Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

Emil Kreutzman

Active learning for annotation and recognition of faces in video

An end-to-end system

Master's Thesis
Helsinki, February 23, 2021

Supervisor: Jorma Laaksonen, D.Sc. (Tech.)
Aalto University

Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

ABSTRACT OF MASTER'S THESIS

Author:	Emil Kreutzman		
Title:	Active learning for annotation and recognition of faces in video — An end-to-end system		
Date:	February 23, 2021	**Pages:**	94
Major:	Machine Learning, Data Science and Artificial Intelligence	**Code:**	SCI3044
Supervisor:	Jorma Laaksonen, D.Sc. (Tech.)		

Data scarcity is often a concern when working with real-world datasets. From a machine learning point of view, it is problematic when datasets are not pre-labeled, so supervised learning cannot be easily performed.

In this thesis, a novel annotation pipeline for video data is introduced that aims to solve this problem. The system consists of a data extraction phase, as well as a labeling architecture and user interface. Various techniques and models from active learning, face detection, face recognition and object tracking are used and improved upon to put together an end-to-end system.

Importantly, the pipeline builds upon many recent advances within face detection and recognition, that utilize deep convolutional neural networks. The single-stage neural face detector used enables accurate trajectories of faces to be formed. Additionally, another neural model is used to create embedding vectors of faces, which are then used for clustering.

The system is evaluated in terms of labeling cost and noise, against a new dataset of Finnish feature films. It is determined that the labeling effort is reduced by orders of magnitude, as compared to a naive approach.

Keywords:	machine learning, artificial intelligence, deep learning, active learning, face detection, face recognition, face tracking, video, film, movies
Language:	English

Aalto-universitetet
Högskolan för teknikvetenskaper
Master's Programme in Computer, Communication and Information Sciences

SAMMANDRAG AV DIPLOMARBETET

Utfört av: Emil Kreutzman

Arbetets namn:
Aktiv inlärning för annotering och identifiering av ansikten i video — Ett helhetstäckande system

Datum:	Den 23 februari 2021	**Sidantal:**	94
Huvudämne:	Machine Learning, Data Science and Artificial Intelligence	**Kod:**	SCI3044
Övervakare:	Jorma Laaksonen, Tekn. Dr.		

Bristfälliga data är ofta ett dilemma då verklighetsbaserade dataset behandlas. Ur ett maskininlärningsperspektiv blir detta problematiskt, ifall data inte är annoterade på förhand. I så fall kan inte väglätt lärande (supervised learning) tillämpas.

I detta diplomarbete introduceras ett nytt annoteringssystem för video, som ämnar att lösa detta problem. Systemet består av en fas för extraktion av data och en arkitektur, samt användargränssnitt, för annotering. Olika tekniker och modeller ifrån aktiv inlärning, ansiktsigenkänning och ansiktsidentifiering används i systemet. Dessutom används och förbättras en teknik från objektspårning (object tracking), och alla dessa metoder sätts ihop till ett helhetstäckande system.

Systemet bygger på flera viktiga framsteg inom ansiktsigenkänning och identifiering, som använder sig av tekniker inom djupa faltningsnätverk (convolutional neural networks). Den neurala ansiktsigenkänningsmodell som används möjliggör att noggranna tidsbanor för ansikten formas. Ytterligare en annan modell används för vektorisering (embedding) av ansiktena. Dessa vektorer utnyttjas därpå för klusterering.

Systemet urvärderas på basis av hur effektivt annoteringen kan utföras, samt hur lite störningar som förekommer i extraherade data. En samling finska filmer används som basmaterial i denna analys. Det fastställs att systemet reducerar annoteringsarbetet mycket i jämförelse med naiva lösningar på samma problem.

Nyckelord:	maskininlärning, artificiell intelligens, djupinlärning, aktiv inlärning, ansiktsigenkänning, ansiktsidentifiering, ansiktsspårning, video, film
Språk:	Engelska

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. Jorma Laaksonen. Dr. Laaksonen came up with the original idea of using pre-clustering for the active learning approach used in this thesis, and actively guided my work with useful feedback. Without your honest interest in the project, the thesis would not have turned out as well as it did.

I am also grateful for having the opportunity to work with the great team of researchers in the Movie Making Finland (MoMaF) research group. Some of you actively participated in this thesis work by being part of the annotation study. However, all of you listened to my monthly ramblings in Zoom, whenever I had added something new to the architecture. Thank you all.

My time at Aalto University has been amazing, truly some of the best years of my life. As this thesis marks the end of my studies, I feel nostalgic, while looking back at these past years with fondness. I have made friends for life, and learned things that I will never forget. Thank you to all my friends who supported me during the studies, and motivated me to keep going with this work whenever I was not feeling it.

Finally, I would like to thank my family. My sister Ida made the effort to correct my horrible grammar, and my brother Wille... Well, he did not do a lot but kept me going by asking about progress. My parents Susanne and Tor, I am grateful that you stuck with me during these past months. Here it is now, the result of all those years of studies, and these past months of intense research.

Helsinki, February 23, 2021

Emil Kreutzman

Abbreviations and Acronyms

ADFV	absolute difference of frame variance
API	application programming interface
ARI	adjusted Rand index
ASFD	Automatic and Scalable Face Detector
CNN	convolutional neural network
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DCN	deformable convolution network
DCNN	deep convolutional neural networks
DCV	discriminative commmon vectors
DPM	deformable parts model
FD	face detection, or face detector
FPN	feature pyramid network
FV	frame variance
GAN	generative adversarial networks
ICA	independent component analysis
KAVI	Kansallinen audiovisuaalinen instituutti, National Audiovisual Institute of Finland
k-NN	k-nearest neighbors
LDA	linear discriminant analysis
LFW	Labeled faces in the wild dataset
MAFD	mean absolute frame difference
MoMaF	Movie Making Finland
MOT	multi-object tracking, or tracker
OLS	ordinary least squares
PCA	principal component analysis
R-CNN	region-based convolutional neural networks
RGB	red, green, blue
SDMAFD	signed difference of MAFD
SQL	structured query language

SVM	support-vector machines
UI	user interface
UMAP	Uniform Manifold Approximation and Projection

Symbols

$b_{i,j}$	rectangular facial bounding box, frame i and in-frame index j
b_{min}	minimum width and height for bounding boxes
B_i	the set of bounding boxes corresponding to frame i
c_i	class label of the actor with index i
c_{max}	maximum size of clusters
c_{size}	ideal cluster size for being shown during annotation
$\boldsymbol{C}$	a clustering, consisting of many individual clusters
C_i	a cluster of trajectories
$\hat{c}$	predicted (actor) class label, $\hat{c} \in \{c_1, c_2, \ldots, c_n\}$
$i(x, y)$	black-and-white intensity value of an image pixel, at (x, y)
$n_{clusters}$	number of output clusters, in clustering algorithms
n_{skip}	in practice, only every n_{skip} frames, from a trajectory, are saved to disk
$\boldsymbol{p}_t$	trajectory-level probability vector
$\boldsymbol{p}_c$	cluster-level probability vector
t_i	the trajectory with index i
t_{min}	minimum length (in frames) for a trajectory to be considered valid
t_{max}	maximum age (in frames) a trajectory can outlast short occlusions
T	the complete set of trajectories seen thus far, during object tracking
T_a	active trajectories, $T_a \subseteq T$
τ_{box}	threshold for bounding boxes to be considered overlapping
τ_{show}	probability threshold for showing actor predictions
$\boldsymbol{v}_{i,j}$	embedding vector corresponding to bounding box $b_{i,j}$
$\bar{\boldsymbol{v}}_j$	representative mean vector for trajectory t_j

Contents

Chapter 1

Introduction

Recent advances in computer vision have enabled human-like performance on face detection and recognition tasks [55]. In no small part, this progress is thanks to ever-better applications of deep convolutional neural networks (DCNN).

While the performance of some newer models (e.g., [74][21][79]) can be considered remarkable, even the best architectures are only as good as the data that is used to train them. To tackle this problem, some research has adopted methods from the field of active learning. In this scenario, a human annotator assists the machine learning system to help it achieve more accurate end results. In some sense, the human in the active learning process extends the original training dataset by providing additional insight.

However, as other researchers have pointed out, this in itself might not be enough to satisfy concerns of data quality. This is because, all too often, active learning research fails to address practical concerns that are involved when building real-world systems [77]. For one, human labelers are imperfect sources of data. People can misunderstand the task they are given, or simply arrive at the wrong answer. Apart from data quality, quantity can also be of concern. For example, it is not uncommon to have access to datasets that have no labels whatsoever.

It is within this context that this thesis sits. Using a dataset of feature films, a solution is devised to allow for accurate classification of actor appearances in movies. However, the problem of final classification is of secondary concern here. Instead, the focus is on acquiring the training data that is ultimately required for a good supervised classifier. To this end, a novel annotation pipeline is introduced. The proposed system tackles the challenges involved in face recognition tasks where data is scarce. In the following sections, the research questions are specified, along with the major contributions made within this work.

1.1 Research questions

The scope of this thesis is broad. An end-to-end active learning system with a complete annotation user interface is introduced, but contributions are also made to individual methods. To narrow down the most important objectives, a set of key research questions are presented here. In order of importance, with the most important question first, the proposed research questions are:

1. How can active learning be used effectively to extract labels for actors' faces in feature films?
2. Which methods can be used to reduce unnecessary manual work by human annotators involved?
3. In terms of labeling cost and quality of the extracted data, what is the performance of the active learning system?
4. How should previous research be adapted and improved upon to achieve the aformentioned goals?

1.2 Contributions

In Chapter 3, the architecture of a new active learning system is presented. As a whole, this can be viewed as a contribution made in this thesis. However, certain aspects of the system are more important than others. These aspects address concerns that previously did not receive much attention within active learning, or they improve the performance of the system significantly. The four major contributions claimed are thus:

1. It is shown how a full active learning pipeline can be applied to raw video and movie data that has no labels to begin with. The pipeline consists of two phases: *extraction* and *annotation*.
2. A full labeling system architecture is presented. Key features of the design are used to speed up the labeling work and reduce noise.
3. The proposed system is highly modular and applicable to generic object detection tasks, but a deliberate choice to focus on face identification has been made here.
4. A pre-clustering approach is used to reduce the annotation effort. As a video-specific addition, trajectories of faces are formed to manage clustering noise and further bundle similar images together before annotation.

