnails and points out similarities most/all of them have. Almost all of the thumbnails contained an action image. Don also mentions that

they usually display league logo, club logos and result of the match. The thumbnails also often used color themes either matching the official league colors or colors of the club publishing the video. Don did not point this out, but the thumbnails usually contained players and there were no presence of motion blur, only defocused/blurry background.

(a) Action close-up

(b) Blurry

(c) Celebrating players with no face

(d) Celebrating players with face

(e) Celebrating crowd

(f) Logo transition

(g) Team staff

(h) Zoomed out

Figure 2.4: Typical types of images a soccer broadcast production usually consists of during a soccer match.

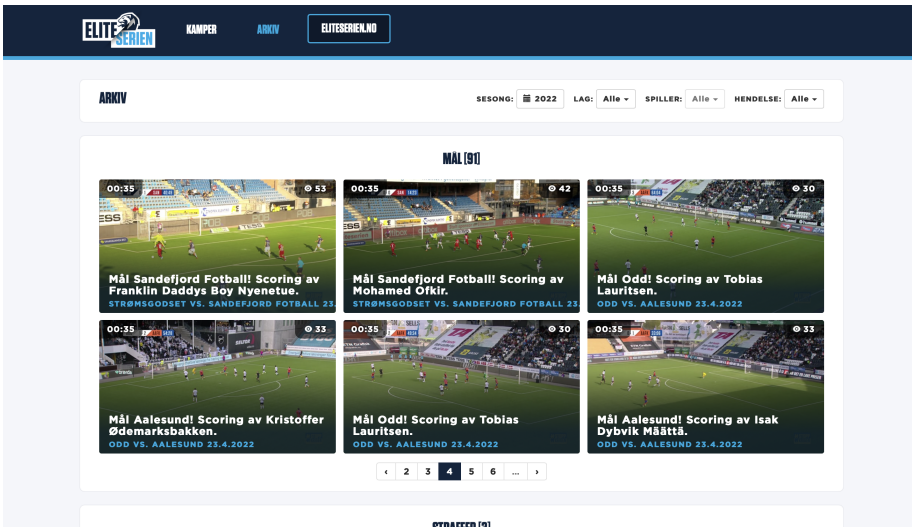
2.3.3 Thumbnail Selection Methods

Manual selection method is the traditional way of selecting thumbnails. For soccer productions, it is typically done in a tagging center, as shown in Figure 2.1d. Manual selection can potentially yield appropriate thumbnails, but the selection operation can be time-consuming and expensive. When selecting thumbnails manually, the choice can be

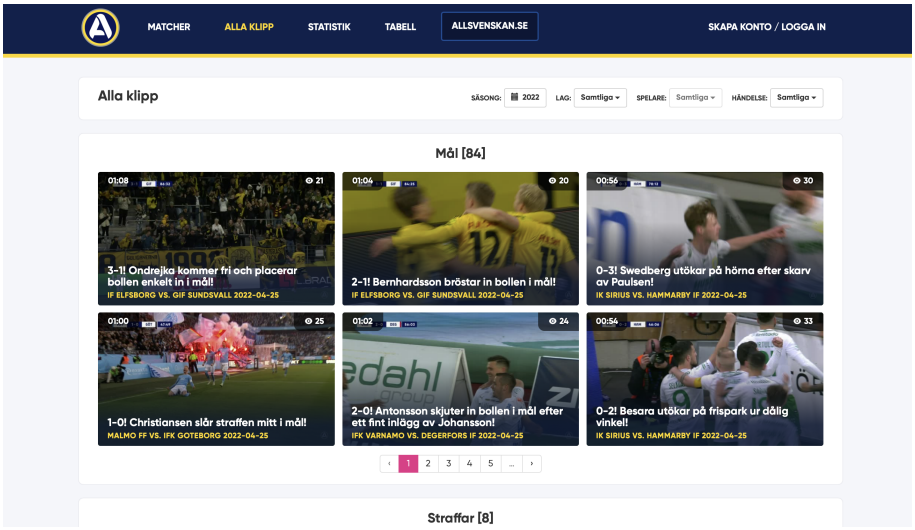
validated by people before being used. Depending on the purpose, the thumbnail can be assessed to fulfill specific requirements that can differ from other purposes.

Static selection method can save resources that is provided in manual selection. Though it is less time consuming and less cost, it does not guarantee quality. There are different ways of doing a static selection of thumbnails. This can be done with selecting a random frame from the given video clip or choosing a frame X seconds into the video clip. By choosing static it can be challenging to make the thumbnail fulfill specific requirements as there is no assessment methods during the selection process.

Automatic selection is a method using metrics or ML-based components to assist the selection process. There is not much work on automatic thumbnail selection specifically for sports videos, but there are a number of related works that focus on thumbnail selection in general. Song et al. [60] propose a generic model called "Hecate" for selecting thumbnails automatically. Their framework uses a video as input, and filters the frames that are qualified as low-quality such as blurry, dark or uniform-colored frames. This is calculated with and decided upon via a threshold value, and not through ML. The framework also filters frames that are related to fading, dissolving or wiping effects in the video, identifying these through a shot boundary detection model. In a second step, frames that are near duplicates are discarded, and finally, frames with highest aesthetic quality are selected. This can be done by selecting the frame from a cluster with the smallest difference value (i.e., the frame that has the least change from the other frames in the same cluster, or the mean frame), where clusters are frames that have visual similarities. The aesthetic quality can be calculated by using a model that assigns a beauty score to a given image. This model has been trained by a set of images that have been annotated with subjective aesthetic scores. Vasudevan et al. [71] present a query-adaptive video summarization model which picks frames from a given video that are relevant to the given query. The model also has the possibility to output a single frame as a thumbnail. The query is a text of what content the end-user would like the frame to contain (e.g., in our context, it could be "soccer" or "goal").



(a) Screenshot of highlight page on official website of Eliteserien [17].



(b) Screenshot of highlight page on offical website of Allsvenskan [3].

Figure 2.5: Screenshots of official highlight pages of Eliteserien and Allsvenskan.

2.4 Machine Learning - Basics

Machine learning is about making computers modify or adapt their actions so that these actions get more accurate. Machine learning is often split into three different categories: supervised, unsupervised and reinforcement learning. For this thesis there will be a focus on supervised learning and more specifically convolutional neural network. The best performing models made for event detection and object detection at the moment are based on convolutional neural network.

Supervised learning is a learning method that require training data where the correct response is provided. An algorithm use the training set as a basis when the algorithm is meant to give a correct answer on possible

inputs. The supervised learning can be elaborated further. It is the most common type of learning method in machine learning. The training data set, which is used on the model for learning, consists of a set of data (x_i,t_i) where x_i is the input value and t_i is the target value. i is the index value which spans from 1 to the amount of tuples N. The model is supposed to be trained so that it can respond correctly with the corresponding t to a given input x. The goal of supervised learning is that it is trained enough so it can predict the right output for new instances. The ability of predicting the output for new instances is also known as generalization.

Unsupervised learning is a learning method where correct responses are not provided. The algorithm has to identify similarities between the inputs. The inputs that has similarities are categorized together.

Reinforcement learning is a learning method where the algorithm gets told when the answer is wrong, but it is not given how to correct the answer. This leads to an exploration for finding the right pattern and the wrong answers will eventually be less.

2.4.1 Data Collection

When decided what problem the machine learning is going to solve, it is necessary to gather data relevant for that given problem. Collecting huge amounts of data can lead to much variety in data format if they are collected from different places. That can lead to extra work in merging the data. One also has to be aware of missing or inaccurate data, as this can be mislead the machine learning algorithm. Another problem to consider when using huge amounts of data for the machine learning algorithm is that it leads to computational costs. However, the model needs significant amounts of data to solve its problem optimally.

2.4.2 Cross Validation

There are different types of data sets that are in use in the machine learning process. They all have the same format (x_i,t_i).

Training set is a data set for training the model and tuning its parameters.

Validation set is an unseen data set for the model to estimate a model's generalization performance while tuning its hyperparameters.

Test set is an unseen data set for the model to estimate a model's generalization performance when finally tuned.

2.4.3 Gradient Descent

Gradient descent is an optimization algorithm for a machine learning model. It iteratively improves the model towards a more optimal model. The algorithm improves the model by using a loss function that measures how well the model predicts the output.

2.4.4 Performance Evaluation

There are terms that are central in describing a model's generalization performance.

Accuracy is a measure for how many correct predictions that has been made of all examined cases.

$$Accuracy = \frac{true\ positives + true\ negatives}{positives + negatives}$$

Precision is a measure of the accuracy on the cases predicted as true. It is describing the reliability when a case is predicted as true.

$$Precision = \frac{true\ positives}{true\ positives + false\ positives}$$

Recall is a measure of how well a generalization predicts true on all the cases that should be true. It is describing how good a model is to detect the relevant cases.

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

F1-score is the harmonic mean of the precision and recall. It gives a better measure of the incorrectly classified cases than the accuracy metric.

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

Matthew's correlation coefficient (MCC) is used as a measure of the quality of binary classifications. The coefficient ranges from -1 to $+1$ where 0 indicates no relationship, 1 indicates perfect agreement and -1 indicates total disagreement. 0 means it is not better than random prediction.

$$MCC = \frac{(tn * tp) - (fn * fp}{\sqrt{(tp + fp) * (tp + fn) * (tn + fp) * (tn + fn)}}$$

Overfitting is a problem to be aware of in machine learning. This is when the model is fitting too well on a particular set of data and corresponds very well on that given data set, but it fails on generalizing on new instances. The main goal for the model is that it manages to predict right output on new instances. Figure 2.6 is illustrating the problem with overfitting.

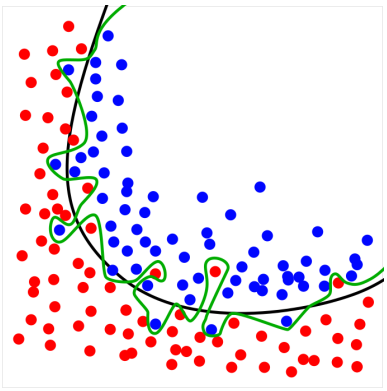


Figure 2.6: This is an example from Wikimedia [8], illustrating the problem with overfitting. The red and blue dots are the two different classes the model are supposed to distinguish. The green line illustrates an overfitted model and the black line illustrates a more moderate model. In this scenario the more moderate model will probably have a better generalization performance on new unseen data.

Underfitting occurs when the algorithm can neither model the training data nor generalize on new data. Being too cautious of overfitting could lead to underfitting, so the model needs to be somewhere between overfitting and underfitting.

2.5 Machine Learning - Families of Algorithms

2.5.1 Regression

Regression is one method to use in supervised learning. Regression is typically used when the output scores are real numbers. When doing regression the goal is to identify the relationship between the input values and the target values that has been given. A regression analysis gives a possibility to find an estimated target value for an arbitrary input value.

Linear regression is a simple regression model where the model finds a linear relationship between input value and target value.

Non-linear regression is a regression model where the model finds a non-linear relationship on the data set. Non-linear regression is used on the more complex data sets compared to the linear. Regression is a method that suits well for predicting values such as temperature, age and price.

2.5.2 Classification

Classification is another method in supervised learning. This is a method that is typically used when the output scores are absolute. The output scores are separated into classes. Classification is a method that suits well for predicting classes such as 'Yes or no' and 'Which country?'.

2.5.3 Support-Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression. It is effective in cases where the number N of dimensions are greater than the number of samples. The SVM is attempting to find a hyperplane that distinctly classifies the data points in the N-dimensional space. The goal is to place the hyperplane with a maximum margin to the nearest datapoints from each class. If there is a very low margin from one datapoint, it increases the chance of miss-classification.

2.5.4 Neural Networks

Neural Network (NN) is a system inspired by the neural network in the brain. It consists of connected neurons or nodes where the neurons can receive signal, process it and transmit it to another neuron. The signal is also known as a real number. The connection between two neurons are known as an edge. The neurons and edges has weights that adjusts as the learning proceeds. A typical NN is illustrated in Figure 2.7.

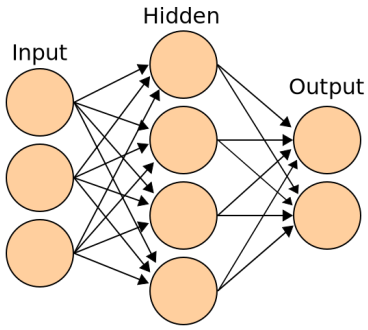


Figure 2.7: An illustration from Wikimedia [7] of a typical NN consisting of three layers: input, hidden and output layer. The circles are the neurons and the black arrows are the edges.

2.5.5 Convolutional Neural Networks

Convolutional Neural Network (CNN) are often used in object detection on images. It typically consists of filters on several layers that are different from one another. The first layers are finding more simple features, but it is gradually becoming more complex in the last layers. The network consists of various types of layers and typically pooling, convolutional, fully-connected layer.

Convolutional layer does the convolutional process in a CNN. It uses a *kernel* or *filter* that is a matrix of values. The kernel is multiplied on a subset of the input, where the subset has the same dimensions as the kernel. The total sum of that multiplication operation is the new value in the output or convolved feature that has smaller dimensions than the input. The convolution process is illustrated in Figure 2.8.

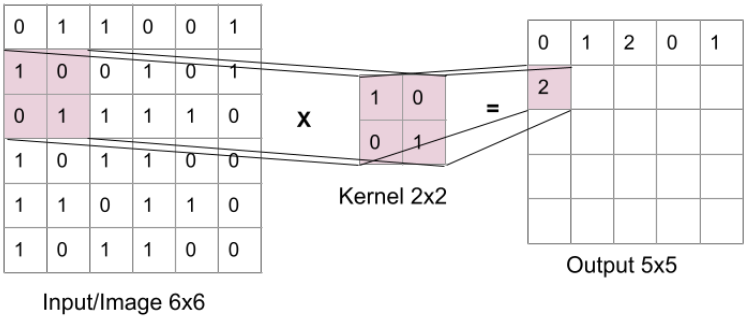


Figure 2.8: An illustration of the convolution process.

Pooling layer does the pooling process which is similar to the convolution. An input grid outputs as a smaller grid. Instead of multiplying subsets with a kernel, there will be done a pooling operation on the subsets. There are three types of pooling operations that are maximum, minimum and average pooling. The maximum pooling operation for instance, outputs the highest value of the subset.

Fully connected layer is a layer where every neuron is connected with all neurons in previous layer. This is typically found in the last layers in a CNN. This layer helps to map representation between input and output.

One simple example of the use of CNN is if there is a cat present in a given frame. There is only necessary with one output node representing the cat class. There will be given a probability number to the output node giving an indication if the animal is present. If the probability score is higher than a given threshold score, the animal is present. A complete CNN is illustrated in Figure 2.9.

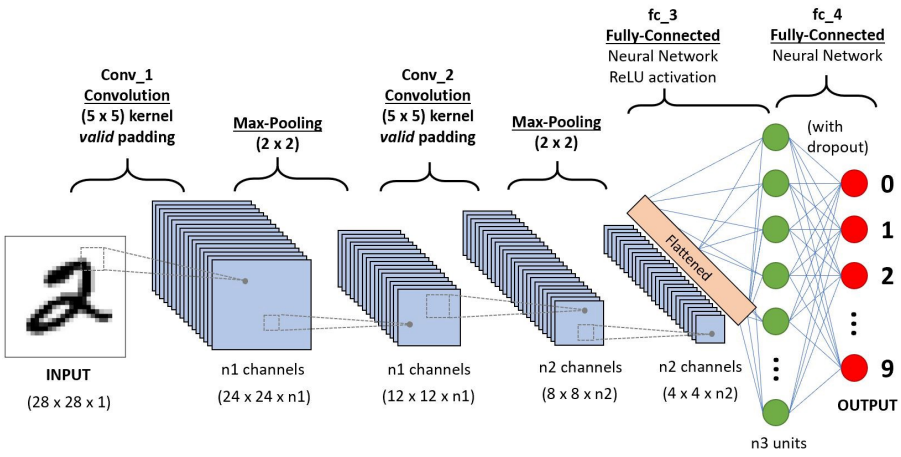


Figure 2.9: A typical example of a CNN from Towards Data Science [56]. This one receives a frame as input and in the output layer the network classifies which number is presented in the frame.

2.6 Machine Learning - Selected Algorithms

2.6.1 Single Shot Multibox Detector

Single Shot Multibox Detector (SSD) is an algorithm for object detection in real-time. It detects several objects simultaneously and classifies them as well. It downsamples the input, but raises the semantic value for each layer. Since the SSD downsamples the input for each layer it has difficulties to detect small objects.

2.6.2 You Only Look Once

You Only Look Once (YOLO) is another algorithm for object detection in real-time. It is a state-of-the-art algorithm, presented for the first time in 2015 [49]. An YOLO type of architecture consists of several convolutional layers and ends with fully connected layers. The reason why it is called 'Only Look Once' is because it processes the input image through the system once from start to end. The algorithm predicts multiple bounding boxes and associated class probabilities.

2.6.3 Feature Pyramid Network

Feature Pyramid Network (FPN) is based on CNN. The idea of using this is to detect small objects. For each convolutional operation the resolution decreases, but the semantic value increases. This describes a bottom-up pathway. This is very helpful for detecting large objects. The problem is that small objects slightly vanishes in several convolution operations. The FPN combines the bottom-up pathway with a top-down pathway as well. The top-down pathway reconstructs the layers with increasing resolution combined with rich semantic (gained from the bottom-up pathway) as well. Figure 2.10 shows an illustration of FPN.

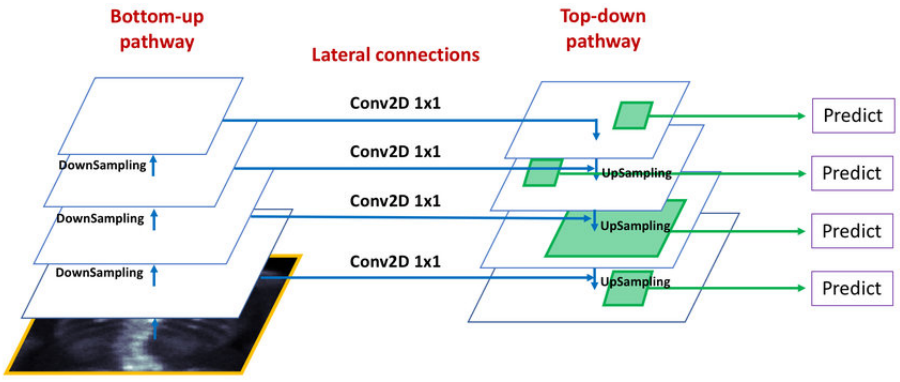


Figure 2.10: Illustration of FPN from Apiparikoon [4]

2.6.4 Surma Image Classifier

The image classifier made by Surma [62] is a model that consists of 2 convolutional layers followed by max pooling, which is repeated 4 times.

The output of the last pooling layer is flattened in the next layer. Then there is 5 layers where the order is dense, dropout, dense, dropout and then dense again. At last there is a activation layer with sigmoid function. For binary classification, the model classifies the input as one class if the probability score is below 0.5, and as the other class if the score is between 0.5 to 1.0. The closer the number is to 1 or 0, the more certain the model is of the input being in the predicted class.

2.7 Object Detection

Object detection is the act of detecting if a classified object appears and where it appears. Objects that could be relevant to detect in this thesis would be object classes such as ball, player, referee, logo, goal, and soccer pitch.

Object detection on soccer ball specifically is challenging and it appears to be many researchers that have not got a solid enough detection system for soccer ball yet [39]. Since the ball has to be detected on a video frame, the detection model is dependent on video quality and it has several challenges detecting ball on a two-dimensional frame according to Komorowski [33]:

- The size of the ball varies in amount of pixels, it can be as small as 8 pixels on the far side of the pitch and as big as 20 pixels on the near side of the pitch.
- The ball often has different shapes from one frame to another. It sometimes appears blurry and elliptical in its shape.
- There could be balls not in play, appearing on the frame.
- The ball can blend in with other objects that has the same color and they overlap with each other, which could make the ball lose its visible shape on a two-dimensional frame.
- There are times when the ball is covered or partially covered from the cameras point of view.

In the article of Komorowski [33], there is made a deep neural network detector named *FootAndBall* detector. It tries to detect ball and players appearing on the screen by the use of FPN design pattern. The detection method is used on single video frames. This model had a average precision score of 0.909 on ball detection. It has remarkably less parameters than a generic deep neural network-based object detector. That helps for "real-time processing of high resolution input video stream [33]". The reason it is possible to reduce the amount of parameters is because this object detector model is dedicated to only detect objects from two classes that are limited in shape and size variance.

2.8 Face Detection

Haar: [74] is an object detection model which is fast, but tends to be prone to false positive detection, compared to other models [55]. This algorithm can be run in real-time, making it possible to detect objects in live video streams. It is possible to train the model for detecting other objects as well as faces. It is capable of detecting objects regardless of their scale and position in an image.

One reason for the Haar being fast is that it is a feature based system instead of pixel based system. On a part of an image a rectangle feature as seen in Figure 2.11 are used to obtain the features from an image. To obtain features in an image one subtract the sum of pixels in the white region from the sum of pixels in the black region. The value after the subtraction indicates if the feature is current or not. This can be used to detect faces as we know that the eye region is often darker than the nose and cheek region. Finding patterns like this can help us predicting if a face is present or not in an image.

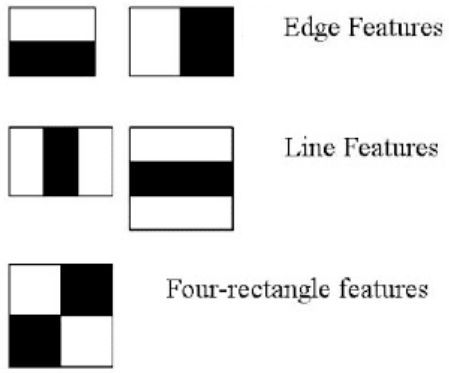


Figure 2.11: Different Haar features extracted from an image [54]

Dlib: Dlib [31] is written in C++, but it has Python bindings so it is possible to run in python. It uses features extracted by HOG and passes them through a SVM. HOG counts the occurrences of gradient orientation on fragments of the picture. The method can be helpful for finding shapes in the picture. Figure 2.12 presents a visualization of how histogram of oriented gradients (HOG) works.

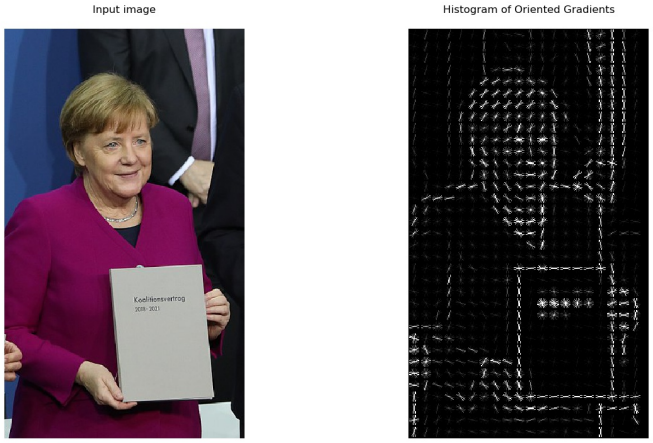


Figure 2.12: A visualization of HOG from Wikimedia [41].

MTCNN: The Multi-Task Cascaded Convolutional Neural Networks (MTCNN) was first introduced in the paper by Zhang in 2016 [81]. The MTCNN has three stages of CNN. In the first stage, it uses a Fully Convolutional Network (FCN) to obtain candidate windows. The second stage, it filters all the false positive candidates. The last stage, a facial landmark detection is performed.

DNN: The DNN [5] face detector in OpenCV is a Caffe model which is based on the SSD and uses ResNet-10 architecture as a backbone.

Haar, Dlib, MTCNN and DNN are all different face detection models compared in performance in an article by Agarwal [1]. Haar is performing worst when it comes to accuracy on detecting faces, but Haar has an advantage of being fast. It was also mentioned that DNN did not perform well on large images. For example images with size of about 3000x3000 pixels. Dlib did not detect any face smaller than 80x80 pixels.

2.9 Shot Boundary Detection

Shot Boundary Detection is supposed to detect when a video switches to a new camera view. Šarić [57] made a Shot Boundary Detection algorithm based on standard twin-comparison. Twin-comparison method compares two consecutive frames to detect change in pixels. A significant change could indicate a transition. There are two types of shot transitions that are described: abrupt and gradual. Abrupt are easier for a method to detect, since the pixel change from one frame to the next are very high. A Shot Boundary Detection method has less accuracy on finding the gradual transitions as the change from one frame to the next are very low compared to the abrupt transition. The proposed algorithm made by Šarić [57] had competitive results and had better precision with very low number of false positives. It did have lower recall than the standard as the proposed

algorithm eliminated several true gradual transitions.

2.10 Image Quality Assessment

There has been made image quality analysis (IQA) models for predicting the subjective and/or objective quality of an image. One that has been made by Jongyoo [30] was tested on detecting distortion types on image, primarily white noise and blur. This is a no-reference image quality assessment (NR-IQA) model meant to be used on assessing images without reference images. In some practical scenarios, there may be no reference images, then it can be useful to have a general IQA.

2.11 Blur detection

Related is also blur detection and OpenCV [5] has programming functions for estimating blur on images. The estimation is given as a value and the higher the value is, the more blurry it is.

Numpy [23] offers a Singular value decomposition (SVD) function that can be used to calculate the blurriness of an image. The function returns an array of vectors and Su [61] uses the vectors for further calculation to get a final blur score. In the array of vectors, the values are sorted in descending order. The way the blur is calculated, is summing only the highest value vectors (default is 10 first vectors) and divide it on the total sum of all vectors on the image. If the sum of the top vectors is close to the total sum of all the vectors, the number indicates high presence of blur as the number is close to 1. But if it is opposite, close to 0, it indicates very little blur.

Our work focus on sports, and there can be a lot of motion in soccer resulting in several frames that end up being blurry. When selecting a single frame from a video, it is important to avoid images that are too blurry for aesthetic reasons, but also keeping the image informative and representative for the event. It should be possible to see what is happening on the image and too much blur could avoid that.

2.12 Soccer Datasets

There have already been made data sets of soccer videos that can come in good use. **SoccerNet** [14] is a proposed dataset, consisting of 500 complete untrimmed soccer matches from different European leagues. There is a total of 300 000 annotations on these videos. The annotation classes are cards, goals, substitutions, penalty, corner, etc.

2.13 Summary

In this chapter, we gave definitions of relevant terminology to begin with. We then introduce the automation of event and highlight production in

soccer. Going through work related to event detection and event clipping. Then, background information related to thumbnail selection is presented, mentioning there is little work related to thumbnail selection specifically for sport videos. Then, there is several sections on machine learning: basic machine learning, elements in machine learning architecture, and then more specific proposed ML algorithms. Next, we present related work on object detection, face detection, shot boundary detection, image quality assessment and blur detection. Then we mention relevant dataset for ML-related work on soccer. As there is missing thumbnail selection frameworks related to sports, the next chapters is motivated on proposing a framework as a solution on the problem statement given in Section 1.2.

Chapter 3

Methodology

In previous chapter, background information and related work to this thesis was presented. Much of the work presented there will be used in this work, and in this chapter, we will introduce the proposed framework for automatic thumbnail selection. First, a ruleset for the thumbnail selection is proposed. The framework is implemented following this ruleset. Moreover, the datasets used for training and testing is presented, followed by the implementation details. Finally, we present a method for evaluating the performance of the pipeline in terms of end-user perception.

3.1 Proposed Framework

To implement an automatic thumbnail selection framework, we must first define what makes a good thumbnail. In Section 2.3, there are mentioned several characteristics that can make a thumbnail appealing. We centered our framework around 3 key principles, namely relevance, content, and image-quality. In the following, we describe these key principles in more detail and explain how they impact our thumbnail selection process.

- **Relevance:** The thumbnail that our proposed framework selects will be a frame from the video it is supposed to represent (i.e., no external images that will be impossible for an automated system to relate to the clip). Video frames from the highlight clip will be used as input to our framework, so the output image is relevant to the video as it is a frame around the event annotation.
- **Content:** Highlight clips are usually presented in a gallery as a grid, where each thumbnail appears in a small size (e.g., 200 pixels) on the screen. It could be difficult for viewers to understand the contents of the image if the thumbnail displays a long-distance view of the soccer field. Therefore, we resolve to use close-up shots in our framework. Close-up shots, in a soccer match video, are usually frames showing the soccer players, spectators, and managers. There could also be frames from the replay of the goal event with shots that are closer than the default long-distance shot. If a frame is identified as a close-up, it will have a higher priority in the thumbnail selection process.

We would also like to omit graphics such as the logo transitions appearing before replays. So, if a frame is identified as containing a logo, it will not be used as a thumbnail. Graphics are generic production elements and are not unique for the specific event.

- **Image quality:** It is possible that there are frames in a video which individually appear aesthetically unpleasing or unclear to the human eye. There can be a lot of motion in soccer games, resulting in video frames being blurry. When selecting a frame from a video clip as a thumbnail, we want to avoid selecting a blurry image. Other characteristics like darkness and fading are not usable as thumbnails either. Therefore, we propose to undertake blur detection and image quality prediction as the final filter.

Based on the above observations, we devise a number of thumbnail selection goals for the framework, which are listed in Table 3.1. These rules do not mean that our definition of a good thumbnail is universal, but rather establish a framework which is tailored for soccer videos, which rules and priorities easily can be changed later. These are further used in our framework consisting of 3 steps: pre-processing, content analysis and priority assignment, and image quality prediction. Figure 3.1 presents these steps and the corresponding components in our framework. A video clip (sequence of video frames) is fed as input to the framework, and the final output is an image, which is a frame from the video, as the suggested thumbnail.

Category	No	Rule
Relevance	1	The thumbnail should be a frame from the video clip itself.
Content	2	The frame should be a close-up shot of people.
	3	The frame should contain a face.
	4	The frame should not contain graphics (e.g., logo).
	5	The frame should not contain visuals of a fading transition.
Image Quality	6	The frame should not be blurry.
	7	The frame should not be dark.

Table 3.1: Thumbnail selection rules.

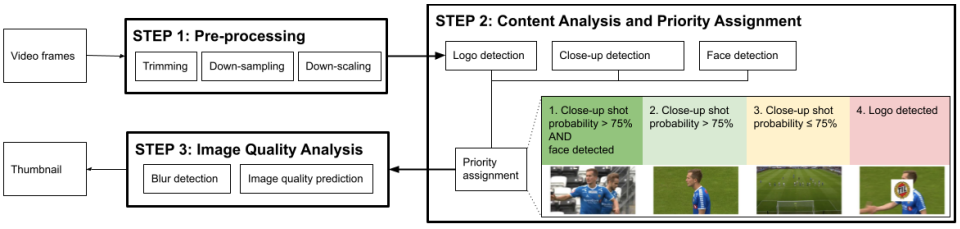


Figure 3.1: Proposed automatic thumbnail selection pipeline.

3.1.1 Step 1. Pre-processing

In the pre-processing step, the sequence of frames[1] can be trimmed, down-sampled, and/or down-scaled. *Trimming* refers to the cutting of a desired number of seconds from the beginning and/or end of the sequence. It is also possible to define a time interval of interest with respect to an event annotation. This allows for increasing the relevance of the thumbnail candidates to the central event in the clip, by ensuring that images come from around the event annotation timestamp. *Down-sampling* refers to the extraction of a lower number of frames as a subset from the full set of frames in the sequence. It allows for decreasing the number of frames that are considered further on in the pipeline, consequently decreasing the amount of processing time. The default number of frames extracted, is 50 frames.[2]

Down-scaling refers to the reduction of image resolution (changing the resolution of individual frames) in terms of a percentage. Table 4.5 shows the influence of downscaling on accuracy of the IQA. Each of these operations are optional, with configurable parameters. The set of frames remaining after pre-processing is passed to the next step.

3.1.2 Step 2. Content Analysis and Priority Assignment

In the second step, the contents of the frames passed on from the first step are analyzed according to rules $2 - 5$ from Table 3.1. For this purpose, independent modules for logo detection, close-up detection, and face detection are used, after which priorities are assigned to each frame and the frames are filtered accordingly.

Logo Detection

The logo detection module is for detecting logos and other graphics that might appear in the video frames. These are frames we would like to eliminate from the thumbnail selection. As a model for this module, we train a CNN based on the architecture presented by Surma [62]. This is a general image classification model that can classify images based on a given dataset. The output of the model is a probability score between 0 and 1, where an output of 0.5 or above indicates that the image contains a logo. 0.5 is the default probability threshold value and it is possible to adjust for improving the performance metrics of the model.

Close-up Detection

The close-up detection module is used to decide whether a video frame depicts a scene coming from a wide-angle camera (zoomed-out, long-

[1]Sequence of frames refers to the frames in the original input (video clip). E.g., a 30 second video clip at 30fps would yield 900 frames.

[2]Extracting 50 frames for a video clip, assumes that the video clip after trimming, is not more than 2 minutes. A video clip longer than 2 minutes is likely to have video content outside the specific event.

distance shot), or a close-up (zoomed-in) shot. Images classified as close-ups are prioritized in thumbnail selection. Our pipeline supports the use of the image classification model by Surma [62] in this module. The threshold value for model certainty is a configurable parameter. Like the logo detection module, images with a probability below this threshold, receive a lower priority.

Face Detection

Face detection is used to detect the appearance of a face on a given image, as well as where the face appears on the image. In our context, we consider a thumbnail image to be more relevant if there is a face appearing in it. Traditionally, face detection models rely on frontal views, and cannot detect a person's head from behind. It is possible to consider images with faces turned to an angle (such as a person's head from the side or behind) to be just as relevant. However, models for detecting such phenomena are more complex and less accurate, so in this work we only consider frontal face views for the sake of simplicity and accuracy. Our pipeline supports 4 alternative models for this module. The models supported are Haar cascade [74], Dlib [31], MTCNN [81] and DNN [5]. Any one of the models can be used in our pipeline for face detection, specified via configuration parameters. The default model for the face detection module is DNN.

Priority Assignment

As shown in Figure 3.1, the results from the logo detection, close-up detection, and face detection modules are passed to a priority assignment module before being filtered. The reasons for the priority levels are to be able to use several metrics concurrently in the evaluation of candidate images, and to control how greedy we would like our framework to be in enforcing our thumbnail selection rules. Our pipeline uses 4 priority levels:

(4) Images that are classified by the logo detection module as containing a logo are assigned to priority level 4, the lowest priority[3]. Images that are classified by the logo detection module as not containing a logo proceed ahead.

(3) Images that are classified by the close-up detection module as close-up image, are sorted in descending order of their probability score (i.e., image with the highest probability score comes first). Images with a score below the threshold value (default=75%) are assigned to priority level 3. Images with a score above the threshold value proceed ahead. Rankings are preserved in both cases.

(2) Images that are classified by the face detection module as not containing any faces are assigned to priority level 2.

[3]If all the images are classified as containing a logo, they are all assigned to the priority level 1 and passed to the image quality prediction module. In this case, the thumbnail presented as the overall result of the framework will be an image with a logo, which, despite being undesirable, is preferred over no output.

(1) Images that are classified as having at least one face are sorted according to the pixel size of the biggest face detected on the image in descending order, and assigned to priority level 1.

Figure 3.2 presents the decision tree used for determining the presented priority levels.

Note that increasing the threshold for the probability score from the close-up detection module can increase the adherence to the established thumbnail selection rules (more rigid enforcement of rule 2 from Table 3.1) and decrease latency (as the face detection module will take shorter if fewer images are passed on from the close-up detector), but also carries the risk of omitting potentially viable thumbnail candidates by assigning more images to the lower priority levels. Another trade-off is related to the final sorting of images which have been assigned to priority level 1, as sorting by face size vs. sorting by close-up probability emphasize different preferences. We propose that this choice be made depending on the accuracy of the models used in the face detection and close-up detection modules. For instance, in Chapter 4, we sort the images assigned to level 1 by face size in descending order, since the face detection model Dlib has better accuracy than the close-up detection model Surma, especially on bigger faces.