All components in the pipeline make for an overall effective labeling system. Individual parts are subjected to comprehensive analysis, by which it is shown that the system performs well in the key areas. The labeling cost is substantially reduced, and human annotators are not subjected to unnecessary noise at the time of labeling.

1.3 Structure of the work

This thesis work divides subject matters into distinct chapters to provide a complete picture of previous work in the area, along with proposed new ideas. In Chapter 2, the relevant related work is surveyed. Mostly, this is about historically significant works within face detection and recognition, as well as active learning.

Chapter 3 introduces the novel active learning pipeline. The pipeline architecture is generic, and does not depend on specific methods to work. To show the effectiveness of the system, a set of good baseline methods are also proposed. Chapter 4 surveys the literature of the baseline methods, while also specifying small improvements made in this thesis.

Finally, the annotation architecture is critically evaluated in Chapter 5. Various methods are used to examine the labeling cost and quality of the data used in annotation. This is followed up by conclusions drawn in Chapter 6, where suggestions for future work are also made.

Chapter 2

Background

This chapter looks back at the fields of face detection and recognition. These domains, along with active learning, form the basis of the methods used in this thesis. There is also a short introduction to the Finnish cinematic research project, within which this work exists.

The first feature film ever produced in Finland was *Salaviinanpolttajat* ("The Moon Shiners"), released in 1907. In the years 1907–1935 a total of 115 movies were released, although only 50 have survived until today. [83]

Since the early days of the Finnish film industry, more than a thousand movies have been produced and are preserved by the National Audiovisual Institute, KAVI[1]. For this purpose the institute provides the Finnish National Filmography, which serves as a resource documenting the development of Finnish society through film. Even if multiple works have been written about the history of Finnish cinema (e.g., [36][86]), no large-scale studies using machine learning have been attempted. This thesis is part of a larger project which aims to do that. That is, treat the Finnish National Filmography as a big data resource and study it using the modern tools of data science.

2.1 Movie Making Finland

Movie Making Finland: Finnish fiction films as audiovisual big data, 1907–2017 (abbreviated MoMaF) is an interdisciplinary research project. The aim is to study Finnish films and the development of Finnish society through the movie medium. This field of research is novel since, thus far, humanities research in the digital era has mostly relied on textual data [90]. Newer works

[1] https://kavi.fi

are however already using data science techniques to study phenomena such as gender equality [42] and architecture [16].

With MoMaF, the areas of research are twofold. First, the aim is to use and improve state-of-the-art tools within audio and natural language processing, as well as computer vision. Speech recognition can be performed to provide automatic subtitles even for older films; a challenging task as older movies tend to have poor sound quality. The latest models in the area are able to produce very good transcriptions in low-noise scenes [68]. Apart from human speech, the project will also utilize other sounds occurring in movies, such as those of vehicles or weapons. Naturally, sound in itself is not interesting without the visuals of cinema. There are multiple aspects to study when looking purely at the picture in films. Some examples would be scene types, genders of actors and objects present on screen. These items can be quantified using machine learning techniques from computer vision. Many computer vision techniques focus mostly on images, something MoMaF extends by its priority on video material. Any efforts in these fields of audio and visual data science can be interesting on their own. Another major goal of MoMaF, though, is the use of these techniques to study cultural phenomena. In what ways has Finland changed in the last 100 years or so, and how is the change visible in the films made during that time? [73]

2.2 Detecting human faces in images

In this section the recent history of face detection is established. The aim is to give the reader an idea of the creative approaches that are used and their most important principles. Extra attention is given to specific works and advances that have shaped the field over the years.

In general *face detection* can be defined as follows. Given an image, the task is to determine whether or not it contains one or more faces. The system should return both the position and area of each face [91]. The problem itself is challenging due to the multitude of imaging parameters affecting the appearance of a face. Some of these include imaging conditions, such as lighting, scale or camera angles, but also facial features, such as pose or facial expressions. Because of the many use cases, the problem of human face detection has been widely studied. Hundreds of papers in the domain have been published within the last decades. [95]

Zafeiriou [93] puts face detection algorithms into two categories based on the implementation principle:

1. Algorithms based on rigid templates, including:
    - Boosting-based schemes such as the Viola-Jones detector [84].
    - Older neural network models [72].
    - Newer CNN-based approaches [98][21].
2. Algorithms using Deformable Parts Models (DPM):
    - This class of algorithms adapts pictorial structures [23] for detecting faces.

Recent work in the field has concentrated largely on architectures using convolutional neural networks (CNN). In the above categorization, these models fit into the rigid template class. RetinaFace [21] is one such model, and it is used as the baseline face detector for this thesis. In RetinaFace, the authors use a tiered pyramid structure of convolutional feature maps to achieve scale-invariance. They also employ multi-task learning, which is claimed to improve performance also on the individual task of bounding box regression. The benefit of multi-task learning was demonstrated earlier in generic object detection, with the introduction of Mask R-CNN [33]. Indeed, borrowing from object detection research has been a trend in face detection recently, also in other works. Apart from RetinaFace, other notable face detection architectures that use CNNs include RefineFace [99], AInnoFace [96] and ASFD [94].

2.2.1 The Viola-Jones algorithm

In the 2000s, perhaps the most influential work within face detection is the algorithm developed by Viola and Jones. This section summarizes the work and its key contributions based on the original work and follow-up revisions [84][85]. The algorithm is not directly applied in this thesis work, but it could certainly be used as a part of the generic architecture presented in Chapter 3. Nevertheless, the method is of great historical significance within the field, which is why it is surveyed here to provide context.

The working principle of the detector is the notion of rigid Haar-like features. Figure 2.1 visualizes these features as grids of black-and-white rectangles within sub-windows of the target image. Let us denote the set of pixels in the white rectangles as w, and the set of pixels in the black

Figure 2.1: Visuals of features used in square sub-windows of an image. The sum of pixels in the black region(s) is subtracted from the sum of pixels in the white region(s) to form a feature. a) and b) features overlaid on a sub-window containing a face. c) 4-rectangle feature, the most complex kind used by Viola-Jones.

rectangles as b. Then, if pixel intensity values are $i(x, y)$, the scalar feature f given a sub-window s is computed as

$$f(s; w, b) = \sum_{x,y \in w} i(x, y) - \sum_{x,y \in b} i(x, y). \tag{2.1}$$

A feature is parametrized by the white w and black b regions within the sub-window s of the full image, as shown in Equation (2.1). Then, given a feature f we are able to create classifiers h^w to detect a face:

$$h^w(s) = \begin{cases} 1 & \text{if } p\, f(s) \leq p\, \theta \\ 0 & \text{otherwise} \end{cases}. \tag{2.2}$$

The superscript in h^w denotes a *weak* classifier, since it is using only a single feature to make predictions. The parameters are p as a parity sign determining the direction of the inequality, and θ as a threshold. However, a classifier such as the one shown in Equation (2.2) is not useful on its own. Thus, N such classifiers are combined to form a stronger one. Let us denote this stronger classifier, without a superscript, as

$$h(s) = \begin{cases} 1 & \text{if } \sum_{n=1}^{N} \alpha_n\, h_n^w(s) \geq \frac{1}{2} \sum_{n=1}^{N} \alpha_n \\ 0 & \text{otherwise} \end{cases}. \tag{2.3}$$

The classifier in Equation (2.3) detects a face in a sub-window s if the returned value is *1*. If the returned value is *0*, no face was detected. The parameters α_n used in the strong classifier are learned weights. The last

paragraphs in this section discuss classifier and feature training. Note that in itself a classifier $h(s)$ like this could be used to detect faces in an image if applied to all sub-windows. The performance would not be great, however, so additional work is needed to put together an even better detector.

In itself this process of using feature extraction was not new at the time when Viola and Jones published their original article [84]. More complex features had been used in object detection even before that [57]. Instead, the Viola-Jones detector claims three key contributions to make it fast and accurate.

The first is the integral image, described in [19]. It makes calculating the intensity sums of Equation (2.1) $\mathcal{O}(1)$ in time complexity. The integral image is pre-computed once for every image that detection is performed on, according to

$$ii(x, y) = \sum_{x'=1}^{x} \sum_{y'=1}^{y} i(x', y'). \tag{2.4}$$

As such, the ii-value at any image coordinate (x, y) equals the sum of all intensity values in the spanning sub-window from $(1, 1)$ to (x, y). In turn, this formulation allows for computing the sum of all pixels in any rectangle $R = (x_1, y_1, x_2, y_2)$ in constant time using

$$\text{sum}(R) = ii(x_2, y_2) + ii(x_1, y_1) - ii(x_2, y_1) - ii(x_1, y_2). \tag{2.5}$$

Since the integral image is pre-computed, the computation requires only four memory references and three additions, regardless of the rectangle coordinates and size.

The second contribution made in the paper is in how classifiers are combined to form a final detector that is fast, yet more accurate than any individual strong classifiers. The idea is referred to as the *attentional cascade*, linking multiple strong classifiers together sequentially, as shown in Figure 2.2.

With the cascade, sub-windows have to pass all classifiers sequentially to get the final classification *1*, a detected face. Any rejection made by a single classifier along the way immediately rejects the whole sub-window. This speeds up computation greatly, since there are over 6000 weak classifiers in the detector as a whole.

To achieve good performance the first classifiers are selected to be simple and yield high true negative rates. The idea here is that the first layer should be able to reject a high number of all the non-face sub-windows that will be processed for any image. Potential false positives that are first let through,

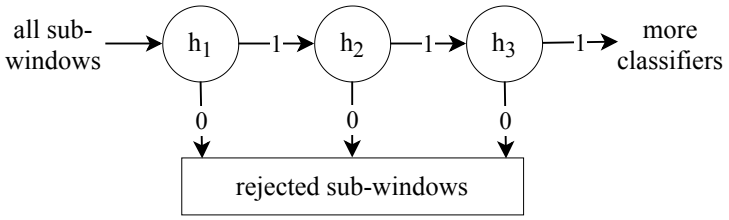

Figure 2.2: The attentional cascade links multiple strong classifiers together.

are then rejected by more complex layers later in the cascade.