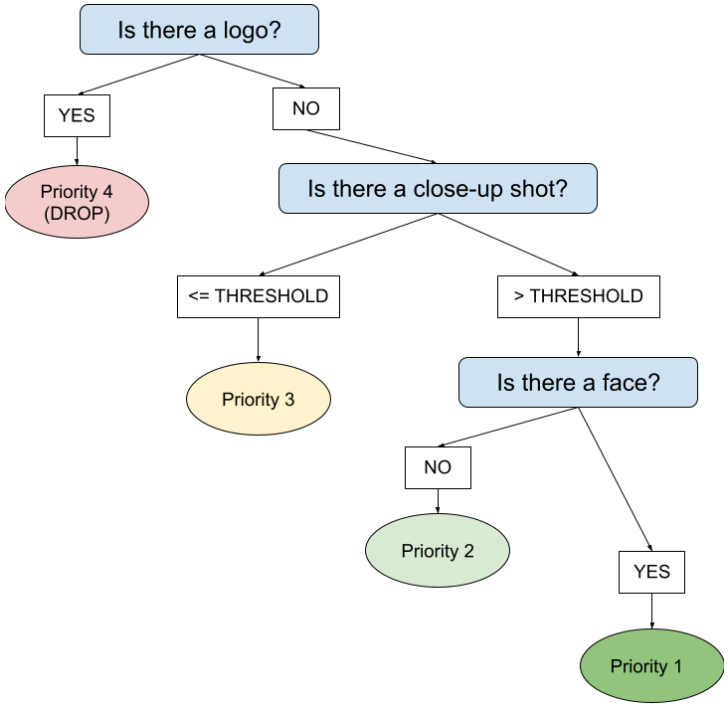


Figure 3.2: HOST-ATS priority assignment decision tree.

3.1.3 Step 3. Image Quality Analysis

The assignment of images to priority levels in the second step gives us a sorted list of thumbnail candidates. Then, one by one from the top,

each candidate goes through the third step. The iteration for selecting a thumbnail candidate starts at the highest priority level, and follows the image order prescribed in the previous step. As mentioned above, we prefer sorting images by the size of the largest face detected in them at this level. If there are no images assigned to the higher priority level, the iteration skips to the next priority level. During the iteration process, a blur detection will be run on each image. Images with higher presence of blur than the given threshold value, will be filtered out. The remaining images will be iterated for the image quality predictor [46]. It is supposed to predict the quality of an image by calculating its blur and distortion. If the BRISQUE score from the quality predictor is below a threshold, the image is chosen as the output thumbnail. The image quality prediction module is only ran until the first image to satisfy this condition[4], and the image will become the final thumbnail. If none of the images in the priority level satisfy the condition, the image with the lowest score becomes the final thumbnail.

3.2 Datasets

In this work, we make use of 2 in-house video datasets generated from the Norwegian and Swedish men's elite soccer leagues[5], focusing on goal events. A public available dataset, SoccerNet, is also included in the datasets. Table 3.2 presents the distribution of samples in our datasets into training, validation, and test splits. The resolution of each dataset is also given in their belonging row.[6].

The **Eliteserien** dataset consists of 300 clips of goal events scored in the Norwegian Eliteserien. Most of these clips are starting 25 seconds before the goal event occurs and ends 50 seconds after the event.

- **Eliteserien for logo detection (E1):** For logo detection, we use the annotations made in [70]. It contains $1,024$ images classified as logos and $7,024$ images classified as background.
- **Eliteserien for close-up detection (E2):** For close-up detection, we create an image dataset containing frames that are extracted from the Eliteserien dataset. This dataset consists of 2 classes, "close-up" and "no-close-up". Images classified as close-ups, often contain players and managers, where others include wide views of the pitch,

[4]The image quality predictor [46] can spend almost up to a second per image. The amount of images processed by the image quality predictor is reduced. This is with the purpose of not spending too much time on selecting a final thumbnail.

[5]The reason we create our own datasets is that each module in our pipeline requires different ground truth labels (e.g., close-up vs. long distance view for one module, logo vs. no-logo for another). Therefore, multiple datasets, or a large dataset with multiple ground truth dimensions is needed. However, there is no publicly available dataset we can use for all modules in our pipeline.

[6]As our pipeline is configurable, different image resolutions can be used for training and testing (different from the original assets and/or different between training and testing). The influence of resolution (or "scale") is further investigated in Tables 4.3-4.5

spectators, etc. Images of spectators celebrating are often not as close as those of players celebrating. Blurriness or image noise are not taken into account when considering which class an image belongs to. Overall, this dataset contains 702 images classified as close-ups and 968 images classified as no-close-ups.

- **Eliteserien for face detection (E3):** For face detection, we create an image dataset containing frames extracted from the Eliteserien dataset, consisting of 2 parts: "Close-up" and "Audience". The "Close-up" part contains 89 images varying in amount of faces appearing on the image. Some of the faces in these images are presumably undetectable by a face detector, as they are turned away from the camera. Having these faces in the dataset as well can be helpful to evaluate the performance of the face detector. The second part of the dataset, "Audience", is comprised of images showing the audience. It has a total of 32 images. The distance from the camera to the audience varies, but the images mostly contain faces that are smaller than the faces appearing in the "Close-up" dataset. This second dataset is generated for the purpose of determining if a model is good at detecting small faces.

The **Allsvenskan** dataset consists of goal clips from the Swedish Allsvenskan. This dataset is similar to the Eliteserien, but there is more variation in video clip length. Most of the clips start approximately 25 seconds before the goal event and end about 60-80 seconds after the goal event. There are 233 clips in the dataset. This dataset is not used for training and has its purpose of being an unseen dataset for the pipeline. This can be useful to evaluate the performance of the pipeline on soccer productions in general.

- **Allsvenskan for logo detection (A1):** For testing the logo detection module, we annotate a number of video clips from the Allsvenskan. Overall, this dataset contains 180 images, where 69 images contain a logo and 111 images do not contain a logo. As mentioned, this dataset is not used for training, but only for testing.

Another dataset that is used is **SoccerNet** [14]. It has 500 annotated games from different professional soccer leagues.

- **SoccerNet for logo detection (S1):** For logo detection, we use the annotations made in [70]. The samples in this set is from the Premier League Season 16/17. This logo dataset has several different logos, but also types of transitions which can make the training set more diverse. The number of samples is also much bigger compared to the logo datasets E1 and A1.

Dataset	Resolution (px)	Class	Train.	Val.	Test	Total
E1	200 x 120	Logo	561	240	223	1024
		No logo	4281	1355	1388	7024
E2	960 x 540	Close-up	564	109	29	702
		No close-up	772	158	38	968
E3	960 x 540	Close-up			89	89
		Audience			32	32
A1	480 x 270	Logo			69	69
		No Logo			111	111
S1	398 x 224	Logo	22240	6938	7004	36182
		No Logo	29378	9302	9102	47782

Table 3.2: The distribution of the samples in our datasets into training, validation, and test sets.

3.3 Implementation

Our automatic thumbnail selection pipeline is implemented using Python `v3.9.7`, tensorflow-cpu `v2.6.0`, keras `v2.6.0`, opencv-python `v4.5.5.64`, numpy `v1.22.3`, pillow `v9.0.1`, argparse `v1.1`, tkinter `v8.6`, cmake `v3.22.3`, dlib `v19.23.1`, MTCNN `v0.1.1`, image-quality `v1.2.7` and scikit-image `v0.18.3` on a DGX-2 server. This server is well suited for heavy computational operations and heavy memory operations. However, it is not necessary to use a machine that is exceptionally suited for heavy operations to run the pipeline. The image classifier model by Surma [62] which is used for logo detection and close-up detection is trained on the same server as well. For training the image classifier, the same versions of Keras and Tensorflow were used, as well as matplotlib `v3.4.3`, livelossplot `v0.5.4` and efficientnet `v1.1.1`. The logo detection and close-up detection models used by the framework are saved in Hierarchical Data Format (HDF) format, and the Haar cascade face detection model is saved in Extensible Markup Language (XML) format. The codebase for our pipeline is publicly accessible as an open-source software repository under[7], and further artifact details are provided in the reproducibility appendix of [26] (see Section A.1). The publicly accessible code, is also possible to run on Code Ocean [24]. A demonstration of our pipeline is presented in [25]. Furthermore, the datasets for logo detection and close-up detection are split into training, validation and test sets. The distribution of the samples are presented in Table 3.2. To evaluate the performance of the classifier-based modules in our pipeline (logo detection, close-up detection, face detection), we use accuracy, precision, and recall as metrics. For the image quality prediction module, we use BRISQUE score, which is a score given to an image to indicate its image quality. A smaller score indicates better perceptual quality [38].

Tensorflow: Tensorflow is an open-source software library for machine learning. It can be used in several programming languages, but it is most optimal in Python [63].

[7] https://github.com/simula/host-ats

KERAS: Keras is a deep learning API written in Python using Tensorflow. It focuses on enabling fast experimentation on deep neural network [10].

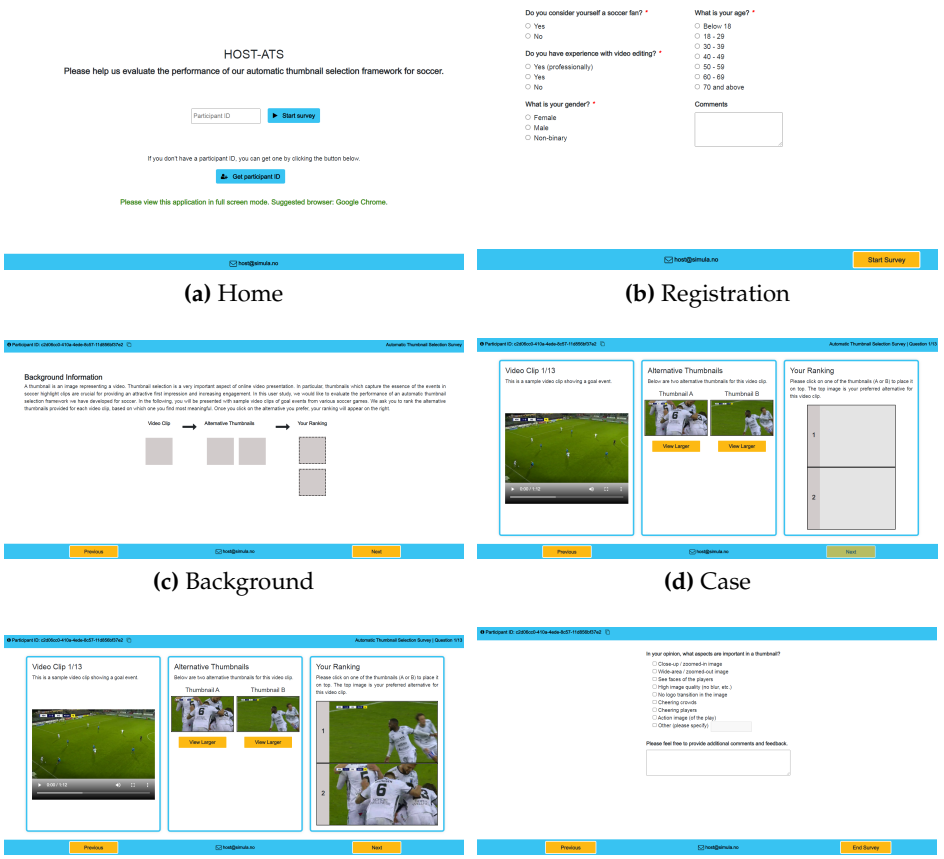
Hyperparameters: All training done on Surma [62] model, have been trained with the default hyperparameters. Input shape is 200x200x3, batch size is 32, number of epochs is 20, and learning rate is 0.0001.

3.4 User Study Framework

In order to evaluate the performance of the pipeline in terms of end-user perception, we provide a dynamic survey which can be used to conduct user studies. For instance, each video clip for which a thumbnail is selected can be presented as a "case", where survey participants are asked to compare the thumbnail selected by our pipeline, and a thumbnail selected by other methods (e.g., static frame selection method used by the industry today). As a user study, we made an online survey, where the goal was to get sufficient data to better understand what a consumer thinks makes a good thumbnail.

This user study can give an answer in form of a metric which thumbnail is favored and it is also possible to get a qualitative answer from the participant what they think could be a preferable thumbnail in soccer goal events. Not all participants are willing to give answers in own sentences. It can then be be useful to propose some different characteristics a thumbnail could have and they can answer whether they agree or not.

`Huldra` [22] is our framework for collecting crowdsourced feedback on multimedia assets. This framework allows for the collection of participant responses in a storage bucket hosted on the cloud, from where they can be retrieved in real-time by survey organizers, using credentials, immediately after the first interaction of each participant. As part of our automatic thumbnail selection system, we customize the Huldra framework and call it HOST-ATS. Figure 3.3 presents screenshots of the main pages of HOST-ATS.



(a) Home **(b)** Registration

(c) Background **(d)** Case

(e) Case (ranked) **(f)** Feedback

Figure 3.3: Screenshots from the main pages of the HOST-ATS user study.

The survey begins with the home page (Figure 3.3a) which allows users who already have a universally unique identifier (UUID) to complete their responses if they have closed the browser involuntarily, or decided to continue later. In both cases, their information remains saved in the browser's local storage. However, if the participant does not have a UUID, they must complete a registration form (Figure 3.3b) where we ask for participant information regarding age, gender, video editing experience, and soccer fandom (mandatory), as well as a free form text field if the participant has other relevant comments to add (optional). Details regarding the registration page input fields are given in Table 1 in the appendix of [26] (see Section A.1). After successful login or registration, the user is redirected to the background page (Figure 3.3c) which introduces the context of the study and shows directions for use with a simple figure. The core of the framework lies in the ranking of multimedia assets.

This functionality is provided by the case page (Figure 3.3d) which is composed of 3 vertical columns as follows:

- The left column presents a sample video clip showing a goal event. We use the npm package React player [12] for playback, which offers an off-the-shelf component for playing a variety of multimedia.

- The middle column presents two alternative thumbnails for the video clip. Both of which could be viewed larger if needed. The user ranks simply by clicking on one of the thumbnails.
- Once a thumbnail is clicked, it is displayed immediately on the top in the right column (Figure 3.3e).

In order to have complete and consistent responses for our study, we make it mandatory to rank a case before proceeding to the next. Users can later go backwards and revisit their answers or change them. After finishing the ranking, the participants are invited to fill out a feedback form (Figure 3.3f) about the aspects that they deem important in a thumbnail. They can mark from a list of alternatives, as well as suggest other facets. We also give them the option to add additional comments and feedback in a text field input. The full list of questions can be found in Table 1 in the appendix of [26] (see Section A.1).

3.5 Summary

In this chapter, we introduce our proposed pipeline for automatic thumbnail selection. The pipeline consists of 3 steps: '1.Pre-processing', '2.Content Analysis and Priority Assignment' and '3.Image Quality Prediction'. The modules 'Logo Detection', 'Close-up Detection' and 'Face detection' are included in the second step and 'Image Quality Prediction' and 'Blur Detection' is included in the third step. We continue to introduce the datasets of Eliteserien, Allsvenskan and SoccerNet [14]. We further describe the hardware and the libraries used for implementation and development of the pipeline. Finally, we present a method for evaluating the performance of the pipeline in terms of end-user perception. This is by providing a dynamic survey which can be used to conduct user studies.

The implementation of the pipeline is separated in two iterations, where the two next chapters, Chapters 4 and 5, represent the experiments for each iteration. The idea of separating the pipeline implementation in two parts, is to evaluate a complete pipeline after the first iteration. The evaluation can give knowledge on what to improve further for the second iteration.

Chapter 4

Experiments and Results (First Iteration)

In previous chapter, there was proposed a framework for automatic thumbnail selection.

In this chapter, the first iteration of implementing the pipeline is presented. All components added in this iteration are inspected with test results. There is also a user study for evaluating the pipeline and get ideas of how to improve the thumbnail selection of the pipeline. At last, we discuss what we can work further on to improve and increment the current framework.

4.1 Pipeline

The final pipeline for the first iteration is visualized in Figure 4.1. This pipeline is without any blur detection in step 3 (Image Quality Analysis). Not including DNN face detection and the logo detection is not trained on SoccerNet.

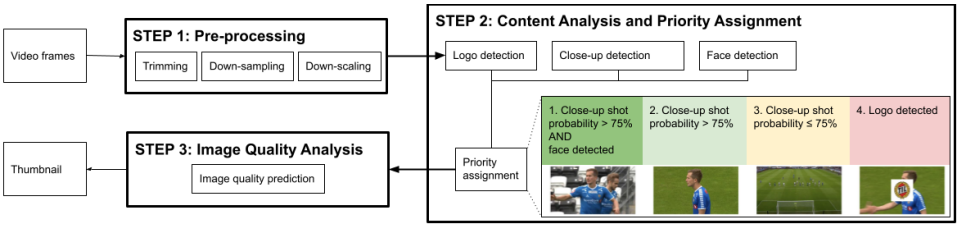


Figure 4.1: Automatic thumbnail selection pipeline in first iteration. This is not the final pipeline like presented in Figure 3.1. This one does not include the blur detection module in step 3.

4.2 Logo Detection Performance

We train the Surma [62] model on the E1 dataset mentioned in Section 3.2. The results are displayed in Table 4.1, where the first row corresponds to

a threshold value[1] of 0.5 (default value). The model achieves an accuracy of 0.994 on the E1 test split. Upon inspection, we see that most of the "no-logo" images have a probability score below 0.01, indicating that the model strongly (and correctly) identifies that the image does not contain a logo, most of the time. The precision is higher than the recall, indicating that there are more false negatives than false positives. To make the model have higher recall and maybe higher overall accuracy, it is possible to consider adjusting the threshold value. It would be reasonable to assume that the accuracy of the model would be improved by lowering the threshold value. For instance, an image with over 0.1 probability is likely to contain a logo, so we test how the threshold value influences the accuracy, using 0.1 and 0.05 in addition to the default value of 0.5. The results for a threshold value of 0.1 are given in the second row of Table 4.1. Even though the threshold value is changed by a large percentage from 0.5 to 0.1, the accuracy of the model is not affected considerably. Precision is lower, as the number of false positives increases, but the recall is improved. We argue that having high recall is more important than having high precision, as classifying images which do not actually contain a logo to contain a logo (false positive) causes the pipeline to be more conservative[2], and possibly drop more thumbnail candidates, rather than cause it to allow for rule 4 in Table 3.1 to be broken. The third row of Table 4.1 displays the results for an even lower threshold value, namely 0.05. The recall remains the same, but the precision is worse. We therefore conclude that adjusting the threshold value further than 0.1 is not necessary for better accuracy.

Overall, from the first part of Table 4.1, we see that the logo detection module performs adequately well on the Eliteserien dataset (E1 test split), as it has been trained on a part of this dataset (E1 training split). However, for this module to be used as part of an automatic thumbnail selection pipeline which is generalizable across different datasets (and correspondingly, different soccer leagues), we would like to see if it performs adequately for different datasets as well. For this purpose, we use the unseen Allsvenskan dataset (A1 test split). The results are displayed in the second part of Table 4.1. On the Allsvenskan dataset, the model is able to detect fewer than half of the logos. The images not containing a logo are all predicted correctly, irrespective of the threshold value. The accuracy of the model for the Allsvenskan set is not as high as for the Eliteserien dataset, where it was 0.99. However, it did not predict any false positives. We observe that a change in the threshold value from 0.5 to 0.1 improves performance for the Allsvenskan dataset as well. The precision is 100% for both cases, but the recall is improved for 0.1 as the threshold value. As we are interested in improving the recall, we further test with 0.05 as for the Eliteserien dataset, and see that there actually is an improvement when the threshold is lowered further. However, since this dataset is smaller than the Eliteserien dataset, we have lower confidence in this conclusion and

[1]The threshold value is described further in the Section 3.1.2.

[2]Conservative in context of the pipeline's behavior, means it is passive in terms of dropping images in its process.

resolve to use a threshold value of 0.1 in the final pipeline.

To test the generalizability of the model further, we test it on SoccerNet dataset as well (S1 test split). The results are displayed in the third part of Table 4.1. On the SoccerNet dataset, it is detecting very few of the logos, looking at the recall score. Adjusting the threshold value to a lower value, improves the accuracy slightly. The accuracy for SoccerNet is even worse than it is for Allsvenskan. The model does achieves its highest accuracy score when the threshold value is adjusted to 0.05.

The performance of the logo detection model can vary on general use. Adjusting the threshold value to 0.1, does give higher accuracy compared to 0.5 for all the tested sets. Since this model is trained on Eliteserien, it would also be reasonable to make the model perform well on the set it is trained on. We resolve to use a threshold value of 0.1 in the pipeline.

Dataset	Threshold	Accuracy	Precision	Recall	F1 score	MCC
Eliteserien (E1)	0.50	0.994	0.986	0.969	0.977	0.974
	0.10	0.995	0.969	0.996	0.982	0.980
	0.05	0.993	0.953	0.996	0.974	0.970
Allsvenskan (A1)	0.50	0.717	1.000	0.261	0.414	0.423
	0.10	0.761	1.000	0.377	0.547	0.521
	0.05	0.794	1.000	0.464	0.634	0.590
SoccerNet (S1)	0.50	0.570	0.717	0.018	0.036	0.060
	0.10	0.571	0.700	0.025	0.049	0.067
	0.05	0.573	0.697	0.031	0.059	0.074

Table 4.1: Performance of the logo detection module using the model by Surma [62], on the test splits from the Eliteserien (E1), Allsvenskan (A1) and SoccerNet (S1) datasets containing 1611, 180 and 16106 images, respectively.

4.3 Close-up Detection Performance

The close-up detection module is trained on the E2 dataset consisting of "close-up" and "no close-up" classes. The binary image classification by Surma [62] originally yields an accuracy of 0.82 on the test split when the probability threshold is set to 0.5. For the close-up detection task, we would like to prioritize precision over recall, because there is a relatively high chance that false positives can end up being the final thumbnail. Increasing the precision will prevent false positives from being prioritized. We test with different threshold values to observe the changes in the performance of the close-up detection module, the results are displayed in Table 4.2. Based on the trade-off between precision and accuracy, we resolve to use a threshold of 0.75 in the final pipeline.

Dataset	Threshold	Accuracy	Precision	Recall	F1 score	MCC
Eliteserien (E2)	0.8	0.761	1.000	0.448	0.619	0.562
	0.75	0.851	0.952	0.690	0.780	0.708
	0.7	0.881	0.920	0.793	0.852	0.759
	0.6	0.851	0.771	0.931	0.844	0.715
	0.5	0.821	0.730	0.931	0.818	0.665

Table 4.2: Performance of the close-up detection module using the model by Surma [62], on the test split from the Eliteserien (E2) dataset containing 67 images.

4.4 Face Detection Performance

The face detection module can be run with different face detection models. In this work, we consider Dlib [31], MTCNN [81], and Haar cascade[74]. According to Agarwal [1], Haar cascade is faster but less accurate, where Dlib is slower but more accurate. To evaluate the performance of the module, we use precision achieved on the E3 dataset, using the original versions of images ("Full" scale) and 50% down-scaled[3] versions ("Half" scale). The metrics we use to evaluate the performance of the different face detection models is precision, which is the number of correct face detections (true positives) divided by the total of face detections (true positives and false positives). There can be several face detections for each image. Table 4.3 presents the results in terms of precision and processing time per image. Each detection is assessed manually and considered if it is correct (true positive) or false (false positive), which gives the necessary numbers to calculate the precision.

To calculate the exact recall score, one need to have the number of false negatives. That would be faces that should be detected, that did not get detected by the face detection model. Every detection is evaluated manually for calculating the performance metrics. Evaluating false positives can be challenging and time consuming, as one has to consider which faces in the image set should be classified as a relevant element for the face detection test. One has to set some rules and for example consider how small a face can be in amount of pixels, for it to be excluded as a relevant element. Instead of counting the amount of false negatives manually, it is possible to compare the amount of correct face detections (true positives) a model achieves compared to the other models. This number indicates its recall relatively to the other models. Following the rule of increase in true positives, increases recall.

To calculate the precision we need to assess all positives (all face detections). Reviewing which are true or false can be challenging, but most of the detections are clear whether they are detecting a face or not. The numbers will still give a good indication of what the precision of the model is even though it is not as accurate as it could be through our calculations.

[3]In this context, down-scaling refers to the reduction in the resolution of (and correspondingly, number of pixels in) the images, and not the encoding quality.

Model	**Image Scale**	**Precision (Close-up)** (TP/TP+FP)	**Precision (Audience)** (TP/TP+FP)	**Time per Image**
Dlib	Full	0.96 (55/57)	0.99 (152/153)	0.63*s*
	Half	0.98 (53/54)	0.50 (1/2)	0.39*s*
MTCNN	Full	0.85 (71/84)	0.94 (321/343)	1.28*s*
	Half	0.91 (73/80)	0.93 (139/150)	0.89*s*
Haar	Full	0.61 (61/100)	0.91 (303/332)	0.20*s*
	Half	0.74 (58/78)	0.94 (34/36)	0.06*s*

Table 4.3: Performance of the face detection module using different models (Dlib, MTCNN, and Haar cascade) and different image scales (full-scaled 960×540 and half-scaled 480×270), on the test splits from the "Close-up" and "Audience" datasets. The precision columns contains the exact number of true positives (TP) and false positives (FP).

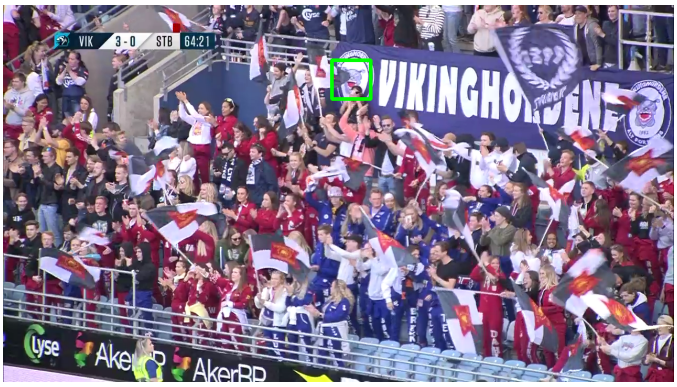
Dlib: From the "Close-up" set of 89 images, the Dlib model detected 55 faces in total when the images were full-scaled. The Dlib model had 53 face detections on the same images when half-scaled. There were a few false positive face detections on the "Close-up" set, where 2 false detections were made on full-scaled and 1 on half-scaled. Both false detections were on a face-shaped skeleton logo shown in Figure 4.2a. On the "Audience" set, the Dlib had a total of 153 detections on full-scaled and only 2 on half-scaled. A false positive face detection appeared here as well that was a logo with a face of a dog in Figure 4.2b.

The trend for the Dlib when doing face detection on half-scaled images was that it could not detect smaller faces. Under our inspection, the Dlib has yet to detect a face smaller than 52x52 pixels. When half-scaling an image, even more faces gets smaller than 52x52 pixels. This makes the Dlib overlook all the smaller faces. Even though the Dlib detected less faces in total for half-scaled than full-scaled. Dlib detected some faces it overlooked on full-scaled. The relative face size to the image scale was bigger for the detections on the half-scaled image set than on the full-scaled set. The Dlib model has also yet to have a false detection on a figure that is not face-shaped. The dog and the skeleton has shapes that are similar to a human face.

MTCNN: Overall, MTCNN has higher recall than the other models for all classes on the dataset. The precision on the "Close-up" dataset is better than Haar cascade, but not as good as Dlib. The model is the best at detecting smaller faces, as it has the highest recall on the "Audience" set. It is also able to detect faces from the side (at a 90 degree angle). 8% of the correct face detections, on both full-scaled and half-scaled, were faces from the side that the other models face detection models could not detect. However, the mean processing time per image for MTCNN is 0.89*s* on half-scaled images and 1.28*s* on full-scaled images. On half-scaled images it is about 15 times slower than Haar cascade and 2 times slower than Dlib.

Haar cascade: On the "Audience" set, Haar cascade has comparable precision to the other models, and higher recall than Dlib. However, it

(a) Detection on a face-shaped skeleton logo in the top left, marked with a green square.



(b) Detection on dog face logo in the top center, marked with a green square.

Figure 4.2: False positive face detections by the Dlib [31] model.

has significantly lower precision on the "Close-up" dataset. It detects more faces than Dlib, but not as much as MTCNN. The model is prone to false positives: on full-scaled images, 39% of the detections are false.

Final pipeline: After evaluating different models, we try to make a decision for the model to use in our final pipeline. In the context of automatic thumbnail selection for soccer video clips, we prioritize high precision and speed. The reason we prioritize precision over recall for the face detection task is because faces appear often and redundantly in soccer videos, so it is not essential that our model manages to detect *all* faces. However, it is essential that the model be certain *when* it does detect a face, because there is a higher probability that this image will be selected as the final thumbnail, according to the prioritization rules shown in Figure 3.1. Regarding speed, we see that a clear benefit of down-scaling is the reduction in processing time. For 50% down-scaling, up to 0.2s can be saved per image for Dlib. For instance, if the pipeline is running Dlib face detection model on 7 frames from a single video clip, 50% down-scaling decreases the time required by the face detection module from 4.4s to 2.7s.

MTCNN has the highest recall and the second highest precision on the "Close-up" set. It can also detect smaller faces, and faces from a side angle

unlike the other models. However, MTCNN uses significantly more time than the other models, and the absolute time it takes for going through, e.g., 7 images per video clip is not feasible for our context. Haar cascade has performed well on detecting small faces, and is the fastest model (can process a video 7 times faster than the Dlib for instance). However, it tends to have many false positives. Dlib has the highest precision on the "Close-up" set and it is faster while processing down-scaled images. The disadvantage is that it cannot detect smaller faces, but we consider the ability to detect bigger faces to be more important. This is discussed further in Section 6.1

We therefore resolve to use the Dlib model as the default model in the face detection module in our pipeline, along with 50% down-scaling of the video frames. The MTCNN model is added as a configurable (non-default) option. As the Haar cascade is very fast compared to Dlib and MTCNN, it is also added as a configurable option. If the automatic thumbnail selector is intended to process a large number of video clips in near-real-time, this model might be more suitable, as it allows for prioritizing speed over precision.

4.5 Image Quality Analysis

The last step in our pipeline comprises image quality analysis on the thumbnail candidates, in the order that the images appear after priority assignment in the previous step, followed by filtering and final thumbnail selection. We run the model by Ocampo [46] for image quality prediction, which is based on model proposed by Jongyoo [30]. The model outputs a BRISQUE score, where the lower the score, the better the predicted quality of the image. The image quality predictor is tested on 65 frames from a single video clip from the Eliteserien dataset.

Looking at the Figure 4.3 a set of frames has been tested on the image quality prediction model. The model outputs a BRISQUE score and the lower the score number is, the better quality is predicted on the image. Figure 4.3e, 4.3f and 4.3j receive a good score and have little distortion. These are all images that qualify quality-wise without thinking of what objects the image is containing (i.e. persons, ball, soccer pitch). All the images receiving higher score than 45, are seemingly more blurry/distorted than the rest that is scoring under.

Another inspection that is been made with the model made by Ocampo [46] is that images with little color variance receives high output score. On a set of images with logo graphics from E1 set, the BRISQUE score tends to be higher when the logo graphic covers more of the screen. In Figure 4.4 all the images receiving score above 50, are logo graphics[4]. Our own proposed pipeline can take advantage of the BRISQUE score if the logo detection model does not manage to detect a logo. A high BRISQUE score does not necessarily mean that the image contains logo graphic, but it is a possibility.

[4]The scores in Figure 4.4 can not be compared to the scores in Figure 4.3, as the resolution is different on each set.

Regardless we want images with a low BRISQUE score as we want to be more confident that the image is well suited for being a good thumbnail.

Influence of compression: While testing the model, we also investigate if the model predicts lower quality when the input image has been saved with lower quality using the imwrite function from cv2. With the imwrite function, it is possible to choose the percentage of how much data from the image is going to be preserved. From Table 4.4, it is possible to see the accuracy of the model for predicting lower quality on actual lower quality images. For the test set of 65 images, it did not achieve 95% accuracy before the chosen image quality was about 50%. Another relevant aspect is how fast the image quality predictor model processes the images when they contain less data. The model by Ocampo has a tendency to consume a lot of time for processing a single image. From Table 4.4 it is possible to see the mean processing time per image with respect to preserved quality. Preserving less data on an image does not seem to make the image faster to process for the image quality predictor model, even though the storage size is reduced to a great extent.

Influence of down-scaling: What seems to reduce the processing time is reducing the scale of the image. The intermediary conclusion is that processing time is dependent on the amount of pixels and not necessarily the amount of data the image contains. We have tested what the image quality predictor [46] predicts for different degrees of down-scaling (Table 4.5). This is to know how the model behaves when it receives a down-scaled image, as we want the processing of an image to be fast in our pipeline. However, even though the processing might be faster, we want to make sure that the model is able to predict on down-scaled images with similar performance compared to the images in original scale. The output thumbnail of our pipeline can be the original or down-scaled version of the image. In the latter case, the BRISQUE score used in the final step will be referring to the down-scaled version of the image.

Assessing the image quality predictor, many of the images receiving a high score (low quality) are often images that are blurry, noisy and even with generic graphics. Ranking the images with lower scores does not seem necessary as an image with 10 in score can look as good as an image with 30 in score to the human eye. The difference is not significant enough as we want to take the runtime into account as well. Eliminating the undesired images altogether seems to be the most helpful course of action with this model.

Preserved Quality	Accuracy	Mean Processing Time	Mean Size
100%	NA	2.95s	128 KB
70%	0.72	2.82s	51 KB
60%	0.89	2.72s	42 KB
50%	0.95	2.65s	35 KB
40%	0.97	2.65s	32 KB
5%	1.00	2.28s	13 KB

Table 4.4: Influence of compression: Accuracy of the Ocampo model [46] in predicting if the image has lower quality, along with mean processing time per image and mean storage size per image, with respect to preserved image quality.

Image Scale	Accuracy	Mean Processing Time	Mean Size
960×540	NA	2.95s	128 KB
800×450	0.88	1.63s	42 KB
600×338	0.86	1.16s	27 KB
500×281	0.84	0.73s	21 KB
300×169	0.92	0.34s	9 KB
100×56	0.85	0.06s	1.9 KB

Table 4.5: Influence of down-scaling: Accuracy of the Ocampo model [46] in predicting if the down-scaled image has better quality than the original image with 960×540 resolution, along with mean processing time per image and mean storage size per image, with respect to image scale.

(a) BRISQUE score: 49.36

(b) BRISQUE score: 48.98

(c) BRISQUE score: 51.12

(d) BRISQUE score: 42.96

(e) BRISQUE score: 39.17

(f) BRISQUE score: 33.31

(g) BRISQUE score: 52.06

(h) BRISQUE score: 43.80

(i) BRISQUE score: 45.09

(j) BRISQUE score: 32.38

Figure 4.3: BRISQUE score for several sample frames from a single video clip, using the image quality predictor by Ocampo [46]. The video clip is from the Eliteserien dataset and the resolution is down-scaled to 480x270 pixels.

Figure 4.4: Images from E1 test set tested on the image quality prediction model made by Ocampo [46]. All images with red frame received a BRISQUE score higher than 60. The images with orange frame had a BRISQUE score between 50 and 60. The BRISQUE score is dependent on the image resolution and the resolution on these images when testing were 200x120 pixels.

4.6 User Study

In this section, we evaluate the quality of the thumbnails selected by our own pipeline in relation to manual and static selected thumbnails, using quantitative and qualitative subjective data gathered from our user survey. First we go through the background information of the participants and look at the distribution of each information. Then the overall results from the survey will be inspected. At last the results of the different groupings of the participants will be inspected. This user study is between the first iteration and second iteration so the data can be used to find ways to improve the thumbnail selection framework further. Having a

user study before the framework is finalized can give both information on how to find improvements and also give measurements on how good the proposed thumbnails already is. Participants of the survey include family, friends and colleagues. It is also published publicly for participation. The information given to the participants are nothing other than what is given in the survey itself.

The survey is structured as described in Section 3.4. Table 1 in the appendix of [26] (see Section A.1), presents the list of registration form fields and feedback questions used in the study. We use the HOST-ATS framework to run 2 user study iterations, the first including 9 cases (Figure 1 in the appendix of [26], Section A.1), and the second including 13 cases (Figure 2 in the appendix of [26], Section A.1). In each case, participants compare 2 alternative thumbnail selection methods. Table 4.6, presents the investigation aspects relevant to the alternative thumbnails for each case. The selection of cases allows us to compare among different thumbnail selection methods, as well as to gain deeper insights into viewer expectations from thumbnails, which will potentially help us improve our framework (e.g., rules and assumptions described in Section 3.1 and Table 3.1) as future work.