At this point one can see how the Viola-Jones detector works to detect faces. All sub-windows of an image are processed and passed through the series of classifiers for detection or rejection. However, the process cannot be effective unless the individual classifiers are selected well. This is the third contribution made by the authors: using machine learning and the AdaBoost algorithm [25] to learn good features and train classifier parameters.

2.3 Recognizing faces

In this section the background of face recognition development is briefly introduced.

The field of automatic, computerized face recognition was pioneered by early works in the 1960s. Back then, the systems were not fully automatic and would still involve human-labeled facial features to assist the computer in finding database matches to a face. [8][7]

Since the early efforts, a lot of research has been published on the topic. This is likely due to the wide range of surveillance and business use cases for the technology. These areas are particularly affected since face recognition is used to identify one or more people in still images or videos. Ideally, the software automatically and accurately maps the indivudals seen in media to known images in a collection. [101]

We can distinguish between two separate tasks performed by face recognition systems. They are *verification* and *identification*. [40]

- Verification: Determine whether a given face is the same as in an image. In this scenario, we are not interested in knowing who exactly the person is. Instead, it is enough to determine if the face matches the picture.

- Identification: Given an image of a person, determine who it is by mapping to a known set of faces in a database.

Similarly to Bledsoe [8], most early works in face recognition continued using crude feature extraction methods to perform recognition. A seminal work with this approach is that by Kanade [45]. Their system uses simple techniques to extract facial features such as ratios between key points or areas. In total, 16 of these features form a vector, which is then compared to other faces using Euclidean distance. Since, many articles have been published with similar approaches based on facial geometry with simple transforms (e.g., [65][30][18]).

In their work on associative memory systems in the 1980s, Kohonen [48] shows a system using image vectors as their input signal. In moderns terms the presented memory system is a linear classifier with a weight matrix encoding "memory". Even though it used only one matrix to map the input, the work can be seen as a one-layer precursor to later face recognition research based on neural networks.

If feature extraction based on geometry is a way of reducing the dimenensionality of the face recognition problem, then so is explicit dimensionality reduction applied on the image space. Perhaps the first work with this approach is by Kirby [47], who studied face images using principal component analysis (PCA). For this case specifically, the authors refer to the principal components as *eigenpictures*. Here, images are vectorized into $\mathbb{R}^{w \cdot h}$ vectors simply by flattening the 2D grayscale image matrices before performing PCA. Figure 2.3 presents three eigenpictures as shown in the original paper [47]. This same method was then used by Turk [82] in their face recognizer, comparing low-dimensional PCA projections of face images using mainly Euclidean distance.

Many other methods expanding on the idea of low-dimensional representations of faces have also been developed. Instead of using PCA, the authors of [53] use linear discriminant analysis (LDA). LDA allows for maximizing separation of ground-truth classes, and not only feature-level variance like PCA. This property is useful in the face recognition task as identity labels can be used to learn a better transformation. Apart from PCA and LDA, similar work uses independent component analysis (ICA) [4] and discriminative common vectors (DCV) [12].

Regardless of the exact underlying methods used, the following is a common pattern used in face recognition: deriving alternative representations for face images as an intermediary step. More generally, this process is known as feature extraction, and is still the dominant way in which face recognition pipelines work. Figure 2.4 outlines this general idea, when feature extraction

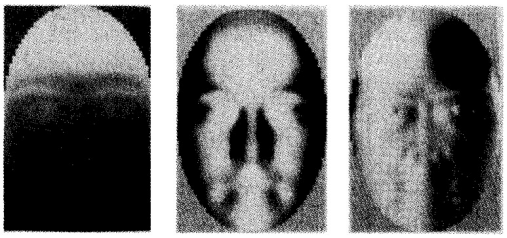

Figure 2.3: The first three *eigenpictures* (principal components) of the data in [47]. The images shown are simply the vectors converted back into 2D images (reshaped) and the zero-level set at mid gray 128. Pixel values in the range $[0, 255]$.

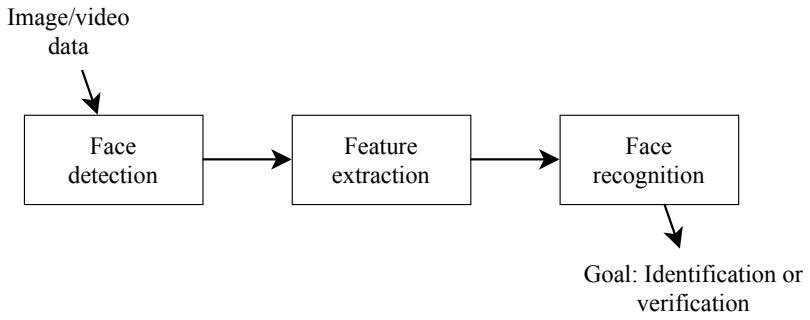

Figure 2.4: The generic face recognition pipeline. Face detection is first performed to localize faces within the image. Various methods are used to perform feature extraction, including DCNN models recently.

is coupled with face detection to perform recognition.

Nevertheless, these methods still struggle with some of the core challenges of face recognition: age, illumination and pose of the subject. Indeed, they often fail to generalize well and require constrained conditions. The performance is poor with faces "in the wild"; faces in varying, hard conditions. [40]

Recently, however, major strides have been made in addressing these issues. Part of the solution is having access to new datasets with millions of labeled faces, such as MS-Celeb [32] and VGGFace2 [11]. These big datasets in conjunction with very large deep convolutional neural networks (DCNN) have led to human-like perfomance on recognition tasks [55]. DCNN-based methods still use the same generic face recognition pipeline as shown in Figure

2.4 and aim to improve the feature extraction step. In their training process, an auxiliary classification task is used to train the network to discriminate between different subjects [55]. As such, the network takes as its input a face image and outputs a vector representation. A notable method in this space is by Taigman [79], in which a nine-layer neural network is used to compute face representations. The authors then use the dot product as a similarity measure to perform recognition. Not relying on classification in its training procedure, the FaceNet [74] model is another example of a DCNN-based facial feature extractor. Instead, a novel approach with triplet loss [75] is used to learn discriminating features. The authors also claim good results in clustering, which is why that model is used within the active learning architecture proposed in this thesis work.

2.4 Active learning

This section defines the concept of active learning, and covers various pieces of research where active learning techniques have been used in the past.

In the context of machine learning, any form of learning where the learning agent has a degree of control in the learning process, can be defined as active learning [15]. This is even applicable to human learners; we *actively* seek out and examine phenomena that we want to learn more about.

More precisely though, active learning can be thought of in terms of *queries* posed to an *oracle*. [76]

- Oracle: The source of truth, mostly a human annotator. The oracle is assumed to know the answers and be able to fill in the gaps where the system is uncertain. For instance, in the case of image classification the oracle will know the correct label when presented with an image. The oracle can also be a time-constrained automatic system. In either case, querying the oracle is expensive or somehow inconvenient.
- Query: The way a learner can access the oracle. A simple query could be in the form of presenting a human labeler with an image that should be labeled. The way queries are chosen is called the *query strategy*.

It is this setting that separates active learning processes from a supervised learning setting where fully labeled data is available. Not all the data is labeled in advance in active learning, and manual labeling is expensive and slow. Thus, the query system design becomes critical as it needs to efficiently

sample queries for the oracle to respond to [1]. This is the central problem of active learning.

Additionally, even the notion of a query is rather vague, which is why the literature on the topic often divides queries into three high-level categories. In the following three sections the categories are defined, along with relevant examples of previous work in the domain. In these sections *sampling* will always refer to the process of selecting a datapoint for queries to an oracle.

2.4.1 Membership query synthesis

Membership query synthesis, sometimes called "concept learning", is perhaps the oldest paradigm within active learning. It refers to any query strategy which aims to synthesize novel entities from available data. These newly created datapoints are then what is presented to the oracle. As such, it is the derived data that is labeled with a membership $x \in L_i$, not a point in the original data distribution. [2]

One example of this type of synthetic queries is by Baum [5]. In their work query learning is used to segment a space of hand-written digits. For instance, given images of digits 3 and 4, a decision boundary has to lie between the points of these two images in the input space. The method synthezises new images for these in-between points and uses human annotators to label them.

A similar method recently has been used to generate synthetic images of human faces instead of digits [46]. This is within the domain of generative adversarial networks (GAN) [28], which is vaguely related, but not active learning. In the case of GANs, a generator model is trained against an adversarial discriminator, mimicing the role of the oracle.

2.4.2 Stream-based sampling

Stream-based, sometimes "selective", sampling differs from query synthesis in that it is sampling from the same dataset that is being studied. Another important difference is the notion of a *stream*. Here, the query system looks at a sequential stream of datapoints. It then decides separately for each one, whether or not to query the oracle for a label. [14]

The key problem in selective sampling is finding a good way to sample so that learning is effective. One way to do it is simply sampling by uncertainty; query when an uncertainty measure is high, otherwise do not. [102]

Stock price prediction is one area that has been studied using stream-based sampling. A recent paper uses sentiment analysis (positive, neutral, negative) in conjunction with a support-vector machine (SVM) [17] classifier to assess the sentiment of social media posts. The selection strategy used is a

simple one whereby datapoints are selected for labeling if they are close to the boundary between two classes. If a sample is highly positive as determined by the agent, it is not selected for labeling by the oracle. If it is at the boundary between positive and neutral, it is passed on to be labeled by a human annotator [78]. This sampling strategy is similar to one used by Baum [5], as mentioned in Section 2.4.1. This is often the case; sampling strategies are rarely limited to one specific paradigm of active learning.

2.4.3 Pool-based sampling

Pool-based sampling [50] assumes the existence of a pool of unlabeled data. In this scenario the full pool is available for sampling: a difference from the streaming scenario where data is available one point at a time. As such, pool-based sampling is applicable in all instances where a large amount of unlabeled data can easily be acquired. However, if the pool is large, effective sampling from the pool is required for the method to be viable. [87]

Many available datasets can be thought of in terms of a pool like this. That is, when a large collection of unlabeled data is available, but labeling it is expensive. Therefore, many studies have been published in the area of pool-based sampling (e.g., [56][41][69][38][81]).

To devise a good query strategy to perform pool-based sampling, methods from the other areas (query synthesis, stream-based sampling) of active learning can be used. For instance, sampling based on uncertainty measures [102] is applicable here as well. Another useful approach is that of density sampling, which attempts to label items that are "representative". The idea here is that dense regions of the input space are the most representative of the input distribution as a whole, and should thus be labeled. [76]

One way to model the density is by clustering the pooled data. Nguyen [64] uses pre-clustering to represent the density of the underlying distribution. In their work, the authors apply the pre-clustering method to an image classification task. By using this approach they avoid labeling examples from the same cluster manually multiple times. In the architecture presented in Chapter 3, this idea is extended to fit the use case of video labeling.

Chapter 3

Architecture

A novel system architecture for active learning within films is introduced in this chapter. The system design is generic, without stringent requirements on individual components, such as the face detector or object tracker. However, a good set of baseline methods is suggested and used to evaluate the system performance.

The system is end-to-end, meaning that it looks at all steps in the problem space, from raw data extraction to classification and manual annotation. Human annotators are used in a labeling step as the oracle. In the proposed solution, the annotation step is given extra attention since it is by far the slowest step in the whole procedure. To this end, the architecture primarily aims to minimize labeling cost with efficient preprocessing and an ad-hoc annotation solution. Within this solution, an approach to iterative classification of actors within films is also shown.

In the following sections, an overview of the system is given that presents the high-level components. Following that, all pieces of the full pipeline are introduced separately.

3.1 Overview

On a high level, we can divide the architecture into two phases, namely data extraction and annotation. As part of the extraction phase, classification is also performed.

Out of these tasks, extraction and classification are highly automatic and the last phase, annotation, is very laborious since it introduces human annotators into the loop. As such, the primary objective of the system design is reducing the amount of work performed by people in the annotation step. Thus, this will be the main focus of the architecture design: a system that

is rigid and provides an efficient way of performing annotation. The architecture is visually depicted in Figure 3.1. While it is not the focus of this thesis, the architecture can easily be deployed using an iterative feedback loop. In this scenario, classification is performed again after every new batch of annotated images is produced in labeling. In this work, a classification approach is introduced that can be used in this way. Still, the primary focus of this research is on the annotation and extraction processes.

Past work in active learning has often focused on the query mechanism specifically, i.e. how good questions can be posed to the oracle. This approach often neglects practical considerations that appear in real-world use of learning algorithms. For instance, the three high-level categories (see Section 2.4) within active learning assume that labels will be assigned by the oracle one by one. In practice, this is often cumbersome; it might be better to label in batches. [77]

Because of the aforementioned reasons, this thesis work heavily applies batch labeling and pre-clustering. The key active learning mechanism is similar to that of [64], but it is further adapted for use with video data. The main addition for video is the use of an object tracker to form trajectories. In Sections 3.2.2 and 3.2.6 these contributions of trajectories and pre-clustering, are further explained.

The next section presents the extraction pipeline, which is the precursor to manual annotation. The extraction pipeline aims to extract data from the raw films, into an intermediary format, that can then be efficiently annotated.

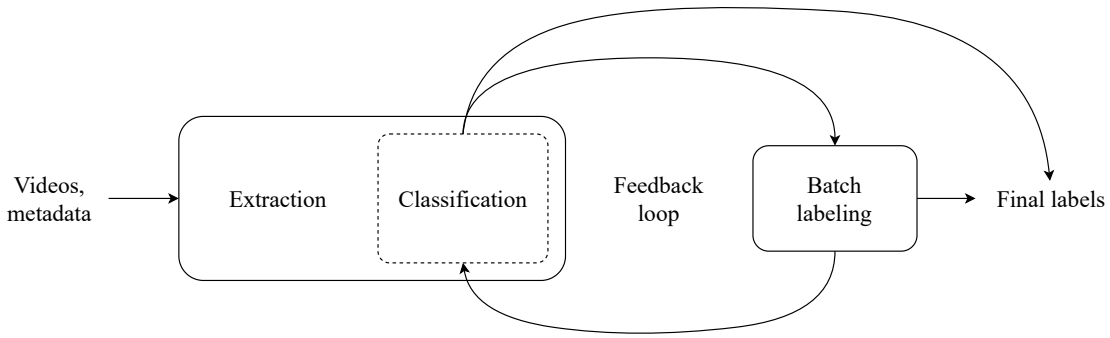

Figure 3.1: The proposed end-to-end active learning pipeline for actor identification in movies.

3.2 Extraction

The purpose of the extraction pipeline is to preprocess the raw video file in order to extract derived data needed for annotation and face recognition. In this section, an overview of the pipeline is given, followed by detailed definitions for each particular subtask that is involved.

The proposed extraction process works on raw video files, processing each frame sequentially. The process is modular, requiring the use of multiple different machine learning models to work. The most important individual model is the face detector, applied on each frame to detect faces of the actors that appear in the film. Using bounding boxes of the detected faces, the multi-object tracker then forms trajectories that span multiple frames. Finally, in an embedded feature space, the faces are clustered and classified. The full schematic in Figure 3.2 shows the many processing modules used.

In a conventional setting, the only interesting output from a data pipeline such as this would be the final classifier results. However, while the proposed pipeline does output predicted actor labels, many more outputs are needed. Combined, all of the saved data that is extracted is used to guide the annotation. Thus, all of the following pieces of data are stored[1] as output:

1. Indices of frames where shot transitions occurred
2. Face images from each frame, and corresponding feature vectors
3. Trajectories
4. Cluster assignment for each trajectory
5. Distribution over predicted actor labels, for each cluster

Note that items 2, 3 and 4 in the above list form a hierarchy. In this hierarchy, each detected face belongs to a complete trajectory of detections across multiple frames. As such, trajectories consist of detected face bounding boxes. In turn, clustering puts each trajectory into a cluster. Thus, clusters consist of one or many trajectories. In the remaining parts of this thesis, these concepts are mentioned a lot. In any event, the hierarchy remains the same throughout.

It is important to note that none of the aforementioned steps require a certain implementation or algorithm to work. They are therefore assumed

[1] Appendix C shows the actual file structures used to save the listed entities in the baseline implementation.

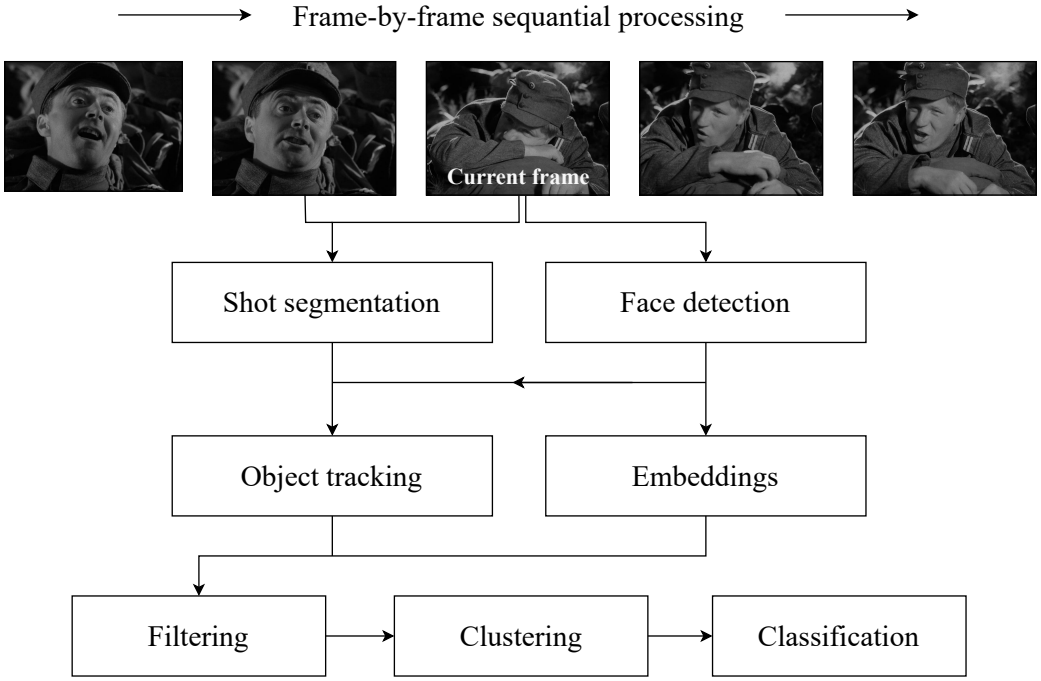

Figure 3.2: Schematic of data flow in the proposed extraction pipeline. Many processing steps use data from one or more other steps, as indicated by the arrows. The multi-object tracker, which is used to track facial bounding boxes, and shot segmentation methods look at data from the current and the previous frames.

to be individually generic and interchangeable. As such, any individual step can optionally be improved by changing to a different algorithm. A good set of baseline options is given in Chapter 4, while in this chapter the minimum requirements are given for the full extraction pipeline.

3.2.1 Face detection

The initial step in the pipeline is face detection. A suitable face detector is only required to detect rectangular bounding boxes of individual faces in each frame of the video. More formally, for a given frame f_i in the movie with k faces in that frame, the face detector extracts the set of bounding boxes

$$B_i = \{b_{i,1}, b_{i,2}, \ldots, b_{i,k}\}. \tag{3.1}$$

If no human faces are found, B_i is the empty set $\varnothing$. Within a set

B_i bounding boxes $b_{i,j}$ are expressed as pixel coordinates of the upper-left (x_1, y_1) and bottom-right (x_2, y_2) points of each rectangle:

$$b_{i,j} = (x_1, y_1, x_2, y_2)_{i,j}. \tag{3.2}$$

Throughout this thesis, many concepts are used to discuss individual bounding boxes as found by face detection in this way. Depending on the context, concepts such as "images", "faces", "feature vectors" or "embeddings" are all used in addition to "bounding boxes". All of these are directly linked to one detected face in a single frame, unless otherwise specified. Thus, phrases such as "images in a trajectory" or "vectors in a trajectory" are often used.

3.2.2 Multi-object tracking and trajectories

The general idea with multi-object tracking is to track individual objects or faces. The tracker exploits the fact that faces appearing roughly in the same place in consequtive frames will together form a trajectory t of faces. This abstraction is useful since it allows us to bundle multiple detected images of a single actor together without doing any clustering at all. This is one of the main contributions of this thesis: using the time dimension of films to group face images efficiently. As shown in Chapter 5, this has big implications on annotation efficiency. By using this idea, most images of faces in a film will never have to be shown to the annotator. Instead, only one or two representative images are shown for a trajectory, which massively compresses the labeling task as a whole.