Case	**Investigation**	**Method**		
		H	M	S
70396	Players celebrating: close-up vs. medium distance	✓	✓	
75861	Player close-up vs. player celebration	✓	✓	
76465	Players celebrating: scorer visible vs. not visible	✓	✓	
80540	Player close-up vs. player celebration	✓	✓	
73375	Player close-up vs. players celebrating	✓		✓
77648	Players celebrating: scorer visible vs. not visible	✓		✓
77651	Players celebrating vs. audience	✓		✓
79994	Players celebrating vs. goal shot	✓		✓
80077	Players celebrating vs. attack action	✓		✓
75143	Player close-up vs. players celebrating	✓		✓
76358	Player close-up vs. attack action	✓		✓
76394	Long distance vs. close-up shot	✓		✓
76407	Player close-up vs. player celebration	✓		✓
76422	Player close-up vs. long distance shot	✓		✓
76821	Player close-up vs. audience	✓		✓
78438	Players celebrating vs. audience	✓		✓
78888	Player close-up vs. players celebrating	✓		✓
79285	Players celebrating vs. long distance shot	✓		✓
79588	Player close-up: players from different teams	✓		✓
79606	Player close-up: front view vs. back view	✓		✓
79789	Player close-up vs. players celebrating	✓		✓
80136	Long distance shot vs. audience	✓		✓

Table 4.6: Investigation aspects for cases in the user study. Evaluated thumbnail selection alternatives are HOST-ATS (H), manual selection (M), and static selection (S).

4.6.1 General Information About the Participants

The number of participants were in total 43. 23 participants were on the first iteration and 20 were on the second iteration.

Age Distribution

In the Figure 4.5, the age distribution is presented. The majority of the participants was between 18-40 in age. The other age groups are under-represented and there are no participants below age 18 and no above 59.

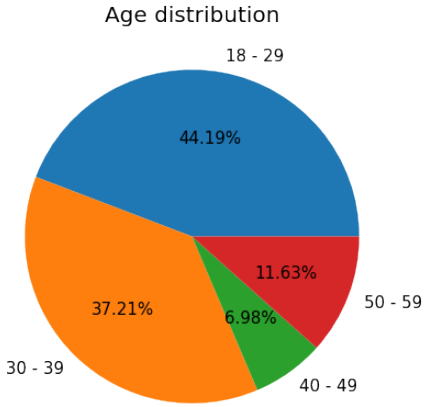


Figure 4.5: Age distribution of the participants of the user study.

Gender Distribution

We can see that there are more males than females that has participated the survey from Figure 4.6.

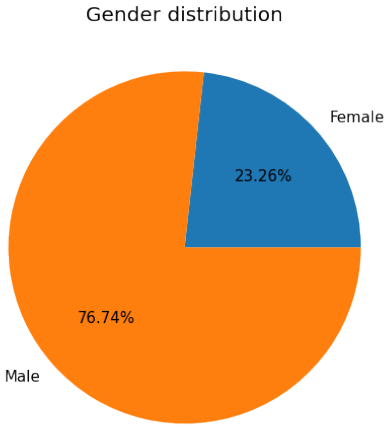


Figure 4.6: Gender distribution of the participants of the user study.

Sports Fandom Distribution

In the Figure 4.7, the distribution of sports fandom is displayed. About half of the participants were sports fans and the rest were not. It can be good to have a variation on the participants interests. The people that are sports fans can give reliable answers as they have much knowledge about the sport and their meanings about thumbnails can be important as they could be the persons that are consuming sports content the most and seeing the thumbnails often. The people that are not sports fans can also have relevant opinions for the survey. There are possibilities that there are some cases they are watching a sport event even if they are not sports fan. Then it can be important to make the content suited for non-fans as well.

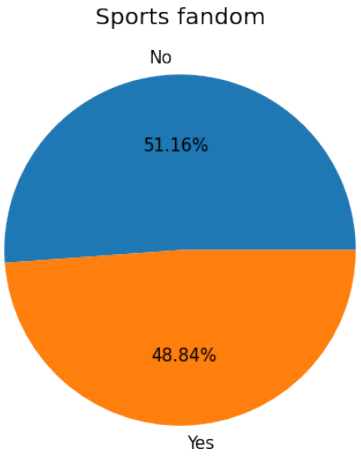


Figure 4.7: Sports fandom distribution of the participants of the user study.

Video Editing Experience Distribution

In Figure 4.8, the video editing experience of the participants is displayed. There is only one with professional experience, but there is a fair share of participants having experience of some degree. Having a variation on the video editing experience as well can be useful information. People with experience have probably more thoughts on what a thumbnail should be, but getting answers from people with no experience is important as well as their thoughts might be more instinctive. It is more important that the thumbnail catches the interest of the majority, which can be assumed to normally not be people experienced with video editing. Getting answers from consumers and creators can make it possible to see if the different groups correspond in their answers. A problem with comparing the different groups is in this case when there is only one representative for the professional video editors. This is too little to represent a whole group.

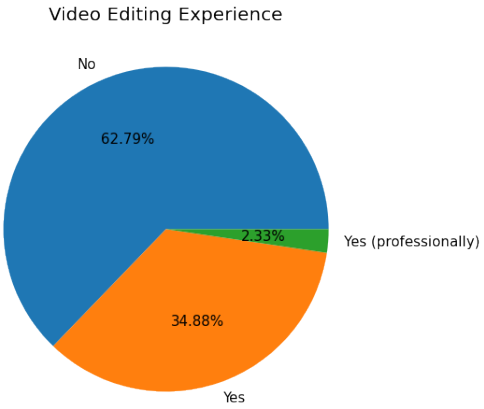


Figure 4.8: Video editing experience of the participants of the user study.

4.6.2 User Study Findings

Figure 4.9 and 4.10 presents the overall results from the iterations of our user study, in terms of the number of votes for different options per case, where "HOST-ATS" indicates the thumbnail selection by our framework, "Static" indicates the thumbnail selection method wherein a frame from the video clip at a fixed timestamp (e.g., X seconds after playback start) is assigned as the thumbnail, and "Manual" indicates manual thumbnail selection undertaken by human operators.

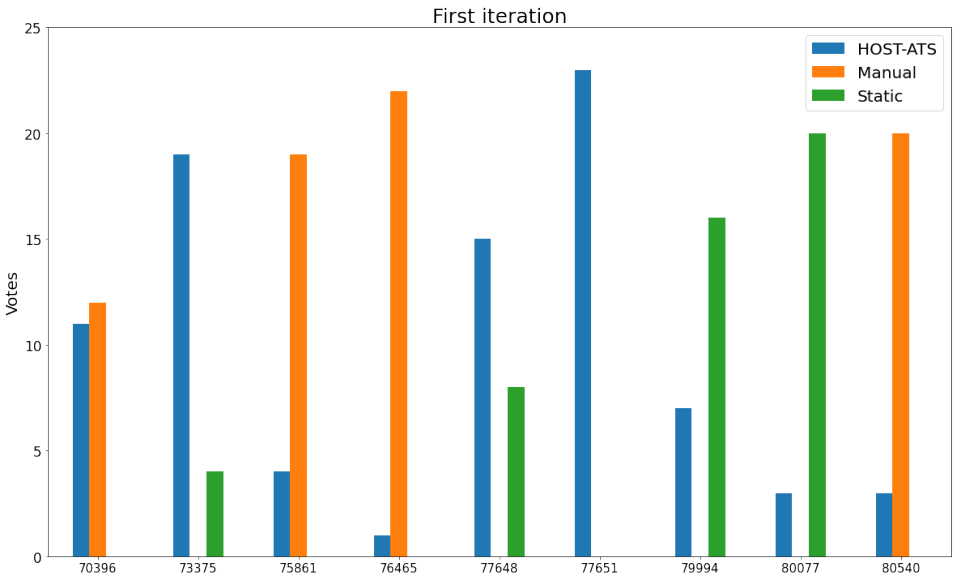


Figure 4.9: Case answers for the first iteration of the user study.

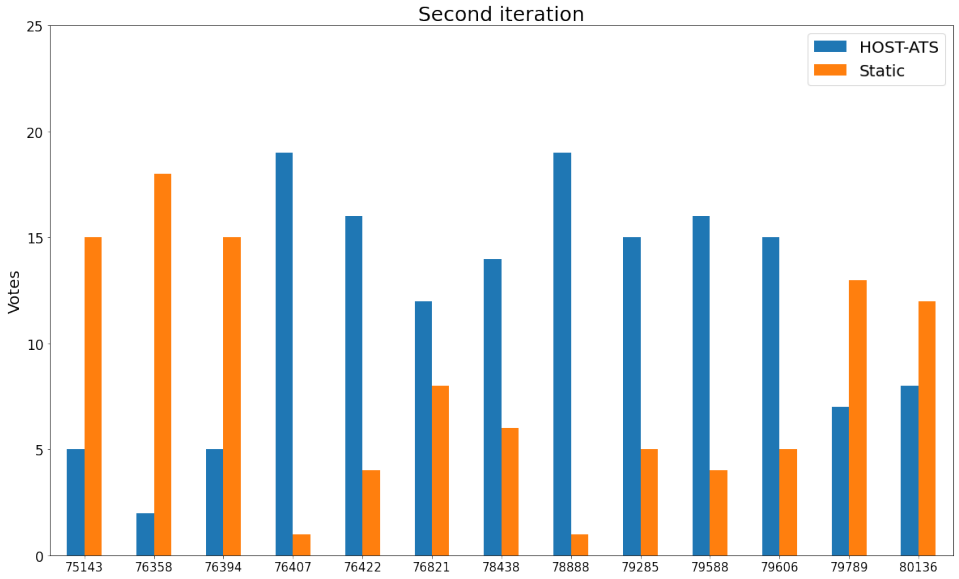


Figure 4.10: Case answers for the second iteration of the user study.

Manual selection is preferred over HOST-ATS (4 out of 4 pairwise comparisons), i.e., meaning that if used, a human operator selects better thumbnails than our automated system. For case 70396, it is a close call, where the medium distance shot of player celebration is preferred over the close-up shot of player celebration. For case 75861, manual selection showing multiple players is preferred by a larger margin against a single player close-up. For case 76465, manual selection where the scorer is visible in front view is preferred by a larger margin against a celebration where the scorer is only visible in back view. For case 80540, manual selection showing multiple players in celebration from the back view is preferred by a larger margin against a close-up view of players where a face is visible more distinctly but there is no celebration action. These are very valuable insights for us to improve our thumbnail selection rules (Table 3.1) with more details regarding the joint evaluation of face detection and close-up detection results, as well as including context-aware priorities with respect to celebrations and action content (e.g., crowded celebration scene preferred over frontal or side view of fewer players with less action content).

HOST-ATS outperforms static selection in most cases (11 out of 18 pairwise comparisons). For cases 73375, 77648, 77651, 76407, 76422, 76821, 78438, 78888, 79285, 79588, and 79606, we see that the rules enforcing higher image quality, preference of close-up shots to long distance shots, and frontal face view are in line with the viewer preferences. Cases where the static selection is preferred include the following: 79994 (clear view of the goal shot, despite being medium/long distance preferred over blurry image of players celebrating), 80077 (attack scene preferred over celebration scene with partial obstruction in image), 75143 (goal scorer's single celebration, despite blur in image, preferred over multiple-player celebration where scorer is not visible), 76358 (attack action from medium

distance preferred over close-up of player), 76394 (long distance shot of field preferred over close-up shot with partial obstruction in image), 79789 (goal scorer from back view preferred over multiple-player celebration with slight blur), and 80136 (long distance shot of the field preferred over blurry audience celebration). These cases show that additional rule adjustments are needed to take image blurriness, partial obstructions, action content, and celebration context (e.g., goal scorer) into account.

Viewers consider high image quality, player faces, and action content as the most important aspects of a thumbnail, followed by close-ups, cheering, and the absence of logo transitions (Figure 4.11). These confirm our initial thumbnail selection rules, and give further insights regarding possible additional rules (e.g., detection of action content and cheering context).

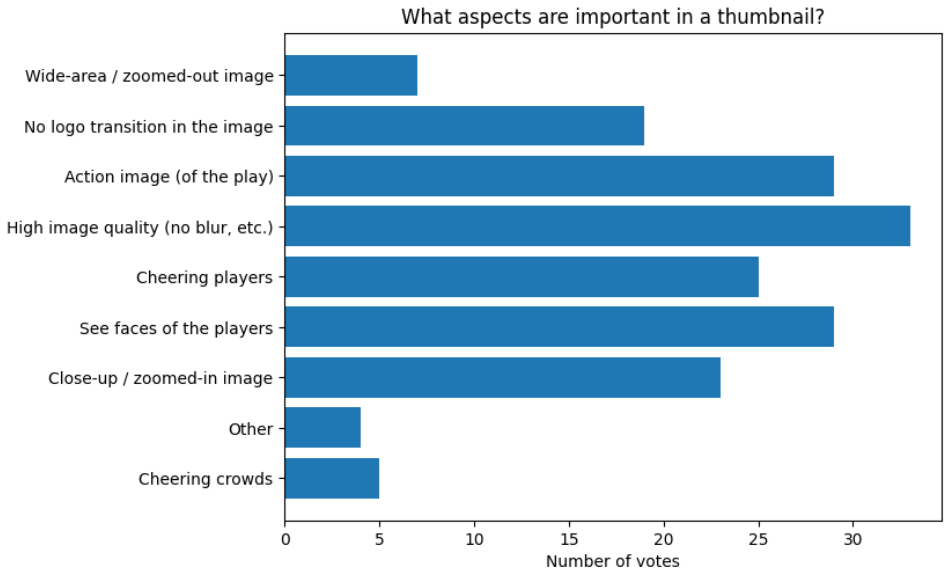


Figure 4.11: The distribution of votes the users gave on the different aspects they wanted a soccer thumbnail to have.

Additional Comments From the Users

14 of the participants did also give a comment in own words what they wanted a thumbnail to be. Characteristics mentioned in the checkbox form, were also rementioned (high quality, action, cheering crowd, cheering players, faces and close-up). This is a summarization of the feedback from the participants:

- Many or all of the thumbnail alternatives presented were bad.
- The images showing action should be close to the goal or the moment the goal occurs.
- Opposing players should not appear on the thumbnail.
- Opposing players could appear if it is relevant.

- The goalscoring player should be appearing on the thumbnail.
- Emotions should be expressed in the thumbnail.
- What kind of goal that has been scored should be conveyed in the thumbnail. (Assuming this could be conveying if the goal was scored from penalty, corner, free kick or open play.)

Vote Distribution by Age Group

The distribution of votes on HOST-ATS and manual thumbnails are seen in Figure 4.12, 4.13 and 4.14. All age groups favors the same thumbnail type in all charts, except for the age group '40-49' in Figure 4.12, which gives the equal amount of votes to manual and HOST-ATS thumbnails. In the first iteration it is only 1 person in the age groups '40-49' and '50-59', which can not be a full representation of these age groups.

In first iteration, the age group '18-29' favors the manual thumbnails more than the '30-39' age group in Figure 4.12. The same age groups also have the same relative vote distribution on the HOST-ATS and static thumbnails in Figure 4.13.

The results from second iteration, seen in Figure 4.14, tell us that the age group '30-39' is favoring the HOST-ATS relatively more than the other groups.

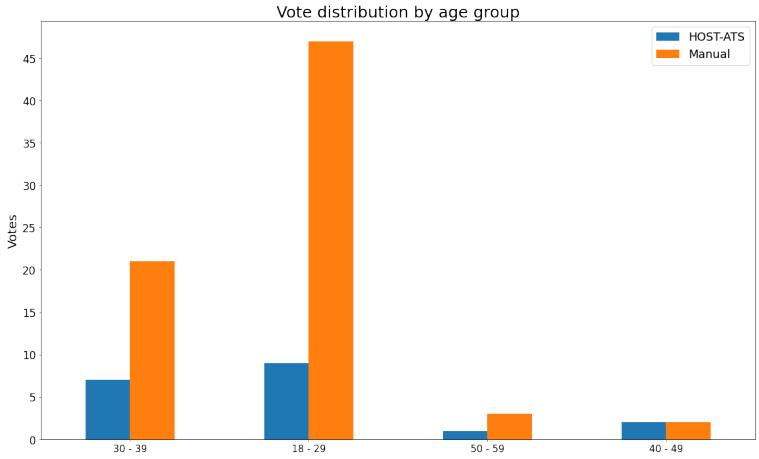


Figure 4.12: The vote distribution by age group on manual and HOST-ATS thumbnails in first iteration.

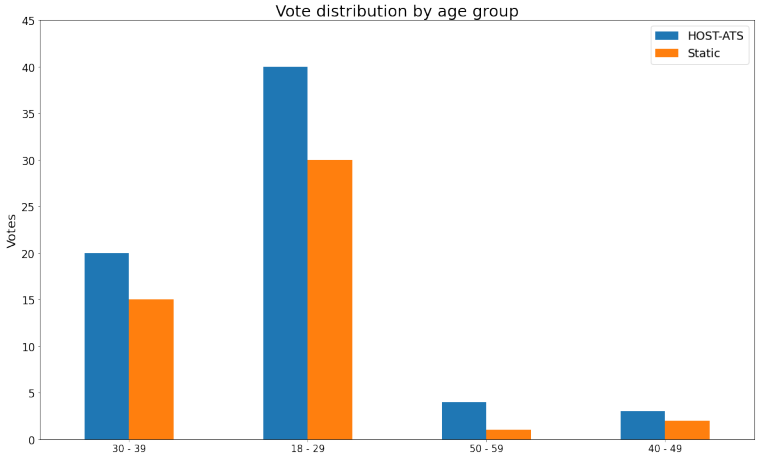


Figure 4.13: The vote distribution by age group on static and HOST-ATS thumbnails in first iteration.

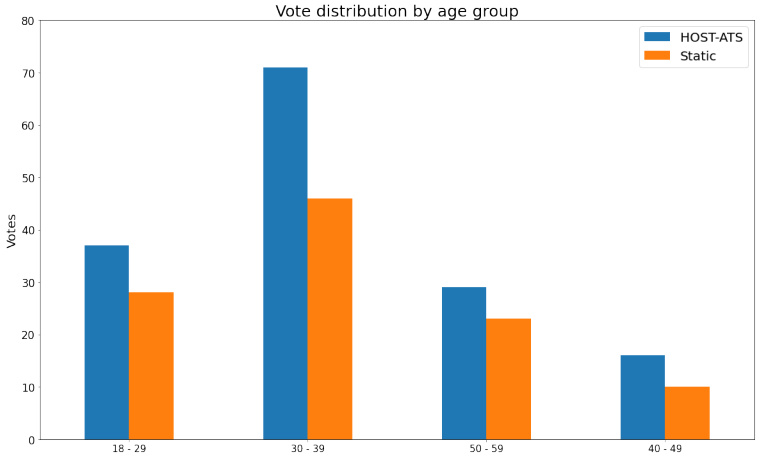


Figure 4.14: The vote distribution by age group on static and HOST-ATS thumbnails in second iteration.

Vote distribution by gender

The distribution of votes by gender are very similar to each other. They are also favoring the same thumbnail type in all charts seen in Figure 4.15, 4.16 and 4.17

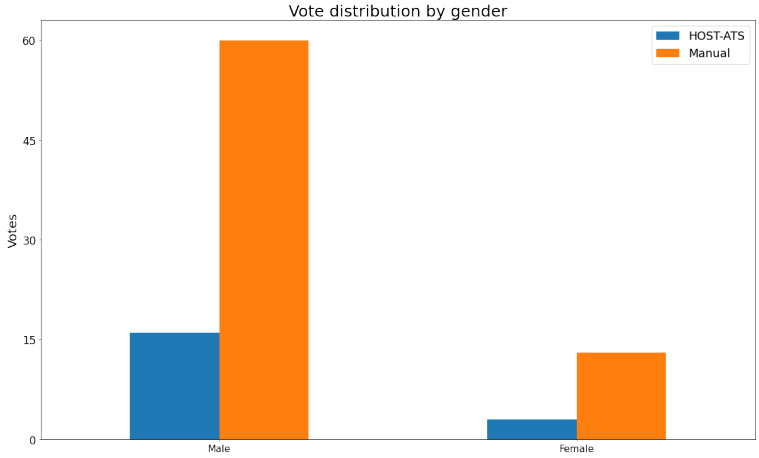


Figure 4.15: The vote distribution by gender on manual and HOST-ATS thumbnails in first iteration.

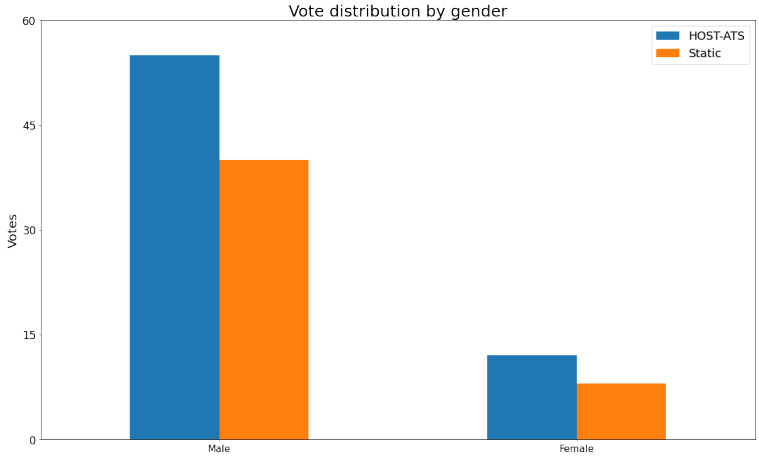


Figure 4.16: The vote distribution by gender on static and HOST-ATS thumbnails in first iteration.

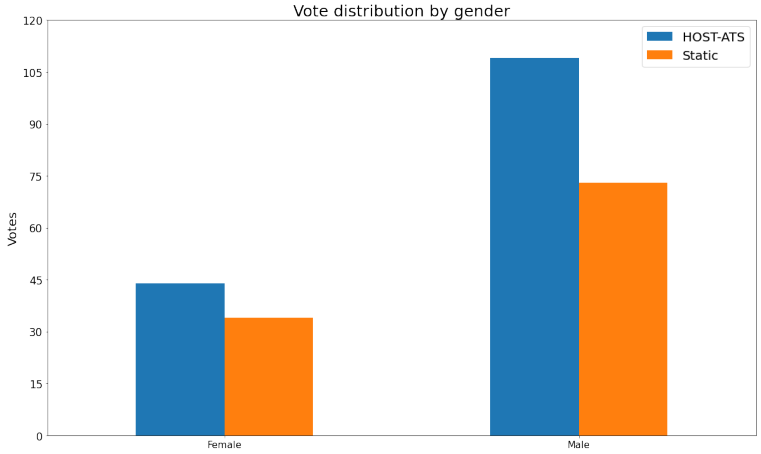


Figure 4.17: The vote distribution by gender on static and HOST-ATS thumbnails in second iteration.

Vote Distribution by Sports Fandom

The vote distribution by sports fandom is seen in Figure 4.18, 4.19 and 4.20. In every figure the same thumbnail type is favored for both groups. Beyond inspection, the group that are fan of sports gave relatively more votes to the static and manual thumbnails than the group that were not fans of sports. Meaning that the HOST-ATS was not as clear favorite for the sports fans as it was for the non-sports fans.

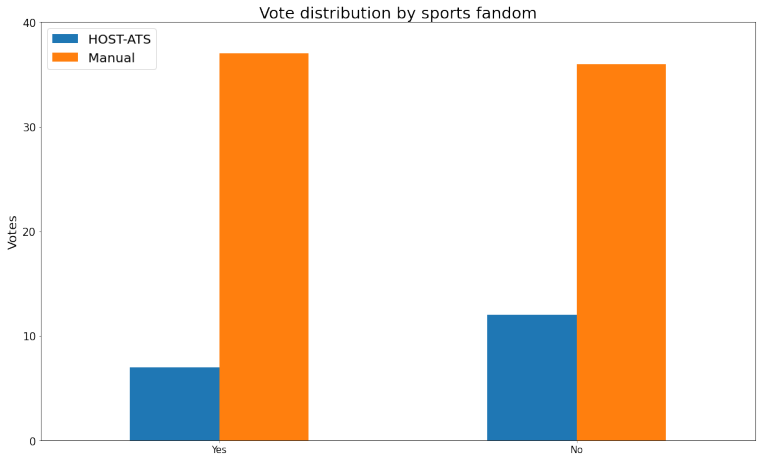


Figure 4.18: The vote distribution by sports fandom on manual and HOST-ATS thumbnails in first iteration.

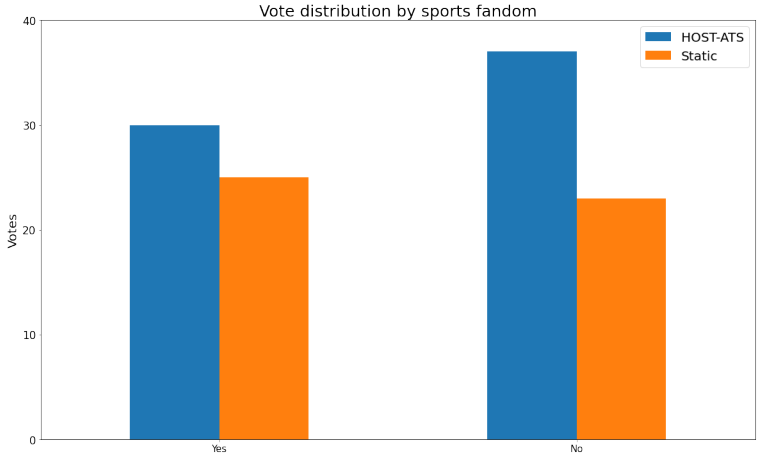


Figure 4.19: The vote distribution by sports fandom on static and HOST-ATS thumbnails in first iteration.

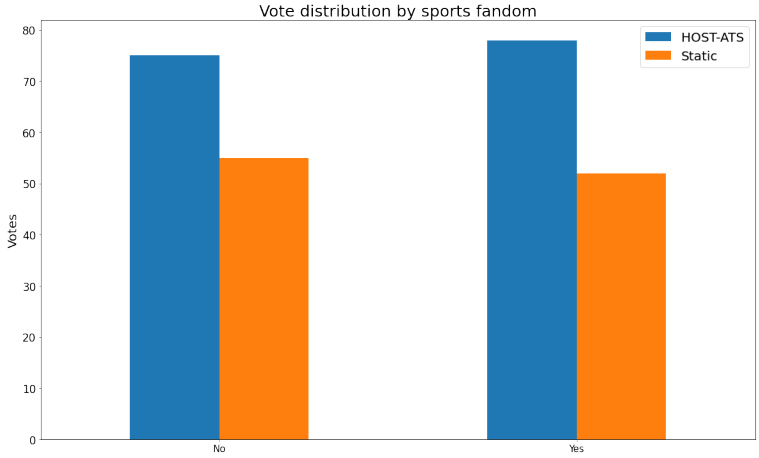


Figure 4.20: The vote distribution by sports fandom on static and HOST-ATS thumbnails in second iteration.

Vote Distribution by Video Editing Experience

The vote distribution by video editing experience is shown in Figure 4.21, 4.22 and 4.23. The difference on the vote distribution on the different groups are not significant. The people with no video editing experience have relatively more votes on the manual thumbnails than the people with experience. In first iteration the non-experienced people are also giving relatively more votes to the static thumbnail. On second iteration it is opposite, the experienced people are giving relatively more votes to the static than the non-experienced. Even though there only was one person with professional video editing experience, this person had exactly the same relative vote distribution as the people with non-professional video editing experience.

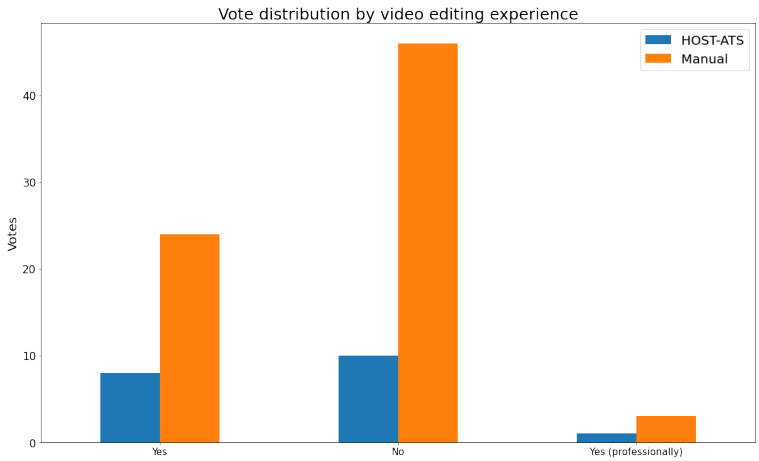


Figure 4.21: The vote distribution by video editing experience on manual and HOST-ATS thumbnails in first iteration.

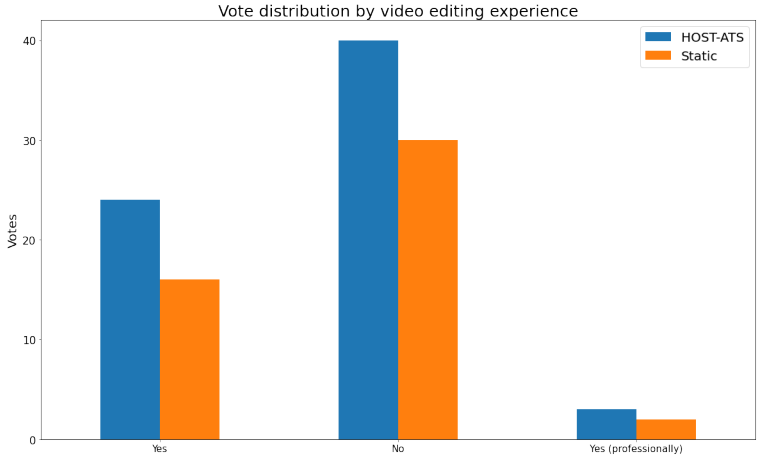


Figure 4.22: The vote distribution by video editing experience on static and HOST-ATS thumbnails in first iteration.

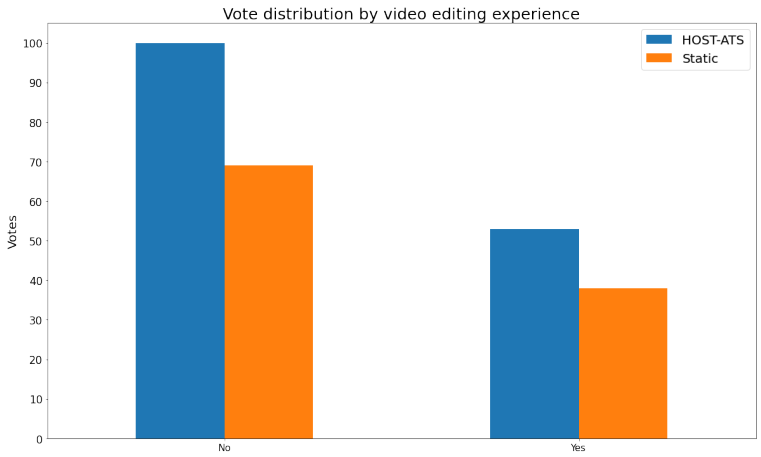


Figure 4.23: The vote distribution by video editing experience on static and HOST-ATS thumbnails in second iteration.

4.7 Lessons Learned and Possible Improvements

The results from the user study is promising, the HOST-ATS apparently chooses better thumbnails than static selected, but not as good as manual selected. We got several ideas from the user study on what kind of thumbnails the pipeline can focus on selecting. We also see that the modules of the pipeline also can be improved. In this context, there are a number of possible improvements which constitute potential future work directions for the second iteration (Chapter 5):

- **Model performance:** Different models can be used in the individual framework components (currently logo detection, close-up detection, face detection, and image quality prediction modules, possible to extend easily in the modular implementation according to the above points), and benchmarked, similar to what we have demonstrated in Section 4.4 for face detection.
- **Blur detection:** Even though the blur detection was not added in first iteration. We still want to inspect it and add it as a module for the second iteration.
- **Complexity analysis:** Evaluate the complexity of the framework with different metrics. Also test on full length soccer match videos, to give an idea of real-time performance of our framework.
- **Frame extraction:** The number of frames extracted from a video clip, can affect the thumbnail selection. The time usage of the pipeline increases, but it could be possible that the output thumbnail could be better.

4.8 Summary

In this chapter, we test all the components added to the pipeline for the first iteration. We also do a user study to evaluate the current pipeline for this iteration. The thumbnails selected by our framework seemed to be favored in front of static selected thumbnails, but not the manual selected ones. At last, we address some of our lessons learned and potential improvements for the next chapter (Chapter 5, as the second iteration of our framework).

Chapter 5

Experiments and Results (Second Iteration)

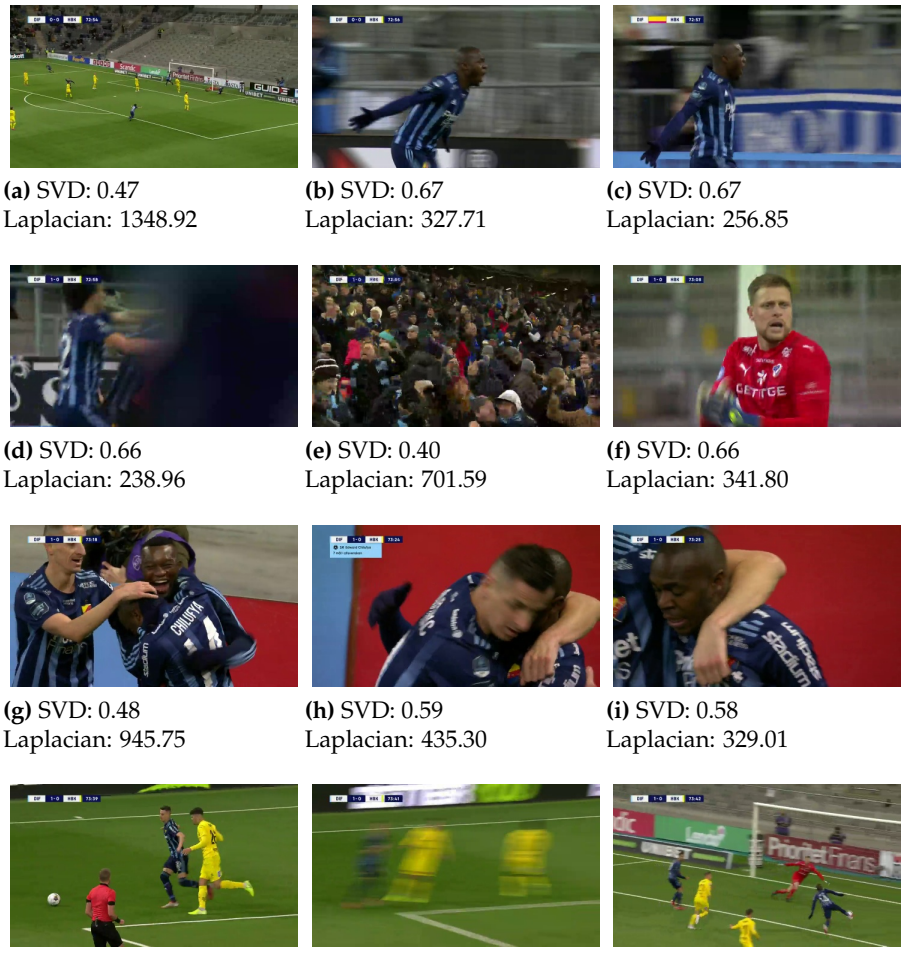
In Chapter 4 (first iteration), there were pointed out possible improvements in Section 4.7. These are worked on in this chapter. We improve as well as adding modules for this iteration. A dashboard for the framework is added as well. We evaluate the framework by complexity analysis and comparing it to first iteration and others work.

5.1 Updated Pipeline Design

The final thumbnail selection pipeline in this chapter is as presented in Chapter 3.

In first iteration (Chapter 4), the blur detection module was excluded. The reason for not adding it in first iteration, was that it did not have high priority and on a quick assessment it did not seem to improve the thumbnail selection pipeline. The discussion around the efficiency of the blur detection is made in Section 6.2. For second iteration the blur detection module is added with the option to choose between the Laplacian method from OpenCV library [5] and the SVD [23] method. Though the blur detection methods can have a tendency to detect high presence of blur on image that only has a blurry background and clear foreground, it is still optional to use it and the pipeline will presumably output higher quality images in general.

In Figure 5.1, it is possible to see the blur score from both SVD and Laplacian. The absolute value can not be compared, but the relative value on the different images in the figure is possible to compare. There are some correspondance on the relative values, but on some images they have a big difference on the relative blur score. Sorting the scores on the images in Figure 5.1 for both SVD and Laplacian gives a similar order on the images. One bigger difference to mention is that the image in Figure 5.1k is predicted the most blurry by SVD, but it has the median value for Laplacian.