Trajectories can be denoted as a sequential list of bounding boxes from different frames. Thus, the required output of a single object tracker will be in the following form:

$$t = \{b_i, b_{i+1}, \ldots, b_{i+n-1}\}, \text{ where } n = |t|. \tag{3.3}$$

In Equation (3.3), the in-frame index j for a bounding box $b_{i,j}$ is omitted for notational clarity. The trajectory length is denoted $|t|$ and equals the number of all bounding boxes in the trajectory. This is also the number of frames, in the movie, that is covered by the trajectory.

In this extraction pipeline the idea of tracking-by-detection is used, where an object or face detector is used to first detect faces separately. Various methods (e.g., [43][35][9]) within this body of research have been proposed, and most will be applicable here.

Within the tracking-by-detection paradigm, an object or face detector is used first to detect individual bounding boxes in frames. The tracker then

stitches them together into a trajectory. This is a convenient approach to apply in this pipeline since a face detector is being used regardless.

After initial detection of faces in a frame, the detected bounding boxes are passed to the multi-object tracker. The tracking algorithm then has to solve the assignment problem to determine whether a detected face can be assigned to a known trajectory given a utility metric. At any given point when processing the movie frames, the object tracker stores a reference to two sets of trajectories. The first set is the complete set T of all observed trajectories so far. The second is the list of *active trajectories* $T_a \subseteq T$ that are still being tracked and appended to. For the active trajectories T_a and a set of bounding boxes B_i from a frame, the multi-object tracker assigns each bounding box $b_{i,j} \in B_i$ such that the aggregate utility function

$$U(B_i) = \sum_{b \in B_i} u(b, f(b)) \tag{3.4}$$

is maximized. The assignment is then determined by the mapping function f which maps individual bounding boxes from B_i to trajectories within T_a. In practice, a good utility function u can be derived from intersection over union (IoU) or the Jaccard index

$$J(b_1, b_2) = \frac{A(b_1) \cap A(b_2)}{A(b_1) \cup A(b_2)}. \tag{3.5}$$

The Jaccard index, in this context, gives the ratio of area intersections to area unions of two bounding boxes b_1 and b_2. For object tracking, one computes the IoU between the last bounding box in a trajectory that is being tracked and the newly detected bounding box whose utility is being measured.

Additionally, note that f can not be a bijection if B_i and T_a are not equal in cardinality. If multiple bounding boxes are mapped to the same trajectory, only the one with the highest utility is added to the trajectory. In this way, any detections that can not be assigned to a current trajectory are used to form the starting point of a new trajectory. As such, f does not map to empty or new trajectories, but new trajectories should be added after the optimal mapping is computed. Similarly, trajectories that have not had a new bounding box assigned for some number of newly processed frames expire. These expired trajectories are passed on in the pipeline, but are no longer followed by the multi-object tracker. The reader is advised to study Section 4.3 for the full algorithm that implements this idea.

3.2.3 Shot segmentation

One problem arising from simple multi-object tracking is that trajectories can cross shot cuts. This happens at sharp shot cuts when an actor's face appears in a similar bounding box to another actor's before the cut. To address this, the extraction pipeline performs shot segmentation of the video to detect the cuts in the video. When at a frame f_i a cut is detected, the multi-object tracker resets the active trajectories to the empty set. Thus, any face detections found in the frame after the shot cut will form new trajectories and not be appended to the previously active ones.

The end result of this procedure is that two trajectories are formed at a point where originally there would have been only one. In practical terms, imagine a video has a trajectory

$$t_1 = \{b_{1,1}, b_{2,1}, b_{3,1}, \ldots, b_{n,1}\}, \tag{3.6}$$

and a shot cut is detected at frame f_3. This indicates that a transition occurred between frames indexed as 2 and 3. With shot segmentation, it is determined that the trajectory should not continue further from frame f_2. Instead, we segment the trajectory into two trajectories

$$\begin{aligned} t_2 &= \{b_{1,1}, b_{2,1}\} \text{ and} \\ t_3 &= \{b_{3,1}, \ldots, b_{n,1}\}. \end{aligned} \tag{3.7}$$

On a high level, shot boundaries can be divided into two categories. An abrupt cut occurs when a frame in one shot is followed by a different frame in a different shot in a completely different frame. The shots could be in the same scene, but the video is cut and discontinuous at that point. An example of an abrupt shot cut is shown in Figure 3.3. The second type of shot cut is a gradual transition, where the shot boundary occurs over multiple frames. [13]

A sufficient shot segmentation algorithm for this pipeline has two main qualities. First, unlike some algorithms for video segmentation (e.g., [59][62]), it can not be based on a specific compression scheme since the dataset (Section 5.1) contains different file formats and encodings. In terms of performance, however, any level of detected shot transitions is helpful. Considering precision and recall, there are different benefits to be had depending on how well the algorithm scores on these metrics. If the algorithm produces very few false positives, it has direct implications on the number of final trajectories. Since any positive detection could split a trajectory into two, false positives increase the number of trajectories in the saved output. This makes the annotation task more complex. As such, a sufficiently high precision score is

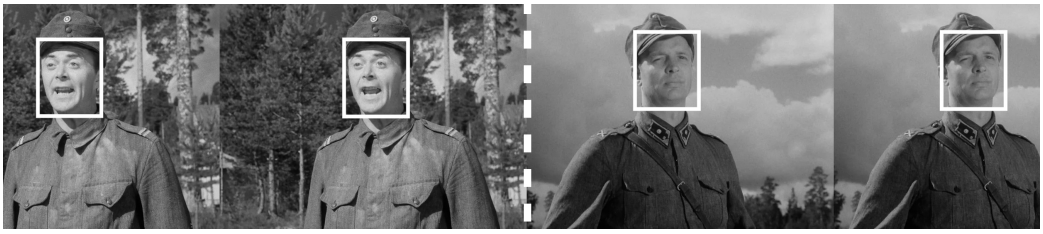

Figure 3.3: Four consequtive frames in a movie with an abrupt shot cut in the middle. The two actors' faces should not be put in the same trajectory, so segmentation is needed.

needed. At the other end, a good recall score helps us avoid long trajectories that erroneously cross a shot boundary.

The baseline shot cut detector presented in Chapter 4 is only able to detect abrupt shot cuts. However, abrupt shot cuts are still the most common type of shot transition. [13]

3.2.4 Face embedding

It has been shown in natural language processing that the use of efficient vector representations of words and n-grams has many useful applications on upstream tasks. [60][66][10]

Similarly for images of human faces, it is not appropriate to cluster and classify faces based on the raw image data. Instead, it is useful to embed the face images into Euclidean-space feature vectors. A function performing the embedding takes as its input the pixel data of the cropped-in image of a face and outputs a real-valued vector. It is desirable that the resulting embeddings have at least the following qualities:

1. The embeddings are invariant to pose and illumination of faces,
2. The embeddings are low in dimensionality,
3. The embeddings of similar faces have a small distance by L2 norm,
4. The embeddings of different faces have a large distance by L2 norm.

These characteristics are useful for a number of reasons. The dimensionality should be comparably low to the raw image vector for computational reasons. This also ensures that the dense vectors have more semantically meaningful features than the original pixel data. The pose and illumination invariance

refers to the embedding function's ability to produce similar vectors regardless of external factors in an image. Color, lighting and pose should not affect the output vectors, but only the face itself. Lastly, for clustering and face recognition, embeddings of similar faces should align closely in a Euclidean space. This allows the use of standard methods like k-means [54] for the clustering or SVMs for the classification (recognition) of faces. [74]

In the extraction pipeline any face embedding function that matches these criteria should work. The embeddings are produced for every detected face that occurs within a complete trajectory. With embeddings, there will be three different representations for each detected face. They are the bounding box $b_{i,j}$, the image of the face, cropped to the bounding box area, and the embedding vector $\boldsymbol{v}_{i,j}$.

3.2.5 Filtering

In some pipeline steps, mild filtering is applied to reduce noise in the overall output. The crucial parameters that are used for various types of filtering, are listed in this section.

Section 3.2.2 discussed the formation of complete trajectories. The object tracker initiates a new trajectory on every new face that is detected if it cannot be assigned to an active trajectory. Then, if no new faces are assigned to a new trajectory, its final length is $|t| = 1$. To avoid very short trajectories being passed on in the pipeline a minimum length (in number of frames) parameter t_{min} is introduced. The recommended default value is $t_{min} = 3$. Trajectories shorter than the minimum length are filtered out.

The baseline object tracker (see Chapter 4) is also able to deal with short occlusions or missed detections. This is useful if the object detector has poor performance and occasionally misses detections in frame sequences that would otherwise form a single trajectory. The maximum age parameter t_{max} specifies how long (in number of frames) a trajectory can be considered active if the object tracker does not assign new bounding boxes to it. The recommended default value is $t_{max} = 5$.

In practice face embeddings can rarely be useful if the original image resolution is too small. FaceNet authors note that embedding quality is slightly reduced at resolutions below 80×80 pixels for input images [74]. To accomodate for this, a minimum size parameter b_{min} with a default value of $b_{min} = 50$ pixels is used for detected faces. The filtering criterion for any detected face with bounding box b in the pipeline is

$$\min(b_{width}, b_{height}) \geq b_{min}. \tag{3.8}$$

For computational reasons it is also convenient not to save and compute embeddings for every detected face in a movie. Nearby frames should serve as reasonable approximations for feature vectors within the same trajectory. With this in mind, extracted images and features are saved (for annotation) every n_{skip} frames in a trajectory. The default value is set at $n_{skip} = 5$. Clustering, probabilistic classification and annotation are then performed on this 20% ($1/n_{skip}$) out of all detections. For the remainder of this thesis many sections discuss various processes being applied to bounding boxes and trajectories. In practice and for the baseline implementation, one can assume that this filter is always applied: only data from every n_{skip}th frame is actually being used.

3.2.6 Clustering

Clustering is an essential part of the active learning approach presented here. This section explains the role of pre-clustering in the extraction pipeline.

As the aim of the extraction pipeline is to make the human image annotation faster, there are two ways in which this is done. First, face images are bundled into trajectories as described in Section 3.2.2. The assumption is made that trajectories mostly contain images of a single actor, with low noise levels. Since trajectories can span hundreds of images, the annotation task is drastically reduced if a single image can be used to represent the full trajectory. Then, trajectories are batched together using clustering. Figure 3.4 showcases the hierarchy that detected faces are put into. In the annotation software, this hierarchy allows for showing full clusters that can be labeled in one instance, instead of separately. At this stage, only complete trajectories are used.

By clustering trajectories, the initial detections from the film are now organized in very big batches. In general, any clustering algorithm can be used here. The input to the algorithm would be the trajectories from a previous step, and the output would be a complete clustering. In practice, however, the algorithm should devise a distance metric between trajectories that is applicable for clustering. Additionally, it is necessary to restrict cluster sizes to a maximum size so that they can be easily shown in the annotation user interface. Section 4.4 shows a generic solution to this, that is built to work with k-means [54], and other well-known clustering algorithms.

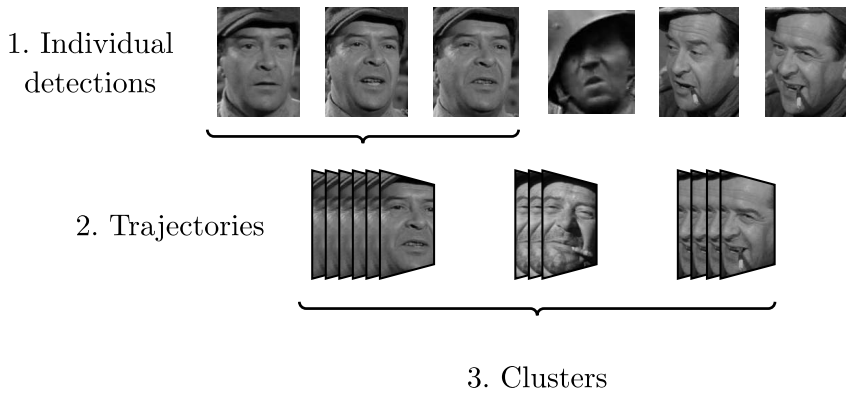

Figure 3.4: The entity hierarchy produced in the extraction pipeline. In a single movie, there can be 20000+ individual detections, approximately 1000 trajectories, and 100+ clusters.

3.2.7 Classification

In order to provide hints for users of the annotation software, supervised probabilistic classification is performed. The output of the final data extraction step will thus be a categorical distribution over class labels. The class labels correspond to actors in the specific movie being processed and the distributions are cluster-specific.

A suitable choice for the classification given the archived actor images from the MoMaF-2020 dataset, described in Section 5.1, is k-nearest neighbors (k-NN) [24]. First, face detection and embedding into vector space is performed on each ground truth actor image. Then, a set number of N vectors are chosen for each actor. Oversampling is performed for classes with fever than N images. However, in the archived images database used in our experiments, many actors do not even have one image that can be used as ground truth in this way. This is a big problem, which makes these preliminary predictions unreliable, particularly for less famous actors. However, the goal here is not to produce a final prediction for every detected face. Instead, the target is to produce baseline predictions to guide the annotation process later. For the target purpose, an incomplete set of ground truth images is deemed sufficient, especially as most main actors do have usable database images available.

Using k-NN, the cluster-level class label distributions are computed in the following steps:

1. We compute the probability distribution for individual detected faces first. For every feature vector corresponding to a single face image the probability of the image showing the actor with the ID c_i, is $P(i) = \frac{m_i}{k}$. In the calculation, k is a static hyperparameter setting the k nearest vectors to count. In turn, m_i is the number, out of the k vectors, with the class c_i. If n is the number of actors in the current movie, the probability mass histogram becomes:

$$p_i = p(x = c_i \,|\, \boldsymbol{m}) = \frac{m_i}{k}, \text{ where } \sum_{i=1}^{n} m_i = k. \tag{3.9}$$

2. For every bounding box in complete trajectories, we compute the mean of all probability vectors $[p_{i,1}, p_{i,2}, \ldots, p_{i,n}]^T$ corresponding to every feature vector in the trajectory. This mean vector over the frames in trajectory t will be the trajectory-level probability vector $\boldsymbol{p}_t$.

3. For every cluster, we compute the mean of all probability vectors $\boldsymbol{p}_t$ of the trajectories in the cluster. These mean vectors will be the cluster-level probability vectors (e.g., $\boldsymbol{p}_c$).

The cluster-level probability vectors then define the categorical distributions that are used by the annotation software to provide a first prediction to annotators.

3.3 Annotation software

Any active learning system involves an oracle. In this pipeline, human annotators are used to assign labels (actor identity) to clusters of detected faces. This is often an overlooked part of active learning systems, and oracles are often assumed to be perfect. In reality though, human labelers are imperfect and assign noisy labels, while also being slow [77]. Here, an annotation system is presented that aims to alleviate some of the aforementioned practical issues.

3.3.1 Labeling workflow

For the system design a web stack is chosen. This is convenient, since annotation tasks can be performed on any modern device that can render web pages. In this setup, a backend server reads the extracted data (as specified

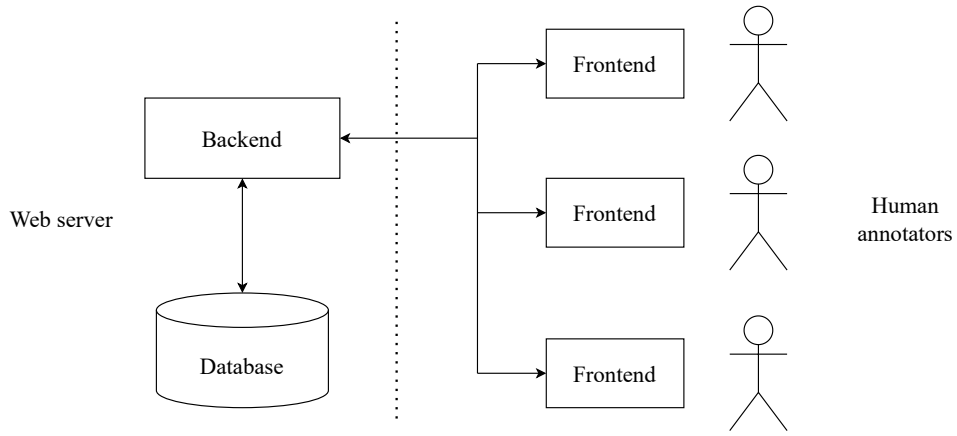

Figure 3.5: The annotation tool architecture. The system is simple, but still enables multiple active users annotating at once.

in Section 3.2) and serves a frontend labeling user interface. When users annotate using the frontend application, results are automatically sent back to the backend and stored in a database. React is used to build the frontend, while the backend uses FastAPI (Python) and a PostgreSQL database. A schematic of this basic architecture is shown in Figure 3.5.

Using a browser to access the system frontend, shown in Figure 3.6, the annotator is then instructed in the labeling task to proceed like:

1. Select a movie in the list to the left.
2. The main task is to select exactly one correct actor for a set of images. By means of clustering in the extraction, most sets will have images of only one person.
3. When presented with a cluster of face images:
   (a) Toggle individual individual images as false positives, if they do not show the main actor.
   (b) For clusters that seem noisy, optionally give the full cluster a *status* of “postponed”, “mixed” or “discarded”.
   (c) Finally, select the main actor from the list to the right.
4. Proceed by labeling more clusters in the movie, or continue to the next movie.

The full instructions given to users are more detailed than this. There are some considerations with this approach that should be highlighted. The

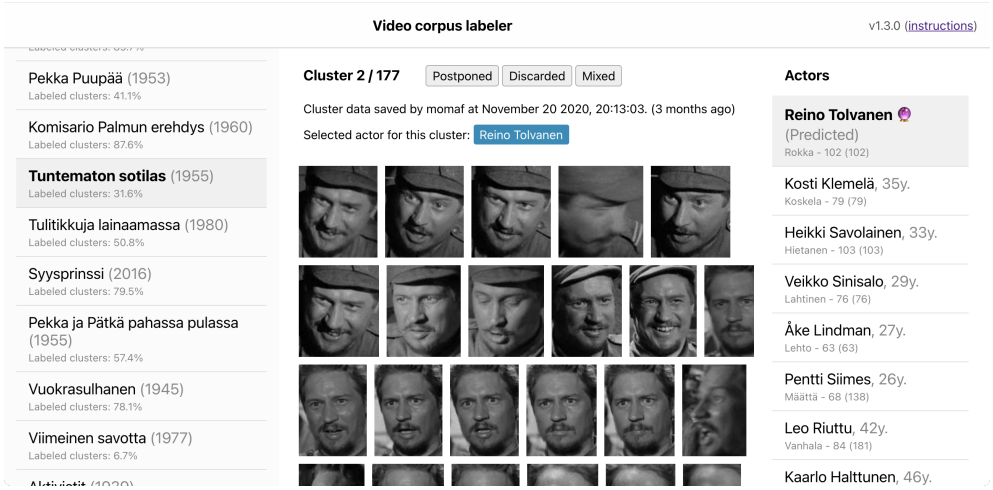

Figure 3.6: Screen grab of the frontend component of the labeling system. Between 1 and 60 face images are shown at once as a cluster to the oracle, but hundreds or thousands are labeled at once since each cluster contains many trajectories.

task is structured as a *batch mode* labeling task, with multiple images being labeled at the same time. Most previous studies in active learning have focused on labeling a single item at once [31]. Due to clustering, however, many images can be batched together into a single task so that annotation overhead is reduced.

Furthermore, since the dataset at hand had no ground-truth labels to begin with, even initial labels have to be created using our annotation tool. This is problematic since there are no benchmark labels, to which the newly created annotations can be compared when the tool is used. Because of this, a broad labeling effort is attempted where annotators first label a lot of clusters in a movie, instead of being provided with a targeted sample to begin with. This broad sample can be seen as a compromise between having a uniform sample of clusters that can be analyzed later and having a straightforward user workflow in the tool.

3.3.2 Noisy oracles

There are two primary sources of noise and corrupted data in the active learning pipeline. In the extraction step errors might occur, e.g. in face detection or when forming trajectories. Even when requiring a high detection probability of 0.95, false positives from face detection will likely end up in the final set of images. In this case, a false positive is a cropped image that is

not a human face. Similarly, trajectories can end up spanning multiple shots or include more than one actor. These are just some examples of noise that can occur in the extraction process. The second source of noise is the people that perform annotation; humans are imperfect labelers so this is taken into account in the annotation tool design.