(a) SVD: 0.47
Laplacian: 1348.92

(b) SVD: 0.67
Laplacian: 327.71

(c) SVD: 0.67
Laplacian: 256.85

(d) SVD: 0.66
Laplacian: 238.96

(e) SVD: 0.40
Laplacian: 701.59

(f) SVD: 0.66
Laplacian: 341.80

(g) SVD: 0.48
Laplacian: 945.75

(h) SVD: 0.59
Laplacian: 435.30

(i) SVD: 0.58
Laplacian: 329.01

(j) SVD: 0.54
Laplacian: 697.21

(k) SVD: 0.76
Laplacian: 399.28

(l) SVD: 0.51
Laplacian: 761.49

Figure 5.1: Blur detection ran on a set of frames from an Allsvenskan goal video. All frames that got a score above 0.60 are included and some other frames to show the models behavior.

One last inspection made, is estimating the time usage of both Laplacian and SVD. The times are received, running the blur detection methods on a set of images from the Allsvenskan dataset consisting of 51 samples with the scale of (480x270). The Laplacian method had an average time usage of 0.00097s and the SVD method had an average time usage of 0.07137s.

After some inspection we decide to have the Laplacian as the default blur detection method and the default threshold value, deciding which images to filter, being 800. The execution time is the main argument for choosing Laplacian.

5.2 Logo Detection Performance

The logo detection model by Surma [62] only got trained on E1 in first iteration (Chapter 4). The S1 dataset has more samples and more diversity

than E1 and could potentially be performing better on unseen logos than the model trained on E1. It will definitely perform better on logos from Premier League and the model can be added as an option on the logo detection module depending on which league the framework are supposed to select thumbnail from.

The test results of the SoccerNet model (logo detection model trained on S1) are displayed in Table 5.1. Comparing it to the test results for the Eliteserien model (logo detection model trained on E1), the SoccerNet model performed better on S1, but worse on E1 and A1. For the SoccerNet model it seems like it receives better accuracy when the probability threshold is 0.5, contrary to Eliteserien model which receives better accuracy on 0.1. It could be the case that the SoccerNet model was trained on significant more samples than the Eliteserien model.

Dataset	Threshold	Accuracy	Precision	Recall	F1 score	MCC
Eliteserien (E1)	0.50	0.839	0.400	0.323	0.357	0.269
	0.10	0.777	0.267	0.350	0.303	0.175
	0.05	0.753	0.240	0.363	0.289	0.152
Allsvenskan (A1)	0.50	0.594	0.167	0.014	0.027	-0.083
	0.10	0.594	0.333	0.058	0.099	-0.027
	0.05	0.600	0.421	0.116	0.182	0.027
SoccerNet (S1)	0.50	0.972	0.969	0.966	0.968	0.943
	0.10	0.953	0.915	0.984	0.948	0.907
	0.05	0.937	0.883	0.987	0.932	0.879

Table 5.1: Performance of the logo detection module using the model by Surma [62] trained on Soccernet, on the test splits from the Eliteserien (E1), Allsvenskan (A1) and SoccerNet (S1) datasets containing 1611, 180 and 16106 images, respectively.

5.3 Face Detection Performance

To improve the face detection module, the DNN face detector from OpenCV library [5] is added as an option to the face detection module. The results for the performance of the DNN is displayed in the last row of the Table 5.2[1]. The metrics of the DNN shows better precision, more detections and faster processing time than Dlib and MTCNN. It is slightly slower than DNN on half-scaled images. Inspecting each detection made of DNN, about 13 of them were faces from the side. This is more than any of the other models as well. After this inspection the DNN is default model for the face detection module.

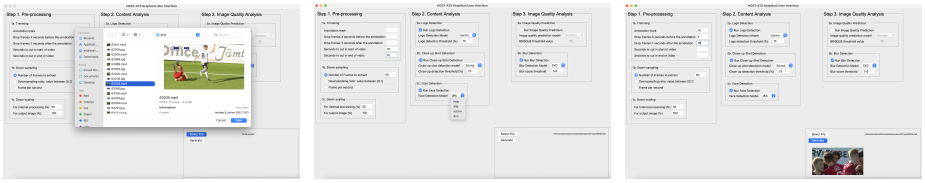
[1]Table 5.2 is an updated version of Table 4.3, where the DNN face detection model is added as well to the table.

Model	**Image Scale**	**Precision (Close-up)** (TP/TP+FP)	**Precision (Audience)** (TP/TP+FP)	**Time per Image**
Dlib	Full	0.96 (55/57)	0.99 (152/153)	0.63*s*
	Half	0.98 (53/54)	0.50 (1/2)	0.39*s*
MTCNN	Full	0.85 (71/84)	0.94 (321/343)	1.28*s*
	Half	0.91 (73/80)	0.93 (139/150)	0.89*s*
Haar	Full	0.61 (61/100)	0.91 (303/332)	0.20*s*
	Half	0.74 (58/78)	0.94 (34/36)	0.06*s*
DNN	Full	0.99 (92/93)	0.95 (19/20)	0.09*s*
	Half	0.99 (92/93)	0.95 (18/19)	0.06*s*

Table 5.2: Performance of the face detection module using different models (Dlib, MTCNN, Haar cascade and DNN) and different image scales (full-scaled 960×540 and half-scaled 480×270), on the test splits from the "Close-up" and "Audience" datasets. The precision columns contains the exact number of true positives (TP) and false positives (FP).

5.4 Dashboard

To make the framework more user friendly, a GUI was created [25]. The dashboard GUI consists of a window with all the interactions in one page. Figure 5.2 shows the page that the GUI displays. All configuration parameters in the pipeline are possible to modify from this page. None of the parameters in steps 1, 2 or 3, which are presented as subcategories in Figure 5.2, are mandatory. If no input is provided for a specific parameter, the default values of the pipeline will be run. The button "Generate" runs the framework and then the output image will appear under the button as seen in the Figure. In Figure 5.3, it is possible to see more detailed images of the interaction with the GUI.



(a) Input video clip selection **(b)** Model configuration **(c)** Output

Figure 5.3: HOST-ATS dashboard details.

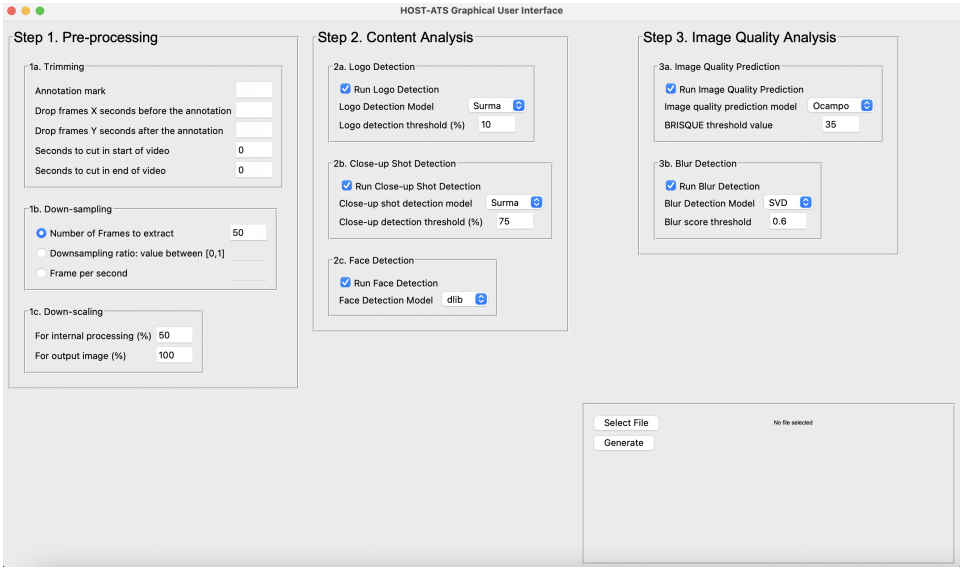


Figure 5.2: HOST-ATS dashboard.

5.5 Complexity Analysis

In order to evaluate the complexity of our framework, we refer to 3 metrics. Our **computational requirements** referring to hardware and software requirements for the full end-to-end execution of the framework were already discussed in Section 3.3. Then, the **model size** refers to the size of the ML models we use, either in terms of storage. Table 5.3 presents respective sizes of various models of the framework on disk (in terms of storage). Finally, the **execution time** refers to the duration it takes for certain components in the framework, or the framework itself, to perform their operations. For example, the time it takes for the pipeline to select a thumbnail after receiving a video as input is an aspect to consider when considering the efficiency of the pipeline. There has been made decisions during the implementation of the pipeline that has made it more time efficient, but less thorough in search for a good thumbnail. Using the image quality predictor BRISQUE [46], it takes about 1 second per image. If you want to receive an image score of all the selected frames from the video, it could take some time. It is ideal that the whole pipeline

does not use more than a few seconds when selecting a thumbnail for a video. Table 5.4 presents the average execution time per clip for each module in the framework, as well as the overall framework in an end-to-end fashion. Here, we use 37 clips from the Allsvenskan dataset, with an average duration of 77 seconds. For the face detection module, only the DNNs time usage is included in the table, as this is the preferred and almost the fastest face detection model in our pipeline. The logo detection module does not include any time usage on any alternative model, since the two alternative models (trained on Eliteserien and SoccerNet) have the same time usage. The pipeline's time usage on each video clip is the time usage on each clip individually. Running the pipeline on a folder of 37 clips at once will avoid loading of models and reduces the runtime for each clip with half a second.

We mentioned in Section 4.7, we wanted to test our runtime on full length videos. We tested the runtime on 45 minute long videos from the SoccerNet dataset and set an annotation mark at the last minute in the video. The mean duration time is 11.5s. This is about 7 seconds more than it is using on 2 minute video clips. What increases the time is that the pipeline has to iterate through each frame to the last minute of the video. If the annotation mark is in the first minute of a 45 minute long video (cutting 20 seconds before and 60 seconds after the annotation mark), it uses the same time as if it only was a 2 minute long video.

Module	**Model**	**Size**
Logo detection	Surma (Eliteserien 2019) [62]	10530KB
	Surma (SoccerNet) [62]	10530KB
Close-up detection	Surma [62]	10530KB
Face detection	Dlib [31]	7365KB
	MTCNN [81]	2256KB
	Haar [74]	930KB
	DNN [5]	10694KB
Image quality prediction	Ocampo [46]	147KB

Table 5.3: Complexity analysis: model size on disk.

Step 1: Pre-processing		**Step 2: Content Analysis and Priority Assignment**				**Step 3: IQA**		**Overall**
Number of frames	Frame extraction	Loading models	Logo detection	Close-up detection	Face detection (DNN)	Blur detection	Image quality prediction	Total
50	1.277*s*	0.524*s*	0.266*s*	0.239*s*	0.329*s*	0.166*s*	1.117*s*	3.939*s*
100	1.431*s*	0.524*s*	0.453*s*	0.413*s*	0.678*s*	0.334*s*	1.228*s*	5.310*s*

Table 5.4: Complexity analysis after second iteration: average execution time per clip for each module in the framework, for 37 video clips from the Allsvenskan dataset (average video clip duration: 77*s*). All frames are 50% down-scaled. "Loading time" refers to the loading of the logo detection and close-up detection models.

5.6 Frame Extraction Impact

When using the framework HOST-ATS, it is possible to specify how many frames it should extract for the analysis steps. The more frames being extracted, the more processing time is needed for the analysis steps. The default amount of frames being extracted is 50 frames and it takes about 4 seconds each video for selecting thumbnail. If there is more time available one could benefit the possibility to extract more than just 50 frames. When adjusting the number of frames extracted, to a higher number, the final thumbnail will very likely be another frame that has received better metric scores.

5.6.1 Frame Extraction on Shorter Videos

We have compared the time difference and thumbnail output on HOST-ATS when running it with 50 frames extracted and all frames extracted (400-450) on goal videos with a mean time of 17 seconds. The thumbnail outputs are seen in Table 5.5. The mean time usage is 4.91 seconds when extracting 50 frames and 24.56 seconds when extracting all frames. The output thumbnails are all different, but in some examples the output frame is only a few frames apart to the other and without any apparent visual difference.

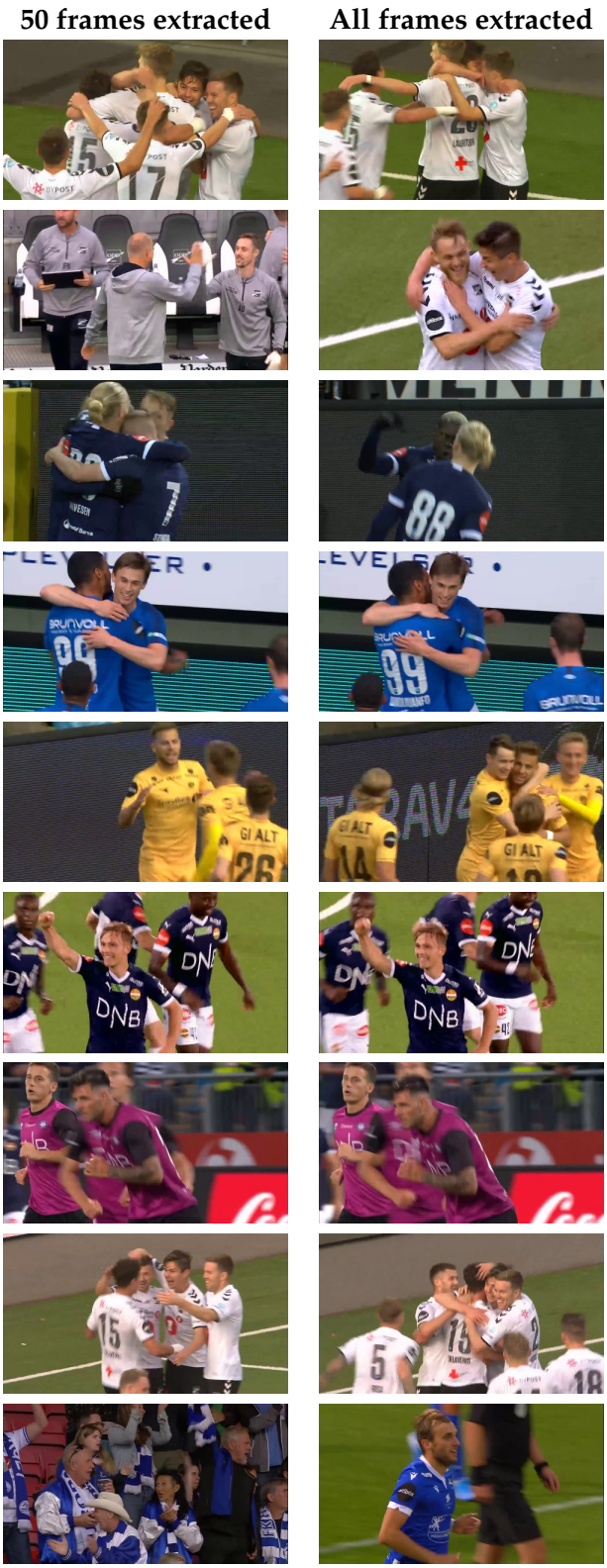


Table 5.5: Comparison of thumbnail output of HOST-ATS when extracting 50 and all frames on videos with mean duration time of 17 seconds. The mean processing time for extracting 50 frames, were 4.91 seconds. The mean processing time for extracting all frames (400-450 frames), were 24.56 seconds.

5.6.2 Frame Extraction on Longer Videos

We have also compared the time difference and thumbnail output on HOST-ATS when running it with 50 frames extracted and all frames extracted (1699-1950) on goal videos with a mean time of 1 minute and 16 seconds. The thumbnail outputs are seen in Table 5.6. The mean time usage is 5.21 seconds when extracting 50 frames and 81.06 seconds when extracting all frames. Again, the output thumbnails are all different and some examples only a few frames apart from the other.

Table 5.6: Comparison of thumbnail output of HOST-ATS when extracting 50 and all frames on videos with mean duration time of 76 seconds. The mean processing time for extracting 50 frames, were 5.21 seconds. The mean processing time for extracting all frames (1699-1950 frames), were 81.06 seconds.

5.7 Comparing First and Second Iteration Thumbnails

After improving different components in the pipeline, we want to compare the thumbnails the pipeline selects in first and second iteration. In Table 5.7, we see selected thumbnails from videos in the Allsvenskan set. Some of the thumbnails the pipeline selects in first and second are similar to eachother. There are at least two thumbnails that are worse in second iteration, following our ruleset in Table 3.1 (row 5 and 7 in Table 5.7). There is a logo appearing on a second iteration thumbnail, but the pipeline has already shown a tendency to not detect logos in the Allsvenskan set as it is not trained on it.

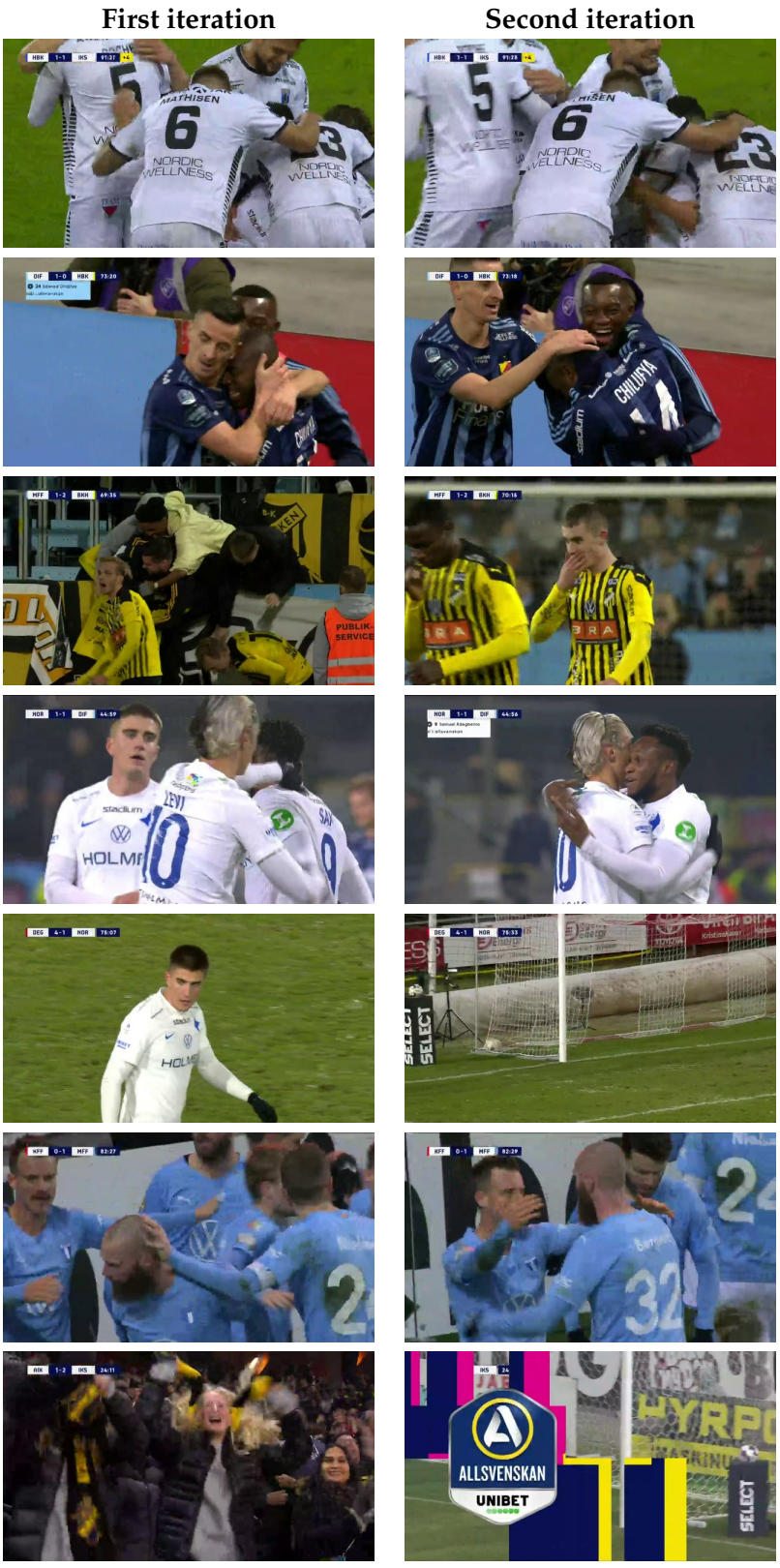


Table 5.7: Comparison of thumbnails selected by the pipeline in first and second iteration.

5.8 Other Frameworks

To compare our framework with other thumbnail selection frameworks, we tried to implement 3 other works as benchmarks.

- **Song [60]** propose a framework called "Hecate", for selecting (generic) thumbnails automatically. Hecate has the possibility to output several thumbnails for a given video, but we have used the option to output a single thumbnail per video (per our non-manual use case).

- **Vasudevan [71]** made a query-adaptive video summarization model that picks frames from a given video that are relevant to the given query. The model also has the possibility to output a single frame as a thumbnail. The query is a text of what content you want the frame to contain, it could be "soccer" or "goal".

  We tried to run the code but did not manage to, as the package "qvsumm" mentioned in the repository was not compatible and it needed code changes and/or requirement installments that were not provided. `README.md` does not include the necessary instructions.

  We downgraded the Python version to Python 2 (from Python 3) to run the file `thumbnail_demo.ipynb`, but it still was problems with the module Lasagne that threw an error. It seems like the environment provided in the `requirements.txt`, did not take all modules into account to make it run successfully. We have sent a message to the authors without any response yet.

- **Agrawal [2]** has a software repository. We have not been able to find a corresponding publication, but the repository itself is referring to the paper by Johnson [28]. From the repository description, it is not clear how the work of Johnson is used in the work of Agrawal (Work of Johnson is intended for dense captioning and not thumbnail selection).

  We tried to run the repository of Agrawal, but we did not receive any access to the pretrained model for running it. There is no information on what kind of model this is or how one can train the model from scratch.

  We have contacted the owner of the repository, but have not received a response yet.

In order to benchmark the actual thumbnail selections by our pipeline, we present a visual comparison with the state-of-the-art thumbnail selector Hecate from Song et al. [60] in Table 5.8 for 4 sample video clips.[2]

Hecate does not prioritize close-up of people, or images that may be more temporally-relevant (e.g., frames closer to a certain event annotation).

[2]Running code could not be obtained for similar generic thumbnail selectors Vasudevan et al. [71] from https://github.com/arunbalajeev/query-video-summary and Agrawal from https://github.com/sumeetag/thumbnails-for-videos.

Therefore, although the selected images might score well on objective metrics such as blur and darkness, they do not appear relevant, i.e., the model does not necessarily capture the semantics of a particular event. For example, Hecate often selects long distance view with no indication of the goal event, compared to the close-ups of the ball inside the goal or players cheering as selected by our proposed pipeline HOST-ATS.

The average end-to-end duration of Hecate is 3.4*s* on 37 video clips from the Allsvenskan set. Running our own proposed framework on the same videos uses minimum 3.9*s* as well as shown in Table 5.4. It is though possible to exclude modules like image quality prediction to receive an execution time of 2.8*s* which is faster.

Table 5.8: Comparison with the state-of-the art thumbnail selector Hecate from Song et al. [60].

5.9 Summary

In this chapter, we present the test results for the models added to the pipeline for this second iteration. Logo detection and face detection module is improved and blur detection is added to the pipeline. We present a dashboard to make the framework more user friendly. Then, a complexity

analysis of the framework is made. We inspect the impact the number of frames extracted from a video clip, affects the thumbnail selection of the pipeline. Then it is a comparison of the thumbnails selected by the pipeline in first and second iteration. At last, we do some assessment in how our framework compares to other thumbnail selection frameworks.

Chapter 6

Discussion

In this chapter, we make discussion of choices made Chapters 4 and 5. We also discuss sections that could be investigated more for a more complete end product.

6.1 Face Detection

We have considered which kind of face detection models to use by considering the time usage and accuracy. Not only by looking at the overall accuracy, but what kind of faces the face detection model tends to detect. The idea of having a "Audience" set is to evaluate the performance of the face detection model on smaller faces. We also evaluated the detections on full-scaled and half-scaled images to see if the same model managed to detect the same faces. The Dlib for instance managed to detect relatively bigger faces on the half-scaled images. Though it detected less faces it also managed to detect some faces that it could not detect on full-scaled. A model detecting relatively bigger faces than other models will be favored in the face detection model. This is because the smaller faces has a bigger chance of being in the background and even an audience face.

6.2 Blur Detection as a Framework Component

Running blur detection on frames from soccer videos has its challenges. Firstly, blur detection models do not necessarily capture the essence of what is really perceived as blur by humans, within a certain context. There might be objects or regions in a frame that are more important to the human eye than others. For instance, a frame where the celebrating soccer player is clear but the background is blurry might be discarded falsely by a blur detection model due to supposed high presence of blur, whereas the frame could actually be considered non-blurry, in terms of the object of interest in the scene. Secondly, blur scores for frames should be considered relatively to other frames from the same video and not in absolute terms. Overall image quality can vary from video to video, which leads to a different viewer tolerance for what is perceived as a blurry frame for a given video

clip.

We inspected the Laplacian operator from OpenCV Library [5], proposed to detect blurriness in an image. The operator receives an image and predicts the presence of blur with a score as output. The lower the score is, the more blur is predicted. Figure 6.1 presents the blur scores according to Laplacian method of different frames from the same video. In Figure 6.1h, we can see a close-up of a player which is clear, but the background is blurry. This frame could have served as a perfectly adequate thumbnail, but scores a high presence of blur, relative to the rest of the frames tested. In contrast, the frames in Figure 6.1a and 6.1b score a low presence of blur. These frames have a clear background but the players in the foreground are a bit blurry as they are in motion. This discrepancy demonstrates the first challenge mentioned above, namely that the blur detection model predicts higher blur the larger the total area covered by blur is, irrespective of regions of interest or layout.

Another example for this challenge is given by Figure 6.2a and 6.2b. These frames are from the same second and the same camera. The player appearing in Figure 6.2a is less blurry than the players appearing in Figure 6.2b. However, the background is more clear in the latter compared to the former. Yet another challenge is to identify why seemingly similar scenes can receive different blur scores, such as Figure 6.1c receiving a score which indicates much more blur than Figures 6.1a and 6.1b.

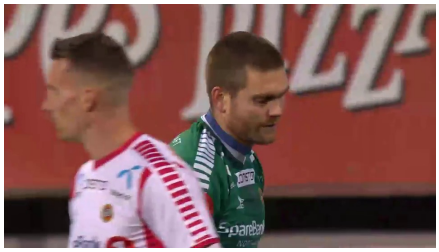
A challenge making it difficult to use the presented blur detection method, is that the overall blur score of an image does not indicate whether it is motion blur or defocus blur. For the pipeline, we want to at least focus on filtering out motion blur on foreground/people. Images with blurry defocused background, is not in our interest to filter out.

By using a blur detection module in our pipeline, we would like to promote frames that look clearer, and consequently improve the thumbnail selection process. However, the frames that existing blur detection models predict as being very clear might not be what we prefer to have as thumbnails.

(a) Blur score: 1324.40

(b) Blur score: 2796.19

(c) Blur score: 968.07



(d) Blur score: 290.10

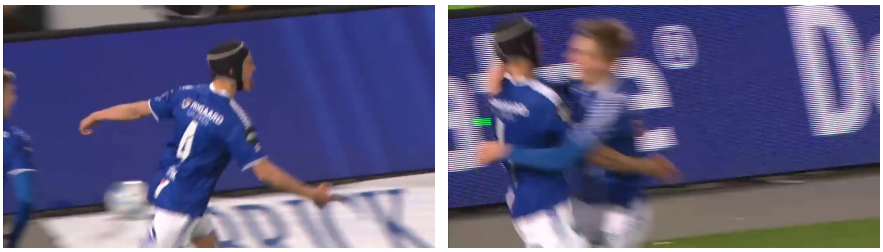
(e) Blur score: 1027.90

(f) Blur score: 690.92

(g) Blur score: 113.53

(h) Blur score: 432.48

Figure 6.1: Blur degree scores recieved using Laplacian function from OpenCV library [5] ran on selected frames from a single video (higher score indicates less blur). The resolution on the images when tested, were 480x270 pixels.



(a) Blur score: 134.31 **(b)** Blur score: 6105.34

Figure 6.2: Blur degree scores recieved using Laplacian function from OpenCV library [5] ran on selected frames from a single video (higher score indicates less blur). The model indicates high presence of blur on the image that has blurry background, and less blur on the image where the players appear blurry, but the background is more clear. The resolution on the images when tested, were 480x270 pixels.

6.3 Hyperparameter Tuning

Logo detection and close-up detection (performance results presented respectively in Section 4.2 and 4.3), are models trained using Surma [62]. The hyperparameters used on training both these models are the default values. The hyperparameters could be tuned to achieve even better performances on both of these models.

6.4 Dataset Availability and Scale

Our experiments are limited in terms of the scale of the datasets we have used for training and evaluation (which are mainly composed of 3 leagues from Norway, Sweden and United Kingdom). The datasets from the Scandinavian leagues are not as large as the SoccerNet (United Kingdom). This might result in incomplete training as mentioned in Section 5.2. Also having a variety on the datasets on for instance containing more leagues, could improve the generalizability of the modules in the pipeline.

6.5 Different Use Cases

Even though the framework is centered for the specific use case of soccer. It is possible to use this framework in different contexts as well. Our ruleset in Table 3.1, was made specific for soccer, and the pipeline by default, is following this ruleset. Since our implementation is modular, modules can be added or dropped to follow a different ruleset. If the focus of the thumbnail selection is to choose panorama pictures, the face detection can be excluded. Even in soccer context, considering the feedback from the user study in Section 4.6.2, face-detection and close-up detection can be excluded. Depending on the use case the framework is used for, the real-time performance could be more essential and dropping modules could

shorten the end-to-end execution time.

6.6 Real-Time Performance

The complexity analysis in Section 5.5, calculates the execution time of the pipeline. The pipeline is modifiable and can exclude modules like image quality prediction and the pipeline have an end-to-end execution time of 2.8s. With the default settings of the pipeline, the pipeline uses 3.9s. The execution time of the framework has to be seen in the context of that it is a thumbnail selection framework. Thumbnail selection in live production, does not necessarily depend on processing done within the same second. Traditionally, the publishing of an event video clip is usually not depending on seconds, but rather minutes. The task required to be done beforehand of a thumbnail selection, as mentioned in Section 2.2, is the event annotation with a timestamp. It is not necessary to do an event clipping (Section 2.2.2) before selecting a thumbnail, as event clipping provide cut points. For the event clip to be published both tasks, event clipping and thumbnail selection needs to be done before publishing it. If the event clipping has a longer execution time than the thumbnail selection, it is not delaying the overall end-to-end pipeline. If the thumbnail selection does become a bottleneck, it could delay the end-to-end pipeline on an average of 3.9 seconds.

6.7 Evaluation of Second Iteration Pipeline

There is no user study on the thumbnails selected by the final (second) pipeline. Although some modules where added or improved metrically, it did not seem to improve the end result significantly or affect the selection by much. Having a user study for the final pipeline could give a more reliable validation and give a more detailed comparison of first and second iteration pipeline. We considered the improvements made in the second iteration, not significant enough to show improvements in a potential user study.

6.8 Deciding on a Ruleset

It is necessary to further investigate what makes a good thumbnail for viewers (in general, for soccer events, and in particular, for goal event). In this respect, we have made note of the following from Section 4.6.2: action content as a higher priority over face detection, blur detection together with contextual information, relaxing of the close-up requirement depending on content and blur, detection of partial obstructions in thumbnail candidates. One of the limitations of our approach is that the thumbnails selected by our pipeline can be very similar to each other, especially for similar video clips, due to our fixed ruleset. For instance, if player scenes are prioritized over audience scenes in the ruleset, such frames will be more

likely to be selected for any given video clip. Updating our ruleset in Table 3.1 to correspond with answers in the user study, could make the thumbnails selected by the framework, more compelling. For instance, to make the pipeline being able to detect more action content frames, it could be possible to detect soccer ball on frames, by using object detection. For instance, YOLO [49] is another promising object detection model that could be integrated and added to our current pipeline.

6.9 Summary

In this chapter, we made discussion on choices we have made and sections that could be investigated more for a more complete end product. We discuss what we weighted in the choice of a face detection model. Then, we discuss the current blur detection models used in this framework and a possible solution to find a better blur detection model suiting our purpose. We discuss the possible improvement of our models by tuning the hyperparameters for training as well as using bigger or more diverse datasets. We also discuss the possibility for using our framework for other use cases than soccer. Then, the real-time performance is evaluated. Afterwards, we discuss the improvement of the pipeline from first and second iteration. At last, we discuss the possible improvement of updating the pipeline to follow a new ruleset corresponding more to the answers received from the user study.

Chapter 7

Conclusion

7.1 Summary

In this thesis, we present an automatic thumbnail selection system for soccer videos, which uses ML to deliver representative thumbnails with high relevance to the video content and high visual quality in near real-time. We propose a software framework that leverages logo detection, close-up detection, face detection, image quality prediction, and blur detection into an automatic thumbnail selection pipeline, and a user study framework for subjective evaluation and verification. We evaluate the proposed pipeline quantitatively using various soccer datasets, as well as qualitatively, through subjective user studies.

Our work combines various independently applicable approaches in an end-to-end system, with an overarching goal and a generalized ruleset. The subjective evaluation campaign using a novel survey framework for crowdsourced feedback collection yields a number of insights. The modular and lightweight implementation of our pipeline allows for the agile integration and benchmarking of various ML methods from literature in each module. The dashboard GUI for controlling our pipeline, and the user study framework can also be demonstrated, as the main components of our system.

Our results showed that an automatic end-to-end system for the selection of thumbnails based on contextual relevance and visual quality can yield highly attractive thumbnails, and can be used in conjunction with existing soccer video production pipelines which require real-time operation. However, the results also indicate that some of our initial rules can be reconsidered and adjusted, as viewers might have different preferences based on context. Nevertheless, our proposed pipeline is shown to work as intended, following the specified rules and priorities, and can run efficiently. This work is therefore a good starting point for even better future automatic thumbnail selection systems.

7.2 Revisiting the Problem statement

In Section 1.2, we made 5 objectives, that would be worked on for the thesis.

- **Objective 1:** We defined a ruleset by identifying properties of a good thumbnail by looking at related work. We also evaluated our ruleset and proposed refinements after looking at the results from our own user study, requesting the users opinion of which properties a thumbnail should contain.
- **Objective 2:** We implemented and end-to-end automated, modular, configurable pipeline for selecting thumbnails, following our given ruleset from objective 1. This pipeline can be controlled via a dashboard with a user-friendly GUI.
- **Objective 3:** We validated the thumbnails selected by our own framework created in objective 2, through a user study. This validation was requesting users to select the thumbnail they found most compelling (without having background information of how the thumbnail was selected) in multiple pair selections. Thumbnails selected by our (HOST-ATS) pipeline were favored over static selected, but not over manual selected.
- **Objective 4:** We did an complexity analysis, going into detail on computational requirements, model size and execution time. We discuss its real-time performance. It is not performing in real-time given that the real-time criteria is within a second. But it can be performing fast enough to avoid becoming a bottleneck in an event and highlight production pipeline in soccer. If the performance can not be considered as real-time, it is not considered as a time-critical task and the delay is not severe either.
- **Objective 5:** We compared our framework with other state-of-the-art automatic thumbnail selection frameworks. The execution time were similar, both using a few seconds to select a thumbnail. The output thumbnails were compared visually and the state-of-the-art framework, selected images that might score well on objective metrics such as blur and darkness, but they did not appear relevant, i.e., the model did not necessarily capture the semantics of a particular event.

Our contributions to these objectives leads back to the main research question:

> 'How can we select good thumbnails from soccer goal event clips automatically?'

Our solution on the main research question is that we have created a modular and configurable pipeline combining multiple ML models, trained on samples from soccer clips. The contributions to each objective shows how we could make it possible to select good thumbnails from soccer goal event clips automatically. That is, define a ruleset for thumbnails, implement a pipeline following to the ruleset, validate the performance of the pipeline, analyze its complexity and compare it to other state-of-the-art thumbnail selection frameworks.

7.3 Other Contributions

Other contributions made, related to the thesis, are the following:

- **Open source software:** We provided the full source code for the automatic thumbnail selection pipeline, the dashboard with the GUI, and the user survey as publicly accessible software under GitHub repositories[1].
- **Dashboard demonstration:** We provided 2 tutorial videos on YouTube to show how the dashboard GUI looks like and how it can be used.[2]
- **Reproducible compute capsule:** We provided a public Code Ocean capsule for reproducibility [24]. Code Ocean is a platform for sharing code and data. It makes it possible to develop, share and publish code using a web browser, without any need for specialized software. This capsule has the necessary environment to run the code, so it is possible to run it on an example video and receive an output thumbnail. It is also possible to modify the already given command for running the automatic thumbnail selection (i.e., excluding modules, modify threshold values).
- **Publications:** We published 3 papers in scientific conferences (ACM Multimedia Systems, MMSys, and Norwegian Artificial Intelligence Research Consortium, NORA)[3]) available in Sections A.1, A.2 and A.3, research track paper badges [26]. Our papers received the following badges from ACM: *Artifacts Available*, *Artifacts Evaluated – Reusable*, *Results Validated - Reproduced*[4]. These badges indicate that the work is documented to the extent that reuse and repurposing is facilitated.