In order to deal with this noise and allow for measuring annotation correctness, several features have been added to the labeling tool. First, on the image level the user can assign a status to individual faces. The allowed, color-coded statuses are defined as "regular", "wrong actor" (blue) and "discarded" (red). In practice, the users select the status by clicking on the cropped actor image in the annotation tool. The default status "regular" is always used for clean images that show the main actor of the cluster. If the image clearly shows a face, but it is not the main actor of the current cluster, the annotator should select the "wrong actor" status instead. False positives from the face detector, when the image does not clearly show a human face, are annotated with the status "discarded". This is one measure to be able to correct noise from the extraction step. Similarly on a cluster-level, the statuses "mixed", "postponed" and "discarded" are provided.

To assess the correctness of trajectories, a certain method is used in the user interface. Since trajectories span hundreds of images, it is not feasible to show all images to the oracle for labeling. Instead the first and last image is shown. That is, if a cluster has 30 trajectories, 60 images in total are shown in the interface. If either image is labeled with a non-regular status, it is concluded that the trajectory is noisy. This method provides a way of evaluating the object tracking performance, without increasing labeling cost too much.

Finally, users themselves are prone to label incorrectly sometimes. One way we may handle this is by labeling clusters multiple times, with different humans as the oracle. As such, the labeling system supports multiple users labeling every instance independently. This incurs additional labeling cost, but adds redundancy and confidence in the final labels. In practice, however, it is not feasible to label all or most clusters multiple times.

3.3.3 Labeling cost

The main way this active learning pipeline reduces labeling cost is achieved in the extraction step using clusters and trajectories. In addition to this, the annotation software itself can be used to make labeling faster. More specifically, we aim to speed up annotation by doing the following:

1. Useful hints, in the form of images, are provided in the user interface. Using these hints, the user will not have to exit the annotation interface to find out about an actor that was previously unknown to them.

2. Pre-classification results are used to clearly highlight names of probable, predicted actors in the user interface.

The key problem that the annotator is solving here is identification; setting the correct actor name for the visible cluster. This task is hard if the user is not familiar with the movie, which is likely especially with older films. In labeling a cluster like this, the annotator would then have to resort to finding example images externally, making the whole process slower. To assist the annotator, archived images of the actor can be shown within our tool, as hints, whenever requested. This process is shown in Figure 3.7. These images have actor labels from the database, but often include many people in one shot, so they are not suitable as such for training a classifier. However, when many of the archive images are shown to a person, they can easily tell which actor appears in all images and assign the correct cluster label. In a similar manner by using their mouse, the annotator can quickly preview corresponding full movie frames of individual face images in a cluster. Figure 3.8 shows an example of this scenario, where the full frame is being shown for a certain face image. There are thus two main type of image hints that can be used: actor archive images and individual frames from the film.

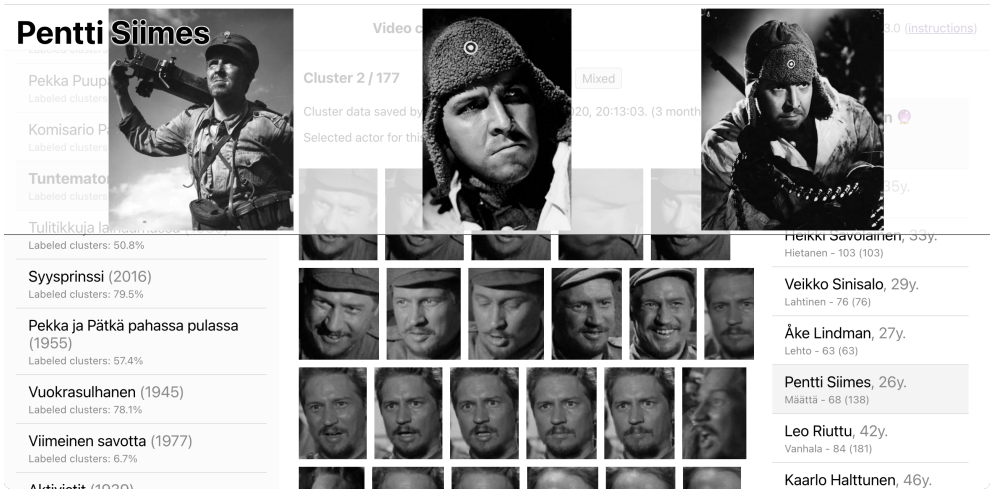

Figure 3.7: At the user's request, the labeling tool can show archive images of an actor.

Figure 3.8: At the user's request, the labeling tool can show the full movie frame that corresponds to a face image in the current cluster.

Finally, to further speed up labeling, pre-classification is used. This takes advantage of precomputed probability distributions as described in Section 3.2.7. If the probability of the cluster showing an actor c_i is greater than a threshold

$$p(X_C = c_i) > \tau_{show}, \tag{3.10}$$

then the specified actor is marked clearly as a probable candidate in the user interface. The actor's name is also moved to the top of the actor list, to the right in the annotation tool. This can speed up the annotation process by guiding the user to the most likely labels first. The heuristic default threshold $\tau_{show} = 0.4$ is chosen so that multiple machine predictions (two at most) can be shown if no class probability is very high. At the same time, we do not want to overwhelm the user with many highlighted possibilities by setting a low threshold and marking a lot of actor names as top choices.

In order to quantitatively measure the labeling cost, the labeling time is also measured at the time of annotation. This piece of data is stored along with class labels in the database, to be used later for evaluation.

Chapter 4

Methods

This chapter surveys the various methods used for the implementation of the extraction pipeline architecture. As stated in the previous chapter, many of the methods used provide stable baselines instead of trying to achieve the best performance possible in their individual tasks.

4.1 Face detection with RetinaFace

Recent methods for detecting faces use convolutional neural networks (CNNs), and RetinaFace [21] is no exception. RetinaFace is the baseline face detector used in the extraction step of the architecture presented in Chapter 3. This single-stage neural face detector is introduced in the following sections, and the main working principles of the model are shown. Face detection is closely related to object detection. Many works referenced here were originally created for object detection, and later adapted for faces.

4.1.1 Feature pyramids

A key quality of any object detector is scale invariance. Traditionally this was achieved using an *image pyramid*, in which the image itself is scaled to or processed at different resolutions. A single-scale detector can then be applied to each image of different scales, combined with a sliding window, to compute useful features on all scales. The image pyramid method has been used in past works such as the Viola-Jones detector (introduced in Section 2.2.1), but also in recent papers [98][27].

However, the image pyramid is a non-ideal method for a number of reasons. First, the scaled images are not semantically more meaningful than the original. Second, computing features based on each layer separately is slow.

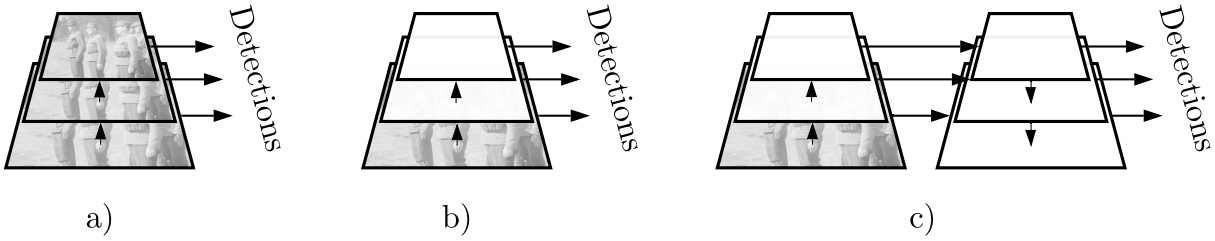

Figure 4.1: Three methods for scale invariance in object detection. White planes indicate convolutional feature maps. a) Image pyramid. b) Basic feature pyramid. c) FPN, such as the one in RetinaFace, with a sub- and upsampling pass. The architecture in the paper has five layers in its feature pyramid.

This is why many (e.g., [100][52][80][26]) new methods have instead opted to use *feature pyramids*. Figure 4.1 contrasts the simple image pyramid scheme against feature pyramids based on CNN-features. The rightmost architecture is the feature pyramid network (FPN) used in RetinaFace. The final upsampling, top-down pass allows the FPN network to encode semantically more meaningful features and to beat object detection benchmarks when used as a replacement for other feature detectors. [51]

The feature pyramid in RetinaFace consists of five layers. Its input is the original image, at a square resolution of 640×640 pixels. The subsampling pass uses pre-trained ResNet-152 [34] feature maps. The upsampling pass uses a 1×1 2D convolution for the lateral connection, and $2\times$ upsampling. Each layer has a constant depth of 256. As such, the (square) feature maps used for detection have dimensions $S_i \times S_i \times 256$, $S_i \in \{160, 80, 40, 20, 10\}$.

4.1.2 Detection from features

The detection part of the model runs independently for each of the five scales in the feature pyramid. The authors call the scale-wise detectors *context modules*, as they are able to use contextual signals through the wide receptive fields captured by the CNN feature maps. A contextual signal here would be, e.g., the presence of a body nearby a human face. For each context module a five-layer deformable convolution network (DCN) [20] is used.

To evaluate different regions of the feature map(s) RetinaFace employs translation-invariant *anchors* [70]. In practice, each anchor ties an $n \times n$ square, sliding window on a feature map to a reference rectangle at a fixed ratio. Thus, the context module input is a sub-window of its feature map,

and its output for bounding boxes is a rectangle $(x_{offset}, y_{offset}, w, h)$ that is *relative* to the anchor reference. Figure 4.2 shows this principle for the feature map on one scale in the feature pyramid; in the complete model there are five scales. For each anchor, there are three square reference rectangles that share their respective midpoints with the anchor point. Each square is bigger than the last by a factor of $2^{\frac{1}{3}} \approx 1.26$, giving 15 (5 feature maps $\times$ 3 scales) total reference rectangle sizes in RetinaFace. In terms of input image pixels, the smallest and largest squares are 16 and 406.37 pixels in width, respectively.