7.4 Future Work

There are several aspects of the solution that has potential for future work. Some of the future work is already discussed in the discussion (Chapter 6):

- **Alternative pipeline:** An alternative approach to our complete pipeline would be to train a ML model on a large dataset of thumbnail images manually selected by experts, and use this model directly, in a single step, on video clips. Such an idea could be preferable to our current pipeline, which needs to use multiple modules to explicitly enforce certain rules. However, the logic behind the selection would

[1]HOST-ATS: https://github.com/simula/host-ats, Huldra: https://github.com/simula/huldra

[2]Demonstration tutorial videos: https://www.youtube.com/watch?v=HHMCdMucorI and https://www.youtube.com/watch?v=VZQaEy2VauQ

[3]Presentation titled "Automatic Thumbnail Selection for Soccer using Machine Learning" at NORA Annual Symposium 2022

[4]https://www.acm.org/publications/policies/artifact-review-badging

remain black box, and the underlying rules might not be possible to explicitly document and reproduce. Secondly, there exists no big-enough dataset to train such a model yet, which might need a lot of data.[5] In this respect, there is a need for open datasets shared between the industry and academia, for facilitating research in this direction.

- **Module improvement:** Referring to the discussion in Sections 6.2 and 6.3, there are several ways to improve the current modules in the pipeline. Blur detection could be improved by having a model taking viewer preferences into account. For instance with the capability to classify blur as defocus or motion blur.

- **Generalizability across leagues:** The models in the current pipeline has proven to perform with different accuracy on different leagues (e.g. Table 5.1). A wider dataset for training (on models) or more generalizable models could make the pipeline more adaptable for general use across leagues (as mentioned in Section 6.4).

- **Different use cases:** Despite its good performance, our pipeline is limited in the sense that its sole focus is on goal events, not considering other soccer events such as player substitutions, yellow and red cards. A further improvement could be to extend our automatic thumbnail selection framework to video clips/streams of other soccer events, such as player substitutions, yellow and red cards. Further extension to other sports and even general use cases is also possible. Mentioned in the discussion Section 6.5, it is already possible to use on different use cases than soccer goal events, but there are no validation or adjustments made on the framework to work optimally on these.

- **Integration of subjective findings:** In Section 6.8, we address the benefit of updating our ruleset given in Table 3.1. The thumbnails selected by our framework could have the potential to become more compelling for the viewer. To evaluate this, we could have another user study to evaluate the performance compared to first iteration pipeline (like mentioned in Section 6.7).

[5]There might also be a need for separate datasets, or different annotations within the same dataset, for different *events* (e.g., thumbnails that experts tend to select for goal event clips might be different from those for card or substitution event clips), and different *leagues* (different jerseys, players, etc. might confuse models trained on one league and tested on another).

Bibliography

[1] Vardan Agarwal. *Face Detection Models: Which to Use and Why?* Ed. by Towards Data Science. 2020. URL: https://towardsdatascience.com/face-detection-models-which-to-use-and-why-d263e82c302c (visited on 13/02/2022).

[2] Sumeet Agrawal. *thumbnails-for-videos*. 2018. URL: https://github.com/sumeetag/thumbnails-for-videos (visited on 01/03/2022).

[3] Allsvenskan. *Highlights*. 2022. URL: https://highlights.allsvenskan.se/ (visited on 21/04/2022).

[4] Terapap Apiparakoon, Nutthaphol Rakratchatakul, Maythinee Chantadisai, Usanee Vutrapongwatana, Kanaungnit Kingpetch, Sasitorn Sirisalipoch, Yothin Rakvongthai, Tawatchai Chaiwatanarat and Ekapol Chuangsuwanich. *FIGURE 2 - MaligNet: Semisupervised Learning for Bone Lesion Instance Segmentation Using Bone Scintigraphy*. 2020. URL: https://www.researchgate.net/figure/Illustration-of-the-feature-pyramid-network-FPN-The-FPN-consists-of-a-bottom-up_fig1_339006612 (visited on 11/05/2021).

[5] Gary Bradski. 'The OpenCV Library'. In: *Dr. Dobb's Journal of Software Tools* (2000).

[6] João Carreira and Andrew Zisserman. 'Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset'. In: vol. abs/1705.07750. 2017. arXiv: 1705.07750. URL: http://arxiv.org/abs/1705.07750.

[7] Cburnett. *File:Artificial neural network.svg*. 2006. URL: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg (visited on 08/05/2021).

[8] Chabacano. *File:Overfitting.svg*. 2008. URL: https://commons.wikimedia.org/wiki/File:Overfitting.svg (visited on 07/05/2021).

[9] Chen-Yu Chen, Jia-Ching Wang, Jhing-Fa Wang and Yu-Hen Hu. 'Motion Entropy Feature and Its Applications to Event-Based Segmentation of Sports Video'. In: *EURASIP Journal on Advances in Signal Processing* 2008 (2008). DOI: 10.1155/2008/460913. URL: https://link.springer.com/content/pdf/10.1155/2008/460913.pdfv.

[10] Francois Chollet. *Keras*. 2015. URL: https://github.com/fchollet/keras (visited on 24/01/2022).

[11] Anthony Cioppa, Adrien Deliege, Silvio Giancola, Bernard Ghanem, Marc Van Droogenbroeck, Rikke Gade and Thomas B. Moeslund. 'A Context-Aware Loss Function for Action Spotting in Soccer Videos'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020. DOI: 10.1109/CVPR42600.2020.01314.

[12] Pete Cook. *react-player*. 2021. URL: https://www.npmjs.com/package/react-player (visited on 21/01/2022).

[13] *Definition of football club*. In: Oxford University Press, 2022. URL: https://www.lexico.com/definition/football_club (visited on 25/04/2022).

[14] Adrien Deliège, Anthony Cioppa, Silvio Giancola, Meisam J. Seikavandi, Jacob V. Dueholm, Kamal Nasrollahi, Bernard Ghanem, Thomas B. Moeslund and Marc Van Droogenbroeck. *SoccerNet-v2 : A Dataset and Benchmarks for Holistic Understanding of Broadcast Soccer Videos*. 2020. arXiv: 2011.13367 [cs.CV].

[15] P.J. Denning, D.E. Comer, D. Gries, M.C. Mulder, A. Tucker, A.J. Turner and P.R. Young. 'Computing as a discipline'. In: vol. 22. 2. 1989, pp. 63–70. DOI: 10.1109/2.19833.

[16] Don. *Top 50 YouTube Thumbnail Ideas for Match Highlights*. 2021. URL: https://unihighlights.com/blog/top-50-youtube-thumbnails-ideas (visited on 20/04/2022).

[17] Eliteserien. *Highlights*. 2022. URL: https://highlights.eliteserien.no/ (visited on 21/04/2022).

[18] *event*. In: *Cambridge Advanced Learner's Dictionary & Thesaurus*. Cambridge University Press, 2021. URL: https://dictionary.cambridge.org/dictionary/english/event (visited on 09/05/2021).

[19] Christoph Feichtenhofer, Axel Pinz and Andrew Zisserman. 'Convolutional Two-Stream Network Fusion for Video Action Recognition'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1933–1941. DOI: 10.1109/CVPR.2016.213.

[20] FIFA.com. *More than half the world watched record-breaking 2018 World Cup*. 2018. URL: https://www.fifa.com/worldcup/news/more-than-half-the-world-watched-record-breaking-2018-world-cup (visited on 23/01/2022).

[21] Silvio Giancola, Mohieddine Amine, Tarek Dghaily and Bernard Ghanem. 'SoccerNet: A Scalable Dataset for Action Spotting in Soccer Videos'. In: vol. abs/1804.04527. 2018. arXiv: 1804.04527. URL: http://arxiv.org/abs/1804.04527.

[22] Malek Hammou, Cise Midoglu, Steven A. Hicks, Andrea Storås, Saeed Shafiee Sabet, Inga Strümke, Michael A. Riegler and Pål Halvorsen. 'Huldra: A Framework for Collecting Crowdsourced Feedback on Multimedia Assets'. In: *13th ACM Multimedia Systems Conference (MMSys '22), June 14–17, 2022, Athlone, Ireland*. New York,

NY, USA: ACM, 2022. ISBN: 978-1-4503-9283-9/22/06. DOI: 10.1145/3524273.3532887. URL: https://doi.org/10.1145/3524273.3532887.

[23] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke and Travis E. Oliphant. 'Array programming with NumPy'. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[24] Andreas Husa and Cise Midoglu. *HOST-ATS Automatic Thumbnail Selection Pipeline*. 2022. URL: https://doi.org/10.24433/CO.2648317.v1 (visited on 10/05/2022).

[25] Andreas Husa, Cise Midoglu, Malek Hammou, Pål Halvorsen and Michael A. Riegler. 'HOST-ATS: Automatic Thumbnail Selection with Dashboard-Controlled ML Pipeline and Dynamic User Survey'. In: *13th ACM Multimedia Systems Conference (MMSys '22), June 14–17, 2022, Athlone, Ireland*. New York, NY, USA: ACM, 2022. ISBN: 978-1-4503-9283-9/22/06. DOI: 10.1145/3524273.3532908. URL: https://doi.org/10.1145/3524273.3532908.

[26] Andreas Husa, Cise Midoglu, Malek Hammou, Steven A. Hicks, Dag Johansen, Tomas Kupka, Michael A. Riegler and Pål Halvorsen. 'Automatic Thumbnail Selection for Soccer Videos using Machine Learning'. In: *Proceedings of the ACM Multimedia Systems Conference (MMSys)*. 2022. DOI: 10.1145/3524273.3528182.

[27] Haroon Idrees, Amir Roshan Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar and Mubarak Shah. 'The THUMOS Challenge on Action Recognition for Videos "in the Wild"'. In: *CoRR* abs/1604.06182 (2016). arXiv: 1604.06182. URL: http://arxiv.org/abs/1604.06182.

[28] Justin Johnson, Andrej Karpathy and Li Fei-Fei. 'DenseCap: Fully Convolutional Localization Networks for Dense Captioning'. In: *CoRR* abs/1511.07571 (2015). arXiv: 1511.07571. URL: http://arxiv.org/abs/1511.07571.

[29] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar and Li Fei-Fei. 'Large-Scale Video Classification with Convolutional Neural Networks'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 1725–1732. DOI: 10.1109/CVPR.2014.223.

[30] Jongyoo Kim, Anh-Duc Nguyen and Sanghoon Lee. 'Deep CNN-Based Blind Image Quality Predictor'. In: *IEEE Transactions on Neural Networks and Learning Systems* 30.1 (2019), pp. 11–24. DOI: 10.1109/TNNLS.2018.2829819.

[31] Davis King. *dlib C++ Library*. 2021. URL: http://dlib.net/ (visited on 24/01/2022).

[32] Ryan Knott. *What Are Video Thumbnails and Why Do They Matter?* Ed. by TechSmith. 2021. URL: https://www.techsmith.com/blog/what-are-video-thumbnails/ (visited on 13/10/2021).

[33] Jacek Komorowski, Grzegorz Kurzejamski and Grzegorz Sarwas. 'FootAndBall: Integrated player and ball detector'. In: *CoRR* abs/1912.05445 (2019). arXiv: 1912.05445. URL: http://arxiv.org/abs/1912.05445.

[34] Harilaos Koumaras, Georgios Gardikis, George Xilouris, Evangelos Pallis and Anastasios Kourtis. 'Shot boundary detection without threshold parameters'. In: *J. Electronic Imaging* 15 (Apr. 2006), p. 020503. DOI: 10.1117/1.2199878.

[35] Thomas J Law. *The Perfect YouTube Thumbnail Size and Best Practices*. 2021. URL: https://www.oberlo.com/blog/youtube-thumbnail-size (visited on 21/04/2022).

[36] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding and Shilei Wen. 'BMN: Boundary-Matching Network for Temporal Action Proposal Generation'. In: *Proceedings of IEEE International Conference on Computer Vision (ICCV)*. 2019.

[37] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang and Ming Yang. 'BSN: Boundary Sensitive Network for Temporal Action Proposal Generation'. In: *Proceedings of the European Conference Computer Vision (ECCV)*. 2018.

[38] MATLAB. *brisque (R2021a)*. 2021. URL: https://se.mathworks.com/help/images/ref/brisque.html (visited on 24/01/2022).

[39] Pier Luigi Mazzeo, Marco Leo, Paolo Spagnolo and Massimiliano Nitti. 'Soccer Ball Detection by Comparing Different Feature Extraction Methodologies'. In: *Advances in Artificial Intelligence* 2012 (2012), p. 12. DOI: 10.1155/2012/512159. URL: https://doi.org/10.1155/2012/512159.

[40] MMSys. *Call For Papers*. 2022. URL: https://mmsys2022.ie/authors/call-for-papers (visited on 12/05/2022).

[41] MTheiler. *File:HOG scikit-image AngelaMerkel.jpeg*. 2019. URL: https://commons.wikimedia.org/wiki/File:HOG_scikit-image_AngelaMerkel.jpeg (visited on 07/12/2021).

[42] Olav Andre Nergård Rongved, Markus Stige, Steven Alexander Hicks, Vajira Lasantha Thambawita, Cise Midoglu, Evi Zouganeli, Dag Johansen, Michael Alexander Riegler and Pål Halvorsen. 'Automated Event Detection and Classification in Soccer: The Potential of Using Multiple Modalities'. In: *Machine Learning and Knowledge Extraction* 3.4 (2021), pp. 1030–1054. DOI: 10.3390/make3040051. URL: https://www.mdpi.com/2504-4990/3/4/51.

[43] NORA. *NORA*. 2022. URL: https://www.nora.ai (visited on 12/05/2022).

[44] NORA. *NORA Annual Conference*. 2022. URL: https://www.nora.ai/nora-annual-conference/ (visited on 12/05/2022).

[45] Research council of Norway. *AI-PRODUCER: AI-based Video Clipping and Summarization of Sport Events*. 2021. URL: https://prosjektbanken.forskningsradet.no/en/project/FORISS/327717 (visited on 02/05/2022).

[46] Ricardo Ocampo. *Deep CNN-Based Blind Image Quality Predictor in Python*. Ed. by Towards Data Science. 2019. URL: https://towardsdatascience.com/deep-image-quality-assessment-with-tensorflow-2-0-69ed8c32f195 (visited on 03/11/2021).

[47] Muhammad Rafiq, Ghazala Rafiq, Rockson Agyeman, Seong-Il Jin and Gyu Sang Choi. 'Scene Classification for Sports Video Summarization Using Transfer Learning'. In: *Sensors* 20 (Mar. 2020), p. 1702. DOI: 10.3390/s20061702.

[48] Arnau Raventos, Raul Quijada, Luis Torres and Francesc Tarres. *Automatic Summarization of Soccer Highlights Using Audio-visual Descriptors*. 2014. arXiv: 1411.6496 [cs.IR].

[49] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick and Ali Farhadi. 'You Only Look Once: Unified, Real-Time Object Detection'. In: *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[50] Reede Ren and Joemon Jose. 'Football Video Segmentation Based on Video Production Strategy'. In: Mar. 2005, pp. 433–446. ISBN: 978-3-540-25295-5. DOI: 10.1007/978-3-540-31865-1_31.

[51] Shaoqing Ren, Kaiming He, Ross Girshick and Jian Sun. 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks'. In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett. Curran Associates, Inc., 2015, pp. 91–99.

[52] Olav A. Nergård Rongved, Steven A Hicks, Vajira Thambawita, Håkon K Stensland, Evi Zouganeli, Dag Johansen, Cise Midoglu, Michael A Riegler and Pål Halvorsen. 'Using 3D Convolutional Neural Networks for Real-time Detection of Soccer Events'. eng. In: *International journal of semantic computing* 15.2 (2021), pp. 161–187. ISSN: 1793-351X.

[53] Olav A. Nergård Rongved, Steven A. Hicks, Vajira Thambawita, Håkon K. Stensland, Evi Zouganeli, Dag Johansen, Michael A. Riegler and Pål Halvorsen. 'Real-Time Detection of Events in Soccer Videos using 3D Convolutional Neural Networks'. In: *Proceedings of the IEEE International Symposium on Multimedia (ISM)*. 2020, pp. 135–144. DOI: 10.1109/ISM.2020.00030.

[54] Adrian Rosebrock. *Figure 4: The 5 different types of Haar-like features extracted from an image patch*. 2021. URL: https://www.pyimagesearch.com/2021/04/12/opencv-haar-cascades/ (visited on 19/10/2021).

[55] Adrian Rosebrock. *OpenCV Haar Cascades*. Ed. by pyimagesearch. 2021. URL: https://www.pyimagesearch.com/2021/04/12/opencv-haar-cascades/ (visited on 21/11/2021).

[56] Sumit Saha. *A CNN sequence to classify handwritten digits*. 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (visited on 11/05/2021).

[57] Matko Šarić, Dujmić Hrvoje and Baričević Domagoj. 'Shot Boundary Detection in Soccer Video using Twin-comparison Algorithm and Dominant Color Region'. In: *Journal of Information and Organizational Sciences* 32 (June 2008).

[58] Zheng Shou, Dongang Wang and Shih-Fu Chang. 'Temporal Action Localization in Untrimmed Videos via Multi-stage CNNs'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1049–1058. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.119.

[59] Karen Simonyan and Andrew Zisserman. 'Two-Stream Convolutional Networks for Action Recognition in Videos'. In: *Proceedings of Advances in Neural Information Processing Systems (NIPS)*. 2014, pp. 568–576.

[60] Yale Song, Miriam Redi, Jordi Vallmitjana and Alejandro Jaimes. *To Click or Not To Click: Automatic Selection of Beautiful Thumbnails from Videos*. 2016. arXiv: 1609.01388 [cs.MM].

[61] Bolan Su, Shijian Lu and Chew Lim Tan. 'Blurred Image Region Detection and Classification'. In: Nov. 2011, pp. 1397–1400. DOI: 10.1145/2072298.2072024.

[62] Greg Surma. *Image Classifier - Cats vs Dogs*. Ed. by Medium. 2018. URL: https://gsurma.medium.com/image-classifier-cats-vs-dogs-with-convolutional-neural-networks-cnns-and-google-colabs-4e9af21ae7a8 (visited on 22/09/2021).

[63] TensorFlow. *API Documentation*. 2021. URL: https://www.tensorflow.org/api_docs/ (visited on 13/12/2021).

[64] Dian Tjondronegoro, Yi-Ping Phoebe Chen and Binh Pham. 'Sports video summarization using highlights and play-breaks'. In: *Proceedings of ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR)*. 2003, pp. 201–208. DOI: 10.1145/973264.973296.

[65] Torrens University Australia. *Why the Sports Industry is Booming in 2020 (and which key players are driving growth)*. 2020. URL: https://www.torrens.edu.au/blog/why-sports-industry-is-booming-in-2020-which-key-players-driving-growth (visited on 20/01/2022).

[66] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani and Manohar Paluri. *Learning Spatiotemporal Features with 3D Convolutional Networks*. 2015. arXiv: 1412.0767 [cs.CV]. URL: https://arxiv.org/abs/1412.0767.

[67] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun and Manohar Paluri. 'A Closer Look at Spatiotemporal Convolutions for Action Recognition'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 6450–6459. DOI: 10.1109/CVPR.2018.00675.

[68] Twitch. *Twitch*. 2022. URL: https://www.twitch.tv (visited on 21/04/2022).

[69] Joakim O. Valand, Haris Kadragic, Steven A. Hicks, Vajira Thambawita, Cise Midoglu, Tomas Kupka, Dag Johansen, Michael A. Riegler and Pål Halvorsen. 'Automated Clipping of Soccer Events using Machine Learning'. In: *2021 IEEE International Symposium on Multimedia (ISM)*. 2021, pp. 210–214. DOI: 10.1109/ISM52913.2021.00042.

[70] Joakim Olav Valand, Haris Kadragic, Steven Alexander Hicks, Vajira Lasantha Thambawita, Cise Midoglu, Tomas Kupka, Dag Johansen, Michael Alexander Riegler and Pål Halvorsen. 'AI-Based Video Clipping of Soccer Events'. In: *Machine Learning and Knowledge Extraction* 3.4 (2021), pp. 990–1008. DOI: 10.3390/make3040049. URL: https://www.mdpi.com/2504-4990/3/4/49.

[71] Arun Balajee Vasudevan, Michael Gygli, Anna Volokitin and Luc Van Gool. *Query-adaptive Video Summarization via Quality-aware Relevance Estimation*. 2017. arXiv: 1705.00581 [cs.CV].

[72] Vimeo. *Watch*. 2022. URL: https://vimeo.com/watch (visited on 21/04/2022).

[73] Vimeo Livestream Blog. *Streaming Stats - 47 Must-Know Live Video Streaming Statistics*. 2022. URL: https://livestream.com/blog/62-must-know-stats-live-video-streaming (visited on 12/03/2022).

[74] P. Viola and M. Jones. 'Rapid object detection using a boosted cascade of simple features'. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. 2001, pp. I–I. DOI: 10.1109/CVPR.2001.990517.

[75] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang and Luc Van Gool. 'Temporal Segment Networks: Towards Good Practices for Deep Action Recognition'. In: *Proceedings of the European Conference Computer Vision (ECCV)*. 2016, pp. 20–36. ISBN: 978-3-319-46484-8.

[76] Huijuan Xu, Abir Das and Kate Saenko. 'R-C3D: Region Convolutional 3D Network for Temporal Activity Detection'. In: *Proceedings of IEEE International Conference on Computer Vision (ICCV)*. 2017.

[77] Peng Xu, Lexing Xie, Shih-Fu Chang, A. Divakaran, A. Vetro and Huifang Sun. 'Algorithms and system for segmentation and structure analysis in soccer video'. In: *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*. 2001, pp. 721–724. DOI: 10.1109/ICME.2001.1237822.

[78] YouTube. *YouTube*. 2022. URL: https://www.youtube.com (visited on 21/04/2022).

[79] Hossam Zawbaa, Nashwa El-Bendary, Aboul Ella Hassanien and Tai-Hoon Kim. 'Event Detection Based Approach for Soccer Video Summarization Using Machine learning'. In: *International Journal of Multimedia and Ubiquitous Engineering (IJMUE)* 7 (Jan. 2012).

[80] Hossam M. Zawbaa, Nashwa El-Bendary, Aboul Ella Hassanien and Ajith Abraham. 'SVM-based soccer video summarization system'. In: *Proceedings of the World Congress on Nature and Biologically Inspired Computing*. 2011, pp. 7–11. DOI: 10.1109/NaBIC.2011.6089409.

[81] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li and Yu Qiao. 'Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks'. In: *CoRR* abs/1604.02878 (2016). arXiv: 1604.02878. URL: http://arxiv.org/abs/1604.02878.

Appendix A

Publications and Presentations

A.1 Research Track paper: Automatic Thumbnail Selection for Soccer Videos using Machine Learning

Authors Andreas Husa, Cise Midoglu, Malek Hammou, Steven A. Hicks, Dag Johansen, Tomas Kupka, Michael A. Riegler, Pål Havlorsen

Published International Conference of Multimedia Systems (MMSYS), Ireland, June 2022 [26]

Short description This paper is presenting the automatic thumbnail selection framework presented in this thesis.

Venue The ACM Multimedia Systems Conference (MMSys) provides a forum for researchers to present and share their latest research findings in multimedia systems. MMSYS specializes in multimedia computing, covering fields such as networking, operating systems, real-time systems, databases, mobile computing, distributed systems, computer vision, and middleware communities [40].

MMSys research track is targeted at enabling authors to present entire multimedia systems or research work that builds on considerable amounts of earlier work in a self-contained manner [40].

Appendices The research track paper "Automatic Thumbnail Selection for Soccer Videos using Machine Learning" includes one research appendix (containing supplementary material related to the user study), and one reproducibility artifact appendix (containing instructions to reproduce the results).

Automatic Thumbnail Selection for Soccer Videos using Machine Learning

Andreas Husa
SimulaMet, Norway

Cise Midoglu
SimulaMet, Norway

Malek Hammou
SimulaMet, Norway

Steven A. Hicks
SimulaMet, Norway

Dag Johansen
UIT The Artic University of Norway

Tomas Kupka
Forzasys AS, Norway

Michael A. Riegler*
SimulaMet, Norway

Pål Halvorsen†‡
SimulaMet, Norway

ABSTRACT

Thumbnail selection is a very important aspect of online sport video presentation, as thumbnails capture the essence of important events, engage viewers, and make video clips attractive to watch. Traditional solutions in the soccer domain for presenting highlight clips of important events such as goals, substitutions, and cards rely on the manual or static selection of thumbnails. However, such approaches can result in the selection of sub-optimal video frames as snapshots, which degrades the overall quality of the video clip as perceived by viewers, and consequently decreases viewership, not to mention that manual processes are expensive and time consuming. In this paper, we present an automatic thumbnail selection system for soccer videos which uses machine learning to deliver representative thumbnails with high relevance to video content and high visual quality in near real-time. Our proposed system combines a software framework which integrates logo detection, close-up shot detection, face detection, and image quality analysis into a modular and customizable pipeline, and a subjective evaluation framework for the evaluation of results. We evaluate our proposed pipeline quantitatively using various soccer datasets, in terms of complexity, runtime, and adherence to a pre-defined ruleset, as well as qualitatively through a user study, in terms of the perception of output thumbnails by end-users. Our results show that an automatic end-to-end system for the selection of thumbnails based on contextual relevance and visual quality can yield attractive highlight clips, and can be used in conjunction with existing soccer broadcast pipelines which require real-time operation.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; *Activity recognition and understanding*; **Video summarization**.

*Also affiliated with UIT The Artic University of Norway
†Also affiliated with Oslo Metropolitan University, Norway
‡Also affiliated with Forzasys AS, Norway

MMSys '22, June 14–17, 2022, Athlone, Ireland

ACM ISBN 978-1-4503-9283-9/22/06.
https://doi.org/10.1145/3524273.3528182

KEYWORDS

blur detection; deep learning; image quality; logo detection; object detection; shot boundary detection; soccer; sports analysis; thumbnail generation; user survey; video

ACM Reference Format:
Andreas Husa, Cise Midoglu, Malek Hammou, Steven A. Hicks, Dag Johansen, Tomas Kupka, Michael A. Riegler, and Pål Halvorsen. 2022. Automatic Thumbnail Selection for Soccer Videos using Machine Learning. In *13th ACM Multimedia Systems Conference (MMSys '22), June 14–17, 2022, Athlone, Ireland.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3524273.3528182

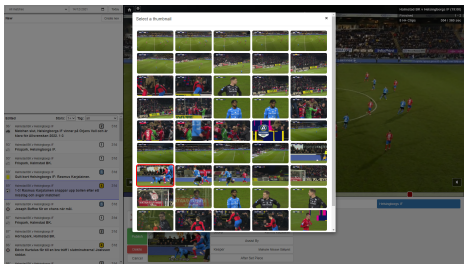
1 INTRODUCTION

Sports broadcasting and streaming are immensely popular, and the interest in viewing videos from sports events grows day by day. Today, live streaming of sports events generates most of the video traffic and is replacing live broadcasting on TV [38]. For example, 3.572 billion viewers tuned in to watch the 2018 FIFA World Cup [9], and as of 2020, soccer had a global market share of about 45% of the $500 billion sports industry [34]. However, the availability of content and the large number of games make systems for extracting highlights and providing summaries in real- or near real-time increasingly important. The generation of video summaries and highlight clips from sports games is of tremendous interest for broadcasters, as a large percent of audiences prefer to view only the main events in a game.

The state-of-the-art systems for generating soccer highlight clips consist of several manual operations. A typical tagging center is shown in Figure 1, where important events such as goals, substitutions, and cards are tagged and annotated before being published into individual clips. A typical pipeline consists of a detection phase where the video is marked with an event such as a goal or card (Figure 1a), relevant frames are cut to generate a highlight clip, and in a second "refinement" phase, the highlight clip is further improved by customized trimming, insertion of additional textual descriptions and tags, and the selection/update of a thumbnail (Figure 1b).

A thumbnail is an image representing a video. For soccer, thumbnails are frequently used in web pages where various highlight clips are presented in gallery form [2, 8], and serve as the first impression meant to attract people to view a highlight clip. As highlight clips summarize certain important events in a soccer game, the challenge is to find appropriate thumbnails for each clip (and not a single thumbnail for the overall game). Thumbnails need to be

(a) An event is detected, classified and annotated.



(b) A thumbnail is selected for the highlight clip.

Figure 1: A live tagging operation by people in a cumbersome, error-prone, and tedious manual process.

selected carefully to be eye-catching and should properly represent the event in the highlight clip, as unattractive thumbnails can cause low engagement (highlight clips might go unwatched due to non-appealing thumbnails). Manual selection can potentially yield appropriate thumbnails, but since the selection operation is time-consuming and expensive, image quality is often not considered extensively. In this sense, automating the thumbnail selection process has the potential to both save resources and improve quality.

In this work, our goal is to replicate the potential performance of manual thumbnail selection with an automated system, which is much faster (applicable in real-time), cheaper, and quantitative (documentable and reproducible). This will not only save resources for the top soccer leagues which already rely on manual selection, but also enable similar services for less resource-capable leagues where automation is the only alternative. We develop an AI-based solution to identify appropriate thumbnail candidates by checking the video frames for logos, scene boundaries, faces, and analyzing image quality. Our proposed approach considers relevance to video content along with visual quality and aesthetic metrics, so that the resulting thumbnail is adequately representative of the video clip. In particular, we make the following contributions:

- We propose a modular and customizable automatic thumbnail selection pipeline, which integrates pre-processing (options for trimming, down-scaling, and down-sampling), logo detection, close-up shot detection, face detection, and image quality analysis (image quality prediction and blur detection). The modular and lightweight implementation allows for the agile integration and benchmarking of various Machine Learning (ML) models from literature in each module[1]. This pipeline can be controlled via a dashboard with a user-friendly Graphical User Interface (GUI).
- We evaluate our proposed pipeline quantitatively using various soccer datasets, in terms of system performance (complexity and runtime) as well as adherence to a pre-defined ruleset, under different configurations for components.
- We run a subjective evaluation (user study) involving 42 participants, to evaluate the performance of the proposed pipeline qualitatively. The subjective evaluation campaign is conducted using a novel survey framework for crowdsourced feedback collection on multimedia assets called HOST-ATS, which is a plug-and-play system component for any future studies as well[2].
- We additionally run a small benchmark study with a state-of-the art generic (non soccer-specific) thumbnail selector.
- We provide a discussion of the generalizability of our approach, along with the limitations and pitfalls of our pipeline, and suggest potential improvements and future work topics.

Overall, the novelty of our work includes: the combination of various independently applicable approaches in an end-to-end automatic thumbnail selection pipeline, with an overarching goal and a generalized ruleset; corresponding quantitative performance and complexity analyses; a novel user study framework for the qualitative evaluation of different thumbnail candidates; as well as the insights gained from the study and subsequent discussions. Our automatic thumbnail selection system is able to reduce production costs by automating a traditionally complex and labor-intensive task, accompanied by end-user validation. Our approach is applicable to various other sports broadcasts, such as skiing, handball, or ice hockey, and presents a viable potential to impact future sports productions.

The rest of this paper is structured as follows. In Section 2, we provide background information and an overview of related work. In Section 3, we describe our proposed thumbnail selection pipeline in detail. In Section 4, we present the results from our quantitative analysis of the proposed pipeline. In Section 5, we introduce a user study framework, and present the results from our qualitative analysis of the proposed thumbnail selection pipeline through subjective evaluation. In Section 6, we discuss a number of relevant aspects including limitations and potential future work. In Section 7, we conclude the paper.

2 BACKGROUND AND RELATED WORK

Soccer video production systems can incorporate research findings from different fields such as object detection [17, 23, 26], shot boundary detection [18, 42], event detection and clasification [5, 13, 20, 21, 24, 27, 28, 30, 35], and event clipping [4, 33, 36, 40]. In this work, our particular focus is on automatic thumbnail selection. As a representative snapshot, thumbnails capture the essence of a video and provide the first impression to the viewers. A good thumbnail makes the video clip more attractive to watch [19, 31]. Various components from the above fields can facilitate the identification of appropriate images that best describe the events in a video sequence and are of high quality.

[1]The proposed pipeline is not limited to the models currently presented in this paper, alternative models can be added with relative ease.

[2]Live deployment of the HOST-ATS subjective evaluation (user study) framework: https://host-ats.herokuapp.com

There is not much work on automatic thumbnail selection specifically for sports videos, but there are a number of related works that focus on thumbnail selection in general. Song et al. [31] propose a generic model called "Hecate" for selecting thumbnails automatically. Their framework uses a video as input, and filters the frames that are qualified as low-quality such as blurry, dark or uniform-colored frames. This is calculated with and decided upon via a threshold value, and not through ML. The framework also filters frames that are related to fading, dissolving or wiping effects in the video, identifying these through a shot boundary detection model. In a second step, frames that are near duplicates are discarded, and finally, frames with highest aesthetic quality are selected. This can be done by selecting the frame from a cluster with the smallest difference value (i.e., the frame that has the least change from the other frames in the same cluster, or the mean frame), where clusters are frames that have visual similarities. The aesthetic quality can be calculated by using a model that assigns a beauty score to a given image. This model has been trained by a set of images that have been annotated with subjective aesthetic scores. Vasudevan et al. [37] present a query-adaptive video summarization model which picks frames from a given video that are relevant to the given query. The model also has the possibility to output a single frame as a thumbnail. The query is a text of what content the end-user would like the frame to contain (e.g., in our context, it could be "soccer" or "goal").

A number of Image Quality Analysis (IQA) models for predicting the subjective and/or objective quality of an image have also been proposed. The model by Jongyoo [14] was tested for detecting distortion types on images, primarily white noise and blur. This is a no-reference image quality assessment (NR-IQA) model meant to be used for assessment without reference images. In some practical scenarios, there may be no reference image, then it can be useful to have a general IQA.

As our focus is on soccer, and there can be a lot of motion in soccer videos resulting in several frames ending up being blurry, blur detection is an important field of research. When selecting a single frame from a video, it is important to avoid images that are too blurry for aesthetic reasons, but also keeping images which are informative and representative of the event. It should be possible to see what is happening in the image, and too much blur could avoid that. In this context, the blur detection operator Laplacian from the OpenCV library [3] is also relevant. The operator outputs a value for a given image and predicts the presence of blur with a score, where the higher the score is, the less blur is predicted.

3 THUMBNAIL SELECTION PIPELINE

Related work indicates that a good thumbnail is relevant to the corresponding video, and appears interesting and attractive in terms of content and image quality [16, 19, 31]. In this regard, we center our proposed thumbnail selection pipeline around 3 key principles, namely relevance, content, and image-quality.

Relevance: The thumbnail that our proposed pipeline selects will be a frame from the video it is supposed to represent (i.e., no external images are considered related to the clip by the automated system). Video frames from the highlight clip will be used as input to our pipeline, so the output image is relevant to the video as it is a frame around the event annotation.

Content: Highlight clips are usually presented in a gallery as a grid, where each thumbnail appears in a small size (e.g., 200 pixels) on the screen. It could be difficult for viewers to understand the contents of the thumbnail if the image displays a long-distance shot of the soccer field. Therefore, we resolve to use close-up shots in our pipeline. Close-up shots are usually frames showing the soccer players, spectators (audience), and managers. There could also be frames from the replay of the event with shots that are closer than the default long-distance shot. If a frame is identified as a close-up shot, it will have a higher priority in the thumbnail selection process. We would also like to omit graphics such as the logo transitions appearing before replays. So, if a frame is identified as containing a logo, it will not be used as a thumbnail.

Image quality: It is possible that there are frames in a video which appear aesthetically unpleasing or unclear to the human eye. For instance, images that are blurry, dark, and/or fading are not usable as thumbnails. Therefore, we propose to undertake image quality analysis as the final filter in our pipeline.

Category	No	Rule
Relevance	1	The thumbnail should be a frame from the video clip itself.
Content	2	The frame should be a close-up shot of people.
	3	The frame should contain a face.
	4	The frame should not contain graphics (e.g., logo).
	5	The frame should not contain visuals of a fading transition.
Image Quality	6	The frame should not be blurry.
	7	The frame should not be dark.

Table 1: Thumbnail selection rules for our proposed pipeline.

Based on the above observations, we devise a number of thumbnail selection goals, which are listed in Table 1. These rules do not mean that our definition of a good thumbnail is universal, but rather establish a framework which is tailored for soccer videos. Our pipeline consists of 3 steps: pre-processing, content analysis and priority assignment, and image quality analysis. Figure 2 presents these steps and the corresponding components in our pipeline. A video clip (sequence of video frames) is fed as input to the pipeline, and the final output is an image, which is a frame from the video clip, as the suggested thumbnail.

3.1 Step 1: Pre-processing

In the pre-processing step, the sequence of frames[3] can be trimmed, down-sampled, and/or down-scaled. *Trimming* refers to the cutting of a desired number of seconds from the beginning and/or end of the sequence. It is also possible to define a time interval of interest with respect to an event annotation. This allows for increasing the relevance of the thumbnail candidates to the central event in the clip, by ensuring that images come from around the event annotation timestamp. *Down-sampling* refers to the extraction of a lower number of frames as a subset from the full set of frames in the sequence. It allows for decreasing the number of frames that are considered further on in the pipeline, consequently decreasing the amount of processing time. *Down-scaling* refers to the reduction of image resolution (changing the resolution of individual frames) in terms of a percentage. Table 8 shows the influence of down-scaling on accuracy. Each of these operations are optional, with

[3]Sequence of frames refers to the frames in the original input (video clip). E.g., a 30 second video clip at 30fps would yield 900 frames.

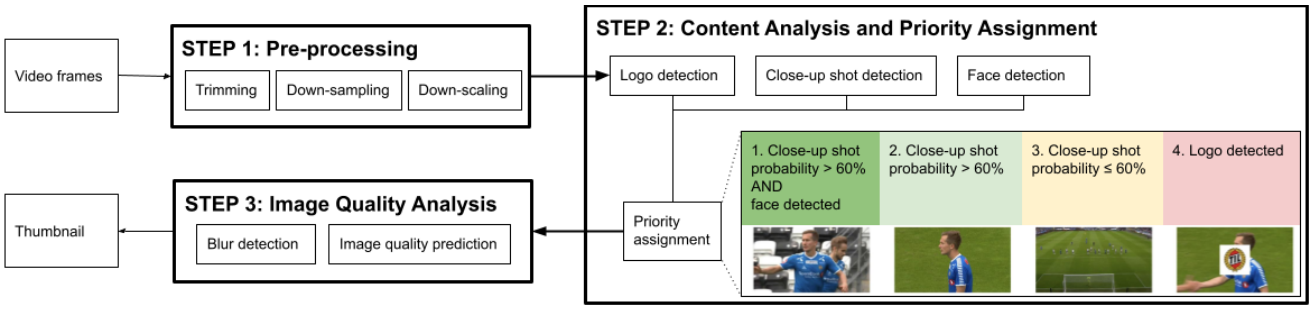


Figure 2: Proposed automatic thumbnail selection pipeline.

configurable parameters. The set of frames remaining after the pre-processing step is passed to the next step.

3.2 Step 2: Content Analysis and Priority Assignment

In the second step, the contents of the frames passed on from the first step are analyzed according to rules 2 – 5 from Table 1. For this purpose, independent modules for logo detection, close-up shot detection, and face detection are used, after which priorities are assigned to each frame and the frames are filtered accordingly.

3.2.1 Logo Detection. The logo detection module is for detecting logos and other graphics that might appear in the video frames. These are frames we would like to eliminate from the thumbnail selection. As an alternative model for this module, we train a convolutional neural network (CNN) based on the architecture presented by Surma [32]. This is a general image classification model that can classify images based on a given dataset. The output of the model is a probability score between 0 and 1, where an output of 0.5 or above indicates that the image contains a logo. Alternatively, the model by Ocampo [25] can also be used, which in turn is based on a model proposed by Jongyoo et al. [14]. The logo detection model and the threshold value are configurable parameters.

3.2.2 Close-up Shot Detection. The close-up shot detection module is used to decide whether a video frame depicts a scene coming from a wide-angle camera (zoomed-out, long-distance shot), or a close-up (zoomed-in) shot. Images classified as close-up shots are prioritized in thumbnail selection. Our pipeline supports the use of the image classification model by Surma [32] in this module. The threshold value for model certainty is a configurable parameter. Unlike the logo detection module, images with a probability below this threshold are not omitted, but receive a lower priority.

3.2.3 Face Detection. Face detection is used to detect the appearance of a face on a given image, as well as where the face appears on the image. In our context, we consider a thumbnail image to be more relevant if there is a face appearing in it. Traditionally, face detection models rely on frontal views, and cannot detect a person's head from behind. It is possible to consider images with faces turned to an angle (such as a person's head from the side or behind) to be just as relevant. However, models for detecting such phenomena are more complex and less accurate, so in this work we only consider frontal face views for the sake of simplicity and accuracy. Our pipeline currently supports 4 alternative models for this module. Haar cascade [39] is an object detection model which is fast, but tends to be prone to false positive detection, compared to other models [29]. This algorithm can be run in real-time, making it possible to detect objects in live video streams. It is possible to train the model for detecting other objects as well as faces. It is capable of detecting objects regardless of their scale and position in an image. Dlib [15] uses features extracted by histogram of oriented gradients (HOG), and passes them through a support vector machine (SVM). HOG counts the occurrences of gradient orientation on fragments of the picture. The method can be helpful for finding shapes in the picture. We support MTCNN [41] where a CNN obtains candidate windows, filters out the false positive candidates, and performs a facial landmark detection. The DNN [3] face detector in OpenCV is a Caffe model which is based on the Single Shot-Multibox Detector (SSD) and uses ResNet-10 architecture as a backbone. DNN is faster, has more detections and is more accurate overall. Agarwal [1] compared the performance of Dlib, Haar, and MTCNN, concluding that Dlib and MTCNN perform much better for face detection in terms of accuracy, but use more time than Haar for processing. Any one of the models can be used in our pipeline for face detection, specified via configuration parameters.

3.2.4 Priority Assignment. As shown in Figure 2, the results from the logo detection, close-up shot detection, and face detection modules are passed to a priority assignment module before being filtered. Our pipeline uses 4 priority levels:

(4) Images that are classified by the logo detection module as containing a logo are assigned to priority level 4, the lowest priority, and dropped[4]. Images that are classified by the logo detection module as not containing a logo proceed ahead.
(3) Images that are classified by the close-up shot detection module as containing a close-up shot are sorted in descending order of their probability score (i.e., image with the highest probability score comes first). Images with a score below the threshold value (default=75%) are assigned to priority level 3. Images with a score above the threshold value proceed ahead. Rankings are preserved in both cases.

[4]If all the images are classified as containing a logo, they are all assigned to the priority level 1 and passed to the image quality prediction module. In this case, the thumbnail presented as the overall result of the framework will be an image with a logo, which, despite being undesirable, is preferred over no output.

(2) Images that are classified by the face detection module as not containing any faces are assigned to priority level 2.
(1) Images that are classified as having at least one face are sorted according to the pixel size of the biggest face detected on the image in descending order, and assigned to priority level 1.

Note that increasing the threshold for the probability score from the close-up shot detection module can increase the adherence to the established thumbnail selection rules (more rigid enforcement of rule 2 from Table 1) and decrease latency (as the face detection module will take shorter if fewer images are passed on from the close-up shot detector), but also carries the risk of omitting potentially viable thumbnail candidates by assigning more images to the lower priority levels. Another trade-off is related to the final sorting of images which have been assigned to priority level 1, as sorting by face size vs. sorting by close-up probability emphasize different preferences. We propose that this choice be made depending on the accuracy of the models used in the face detection and close-up shot detection modules. For instance, in Section 4, we sort the images assigned to level 1 by face size in descending order, since the face detection model Dlib has better accuracy than the close-up shot detection model Surma, especially on bigger faces.

3.3 Step 3: Image Quality Analysis (IQA)

The assignment of images to priority levels in the second step gives us a sorted list of thumbnail candidates. Then, one by one from the top, each candidate goes through the third step. The iteration for selecting a thumbnail candidate starts at the highest priority level, and follows the image order prescribed in the previous step. As mentioned above, we prefer sorting images by the size of the largest face detected in them at this level. If there are no images assigned to the higher priority level, the iteration skips to the next priority level. During the iteration process, an image quality predictor [25] will be run on each image. It is supposed to predict the quality of an image by calculating its blur and distortion. If the Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) score from the quality predictor is below a threshold, the image is chosen as the output thumbnail. The image quality prediction module is only ran until the first image to satisfy this condition, and this image will become the final thumbnail. For instance, in Figure 2, the sample image in the fourth category would be omitted due to having a logo, and not passed to the third step. The sample image in the first category would not be omitted, and has a high chance of being selected as the final thumbnail in the third step.

4 EXPERIMENTS AND RESULTS

4.1 Datasets

In this work, we make use of 2 in-house video datasets generated from the Norwegian and Swedish men's elite soccer leagues[5], focusing on goal events. Table 2 presents the distribution of samples in our datasets into training, validation, and test splits. Table 3 presents the corresponding image resolutions per dataset, for the results presented in this section[6].

Dataset	Class	Train.	Val.	Test	Total
E1	Logo	561	240	223	1024
	No logo	4281	1355	1388	7024
E2	Close-up	564	109	29	702
	No close-up	772	158	38	968
E3	Face (various)	-	-	56	56
	Face (audience)	-	-	32	32
	No face (various)	-	-	33	33
A1	Logo	-	-	69	69
	No logo	-	-	111	111

Table 2: Distribution of samples per dataset: training, validation, and test splits.

Dataset	Original	Training	Testing
E1	960×540	200×120	200×120
E2	960×540	960×540	960×540
E3	960×540	-	960×540
A1	1280×720	-	1280×720

Table 3: Image resolution per dataset: original, used in training, and used in testing / thumbnail selection.

The **Eliteserien** dataset [36] consists of 300 clips of goal events scored in the Norwegian Eliteserien. Most of these clips start 25 seconds before the goal event and end 50 seconds after the event. **Eliteserien for logo detection (E1):** For logo detection, we use the annotations made in [36] (logo images from the Norwegian Eliteserien), and augment these with logos from the English Premier League [7]. Overall, this dataset contains 1,024 images classified as logos and 7,024 images classified as background. **Eliteserien for close-up shot detection (E2):** For close-up shot detection, we create an image dataset containing frames that are extracted from the Eliteserien dataset. This dataset consists of 2 classes, "close-up shot" and "no-close-up shot". Images classified as close-up shots often contain players and managers, where others include wide views of the pitch, spectators, etc. Shots of spectators celebrating are often not as close as those of players celebrating. Blurriness or image noise are not taken into account when considering which class an image belongs to. Overall, this dataset contains 702 images classified as close-up shots and 968 images classified as no-close-up shots. **Eliteserien for face detection (E3):** For face detection, we create an image dataset containing frames extracted from the Eliteserien dataset, consisting of 2 parts: "Various" and "Audience". The "Various" part contains 56 images with one or more faces, and 33 images without any faces. The images not containing faces are either long distance shots, meaning the faces are small, or have heads turned away from the camera. The second part of the dataset, "Audience", is comprised of images showing the audience. It has a total of 32 images. The distance from the camera to the audience varies, but the images mostly contain faces that are smaller than the faces appearing in the "Various" dataset. The second part of the dataset is generated for the purpose of determining if a model is good at detecting small faces.

The **Allsvenskan** dataset consists of goal clips from the Swedish Allsvenskan. This dataset is similar to the Eliteserien, but there

[5]The reason we create our own datasets is that each module in our pipeline requires different ground truth labels (e.g., close-up shot vs. long distance shot for one module, logo vs. no-logo for another). Therefore, multiple datasets, or a large dataset with multiple ground truth dimensions is needed. However, there is no publicly available dataset we can use for all modules in our pipeline.

[6]As our pipeline is configurable, different image resolutions can be used for training and testing (different from the original assets and/or different between training and testing). The influence of resolution ("scale") is further investigated in Tables 6 and 8.

is more variation in video clip length. Most of the clips start approximately 25 seconds before the goal event and end about 60-80 seconds after the goal event. There are 233 clips in the dataset.
Allsvenskan for logo detection (A1): For testing the logo detection module, we annotate a number of video clips from the Allsvenskan dataset. Overall, this dataset contains 180 images, where 69 images contain a logo and 111 images do not contain a logo. We do not use this dataset for training, but only for testing how the logo detection module performs on completely unseen data (see Section 4.3).

4.2 Implementation Details

Our automatic thumbnail selection pipeline was implemented using Python `v3.9.7`, Tensorflow `v2.6.0`, Keras `v2.6.0`, cv2 `v4.5.3`, Dlib `v19.22.1`, mtcnn `v0.1.0` and Imquality `v1.2.7` on a DGX-2 server. This server is well suited for heavy computational and memory operations. However, it is not necessary to use a machine that is exceptionally suited for heavy operations to run the pipeline. The image classifier model by Surma [32] which is used for logo detection and close-up shot detection was trained on the same server, with the same versions of Keras and Tensorflow along with Matplotlib `v3.4.3`, Livelossplot `v0.5.4` and Efficientnet `v1.1.1`. The codebase for our pipeline is publicly accessible as an open-source software repository under[7], and further artifact details are provided in the reproducibility Appendix. A demonstration of our pipeline is presented in [12].

The datasets for logo detection and close-up shot detection are split into training, validation and test sets. The distribution of the samples are presented in Table 2. To evaluate the performance of the classifier-based modules in our pipeline (logo detection, close-up shot detection, face detection), we use accuracy, precision, and recall as metrics. For the image quality prediction module, we use the BRISQUE score, which is a score indicating image quality. A smaller score indicates better perceptual quality [22].

4.3 Logo Detection Performance

We train the Surma [32] model on the E1 dataset. The results are displayed in Table 4, where the first row corresponds to a threshold value of 0.5 (default value). The model achieves an accuracy of 0.994 on the E1 test split. Upon inspection, we see that most of the "no-logo" images have a probability score below 0.01, indicating that the model strongly (and correctly) identifies that the image does not contain a logo, most of the time. The precision is higher than the recall, indicating that there are more false negatives than false positives. To make the model have higher recall and maybe higher overall accuracy, it is possible to consider adjusting the threshold value. It would be reasonable to assume that the accuracy of the model would be improved by lowering the threshold value. For instance, an image with over 0.1 probability is likely to contain a logo, so we test how the threshold value influences the accuracy, using 0.1 and 0.05 in addition to the default value of 0.5. The results for a threshold value of 0.1 are given in the second row of Table 4. Even though the threshold value is changed by a large percentage from 0.5 to 0.1, the accuracy of the model is not affected considerably. Precision is lower, as the number of false positives increases, but the recall is improved. We argue that having high recall is more important than having high precision, as classifying images which do not actually contain a logo to contain a logo (false positive) causes the pipeline to be more conservative, and possibly drop more thumbnail candidates, rather than cause it to allow for rule 4 in Table 1 to be broken. The third row of Table 4 displays the results for an even lower threshold value, namely 0.05. The recall remains the same, but the precision is worse. We therefore conclude that adjusting the threshold value further than 0.1 is not necessary for better accuracy.

Dataset	Threshold	Accuracy	Precision	Recall
Eliteserien (E1)	0.5	0.994	0.986	0.969
	0.1	0.995	0.969	0.996
	0.05	0.993	0.952	0.996
Allsvenskan (A1)	0.5	0.717	1.000	0.261
	0.1	0.761	1.000	0.377
	0.05	0.794	1.000	0.464

Table 4: Performance of the logo detection module using the model by Surma [32], on the test splits from the Eliteserien (E1) and Allsvenskan (A1) datasets containing 1611 **and** 180 **images, respectively.**

Overall, from the first part of Table 4, we see that the logo detection module performs adequately well on the Eliteserien dataset (E1 test split), as it has been trained on a part of this dataset (E1 training split). However, for this module to be used as part of an automatic thumbnail selection pipeline which is generalizable across different datasets (and correspondingly, different soccer leagues), we would like to see if it performs adequately for different datasets as well. For this purpose, we use the unseen Allsvenskan dataset (A1 test split). The results are displayed in the second part of Table 4. On the Allsvenskan dataset, the model is able to detect fewer than half of the logos. The images not containing a logo are all predicted correctly, irrespective of the threshold value. The accuracy of the model for the Allsvenskan set is not as high as for the Eliteserien dataset, where it was 0.99. However, it did not predict any false positives. We observe that a change in the threshold value from 0.5 to 0.1 improves performance for the Allsvenskan dataset as well. The precision is 100% for both cases, but the recall is improved for 0.1 as the threshold value. As we are interested in improving the recall, we further test with 0.05 as for the Eliteserien dataset, and see that there actually is an improvement when the threshold is lowered further. However, since this dataset is smaller than the Eliteserien dataset, we have lower confidence in this conclusion and resolve to use a threshold value of 0.1 in the final pipeline.

4.4 Close-up Shot Detection Performance

The close-up shot detection module is trained on the E2 dataset consisting of "close-up shot" and "no close-up shot" classes. The binary image classification by Surma [32] originally yields an accuracy of 0.82 on the test split when the probability threshold is set to 0.5. For the close-up shot detection task, we would like to prioritize precision over recall, because there is a relatively high chance that false positives can end up being the final thumbnail. Increasing the precision will prevent false positives from being prioritized. We test with different threshold values to observe the changes in the performance of the close-up shot detection module, the results are

[7] https://github.com/simula/host-ats

displayed in Table 5. Based on the trade-off between precision and accuracy, we resolve to use a threshold of 0.75 in the final pipeline.

Dataset	Threshold	Accuracy	Precision	Recall
Eliteserien (E2)	0.8	0.761	1.000	0.448
	0.75	0.866	0.955	0.724
	0.7	0.881	0.920	0.793
	0.6	0.836	0.750	0.931
	0.5	0.821	0.730	0.931

Table 5: Performance of the close-up shot detection module using the model by Surma [32], on the test split from the Eliteserien (E2) dataset containing 67 images.

4.5 Face Detection Performance

The face detection module can be run with different face detection models. In this section, we consider Dlib [15], MTCNN [41], and Haar cascade [39]. According to Agarwal [1], Haar cascade is faster but less accurate, where Dlib is slower but more accurate. To evaluate the performance of the module, we use precision achieved on the E3 dataset, using the original versions of images ("Full" scale) and 50% down-scaled[8] versions ("Half" scale). Table 6 presents the results in terms of precision and processing time per image.

Dlib: From the "Various" set of 89 images, the Dlib model detected 55 faces in total when the images were full-scaled. The Dlib model had 53 face detections on the same images when half-scaled. There were a few false positive face detections on the "Various" set, where 2 false detections were made on full-scaled and 1 on half-scaled. Both false detections were on a face-shaped skeleton logo. On the "Audience" set, the Dlib had a total of 153 detections on full-scaled and only 2 on half-scaled. A false positive face detection appeared here as well that was a logo with a face of a dog. The trend for the Dlib when doing face detection on half-scaled images was that it could not detect smaller faces. When half-scaling the images, the faces are covering half the amount of pixels compared to the original. This makes the Dlib overlook all the smaller faces.

MTCNN: Overall, MTCNN has higher recall than the other models for all classes on the dataset. The precision on the "Various" dataset is better than Haar cascade, but not as good as Dlib. The model is the best at detecting smaller faces, as it has the highest recall on the "Audience" set. It is also able to detect faces from the side (at a 90 degree angle). However, the mean processing time per image for MTCNN is $0.96s$ on half-scale images and $1.53s$ on full-scale images. This is about 19 times slower than Haar cascade and 3 times slower than Dlib.

Haar cascade: On the "Audience" set, Haar cascade has comparable precision to the other models, and higher recall than Dlib. But, it has significantly lower precision on the "Various" dataset. It detects more faces than Dlib, but not as much as MTCNN. The model is prone to false positives: on full-scaled images, 39% of the detections are false.

Final pipeline: After evaluating different models, we try to make a decision for the model to use in our final pipeline. In the context of automatic thumbnail selection for soccer video clips, we prioritize high precision and speed. The reason we prioritize precision over recall for the face detection task is because faces appear often and redundantly in soccer videos, so it is not essential that our model manages to detect *all* faces. However, it is essential that the model be certain *when* it does detect a face, because there is a higher probability that this image will be selected as the final thumbnail, according to the prioritization rules shown in Figure 2. Regarding speed, we see that a clear benefit of down-scaling is the reduction in processing time. For 50% down-scaling, up to $0.2s$ can be saved per image. For instance, if the pipeline is running face detection on 7 frames from a single video clip, 50% down-scaling decreases the time required by the face detection module using the Dlib model from $3.9s$ to $2.5s$.

Model	Image Scale	Precision (Various)	Precision (Audience)	Time per Image
Dlib	Full	0.96	0.99	$0.55s$
	Half	0.98	0.50	$0.36s$
MTCNN	Full	0.85	0.94	$1.53s$
	Half	0.91	0.93	$0.96s$
Haar	Full	0.61	0.91	$0.08s$
	Half	0.74	0.94	$0.05s$

Table 6: Performance of the face detection module using different models (Dlib, MTCNN, and Haar cascade) and different image scales (full size 960×540 and half size 480×270), on the test splits from the "Various" and "Audience" datasets.

MTCNN has the highest recall and the second highest precision on the "Various" set. It can also detect smaller faces, and faces from a side angle unlike the other models. However, MTCNN uses significantly more time than the other models, and the absolute time it takes for going through, e.g., 7 images per video clip is not feasible for our context. Haar cascade has performed well on detecting small faces, and is the fastest model (can process a video 7 times faster than the Dlib for instance). However, it tends to have many false positives. Dlib has the highest precision on the "Various" set and it is faster while processing down-scaled images. The disadvantage is that it cannot detect smaller faces, but we consider the ability to detect bigger faces to be more important.

We therefore resolve to use the Dlib model as the default model in the face detection module in our pipeline, along with 50% down-scaling of the video frames. The MTCNN model is added as a configurable (non-default) option. As the Haar cascade is very fast compared to Dlib and MTCNN, it is also added as a configurable option. If the automatic thumbnail selector is intended to process a large number of video clips in near-real-time, this model might be more suitable, as it allows for prioritizing speed over precision.

4.6 Image Quality Prediction

The last step in our pipeline comprises image quality analysis on the thumbnail candidates, in the order that the images appear after priority assignment in the previous step, followed by filtering and final thumbnail selection. We run the model by Ocampo [25] for image quality prediction, which is based on model proposed by Jongyoo [14]. The model outputs a BRISQUE score, where the lower the score, the better the predicted quality of the image. The image quality predictor is tested on 65 frames from a single video clip from the Eliteserien dataset.

Influence of compression: While testing the model, we also investigate if the model predicts lower quality when the input image has been saved with lower quality using the imwrite function

[8] In this context, down-scaling refers to the reduction in the resolution of (and correspondingly, number of pixels in) the images, and not the encoding quality.

from cv2. With the imwrite function, it is possible to choose the percentage of how much data from the image is going to be preserved. From Table 7, it is possible to see the accuracy of the model for predicting lower quality on actual lower quality images. For the test set of 65 images, it did not achieve 95% accuracy before the chosen image quality was about 50%. Another relevant aspect is how fast the image quality predictor model processes the images when they contain less data. The model by Ocampo has a tendency to consume a lot of time for processing a single image. From Table 7 it is possible to see the mean processing time per image with respect to preserved quality. Preserving less data on an image does not seem to make the image faster to process for the image quality predictor model, even though the storage size is reduced to a great extent.

Preserved Quality	Accuracy	Mean Processing Time	Mean Size
100%	NA	2.95*s*	128 KB
70%	0.72	2.82*s*	51 KB
60%	0.89	2.72*s*	42 KB
50%	0.95	2.65*s*	35 KB
40%	0.97	2.65*s*	32 KB
5%	1.00	2.28*s*	13 KB

Table 7: Influence of compression: accuracy of the Ocampo model [25] in predicting if the image has lower quality, along with mean processing time per image and mean storage size per image, with respect to preserved image quality.

Influence of down-scaling: What seems to reduce the processing time is reducing the scale of the image. The intermediary conclusion is that processing time is dependent on the amount of pixels and not necessarily the amount of data the image contains. We have tested what the image quality predictor [25] predicts for different degrees of down-scaling (Table 8). This is to know how the model behaves when it receives a down-scaled image, as we want the processing of an image to be fast in our pipeline. However, even though the processing might be faster, we want to make sure that the model is able to predict on down-scaled images with similar performance compared to the images in original scale. The output thumbnail of our pipeline can be the original or down-scaled version of the image. In the latter case, the BRISQUE score used in the final step will be referring to the down-scaled version of the image.

Image Scale	Accuracy	Mean Processing Time	Mean Size
960×540	NA	2.95*s*	128 KB
800×450	0.88	1.63*s*	42 KB
600×338	0.86	1.16*s*	27 KB
500×281	0.84	0.73*s*	21 KB
300×169	0.92	0.34*s*	9 KB
100×56	0.85	0.06*s*	1.9 KB

Table 8: Influence of down-scaling: accuracy of the Ocampo model [25] in predicting if the down-scaled image has better quality than the original image with 960×540 resolution, along with mean processing time per image and mean storage size per image, with respect to image scale.

Assessing the image quality predictor, many of the images receiving a high score (low quality) are often images that are blurry, noisy and even with generic graphics. Ranking the images with lower scores does not seem necessary as an image with 10 in score can look as good as an image with 30 in score to the human eye. The difference is not significant enough as we want to take the runtime into account as well. Eliminating the undesired images altogether seems to be the most helpful course of action with this model.

4.7 Complexity Analysis

In order to evaluate the complexity of our pipeline, we refer to 3 metrics. The **computational requirements** referring to hardware and software requirements for the complete end-to-end execution of the pipeline were already discussed in Section 4.2. The **model size** refers to the size of the ML models we use in terms of storage. Table 9 presents respective sizes of various models of the pipeline on disk (in terms of storage). Finally, the **execution time** refers to the duration it takes for certain components in the pipeline, or the pipeline itself, to perform their operations. For example, the time it takes for the pipeline to select a thumbnail after receiving a video as input is an aspect to consider when considering the efficiency of the pipeline. There have been certain decisions made during the implementation of the pipeline rendering it more time-efficient, but less thorough in search for a good thumbnail. Using the image quality predictor BRISQUE [25] takes about 1 second per image. If we want to receive an image quality score for all frames from the video clip, it could take more time. However, it is ideal if the complete pipeline does not use more than a few seconds to select a thumbnail for a given video clip. Table 11 presents the average execution time per video clip for each module in the pipeline, as well as the overall pipeline in an end-to-end fashion. Here, we use 37 clips from the Allsvenskan dataset, with an average duration of 77 seconds. As a benchmark for execution time, we run the state-of the-art thumbnail selector called Hecate from Song et al. [31] that has an average end-to-end execution time of 3.4*s*. Running our proposed pipeline on the same video clips uses minimum 3.8*s* as shown in Table 11.

Module	Model	Size
Logo detection	Surma [32]	10530KB
Close-up shot detection	Surma [32]	10530KB
Face detection	Dlib [15]	7365KB
	MTCNN [41]	2256KB
	Haar [39]	930KB
Image quality prediction	Ocampo [25]	147KB

Table 9: Complexity analysis: model size on disk (all models trained on the respective Eliteserien dataset).

4.8 Comparison with a State-of-the-Art Model

In order to benchmark the actual thumbnail selections by our pipeline, we present a visual comparison with the state-of-the-art thumbnail selector Hecate from Song et al. [31] in Table 10 for 4 sample video clips[9]. Hecate does not prioritize close-up shots of people, or images that may be more temporally-relevant (e.g., frames closer to a certain event annotation). Therefore, although the selected images might score well on objective metrics such as blur and darkness, they do not appear relevant, i.e., the model does not necessarily capture the semantics of a particular event. For example, Hecate often selects far away shots with no indication of

[9]Functional code could not be obtained for similar generic thumbnail selectors Vasudevan et al. [37] from https://github.com/arunbalajeev/query-video-summary and Agrawal from https://github.com/sumeetag/thumbnails-for-videos.

the goal event, compared to the close-up shots of the ball inside the goal or players cheering as selected by our proposed pipeline.

Song et al. (Hecate)	Proposed (HOST-ATS)
	
	
	
	

Table 10: Comparison with the state-of-the art thumbnail selector Hecate from Song et al. [31].

5 SUBJECTIVE EVALUATION

5.1 User Study Framework

`Huldra` is an open-source framework for collecting crowdsourced feedback on multimedia assets [10]. This framework allows for the collection of participant responses in a storage bucket hosted on the cloud, from where they can be retrieved in real-time by survey organizers, using credentials, immediately after the first interaction of each participant. As part of our automatic thumbnail selection system, we customize the Huldra framework and call it HOST-ATS. Figure 3 presents screenshots of the main pages of HOST-ATS.

The survey begins with the home page (Figure 3a) which allows users who already have a universally unique identifier (UUID) to complete their responses if they have closed the browser involuntarily, or decided to continue later. In both cases, their information remains saved in the browser's local storage. However, if the participant does not have a UUID, they must complete a registration form (Figure 3b) where we ask for participant information regarding age, gender, video editing experience, and soccer fandom (mandatory), as well as a free form text field if the participant has other relevant comments to add (optional). Details regarding the registration page input fields are given in [11]. After successful login or registration, the user is redirected to the background page (Figure 3c) which introduces the context of the study and presents directions for use with a simple figure. The core of the framework lies in the ranking of multimedia assets. This functionality is provided by the case page

Figure 3: Screenshots from the HOST-ATS user study.

(Figure 3d) which is composed of 3 vertical columns as follows: The left column presents a sample video clip showing a goal event. We use the npm package React player [6] for playback, which offers an off-the-shelf component for playing a variety of multimedia. The middle column presents two alternative thumbnails for the video clip. Both of which could be viewed larger if needed. The user ranks the alternatives simply by clicking on one of the thumbnails. Once a thumbnail is clicked, it is displayed immediately on the top in the right column (Figure 3e).

In order to have complete and consistent responses for our study, we make it mandatory to rank a case before proceeding to the next. Users can later go backwards and revisit their answers or change them. After finishing the ranking, the participants are invited to fill out a feedback form (Figure 3f) about the aspects that they deem important in a thumbnail. They can mark from a list of alternatives, as well as suggest other facets. We also give them the option to provide additional comments and feedback in a text field input.

We use the HOST-ATS framework to run 2 user study iterations, the first including 9 cases, and the second including 13 cases. In each case, participants compare 2 alternative thumbnail selection methods. Table 12 presents the investigation aspects relevant to the alternative thumbnails for each case. The selection of cases allows us to compare among different thumbnail selection methods, as well as to gain deeper insights into viewer expectations from thumbnails, which will potentially help us improve our framework (e.g., rules and assumptions described in Section 3 and Table 1) as future work.

5.2 User Study Findings

Figures 4 and 5 present the overall results from the iterations of our user study, in terms of the number of votes for different options per case, where "HOST-ATS" indicates the thumbnail selection by

Step 1: Pre-processing		Step 2: Content Analysis and Priority Assignment					Step 3: IQA	Overall
# frames	Frame extraction	Loading models	Logo detection	Close-up shot detection	Face detection		Quality prediction	Total
50	1.767s	0.064s	0.249s	0.223s	Dlib	2.669s	1.156s	6.221s
					Haar	0.288s		3.805s
100	1.935s	0.064s	0.423s	0.387s	Dlib	5.383s	1.429s	9.449s
					Haar	0.572s		4.917s

Table 11: Complexity analysis: average execution time per clip for each module in the framework, for 37 video clips from the Allsvenskan dataset (average video clip duration: 77s). All frames are 50% down-scaled. "Loading time" refers to the loading of the logo detection and close-up shot detection models.

Table 12: Investigation aspects for cases in the user study. Evaluated thumbnail selection alternatives are HOST-ATS (H), manual selection (M), and static selection (S).

Case	Investigation	Method		
		H	M	S
70396	Players celebrating: close-up vs. medium distance	✓	✓	
75861	Player close-up vs. player celebration	✓	✓	
76465	Players celebrating: scorer visible vs. not visible	✓	✓	
80540	Player close-up vs. player celebration	✓	✓	
73375	Player close-up vs. players celebrating	✓		✓
77648	Players celebrating: scorer visible vs. not visible	✓		✓
77651	Players celebrating vs. audience	✓		✓
79994	Players celebrating vs. goal shot	✓		✓
80077	Players celebrating vs. attack action	✓		✓
75143	Player close-up vs. players celebrating	✓		✓
76358	Player close-up vs. attack action	✓		✓
76394	Long distance vs. close-up shot	✓		✓
76407	Player close-up vs. player celebration	✓		✓
76422	Player close-up vs. long distance shot	✓		✓
76821	Player close-up vs. audience	✓		✓
78438	Players celebrating vs. audience	✓		✓
78888	Player close-up vs. players celebrating	✓		✓
79285	Players celebrating vs. long distance shot	✓		✓
79588	Player close-up: players from different teams	✓		✓
79606	Player close-up: front view vs. back view	✓		✓
79789	Player close-up vs. players celebrating	✓		✓
80136	Long distance shot vs. audience	✓		✓

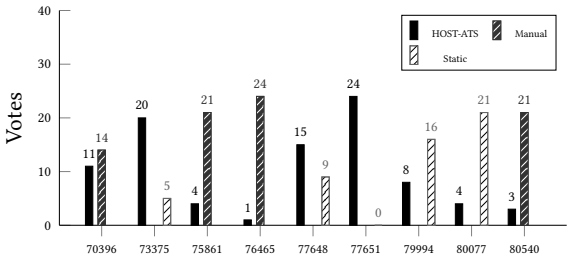


Figure 4: Case answers for the first user study.

our framework, "Static" indicates the thumbnail selection method wherein a frame from the video clip at a fixed timestamp (e.g., 5 seconds after playback start) is assigned as the thumbnail, and "Manual" indicates manual thumbnail selection undertaken by human operators. The study received 27 responses for the first iteration, and 21 responses for the second iteration, with a total of 42 responses remaining after filtering. We make the following general observations.

Manual selection is preferred over HOST-ATS (4 out of 4 pairwise comparisons), i.e., a human operator can select better thumbnails than our automated system. For case `70396`, it is a close call, where the medium distance shot of player celebration is preferred over the close-up shot of player celebration. For case `75861`, manual selection showing multiple players is preferred by a larger margin against a single player close-up. For case `76465`, manual selection where the scorer is visible in front view is preferred by a larger margin against a celebration where the scorer is only visible in back view. For case `80540`, manual selection showing multiple players in celebration from the back view is preferred by a larger margin against a close-up view of players where a face is visible more distinctly but there is no celebration action. These are very valuable insights for us to improve our thumbnail selection rules (Table 1) with more details regarding the joint evaluation of face detection and close-up shot detection results, as well as including context-aware priorities with respect to celebrations and action content (e.g., crowded celebration scene preferred over frontal or side view of fewer players with less action content).

HOST-ATS outperforms static selection in most cases (11 out of 18 pairwise comparisons). For cases `73375`, `77648`, `77651`, `76407`, `76422`, `76821`, `78438`, `78888`, `79285`, `79588`, and `79606`, we see that the rules enforcing higher image quality, preference of close-up shots to long distance shots, and frontal face view are in line with viewer preferences. Cases where the static selection is preferred include the following: `79994` (clear view of the goal shot, despite being medium/long distance preferred over blurry image of players celebrating), `80077` (attack scene preferred over celebration scene with partial obstruction in image), `75143` (goal scorer's single celebration, despite blur in image, preferred over multiple-player celebration where scorer is not visible), `76358` (attack action from medium distance preferred over close-up shot of player), `76394` (long distance shot of field preferred over close-up shot with partial obstruction in image), `79789` (goal scorer from back view preferred over multiple-player celebration with slight blur), and `80136` (long distance shot of the field preferred over blurry audience celebration). These cases show that additional rule adjustments are needed to take image blurriness, partial obstructions, action content, and celebration context (e.g., goal scorer) into account.

Viewers consider high image quality, player faces, and action content as the most important aspects of a thumbnail, followed by close-up shots, cheering, and the absence of logo transitions (Figure 6). These confirm our initial thumbnail selection rules, and give further insights regarding possible additional rules (e.g., detection of action content and cheering context).

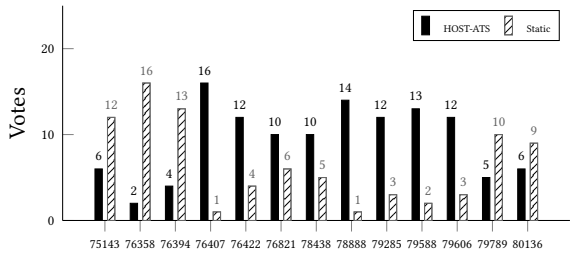


Figure 5: Case answers for the second user study.

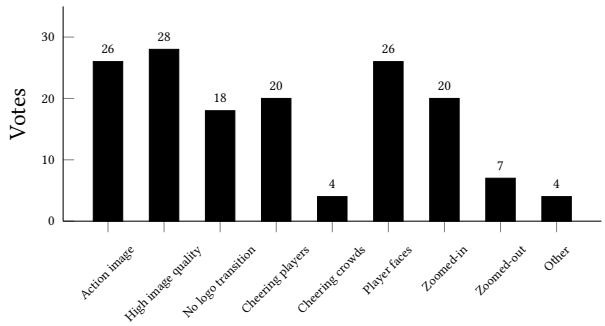


Figure 6: Feedback form answers for question F1 [11] (first and second user study combined).

6 DISCUSSION

6.1 Blur Detection as a Framework Component

There can be a lot of motion in soccer games, resulting in video frames being blurry. When selecting a frame from a video clip as a thumbnail, it is important to avoid images that are too blurry, both for aesthetic reasons and also for keeping the information content and representativity of the thumbnail high. It should be possible to see what is happening in the image, at least in the foreground.

Running blur detection on frames from soccer videos has its challenges. Firstly, blur detection models do not necessarily capture the essence of what is really perceived as blur by humans, within a certain context. There might be objects or regions in a frame that are more important to the human eye than others. For instance, a frame where the celebrating soccer player is clear but the background is blurry might be discarded falsely by a blur detection model due to supposed high presence of blur, whereas the frame could actually be considered non-blurry, in terms of the object of interest in the scene. Secondly, blur scores for frames should be considered relatively to other frames from the same video and not in absolute terms. Overall image quality can vary from video to video, which leads to a different viewer tolerance for what is perceived as a blurry frame for a given video clip.

We integrate and experiment with alternative blur detection models in our automatic thumbnail selection pipeline. One of these is the Laplacian operator from OpenCV Library [3]. Figure 7 presents the blur scores according to [3] of different frames from the same video. In Figure 7d, we can see a close-up shot of a player which is clear, but the background is blurry. This frame could have served as a perfectly adequate thumbnail, but scores a high presence of blur, relative to the rest of the frames tested. In contrast, the frames in Figures 7a and 7b) score a low presence of blur. These frames have a clear background but the players in the foreground are a bit blurry as they are in motion. This discrepancy demonstrates the first challenge mentioned above, namely that the blur detection model predicts higher blur the larger the total area covered by blur is, irrespective of regions of interest or layout. Another example for this challenge is given by Figures 7e and 7f. These frames are from the same second and the same camera. The player appearing in Figure 7e is less blurry than the players appearing in Figure 7f. However, the background is more clear in the latter compared to the former. Yet another challenge is to identify why seemingly similar scenes can receive different blur scores, such as Figure 7c receiving a score which indicates much more blur than Figures 7a and 7b.

By using a blur detection module in our pipeline, we would like to promote frames that look clearer, and consequently improve the thumbnail selection process. However, the frames that existing blur detection models predict as being very clear might not be what we prefer to have as thumbnails. It seems that the direct use of existing blur detection models might not be the right approach for capturing what viewers actually perceive as a degradation within the context of soccer highlight clips. Therefore, the blur detection module is an optional component in our pipeline (which can be configured to use either of 2 alternative models, or not activated at all), and a topic for further investigation.

6.2 Limitations

Our proposed automatic thumbnail selection system has a number of limitations. One of the limitations of our approach is that the thumbnails selected by our pipeline can be very similar to each other, especially for similar video clips, due to our fixed ruleset. For instance, if player scenes are prioritized over audience scenes in the ruleset, such frames will be more likely to be selected for any given video clip. Along with our insights from the user studies described in Section 5, this motivates our current work concerning a finer, more elaborate ruleset. We are currently working on moving away from a plain decision tree based approach towards a more complex approach based on custom scores for each candidate frame, considering more diverse aspects (action content, face angle, context such as celebration, etc.). Despite its good performance, our pipeline is limited in the sense that its sole focus is on goal events, not considering other soccer events such as player substitutions, yellow and red cards. Our experiments are also limited in terms of the scale of the datasets we have used for training and evaluation (which are mainly composed of 2 leagues from Norway and Sweden, with a relatively low number of usable samples in the second dataset).

6.3 Possible Improvements and Future Work

There are a number of possible improvements which constitute potential future work directions:

- **Blur detection:** As shown in the previous section, blur detection is challenging to integrate directly into an automatic thumbnail selection pipeline. In order to identify blur in line with viewer preferences, foreground-background detection

and weighted analysis using a grid to identify important pixels in an image might be required.

- **Integration of subjective findings:** It is necessary to further investigate what makes a good thumbnail for viewers (in general, for soccer events, and in particular, for goals). In this respect, we have made note of the following: action content as a higher priority over face detection, blur detection together with contextual information, relaxing of the close-up shot requirement depending on content and blur, detection of partial obstructions in thumbnail candidates.
- **Model performance:** Different models can be used in the individual framework components for logo detection, close-up shot detection, face detection, and image quality prediction, and benchmarked, similar to what we have demonstrated in Section 4 for face detection. For instance, YOLO [26] is another promising object detection model that we plan to integrate and test in future versions of our pipeline.
- **Generalizability:** Cross-dataset analysis of our proposed pipeline, involving larger open soccer datasets, can yield better insights into the generalizability of our assumptions and findings.
- **Extended complexity analysis:** Tests with different video clip durations, and even full live streams can give a better idea of the real-time performance of our pipeline.
- **Different events:** Extending our automatic thumbnail selection framework to other soccer events, such as player substitutions, yellow and red cards can lead to a more practically relevant and complete system. Developing different versions focusing on other sports is also possible.
- **Alternative pipeline:** An alternative approach to our complete pipeline would be to train an ML model on a large dataset of thumbnail images manually selected by experts, and use this model directly, in a single step, on video clips. Such an idea could be preferable to our current pipeline, which needs to use multiple modules to explicitly enforce certain rules. However, the logic behind the selection would remain black box, and the underlying rules might not be possible to explicitly document and reproduce. Secondly, there exists no big-enough dataset to train such a model yet, which might need a lot of data[10]. In this respect, there is a need for open datasets shared between the industry and academia, for facilitating research in this direction.

7 CONCLUSION

In this paper, we present an automatic thumbnail selection system for soccer videos called HOST-ATS, which uses ML to deliver representative thumbnails with high relevance to the video content and high visual quality in near real-time. HOST-ATS comprises a software framework that leverages logo detection, close-up shot detection, face detection, image quality prediction, and blur detection into an automatic thumbnail selection pipeline, and a user study framework for subjective evaluation and validation. We evaluate the proposed pipeline quantitatively using various soccer datasets, as well as qualitatively, through subjective user studies.

Our work combines various independently applicable approaches in an end-to-end system, with an overarching goal and a generalized ruleset. The modular and lightweight implementation of the thumbnail selection pipeline allows for the agile integration and benchmarking of various ML methods from literature in each module. The subjective evaluation campaign using a novel survey framework for crowdsourced feedback collection yields a number of insights into viewer preferences. The components of the HOST-ATS system, namely the pipeline (as controlled by a dashboard with a GUI) and the user study framework can be demonstrated.

Our results show that an automatic end-to-end system for the selection of thumbnails based on contextual relevance and visual quality can yield highly attractive thumbnails, and can be used in conjunction with existing soccer video production pipelines which require real-time operation. However, the results also indicate that some of our initial rules can be reconsidered and adjusted, as viewers might have different preferences based on context. Nevertheless, our proposed pipeline is shown to work as intended, following the specified rules and priorities, and can run efficiently. This work is therefore a good starting point for even better future automatic thumbnail selection systems.

(a) Blur score: 11.72 (b) Blur score: 34.24
(c) Blur score: 3.75 (d) Blur score: 1.82
(e) Blur score: 1.76 (f) Blur score: 28.65

Figure 7: Results using the OpenCV Laplacian operator [3] on selected frames (higher score indicates less blur).

ACKNOWLEDGMENTS

This research was funded by the Research Council of Norway, project number 327717 (AI-producer).

[10]There might also be a need for separate datasets, or different annotations within the same dataset, for different *events* (e.g., thumbnails that experts tend to select for goal event clips might be different from those for card or substitution event clips), and different *leagues* (different jerseys, players, etc. might confuse models trained on one league and tested on another).

REFERENCES

[1] Vardan Agarwal. 2021. *Face Detection Models: Which to Use and Why?* https://towardsdatascience.com/face-detection-models-which-to-use-and-why-d263e82c302c

[2] Allsvenskan. 2022. Highlights. https://highlights.allsvenskan.se/.

[3] Gary Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[4] Chen-Yu Chen, Jia-Ching Wang, Jhing-Fa Wang, and Yu-Hen Hu. 2008. Motion Entropy Feature and Its Applications to Event-Based Segmentation of Sports Video. *EURASIP Journal on Advances in Signal Processing* 2008 (2008). https://doi.org/10.1155/2008/460913

[5] Anthony Cioppa, Adrien Deliege, Silvio Giancola, Bernard Ghanem, Marc Van Droogenbroeck, Rikke Gade, and Thomas B. Moeslund. 2020. A Context-Aware Loss Function for Action Spotting in Soccer Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/CVPR42600.2020.01314

[6] Pete Cook. 2021. react-player. https://www.npmjs.com/package/react-player.

[7] Adrien Deliège, Anthony Cioppa, Silvio Giancola, Meisam J. Seikavandi, Jacob V. Dueholm, Kamal Nasrollahi, Bernard Ghanem, Thomas B. Moeslund, and Marc Van Droogenbroeck. 2020. SoccerNet-v2 : A Dataset and Benchmarks for Holistic Understanding of Broadcast Soccer Videos. arXiv:2011.13367 [cs.CV]

[8] Eliteserien. 2022. Highlights. https://highlights.eliteserien.no/.

[9] FIFA.com. 2018. More than half the world watched record-breaking 2018 World Cup. https://www.fifa.com/worldcup/news/more-than-half-the-world-watched-record-breaking-2018-world-cup

[10] Malek Hammou, Cise Midoglu, Steven A. Hicks, Andrea Storås, Saeed Shafiee Sabet, Inga Strümke, Michael A. Riegler, and Pål Halvorsen. 2022. Huldra: A Framework for Collecting Crowdsourced Feedback on Multimedia Assets. In *13th ACM Multimedia Systems Conference (MMSys '22), June 14–17, 2022, Athlone, Ireland*. ACM, New York, NY, USA. https://doi.org/10.1145/3524273.3532887

[11] Andreas Husa. 2022. *Automated Thumbnail Selection for Soccer Videos with Machine Learning*. Master's thesis. University of Oslo, Oslo, Norway.

[12] Andreas Husa, Cise Midoglu, Malek Hammou, Pål Halvorsen, and Michael A. Riegler. 2022. HOST-ATS: Automatic Thumbnail Selection with Dashboard-Controlled ML Pipeline and Dynamic User Survey. In *13th ACM Multimedia Systems Conference (MMSys '22), June 14–17, 2022, Athlone, Ireland*. ACM, New York, NY, USA. https://doi.org/10.1145/3524273.3532908

[13] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-Scale Video Classification with Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1725–1732. https://doi.org/10.1109/CVPR.2014.223

[14] Jongyoo Kim, Anh-Duc Nguyen, and Sanghoon Lee. 2019. Deep CNN-Based Blind Image Quality Predictor. *IEEE Transactions on Neural Networks and Learning Systems* 30, 1 (2019), 11–24. https://doi.org/10.1109/TNNLS.2018.2829819

[15] Davis King. 2021. dlib C++ Library. http://dlib.net/. Last accessed 2022-01-24.

[16] Ryan Knott. 2021. *What Are Video Thumbnails and Why Do They Matter?* https://www.techsmith.com/blog/what-are-video-thumbnails/

[17] Jacek Komorowski, Grzegorz Kurzejamski, and Grzegorz Sarwas. 2019. FootAndBall: Integrated player and ball detector. *CoRR* abs/1912.05445 (2019). arXiv:1912.05445 http://arxiv.org/abs/1912.05445

[18] Harilaos Koumaras, Georgios Gardikis, George Xilouris, Evangelos Pallis, and Anastasios Kourtis. 2006. Shot boundary detection without threshold parameters. *J. Electronic Imaging* 15 (4 2006), 020503. https://doi.org/10.1117/1.2199878

[19] Thomas J Law. 2021. The Perfect YouTube Thumbnail Size and Best Practices. https://www.oberlo.com/blog/youtube-thumbnail-size.

[20] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. 2019. BMN: Boundary-Matching Network for Temporal Action Proposal Generation. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*.

[21] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. 2018. BSN: Boundary Sensitive Network for Temporal Action Proposal Generation. In *Proceedings of the European Conference Computer Vision (ECCV)*.

[22] MATLAB. 2021. *brisque (R2021a)*. https://se.mathworks.com/help/images/ref/brisque.html

[23] Pier Luigi Mazzeo, Marco Leo, Paolo Spagnolo, and Massimiliano Nitti. 2012. Soccer Ball Detection by Comparing Different Feature Extraction Methodologies. *Advances in Artificial Intelligence* 2012 (2012), 12. https://doi.org/10.1155/2012/512159

[24] Olav Andre Nergård Rongved, Markus Stige, Steven Alexander Hicks, Vajira Lasantha Thambawita, Cise Midoglu, Evi Zouganeli, Dag Johansen, Michael Alexander Riegler, and Pål Halvorsen. 2021. Automated Event Detection and Classification in Soccer: The Potential of Using Multiple Modalities. *Machine Learning and Knowledge Extraction* 3, 4 (2021), 1030–1054. https://doi.org/10.3390/make3040051

[25] Ricardo Ocampo. 2021. *Deep CNN-Based Blind Image Quality Predictor in Python*. https://towardsdatascience.com/deep-image-quality-assessment-with-tensorflow-2-0-69ed8c32f195

[26] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. 2015. You Only Look Once: Unified, Real-Time Object Detection. *CoRR* abs/1506.02640 (2015). arXiv:1506.02640 http://arxiv.org/abs/1506.02640

[27] Olav A. Nergård Rongved, Steven A. Hicks, Vajira Thambawita, Håkon K. Stensland, Evi Zouganeli, Dag Johansen, Cise Midoglu, Michael A. Riegler, and Pål Halvorsen. 2021. Using 3D Convolutional Neural Networks for Real-time Detection of Soccer Events. *International Journal of Semantic Computing* 15, 02 (2021), 161–187. https://doi.org/10.1142/S1793351X2140002X

[28] Olav A. Nergård Rongved, Steven A. Hicks, Vajira Thambawita, Håkon K. Stensland, Evi Zouganeli, Dag Johansen, Michael A. Riegler, and Pål Halvorsen. 2020. Real-Time Detection of Events in Soccer Videos using 3D Convolutional Neural Networks. In *Proceedings of the IEEE International Symposium on Multimedia (ISM)*. 135–144. https://doi.org/10.1109/ISM.2020.00030

[29] Adrian Rosebrock. 2021. *OpenCV Haar Cascades*. https://www.pyimagesearch.com/2021/04/12/opencv-haar-cascades/

[30] Karen Simonyan and Andrew Zisserman. 2014. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*. 568–576.

[31] Yale Song, Miriam Redi, Jordi Vallmitjana, and Alejandro Jaimes. 2016. To Click or Not To Click: Automatic Selection of Beautiful Thumbnails from Videos. arXiv:1609.01388 [cs.MM]

[32] Greg Surma. 2018. *Image Classifier - Cats vs Dogs*. https://gsurma.medium.com/image-classifier-cats-vs-dogs-with-convolutional-neural-networks-cnns-and-google-colabs-4e9af21ae7a8

[33] Dian Tjondronegoro, Yi-Ping Phoebe Chen, and Binh Pham. 2003. Sports video summarization using highlights and play-breaks. In *Proceedings of ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR)*. 201–208. https://doi.org/10.1145/973264.973296

[34] Torrens University Australia. 2020. Why the Sports Industry is Booming in 2020 (and which key players are driving growth). https://www.torrens.edu.au/blog/why-sports-industry-is-booming-in-2020-which-key-players-driving-growth

[35] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. 2018. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 6450–6459. https://doi.org/10.1109/CVPR.2018.00675

[36] Joakim Olav Valand, Haris Kadragic, Steven Alexander Hicks, Vajira Lasantha Thambawita, Cise Midoglu, Tomas Kupka, Dag Johansen, Michael Alexander Riegler, and Pål Halvorsen. 2021. AI-Based Video Clipping of Soccer Events. *Machine Learning and Knowledge Extraction* 3, 4 (2021), 990–1008. https://doi.org/10.3390/make3040049

[37] Arun Balajee Vasudevan, Michael Gygli, Anna Volokitin, and Luc Van Gool. 2017. Query-adaptive Video Summarization via Quality-aware Relevance Estimation. arXiv:1705.00581 [cs.CV]

[38] Vimeo Livestream Blog. 2022. Streaming Stats - 47 Must-Know Live Video Streaming Statistics. https://livestream.com/blog/62-must-know-stats-live-video-streaming.

[39] P. Viola and M. Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1. I–I. https://doi.org/10.1109/CVPR.2001.990517

[40] Hossam M. Zawbaa, Nashwa El-Bendary, Aboul Ella Hassanien, and Ajith Abraham. 2011. SVM-based soccer video summarization system. In *Proceedings of the World Congress on Nature and Biologically Inspired Computing*. 7–11. https://doi.org/10.1109/NaBIC.2011.6089409

[41] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. 2016. Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks. *CoRR* abs/1604.02878 (2016). arXiv:1604.02878 http://arxiv.org/abs/1604.02878

[42] Matko Šarić, Dujmić Hrvoje, and Baričević Domagoj. 2008. Shot Boundary Detection in Soccer Video using Twin-comparison Algorithm and Dominant Color Region. *Journal of Information and Organizational Sciences* 32 (06 2008).

A SUPPLEMENTARY MATERIAL RELATED TO THE USER STUDY

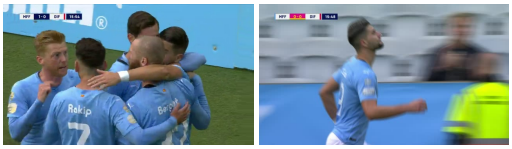
Figure 1 presents the alternative thumbnails used for the cases in the first iteration of the user study.

(a) Case 70396 - HOST-ATS (b) Case 70396 - Manual

(c) Case 75861 - HOST-ATS (d) Case 75861 - Manual

(e) Case 76465 - HOST-ATS (f) Case 76465 - Manual

(g) Case 80540 - HOST-ATS (h) Case 80540 - Manual



(i) Case 73375 - HOST-ATS (j) Case 73375 - Static

(k) Case 77648 - HOST-ATS (l) Case 77648 - Static

(m) Case 77651 - HOST-ATS (n) Case 77651 - Static

(o) Case 79994 - HOST-ATS (p) Case 79994 - Static

(q) Case 80077 - HOST-ATS (r) Case 80077 - Static

Figure 1: Cases from the first iteration of the user study.

Figure 2 presents the alternative thumbnails used for the cases in the second iteration of the user study.

(a) Case 75143 - HOST-ATS (b) Case 75143 - Static

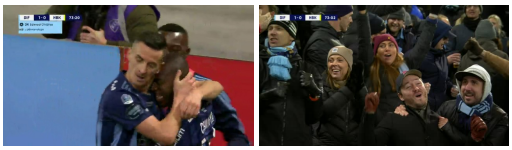
(c) Case 76358 - HOST-ATS (d) Case 76358 - Static

(e) Case 76394 - HOST-ATS (f) Case 76394 - Static

(g) Case 76407 - HOST-ATS (h) Case 76407 - Static

(i) Case 76422 - HOST-ATS (j) Case 76422 - Static

(k) Case 76821 - HOST-ATS (l) Case 76821 - Static

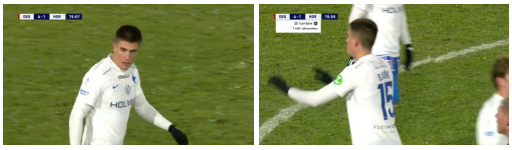


(m) Case 78438 - HOST-ATS (n) Case 78438 - Static

(o) Case 78888 - HOST-ATS (p) Case 78888 - Static

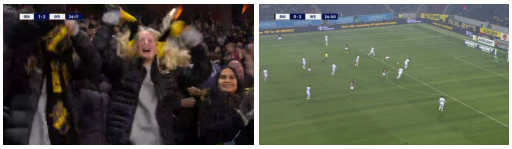
(q) Case 79285 - HOST-ATS (r) Case 79285 - Static

(s) Case 79588 - HOST-ATS (t) Case 79588 - Static



(u) Case 79606 - HOST-ATS (v) Case 79606 - Static

(w) Case 79789 - HOST-ATS (x) Case 79789 - Static



(y) Case 80136 - HOST-ATS (z) Case 80136 - Static

Figure 2: Cases from the second iteration of the user study.

Table 1 presents the list of registration form fields and feedback questions used in the study.

Table 1: Registration form fields and feedback form questions in the HOST-ATS user study.

Page	ID	Type	Mandatory	Question	Details
Registration	R1	Multiple choice	✓	Do you consider yourself a soccer fan?	Options: - Yes - No
	R2	Multiple choice	✓	Do you have experience with video editing?	Options: - Yes (professionally) - Yes - No
	R3	Multiple choice	✓	What is your gender?	Options: - Female - Male - Non-binary
	R4	Multiple choice	✓	What is your age?	Options: - Below 18 - 18-29 - 30-39 - 40-49 - 50-59 - 60-69 - 70 and above
	R5	Free form		Comments	
Feedback	F1	Tickboxes		In your opinion, what aspects are important in a thumbnail?	Options: - Close-up / zoomed-in image - Wide-area / zoomed-out image - See faces of the players - High image quality (no blur, etc.) - No logo transition in the image - Cheering crowds - Cheering players - Action image (of the play) - Other (please specify)
	F2	Free form		Please feel free to provide additional comments and feedback.	

A. Artifact Appendix

A.1 Abstract

The artifact contains the source code for the HOST-ATS automatic thumbnail selection pipeline, tested on a Linux `Ubuntu 20.04.4` environment with `Python 3.8.5` and `pip 20.2.4`, along with installation and running instructions. Thumbnails in the HOST-ATS user study[1] cannot be reproduced since the corresponding video clips are not publicly available, however, a different set of video clips, e.g., from the SoccerNet dataset[2] can be used. The HOST-ATS public repository can be found under[3].

A.2 Artifact check-list (meta-information)

- **Run-time environment:** Ubuntu 20.04.4 with Python 3.8.5 and pip 20.2.4
- **Output:** Thumbnail image (`jpg` file).
- **How much disk space required (approximately)?:** 45KB (code), 45MB (models), 2.5MB (per video clip of 45s duration).
- **How much time is needed to prepare workflow?:** Around 4 minutes.
- **How much time is needed to complete experiments?:** Around 5s per video clip of 45s.
- **Publicly available?:** Yes.
- **Code licenses?:** MIT license.

A.3 Description

A.3.1 How delivered

The source code and instructions are available in a public software repository on GitHub[3].

A.3.2 Software dependencies

The following packages can be installed via `apt-get`:

```
ffmpeg=7:4.2.4-1ubuntu0.1
libsm6=2:1.2.3-1
libxext6=2:1.3.4-0ubuntu1
```

The following Python libraries can be installed via `pip`:

```
cmake==3.22.3
dlib==19.23.1
image-quality==1.2.7
mtcnn==0.1.1
numpy==1.22.3
opencv-python==4.5.5.64
pillow==9.0.1
scikit-image==0.18.3
tensorflow-cpu==2.8.0
```

[1] `https://host-ats.herokuapp.com`

[2] `https://www.soccer-net.org/`

[3] `https://github.com/simula/host-ats/`

A.3.3 Data sets

Video clips for which thumbnails are to be generated can be placed under the `data/videos` folder in the runtime directory. The pipeline can be run with any video (e.g., goal clips from the SoccerNet[2] dataset, highlight galleries for the Norwegian Eliteserien[4] or the Swedish Allsvenskan[5]).
Models to be used in the various modules of the pipeline can be placed under the `data/models` folder in the runtime directory. Examples include:

Corresponding Module	Model
Face detection	`haarcascade_frontalface_default.xml`[6]
Face detection	`res10_300x300_ssd_iter_140000.caffemodel`[7]
Face detection	`deploy.prototxt.txt`[8]
Logo detection	`logo_eliteserien.h5`[3]
Logo detection	`logo_soccernet.h5`[3]
Close-up shot detection	`close_up_model.h5`[3]

A.4 Installation

The software repository under[3] should be cloned, and the instructions provided in the README file should be followed. The repository includes the following:

- **README.md** provides instructions on how to run the pipeline.
- **code** folder contains the scripts to run the pipeline (either from the terminal, or as a dashboard with an interactive Graphical User Interface (GUI)).
- **data** folder for storing the input video clip(s) and the models to be used in the various modules of the pipeline.
- **results** is the default output folder, both for temporary files (e.g., frames extracted as per Step 1 of the pipeline) as well as the final output (selected thumbnail image).
- **packages.txt** includes the list of packages which can be installed via `apt-get`.
- **requirements.txt** includes the list of Python libraries which can be installed via `pip`.

A.5 Evaluation and expected result

Once all the dependencies are installed and the script `create_thumbnail.py` is called (see Section A.6 for command line options), the pipeline is executed. Running the pipeline outputs one image file (`jpg`) per input video clip, into the `results/` folder. This is the thumbnail selected for the particular video clip. If a complete folder is specified as input, the pipeline iterates through all video files in the folder, and outputs one image per video file.

[4] `https://highlights.eliteserien.no`

[5] `https://highlights.allsvenskan.se`

[6] `https://github.com/opencv/opencv/tree/master/data/haarcascades`

[7] `https://github.com/sr6033/face-detection-with-OpenCV-and-DNN`

[8] `https://github.com/opencv/opencv/tree/master/samples/dnn/face_detector`

A.6 Experiment customization

Users can call the `create_thumbnail.py` script on the terminal with different options to adjust the configuration parameters of the pipeline, as follows. A detailed list of command line options is provided in Table 1.

```
create\_thumbnail.py
[-h]
[-LEliteserien2019 | -LSoccernet | -xl]
[-CSurma | -xc] [-IQAOcampo | -xi]
[-BSVD | -BLaplacian | -xb]
[-dlib | -haar | -mtcnn | -dnn | -xf]
[-cuthr CLOSEUPTHRESHOLD]
[-brthr BRISQUETHRESHOLD]
[-logothr LOGOTHRESHOLD]
[-svdthr SVDTHRESHOLD]
[-lapthr LAPLACIANTHRESHOLD]
[-css CUTSTARTSECONDS]
[-ces CUTENDSECONDS]
[-nfe NUMBEROFFRAMESTOEXTRACT | -fre FRAMERATETOEXTRACT | -fpse FPSEXTRACT]
[-ds DOWNSCALEPROCESSINGIMAGES]
[-dso DOWNSCALEOUTPUTIMAGE]
[-as ANNOTATIONSECOND]
[-bac BEFOREANNOTATIONSECONDSCUT]
[-aac AFTERANNOTATIONSECONDSCUT]
[-st STATICTHUMBNAILSEC]
destination
```

Command Line Option	Description
destination	Path to the input file or folder to be processed.
-h, --help	Show help message (contents of this table) and exit.
-LEliteserien2019	Surma model used for logo detection, trained on Eliteserien 2019. This is the default model.
-LSoccernet	Surma model used for logo detection, trained on SoccerNet.
-xl, --xLogoDetection	Don't run logo detection.
-CSurma	Surma model used for close-up shot detection. This is the default model.
-xc, --xCloseupDetection	Don't run close-up shot detection.
-IQAOcampo	Ocampo model used for image quality prediction. This is the default model.
-xi, --xIQA	Don't run image quality prediction.
-BSVD	SVD method used for blur detection. This is the default model.
-BLaplacian	Laplacian method used for blur detection.
-xb, --xBlurDetection	Don't run blur detection.
-dlib	Dlib model for face detection. This model is slow, but precise.
-haar	Haar model for dace detection. This model is fast, but unprecise.
-mtcnn	MTCNN model for face detection. This model is slow, but precise.
-dnn	DNN model for face detection. This model is fast and precise. This is the default model.
-xf, --xFaceDetection	Don't run face detection.
-cuthr, --closeUpThreshold	Threshold value for the close-up shot detection model. The value must be between 0 and 1. The default value is: 0.75.
-brthr, --brisqueThreshold	Threshold value for the image quality prediction model. The default value is: 35.
-logothr, --logoThreshold	Threshold value for the logo detection model. The value must be between 0 and 1. The default value is: 0.1.
-svdthr, --svdThreshold	Threshold value for the SVD blur detection. The default value is: 0.65.
-lapthr, --laplacianThreshold	Threshold value for the Laplacian blur detection. The default value is: 1000.
-css, --cutStartSeconds	Number of seconds to cut from the start of the video. These seconds of video will not be processed in the pipeline. The default value is: 0.
-ces, --cutEndSeconds	Number of seconds to cut from the end of the video. These seconds of video will not be processed in the pipeline. The default value is: 0.
-nfe, --numberOfFramesToExtract	Absolute number of frames to be extracted from the input video. The default value is: 50.
-fre, --framerateToExtract	Down-sampling ratio for determining the number of frames to be extracted from the input video.
-fpse, --fpsExtract	Number of frames per second to extract from the input video.
-ds, --downscaleProcessingImages	Down-scaling ratio for the images to be processed. The default value is: 0.5.
-dso, --downscaleOutputImage	Down-scaling ratio for the output thumbnail image. The default value is: 1.0.
-as, --annotationSecond	The second at which there is an event annotation in the video.
-bac, --beforeAnnotationSecondsCut	Seconds before the event annotation to trim from the input video during frame extraction.
-aac, --afterAnnotationSecondsCut	Seconds after the event annotation to trim from the input video during frame extraction.
-st, --staticThumbnailSec	Flag used for generating a static thumbnail from the input video. Should be followed by the second at which the frame should be captured. Running this flag ignores all other flags.

Table 1. Command line options for the script `create_thumbnail.py` which can be used to set pipeline configuration parameters.

A.2 Demo Track paper: HOST-ATS: Automatic Thumbnail Selection with Dashboard-Controlled ML Pipeline and Dynamic User Survey

Authors Andreas Husa, Cise Midoglu, Malek Hammou, Pål Halvorsen, Michael A. Riegler

Published International Conference of Multimedia Systems (MMSYS), Ireland, June 2022 [25]

Short description This paper is presenting the system HOST-ATS. A system consisting of: a dashboard-controlled ML pipeline, and a dynamic user survey. This is used in the thesis as well.

Venue The ACM Multimedia Systems Conference (MMSys) provides a forum for researchers to present and share their latest research findings in multimedia systems. MMSYS specializes in multimedia computing, covering fields such as networking, operating systems, real-time systems, databases, mobile computing, distributed systems, computer vision, and middleware communities [40].

MMSys Demo & Industry track is targeted at bridging the gap between research and industry by showcasing demonstrations of innovative multimedia concepts. Researchers and industry partners has the opportunity to share their prototypes and proog-of-concepts [40].

HOST-ATS: Automatic Thumbnail Selection with Dashboard-Controlled ML Pipeline and Dynamic User Survey

Andreas Husa
SimulaMet, Norway

Cise Midoglu
SimulaMet, Norway

Malek Hammou
SimulaMet, Norway

Pål Halvorsen*†
SimulaMet, Norway

Michael A. Riegler‡
SimulaMet, Norway

ABSTRACT

We present HOST-ATS, a holistic system for the automatic selection and evaluation of soccer video thumbnails, which is composed of a dashboard-controlled machine learning (ML) pipeline, and a dynamic user survey. The ML pipeline uses logo detection, close-up shot detection, face detection, image quality prediction, and blur detection to automatically select thumbnails from soccer videos in near real-time, and can be configured via a graphical user interface. The web-based dynamic user survey can be employed to qualitatively evaluate the thumbnails selected by the pipeline. The survey is fully configurable and easy to update via continuous integration, allowing for the dynamic aggregation of participant responses to different sets of multimedia assets. We demonstrate the configuration and execution of the ML pipeline via the custom dashboard, and the agile (re-)deployment of the user survey via Firebase and Heroku cloud service integrations, where the audience can interact with configuration parameter updates in real-time. Our experience with HOST-ATS shows that an automatic thumbnail selection system can yield highly attractive highlight clips, and can be used in conjunction with existing soccer broadcast practices in real-time.

CCS CONCEPTS

• **Computing methodologies** → **Video summarization**; Machine learning; • **Human-centered computing** → **Empirical studies in interaction design**; **User studies**; • **General and reference** → **Empirical studies**.

KEYWORDS

crowdsourcing; blur detection; dashboard; deep learning; graphical user interface; image quality; logo detection; object detection; shot boundary detection; soccer; survey; thumbnail generation; user study; video

*Also affiliated with Oslo Metropolitan University, Norway
†Also affiliated with Forzasys AS, Norway
‡Also affiliated with UIT The Artic University of Norway, Norway

MMSys '22, June 14–17, 2022, Athlone, Ireland

ACM ISBN 978-1-4503-9283-9/22/06.
https://doi.org/10.1145/3524273.3532908

ACM Reference Format:
Andreas Husa, Cise Midoglu, Malek Hammou, Pål Halvorsen, and Michael A. Riegler. 2022. HOST-ATS: Automatic Thumbnail Selection with Dashboard-Controlled ML Pipeline and Dynamic User Survey. In *13th ACM Multimedia Systems Conference (MMSys '22), June 14–17, 2022, Athlone, Ireland.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3524273.3532908

1 INTRODUCTION

Sports broadcasting is immensely popular, and the interest in viewing videos from sports games grows day by day. Consequently, the amount of worldwide content, such as video footage and audio commentaries, is enormous and rapidly growing. The huge availability of content and the number of games makes it increasingly important to design systems for extracting highlights in real- or near real-time. As a large percent of audiences prefer to view only the main events in a game, the generation of highlight clips and video summaries is of tremendous interest for broadcasters.

For an end-to-end soccer video production system capable of delivering game highlights, existing work in the areas of event detection [6, 10, 17, 18, 20, 23, 25, 30] and event clipping [5, 29, 31, 34] can be complemented by a thumbnail selection operation. A thumbnail is an image representing a video. Thumbnails can be used in galleries where various highlight clips are presented, and serve as the first impression meant to attract people to view a clip[1]. Thumbnails need to be selected carefully to be eye-catching and to properly represent the event in the highlight clip. However, the thumbnail selection operation is time-consuming and expensive as there are many frames in a video to select among, and image quality is often not considered extensively due to time limitations and costs. Therefore, automating the thumbnail selection process has the potential to both save resources and improve quality.

We present HOST-ATS, a holistic system for the automatic selection and evaluation of soccer video thumbnails. HOST-ATS is composed of: (1) a dashboard-controlled Machine Learning (ML) pipeline, and (2) a dynamic user survey. The pipeline [9] uses ML to automatically select thumbnails for soccer videos in near real-time, and can be configured via a Graphical User Interface (GUI). It combines logo detection, close-up shot detection, face detection, image quality prediction, and blur detection. HOST-ATS also includes a dynamic user survey for evaluating the results of the pipeline qualitatively (through user studies). This web-based survey is fully configurable and easy to update via Continuous Integration (CI), allowing for the dynamic aggregation of participant responses to different sets of multimedia assets.

[1]Example highlight galleries from the Norwegian Eliteserien: https://highlights.eliteserien.no/ and Swedish Allsvenskan: https://highlights.allsvenskan.se/

We demonstrate the configuration and execution of the HOST-ATS pipeline via the custom dashboard, and the agile (re-)deployment of the HOST-ATS user survey via Firebase[2] and Heroku[3] cloud service integrations. The audience can interact with configuration parameter updates in real-time. Overall, our experience with HOST-ATS shows that an automatic thumbnail selection system can yield highly attractive highlight clips, and can be used in conjunction with existing soccer broadcast practices in real-time.

2 BACKGROUND

Automated soccer video production systems can cover research fields such as object detection [14, 19, 22], shot boundary detection [15, 36], event detection and classification [6, 10, 17, 18, 20, 23, 25, 30], and event clipping [5, 29, 31, 34]. In this context, thumbnail selection refers to the task of finding an appropriate thumbnail image for event highlight clips.

As a representative snapshot, thumbnails capture the essence of a video and provide the first impression to the viewers. A good thumbnail makes the video clip more attractive to watch [16, 26]. There is not much work on thumbnail selection specifically for sports videos, but there are a number of studies that target thumbnail selection in general. Song et al. [26] propose a generic model called "Hecate" for selecting thumbnails automatically. Their framework uses a video as input, and filters the frames that are qualified as low-quality such as blurry, dark or uniform-colored frames. This is calculated with and decided upon via a threshold value, and not through ML. The framework also filters frames that are related to fading, dissolving or wiping effects in the video, identifying these through a shot boundary detection model. In a second step, frames that are near duplicates are discarded, and finally, frames with highest aesthetic quality are selected. This model has been trained by a set of images that have been annotated with subjective aesthetic scores. Vasudevan et al. [32] present a query-adaptive video summarization model which picks frames from a given video that are relevant to the given query. The model also has the possibility to output a single frame as a thumbnail. The query is a text of what content the end-user would like the frame to contain (e.g., in our context, it could be "soccer" or "goal"). Our analysis of such models has shown that generic approaches might not necessarily work well for soccer highlight clips [9].

3 HOST-ATS DASHBOARD-CONTROLLED ML PIPELINE

The first component of HOST-ATS is an ML-based pipeline which identifies appropriate images to be used as thumbnails, by checking for logos, scene boundaries, faces, and analyzing image quality. This pipeline selects attractive thumbnails by considering visual quality and aesthetic metrics along with relevance to video content, thus making the resulting thumbnails more representative of the video.

Related work indicates that a good thumbnail is relevant to the corresponding video, and appears interesting and attractive in terms of content and image quality [13, 16, 26]. In this regard, we centered our pipeline around 3 key principles, namely relevance, content, and image-quality.

[2]https://firebase.google.com/
[3]https://heroku.com/

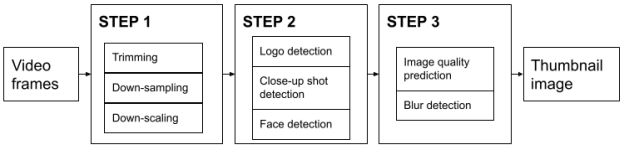


Figure 1: HOST-ATS automatic thumbnail selection pipeline.

- **Relevance:** The thumbnail that our pipeline selects is a frame from the video it is supposed to represent. Along with ensuring relevance, this bypasses the time it would take to find external images, such as graphics.
- **Content:** Highlight clips are usually presented in a gallery as a grid, where each thumbnail appears in a small size (e.g., 200 pixels) on the screen. It could be difficult for viewers to understand the contents of the image if the thumbnail displays a long-distance shot of the soccer field. Therefore, our pipeline prioritizes close-up shots. Close-up shots are usually frames showing the soccer players, spectators, and managers. There could also be frames from the replay of the goal event with shots that are closer than the default long-distance shot. If a frame is identified as a close-up shot, it will have a higher priority in the thumbnail selection process. We would also like to omit graphics such as the logo transitions appearing before replays. So, if a frame is identified as containing a logo, it will not be used as a thumbnail.
- **Image quality:** It is possible that there are frames in a video which individually appear aesthetically unpleasing or unclear to the human eye. For instance, images that are blurry, dark, and/or fading are not usable as thumbnails. Therefore, our pipeline undertakes image quality prediction as the final filter.

The end-to-end execution of our pipeline consists of 3 steps: (1) pre-processing, (2) content analysis and priority assignment, and (3) image quality analysis. Figure 1 presents these steps and the corresponding components in our framework. A video clip (sequence of video frames) is fed as input to the framework, and the final output is an image, which is a frame from the video, as the suggested thumbnail.

3.1 Step 1. Pre-processing

In this step, the input video is trimmed with respect to the start-end of the clip or an event annotation, it is down-sampled to extract a certain number of frames (default=50[4]) and frames are optionally down-scaled in terms of resolution. Each of the remaining frames are candidates for being the final thumbnail, and are fed into the next step. The left column in Figure 2 presents the configuration parameters related to this step in the dashboard GUI.

3.2 Step 2. Content Analysis

In this step, the contents of the candidate frames are analyzed. For this purpose, independent modules for logo detection, close-up shot

[4]The default number of frames might seem for a full soccer game, i.e., 90 minute video, but the pipeline is optimized for shorter highlight clips, which are traditionally no longer than 90 seconds. The number of frames is a configurable parameter.

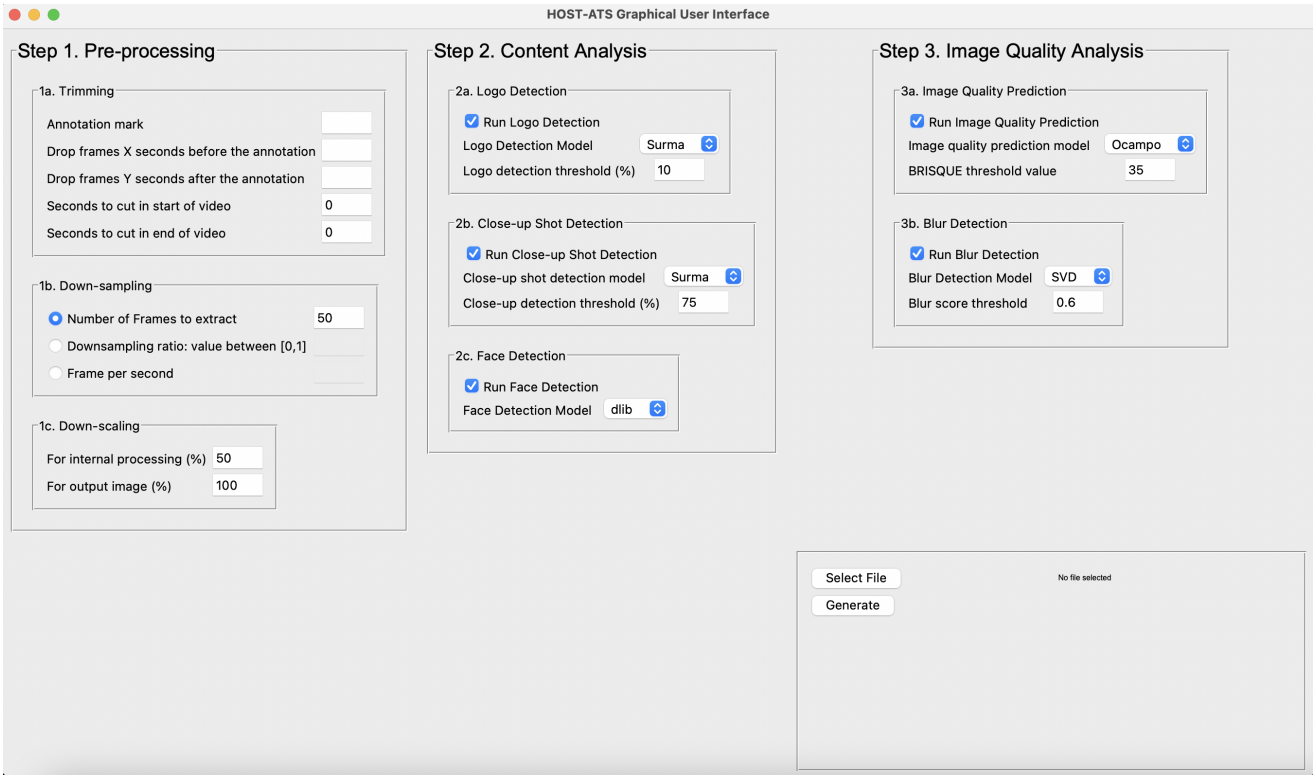
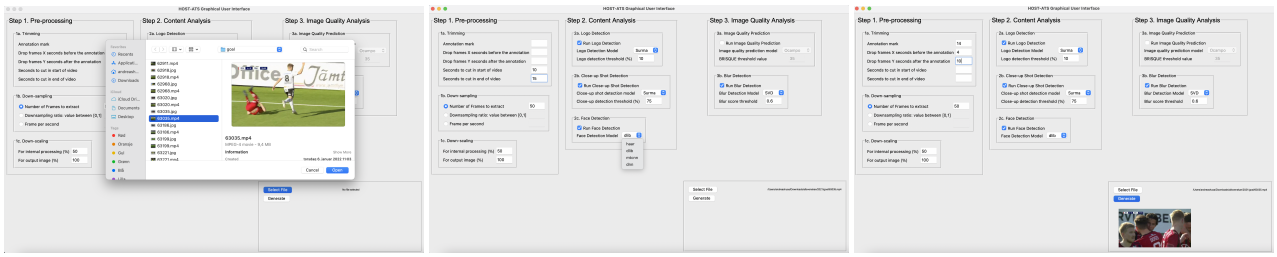


Figure 2: HOST-ATS dashboard.



(a) Input video clip selection **(b) Model configuration** **(c) Output**

Figure 3: HOST-ATS dashboard details.

detection, and face detection are used, after which priorities are assigned to each frame. The middle column in Figure 2 presents the configuration parameters related to this step in the dashboard GUI.

Logo Detection. The logo detection module is used to detect logos that appear in the video frames. When a logo appears in a soccer video, it usually indicates that a graphic is about to appear. These are frames we would like to eliminate from the thumbnail selection. For logo detection, the pipeline can use a Convolutional Neural Network (CNN) based on the architecture presented by Surma [28]. This is a general image classification model that can train and classify images based on a given dataset. The output of the model is a probability score between 0 and 1, where an output of 0.5 or above indicates that the image contains a logo. The closer the number is to 1 or 0, the more certain the model is of the input being in the predicted class. Alternatively, the pipeline can use the model by Ocampo [21] for logo detection, which is based on model proposed by Jongyoo [11]. The logo detection model and the threshold value are configurable parameters in the dashboard GUI as seen in Figure 3b.

Close-up Shot Detection. The close-up shot detection module is used to decide whether a video frame depicts a scene coming from a wide-angle camera (zoomed-out, i.e., medium or long-distance shot), or a close-up (zoomed-in) shot. Images classified as "close-up shot"s are prioritized in thumbnail selection. The image classification model

by Surma [28] can be used in this module. The model certainty threshold value is a configurable parameter in the dashboard GUI. The images with a close-up shot probability below the set threshold are not ignored, as in the case of logo detection, but receive a lower priority.

Face Detection. Face detection is used to detect the appearance of a face on a given image, as well as where the face appears on the image. In the context of our work, we consider a thumbnail image to be more relevant if there is a face appearing in it. Traditionally, face detection models rely on frontal views, and cannot detect a person's head from behind. It is possible to consider images with faces turned to an angle (such as a person's head from the side or behind) to be just as relevant. However, models for detecting such phenomena can be more complex and less accurate.

Our pipeline integrates 4 alternative models for face detection. Haar cascade [33] is an object detection model which is fast, but which tends to be prone to false positive detection, compared to other models [24]. This algorithm can be run in real-time, making it possible to detect objects in live video streams. It is possible to train the model for detecting other objects as well as faces. It is capable of detecting objects regardless of their scale and position in an image. Dlib [12] uses features extracted by Histogram of Oriented Gradients (HOG), and passes them through a Support Vector Machine (SVM). Histogram of oriented gradients (HOG) counts the occurrences of gradient orientation on fragments of the image. The method can be helpful for finding shapes in an image. MTCNN [35] where a CNN obtains candidate windows, filters out the false positive candidates, and performs a facial landmark detection. The DNN [4] face detector in OpenCV is a Caffe model which is based on the Single Shot-Multibox Detector (SSD) and uses ResNet-10 architecture as its backbone. DNN is faster, has more detection and is more accurate. Agarwal [3] compared the performance of Dlib, Haar, and MTCNN, concluding that Dlib and MTCNN perform much better for face detection in terms of accuracy, but use more time than Haar for processing. All models can be used in our automatic thumbnail selection pipeline as alternative face detectors which can be selected via configuration parameters. Other models, such as You Only Look Once (YOLO) by [22] can also be integrated in the future, to address the challenges associated with existing models such as sensitivity to facial orientation.

Priority Assignment. The results from the logo detection, close-up shot detection, and face detection modules are passed through priority assignment before filtering. The reasons for the priority levels are to be able to use several metrics concurrently in the evaluation of candidate images, and to control how greedy we would like our framework to be in enforcing our thumbnail selection rules. Figure 4 presents the decision tree used for determining the priority levels.

3.3 Step 3. Image Quality Analysis (IQA)

The iteration for selecting a thumbnail candidate starts at the highest priority level, and follows the image order prescribed in the previous step. The pipeline sorts images by the size of the largest face detected in them at this level. If there are no images assigned to the higher priority level, the iteration skips to the next priority level.

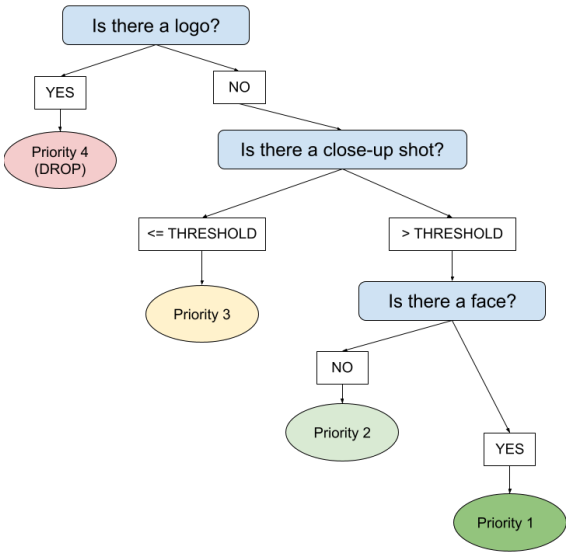


Figure 4: HOST-ATS priority assignment decision tree.

During the iteration process, an image quality predictor [21] can be run on each image. It is supposed to predict the quality of an image by calculating its blur and distortion. If the Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) score from the quality predictor is above the threshold, the image will not be chosen as the output thumbnail. The image quality prediction module is followed by a dedicated blur detection module, which integrates 2 alternative models. Singular Value Decomposition (SVD) is inspired by code from Su [27], and calculated using Numpy. It has a default threshold value of 0.60. The Laplacian using OpenCV highlights regions of an image containing rapid intensity changes. Too low thresholds lead to the incorrect marking of images as blurry when they are not, and too high thresholds lead to the marking of images that are actually blurry as not blurry. The image quality prediction module is only ran until the first image to satisfy this condition, and the image will become the final thumbnail.

3.4 Implementation Details and Demonstration

Our automatic thumbnail selection pipeline was implemented using Python `v3.9.7`, Tensorflow `v2.6.0`, Keras `v2.6.0`, cv2 `v4.5.3`, Dlib `v19.22.1`, mtcnn `v0.1.0` and Imquality `v1.2.7` on a DGX-2 server. This server is well suited for heavy computational operations and heavy memory operations. However, it is not necessary to use a machine that is exceptionally suited for heavy operations to run the thumbnail generator. The image classifier model by Surma [28] which is used for logo detection and close-up shot detection was trained on the same server as well. For training the image classifier, the same versions of Keras and Tensorflow were used, as well as Matplotlib `v3.4.3`, Livelossplot `v0.5.4` and Efficientnet `v1.1.1`. The logo detection and close-up shot detection models used by the framework are saved in Hierarchical Data Format (HDF) format, and the Haar cascade face detection model is saved in Extensible Markup Language (XML) format. These models and the complete

codebase for HOST-ATS are publicly accessible as an open-source software repository under[5].

The dashboard GUI consists of a window with all the interactions in one page. Figure 2 shows the page displayed by the GUI. All configuration parameters in the pipeline are possible to modify from this page. None of the parameters in steps 1, 2 or 3, which are presented as subcategories in Figure 2, are mandatory. If no input is provided for a specific parameter, the default values of the pipeline will be run. The "Generate" button executes the pipeline, and the output image is displayed under the button as seen in the Figure. Sample videos presenting the dashboard layout and consecutive executions with different configuration parameters can be found under[6] and[7]. A deeper analysis of the influence of configuration parameters, providing insights on how practitioners can customize the framework for best results, can be found in [9].

4 HOST-ATS DYNAMIC USER SURVEY

In order to evaluate the performance of the pipeline in terms of end-user perception, HOST-ATS provides a dynamic survey which can be used to conduct user studies. For instance, each video clip for which a thumbnail is selected can be presented as a "case", where survey participants are asked to compare the thumbnail selected by our pipeline, and a thumbnail selected by other methods (e.g., static frame selection method used by the industry today).

4.1 Survey Framework

`Huldra` [8] is a framework for collecting crowdsourced feedback on multimedia assets. This framework allows for the collection of participant responses in a storage bucket hosted on the cloud, from where they can be retrieved in real-time by survey organizers, using credentials, immediately after the first interaction of each participant. The framework is completely configurable through the use of a single configuration file, which allows for the deployment of custom surveys in a very short amount of time.

Huldra uses 3 third-party integrations. **Google Cloud Platform (GCP)** is a suite of cloud computing services that runs on Google infrastructure [1]. Alongside a set of management tools, it provides a series of modular cloud services including computing, data storage, data analytics and machine learning. Huldra uses a GCP S3 bucket for cloud storage, both for storing the multimedia assets and the participant responses. Access to the bucket is maintained through Firebase integration, where credentials can be specified as environment variables in the Heroku configuration (configuration of access to the storage bucket can also be undertaken using the `config.json` file in the codebase, but this is not advisable due to privacy reasons, unless a public bucket is used on purpose). **Firebase** is a platform for creating mobile and web applications [2]. Used in connection with the GCP S3 bucket, Firebase allows Huldra to fetch the multimedia assets to be used in the study seamlessly from, and write participant responses to, configurable locations in this bucket. Huldra uses **Heroku** as a cloud Platform as a Service (PaaS) for web deployment, which supports the triggering of automatic deployments from a GitHub repository. Applications deployed on the web via Heroku can be accessed via a URL of the form: `https://<application-name>.herokuapp.com`. All of the Huldra third-party integration functionalities can be used with free plans as well as paid plans: minimum requirements are a free Google account for GCP and Firebase, and free hobby dynos for a Heroku personal application.

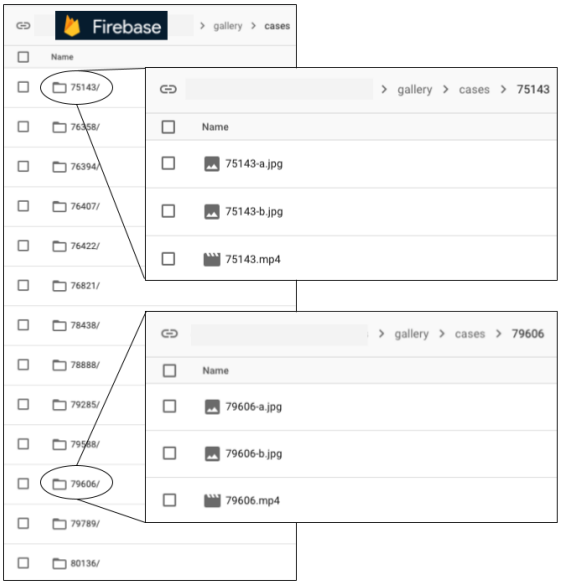


Figure 5: Firebase integration and folder structure.

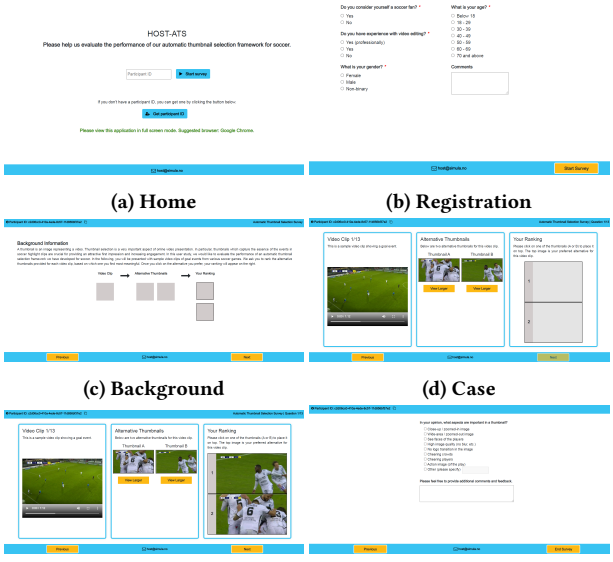


Figure 6: Screenshots of the HOST-ATS user study.

[5]HOST-ATS repository: https://github.com/simula/host-ats
[6]HOST-ATS Dashboard-Controlled ML Pipeline - Part 1/2: https://youtu.be/HHMCdMucorI
[7]HOST-ATS Dashboard-Controlled ML Pipeline - Part 2/2: https://youtu.be/VZQaEy2VauQ

For HOST-ATS, we customize the Huldra framework, specify the multimedia assets in Firebase as presented in Figure 5, and use custom participant registration and feedback form questions. A running instance can be found under[8]. Figure 6 presents screenshots of the main pages. The survey begins with the home page (Figure 6a) which allows users who already have a universally unique identifier (UUID) to complete their responses if they have closed the browser involuntarily, or decided to continue later. In both cases, their information remains saved in the browser's local storage. However, if the participant does not have a UUID, they must complete a registration form (Figure 6b) where we ask for participant information regarding age, gender, video editing experience, and soccer fandom (mandatory), as well as a free form text field if the participant has other relevant comments to add (optional). After successful login or registration, the user is redirected to the background page (Figure 6c) which introduces the context of the study and shows directions for use with a simple figure.

The core of the framework is the ranking of multimedia assets which can be easily updated using the Firebase integration. This functionality is provided by the case page (Figure 6d) which is composed of 3 vertical columns. The left column presents a sample video clip showing a goal event. HOST-ATS uses the npm package React player [7] for playback, which offers an off-the-shelf component for playing a variety of multimedia. The middle column presents two alternative thumbnails for the video clip. Both of which could be viewed larger if needed. The user ranks simply by clicking on one of the thumbnails. Once a thumbnail is clicked, it is displayed immediately on the top in the right column (Figure 6e). In order to have complete and consistent responses for our study, it is mandatory to rank a case before proceeding to the next. Users can later go backwards and revisit their answers or change them. After finishing the ranking, the participants are invited to fill out a feedback form (Figure 6f) about the aspects that they deem important in a thumbnail. They can mark from a list of alternatives, as well as suggest other facets. They also have the option to add additional comments and feedback in a text field input.

User studies conducted with HOST-ATS allow for the comparison of different thumbnail selection methods, as well as provide deeper insights into viewer expectations from thumbnails, which can potentially help improve the selection mechanisms of the ML pipeline itself (e.g., rules and assumptions which motivate the use of various modules) as future work. [9] presents a preliminary analysis of the qualitative results obtained from the survey responses of 42 participants.

4.2 Deployment and Demonstration

Assuming that the third-party service accounts are already established, the below steps can be used to run a HOST-ATS instance. The instance is automatically updated via CI after every push to the repository branch connected to the Heroku instance. It can also be updated (without re-deployment) by updating the multimedia assets in the Firebase bucket, allowing for multiple versions of the survey to be run without any code changes. More information on system setup and reproducibility aspects can be found in [9].

[8] https://host-ats.herokuapp.com

1. **Assets:** Set up the necessary folder structure in the S3 bucket, prepare and upload the multimedia assets corresponding to your desired "case"s.
2. **Codebase:** Clone the Huldra repository.
3. **Configuration:** Update configuration parameters in the `config.json` file as needed, to customize your instance.

4a. **Deployment (Heroku):** Enter the Firebase connection parameters in your app's Heroku configuration. Deploy the relevant branch of your repository. The survey will be accessible at `https://<app-name>.herokuapp.com` by default.

4b. **Deployment (local):** Enter the Firebase connection parameters in your local environment variables. Run `npm install` and `npm start` to start your local server. The survey will be accessible at `localhost:3000` by default.

5 **Outputs:** Retrieve participant response files from the bucket.

5 CONCLUSION

Thumbnail selection is a very important aspect of online video presentation. Thumbnails are needed to capture the essence of a video clip and provide a good first impression to viewers, making clips more attractive to watch. Traditional solutions for soccer highlight clips, which display important events such as goals and cards, rely on the static or manual selection of thumbnails. However, static approaches can result in the selection of sub-optimal video frames as snapshots, which degrades the overall quality of the clip as perceived by the viewers, consequently decreasing viewership, and manual approaches are expensive.

In this demonstration, we present an automatic thumbnail selection system for soccer called HOST-ATS, which comprises a ML pipeline to deliver representative thumbnails with high relevance to the video content and high visual quality in near real-time, and a dynamic user survey for evaluating the selected thumbnails qualitatively. The HOST-ATS ML pipeline leverages logo detection, close-up shot detection, face detection, image quality prediction, and blur detection. The HOST-ATS user survey allows for the evaluation of this pipeline through subjective user studies. Our demonstration will show that an automatic system for the selection of thumbnails based on contextual relevance and visual quality, complemented with a tool for validation studies, can yield highly attractive highlight thumbnails, and can be used in conjunction with existing soccer broadcast practices. Automating the traditionally complex and labor-intensive task of thumbnail selection can reduce production costs. It will also enable the research community to conduct further studies with minimal programming knowledge and time investment, studies which provide insights into various factors influencing the perception of soccer video clips by viewers. We hope that our holistic system for automatic thumbnail selection and evaluation can facilitate both academic and practical applications in the domain of multimedia content generation and presentation. The HOST-ATS system is applicable to various other sports broadcasts, such as skiing, handball, or ice hockey, and presents a viable potential to impact future sports productions.

ACKNOWLEDGMENTS

This research was funded by the Norwegian Research Council, project number 327717 (AI-producer).

REFERENCES

[1] 2022. Cloud Computing Services - Google Cloud. https://cloud.google.com/.

[2] 2022. Firebase. https://firebase.google.com/.

[3] Vardan Agarwal. 2021. *Face Detection Models: Which to Use and Why?* https://towardsdatascience.com/face-detection-models-which-to-use-and-why-d263e82c302c

[4] Gary Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[5] Chen-Yu Chen, Jia-Ching Wang, Jhing-Fa Wang, and Yu-Hen Hu. 2008. Motion Entropy Feature and Its Applications to Event-Based Segmentation of Sports Video. *EURASIP Journal on Advances in Signal Processing* 2008 (2008). https://doi.org/10.1155/2008/460913

[6] Anthony Cioppa, Adrien Deliège, Silvio Giancola, Bernard Ghanem, Marc Van Droogenbroeck, Rikke Gade, and Thomas B. Moeslund. 2019. A Context-Aware Loss Function for Action Spotting in Soccer Videos. *CoRR* abs/1912.01326 (2019). arXiv:1912.01326 http://arxiv.org/abs/1912.01326

[7] Pete Cook. 2021. react-player. https://www.npmjs.com/package/react-player.

[8] Malek Hammou, Cise Midoglu, Steven Alexander Hicks, Andrea Storås, Saeed Shafiee Sabet, Inga Strümke, Michael Alexander Riegler, and Pål Halvorsen. 2022. Huldra: A Framework for Collecting Crowdsourced Feedback on Multimedia Assets. In *Proceedings of the ACM Multimedia Systems Conference (MMSys)*.

[9] Andreas Husa, Cise Midoglu, Malek Hammou, Steven A. Hicks, Dag Johansen, Tomas Kupka, Michael A. Riegler, and Pål Halvorsen. 2022. Automatic Thumbnail Selection for Soccer Videos using Machine Learning. In *13th ACM Multimedia Systems Conference (MMSys '22), June 14–17, 2022, Athlone, Ireland.* ACM, New York, NY, USA. https://doi.org/10.1145/3524273.3528182

[10] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-Scale Video Classification with Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1725–1732. https://doi.org/10.1109/CVPR.2014.223

[11] Jongyoo Kim, Anh-Duc Nguyen, and Sanghoon Lee. 2019. Deep CNN-Based Blind Image Quality Predictor. *IEEE Transactions on Neural Networks and Learning Systems* 30, 1 (2019), 11–24. https://doi.org/10.1109/TNNLS.2018.2829819

[12] Davis King. 2021. dlib C++ Library. http://dlib.net/. Last accessed 2022-01-24.

[13] Ryan Knott. 2021. *What Are Video Thumbnails and Why Do They Matter?* https://www.techsmith.com/blog/what-are-video-thumbnails/

[14] Jacek Komorowski, Grzegorz Kurzejamski, and Grzegorz Sarwas. 2019. FootAndBall: Integrated player and ball detector. *CoRR* abs/1912.05445 (2019). arXiv:1912.05445 http://arxiv.org/abs/1912.05445

[15] Harilaos Koumaras, Georgios Gardikis, George Xilouris, Evangelos Pallis, and Anastasios Kourtis. 2006. Shot boundary detection without threshold parameters. *J. Electronic Imaging* 15 (4 2006), 020503. https://doi.org/10.1117/1.2199878

[16] Thomas J Law. 2021. The Perfect YouTube Thumbnail Size and Best Practices. https://www.oberlo.com/blog/youtube-thumbnail-size.

[17] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. 2019. BMN: Boundary-Matching Network for Temporal Action Proposal Generation. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*.

[18] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. 2018. BSN: Boundary Sensitive Network for Temporal Action Proposal Generation. In *Proceedings of the European Conference Computer Vision (ECCV)*.

[19] Pier Luigi Mazzeo, Marco Leo, Paolo Spagnolo, and Massimiliano Nitti. 2012. Soccer Ball Detection by Comparing Different Feature Extraction Methodologies. *Advances in Artificial Intelligence* 2012 (2012), 12. https://doi.org/10.1155/2012/512159

[20] Olav Andre Nergård Rongved, Markus Stige, Steven Alexander Hicks, Vajira Lasantha Thambawita, Cise Midoglu, Evi Zouganeli, Dag Johansen, Michael Alexander Riegler, and Pål Halvorsen. 2021. Automated Event Detection and Classification in Soccer: The Potential of Using Multiple Modalities. *Machine Learning and Knowledge Extraction* 3, 4 (2021), 1030–1054. https://doi.org/10.3390/make3040051

[21] Ricardo Ocampo. 2021. *Deep CNN-Based Blind Image Quality Predictor in Python.* https://towardsdatascience.com/deep-image-quality-assessment-with-tensorflow-2-0-69ed8c32f195

[22] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. 2015. You Only Look Once: Unified, Real-Time Object Detection. *CoRR* abs/1506.02640 (2015). arXiv:1506.02640 http://arxiv.org/abs/1506.02640

[23] Olav A. Nergård Rongved, Steven A. Hicks, Vajira Thambawita, Håkon K. Stensland, Evi Zouganeli, Dag Johansen, Michael A. Riegler, and Pål Halvorsen. 2020. Real-Time Detection of Events in Soccer Videos using 3D Convolutional Neural Networks. In *Proceedings of the IEEE International Symposium on Multimedia (ISM)*. 135–144. https://doi.org/10.1109/ISM.2020.00030

[24] Adrian Rosebrock. 2021. *OpenCV Haar Cascades.* https://www.pyimagesearch.com/2021/04/12/opencv-haar-cascades/

[25] Karen Simonyan and Andrew Zisserman. 2014. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*. 568–576.

[26] Yale Song, Miriam Redi, Jordi Vallmitjana, and Alejandro Jaimes. 2016. To Click or Not To Click: Automatic Selection of Beautiful Thumbnails from Videos. arXiv:1609.01388 [cs.MM]

[27] Bolan Su, Shijian Lu, and Chew Lim Tan. 2011. Blurred Image Region Detection and Classification. 1397–1400. https://doi.org/10.1145/2072298.2072024

[28] Greg Surma. 2018. *Image Classifier - Cats vs Dogs.* https://gsurma.medium.com/image-classifier-cats-vs-dogs-with-convolutional-neural-networks-cnns-and-google-colabs-4e9af21ae7a8

[29] Dian Tjondronegoro, Yi-Ping Phoebe Chen, and Binh Pham. 2003. Sports video summarization using highlights and play-breaks. In *Proceedings of ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR)*. 201–208. https://doi.org/10.1145/973264.973296

[30] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. 2018. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 6450–6459. https://doi.org/10.1109/CVPR.2018.00675

[31] Joakim Olav Valand, Haris Kadragic, Steven Alexander Hicks, Vajira Lasantha Thambawita, Cise Midoglu, Tomas Kupka, Dag Johansen, Michael Alexander Riegler, and Pål Halvorsen. 2021. AI-Based Video Clipping of Soccer Events. *Machine Learning and Knowledge Extraction* 3, 4 (2021), 990–1008. https://doi.org/10.3390/make3040049

[32] Arun Balajee Vasudevan, Michael Gygli, Anna Volokitin, and Luc Van Gool. 2017. Query-adaptive Video Summarization via Quality-aware Relevance Estimation. arXiv:1705.00581 [cs.CV]

[33] P. Viola and M. Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/CVPR.2001.990517

[34] Hossam M. Zawbaa, Nashwa El-Bendary, Aboul Ella Hassanien, and Ajith Abraham. 2011. SVM-based soccer video summarization system. In *Proceedings of the World Congress on Nature and Biologically Inspired Computing*. 7–11. https://doi.org/10.1109/NaBIC.2011.6089409

[35] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. 2016. Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks. *CoRR* abs/1604.02878 (2016). arXiv:1604.02878 http://arxiv.org/abs/1604.02878

[36] Matko Šarić, Dujmić Hrvoje, and Baričević Domagoj. 2008. Shot Boundary Detection in Soccer Video using Twin-comparison Algorithm and Dominant Color Region. *Journal of Information and Organizational Sciences* 32 (06 2008).

A.3 Poster Presentation: Automatic Thumbnail Selection for Soccer using Machine Learning

Authors Andreas Husa, Cise Midoglu, Pål Halvorsen

Published NORA Annual Smyposium 2022

Short description The poster presentation at NORA Annual Conference is directly related to the works of the papers and the thesis.

Venue Norwegian Artificial Intelligence Research Consortium (NORA) is a Norwegian collaboration between universities, university colleges and research institutes within AI, machine learning and robotics. Aiming to strengthen Norwegian research, education and innovation within these fields [43].

The NORA Annual Conference is an annual event that aims to gather the Norwegian research community within the field of Artificial Intelligence and create a platform where invited speakers and participants can share research, ideas, theories, models and new perspectives, and interact with peers from the field [44].

Automatic Thumbnail Selection for Soccer using Machine Learning

Andreas Husa[a], Cise Midoglu[a], Pål Halvorsen[a,b]

[a]*Simula Metropolitan Center for Digital Engineering (SimulaMet), Oslo, Norway*
[b]*Oslo Metropolitan University (OsloMet), Oslo, Norway*

Keywords: *blur detection, dashboard, deep learning, graphical user interface, image quality, logo detection, object detection, shot boundary detection, soccer, thumbnail generation, video.*

Sports broadcasting is immensely popular, and the interest in viewing videos from sports events grows day by day. Today, live streaming of sports events generates most of the video traffic and is replacing live broadcasting on TV. However, the availability of content and the large number of games make the generation of summaries and highlight clips in real- or near real-time increasingly important, as a large percent of audiences prefer to view only the main highlights in a game.

A thumbnail is an image representing a video. Thumbnails are often used in soccer highlight clip galleries, where they provide the first impression to attract people to view the clips (example galleries from the Norwegian Eliteserien: https://highlights.eliteserien.no/, and the Swedish Allsvenskan: https://highlights.allsvenskan.se/). The thumbnail selection operation is time-consuming and expensive as there are many frames in a video to select among, and image quality is often not considered extensively due to time limitations and cost. Traditional solutions rely on the manual or static selection of thumbnails. However, such approaches can result in the selection of sub-optimal video frames as snapshots, which degrades the overall quality of the clip as perceived by the viewers, and consequently decreases viewership. Therefore, automating the thumbnail selection process has the potential to both save resources and improve quality.

We present an automatic thumbnail selection system for soccer videos which leverages machine learning to deliver representative thumbnails with high relevance to the video content and high visual quality in near real-time. Our proposed pipeline combines logo detection, close-up shot detection, face detection, image quality prediction, and blur detection. We evaluate the proposed framework quantitatively using various soccer datasets, as well as qualitatively, through a user study (https://host-ats.herokuapp.com). Apart from being run locally on a terminal, our pipeline can be controlled by a dashboard with a user-friendly graphical user interface (GUI) as shown in Figure 1, which we will demonstrate interactively to a live audience. Our demonstration will show that an AI-based automatic thumbnail selection system can yield highly attractive highlight clip thumbnails, and can be used in conjunction with existing soccer broadcast practices due to its ability to function in real-time.

Figure 1. Graphical user interface (GUI) of the dashboard for our automatic thumbnail selection system. Left: an input video clip is selected. Middle: machine learning (ML) pipeline is configured. Right: the pipeline is executed and the output is a thumbnail image.