However, the final output of the model is not merely bounding box proposals. Instead, four tasks are learned jointly for the model. Predicting multiple tasks in parallel improves overall detection performance [33]. Table 4.1 lists all tasks that are learned in parallel for each context module. For the purposes of this thesis, we are mostly interested in the classification and bounding box regression tasks. Apart from this, RetinaFace outputs facial keypoints and a 3D mesh of the predicted face. The keypoints are five recognizable points on a human face: left eye, right eye, nose and the two corner points of the mouth.

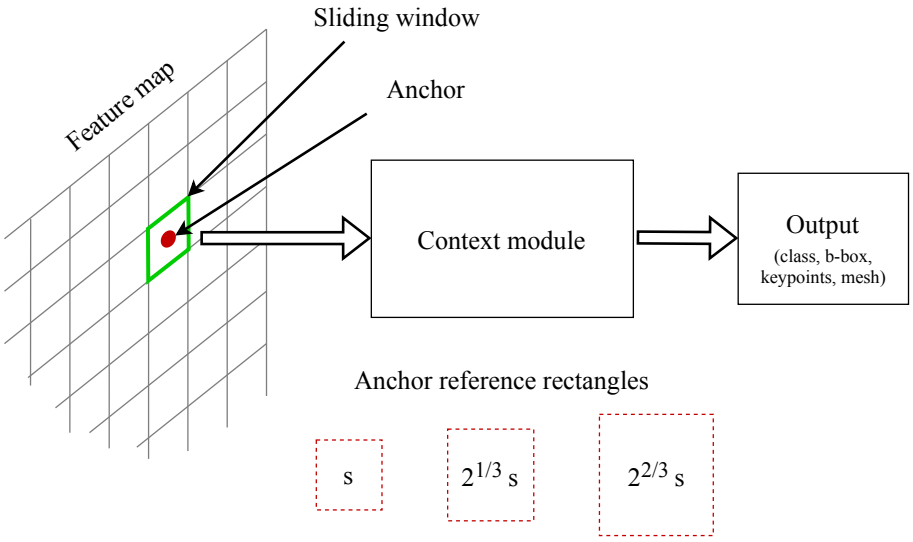

Figure 4.2: Sliding anchors in RetinaFace. The authors use a 1×1 sliding window on all scales of the feature pyramid. Each anchor is linked to three reference rectangles of different scales, further providing granular scales for the detector.

Task	Type	Size	Output explanation
classification	C	2	Face classification task. Output probabilities $(p_{face}, p_{background})$.
bounding box	R	4	Face bounding box $(x_{offset}, y_{offset}, w, h)$ in coordinates relative to the anchor reference.
keypoints	R	10	$(p_{l\text{-}eye}, p_{r\text{-}eye}, p_{nose}, p_{l\text{-}mouth}, p_{r\text{-}mouth})$. Coordinates of 5 key points on the face $p = (x, y)$.
3D mesh	R	144	An $\mathbb{R}^{128}$ vector for the shape and texture of a 3D mesh for the face. An $\mathbb{R}^7$ camera vector. An $\mathbb{R}^9$ illumination vector.

Table 4.1: Multi-task learning objectives in RetinaFace. The type column refers to classification (C) and regression (R) tasks.

4.1.3 Training and loss

For training RetinaFace, a multi-task loss is used that optimizes for all four objectives in parallel. The full loss function has the form

$$\mathcal{L} = \mathcal{L}_{cls}(p_i, p_i^*) + \lambda_1 \, p_i^* \, \mathcal{L}_{box}(b_i, b_i^*) + \lambda_2 \, p_i^* \, \mathcal{L}_{pts}(l_i, l_i^*) + \lambda_3 \, p_i^* \, \mathcal{L}_{pixel} \, . \quad (4.1)$$

The individual loss functions of the multi-task loss in Equation (4.1) differ somewhat. The loss function $\mathcal{L}_{pixel}$ is self-supervised; based on regression loss of a 2D projection of the predicted 3D mesh. For the bounding box loss $\mathcal{L}_{box}$ and the keypoints loss $\mathcal{L}_{pts}$, the smooth L1-loss function [27] is used. Face classification is a binary classification task, using binary cross-entropy loss. The weights λ_i weigh the auxiliary losses against the classification loss $\mathcal{L}_{cls}$. The ground truth class of sample i, denoted p_i^*, is 1 for positive samples and 0 otherwise. For negative samples, only the classification loss $\mathcal{L}_{cls}$ applies.

However, training the context modules using the multi-task loss is not straightforward, since negative samples outweigh positive ones by a substantial amount. That is to say: for the many possible sub-windows in an image, very few include a face. To combat bias, the authors of [21] selectively sample ground truth rectangles used for training. For a given anchor, the reference rectangles are compared to the ground truth face bounding boxes in the dataset using IoU, similarly to Equation (3.5). If the IoU value is larger than 0.5, the bounding box is considered a positive sample. If it is less than 0.3, the reference rectangle is a negative sample. By this procedure, the samples are selected so that the positive-to-negative ratio is at least 1:3. The model is trained using stochastic gradient descent (SGD) and backpropagation.

4.2 Embeddings with FaceNet

Schroff proposed a method, FaceNet, by which practical embeddings can be learned for faces [74]. This section summarizes the main working principles of the FaceNet model, that achieves state-of-the-art performance in face recognition.

4.2.1 Training system

While it can be thought of as a model, the authors of FaceNet describe their work as a "system"; a blueprint for how to train a model that achieves certain goals. Even multiple choices for the exact architecture are considered. Given an image x and an embedding $f(x)$ in $\mathbb{R}^d$, the main goals are laid out:

1. The Euclidean distance between images of the same person should be small,
2. For images of different people, the distance should be large.

It is noteworthy that if a model can produce embeddings such as these, it has two major benefits. One is to be able to use Euclidean distance as the face verification metric, and the other in that clustering of faces becomes possible with traditional methods.

However, the goals should be satisified regardless of external factors such as pose and illumination. To this end, the authors propose the use of a *triplet loss* which formalizes the design criteria as the training objective. This is similar to what is proposed by Weinberger [89]. The general idea is visualized in Figure 4.3.

The process involves *triplet* samples consisting of an anchor image, as well as a positive and a negative image. The positive image has the same identity as the anchor, and the negative does not. Then, in the training process, the embedding is trained so that the network learns to separate the anchor image from the negative image. On the opposite end, the positive image is brought closer in the Euclidan space. The triplet loss takes the form

$$\mathcal{L}_{triplet} = \sum_{i=1}^{N} \Big[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \Big]_+ , \tag{4.2}$$

where x_i^p, x_i^a and x_i^n denote the positive, negative and anchor images, respectively. The parameter α is the margin we want to enforce between the positive and negative samples.

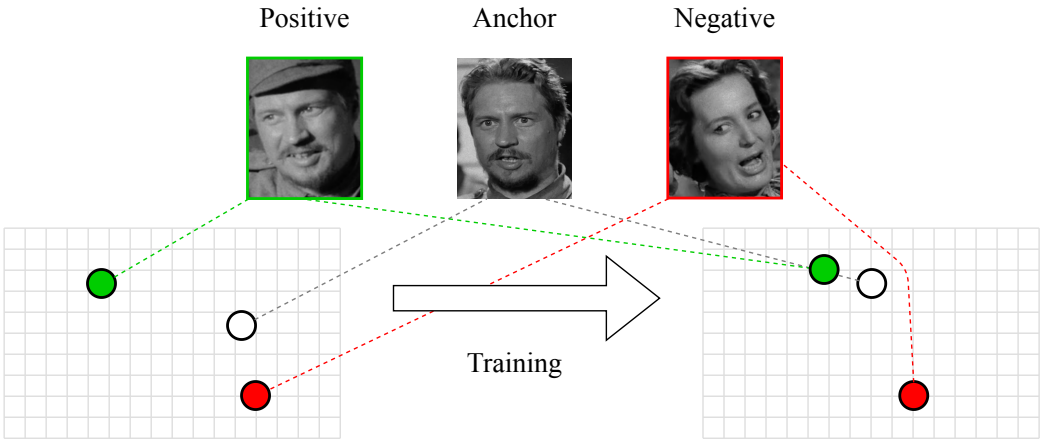

Figure 4.3: The triplet loss in FaceNet. The positive embedding moves closer to the anchor, while the negative gets "pushed away" during training.

For training the model, the authors discuss crucial steps in selecting the triplets. The focus is on selecting *hard* triplets, i.e. those that break the margin criterion set by α. A mini-batch size of 40 triplets is used in training, with stochastic gradient descent (SGD) as the optimizer.

4.2.2 Model

In practice, FaceNet can be trained with different architectures. In their paper, the authors evaluate two different architectures. One architecture for more powerful computers and one for mobile devices. The main difference is in model size and number of hidden layers. The high-level architecture is given by Figure 4.4. In this network, input images are first fed through a deep CNN that acts as the embedding. The output vector is constrained by L2 normalization so that $\|f(x)\|_2 = 1$.

The bigger model presented has a total of 22 layers. Among these, there are six 2D convolutional layers, four maxout [29] pooling layers and three fully connected layers at the end. All layers use the rectified linear unit

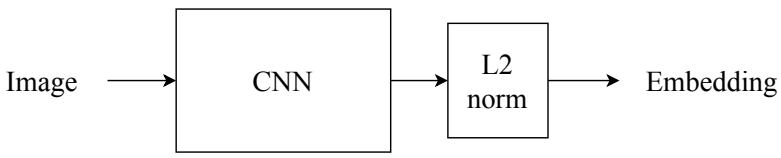

Figure 4.4: The FaceNet architecture.

(ReLU) [61] as their non-linear activation function. When evaluated for face verification against the "Labeled faces in the wild" (LFW) dataset [37], this model achieves 99.63% in accuracy.

4.3 Face tracking methodology

The multi-object tracker (MOT) is described here. It is the baseline method by which we are able to track faces across multiple frames to form trajectories. Inspired by the pragmatic approach of Bewley [6], an MOT with small ad-hoc improvements is devised. Specifically, the following modifications are made:

1. The RetinaFace face detector is used for accurate bounding box detections.
2. The ability to deal with short occlusions is added. Instead of breaking trajectories within one frame of lost detection, bounding box predictions are used to follow the face for up to $t_{max} = 5$ frames. In our use case of films the scenes are not often as busy as, say, a video feed from a surveillance camera.
3. The bounding box width-to-height ratio is not assumed to be constant.
4. A minimum of $t_{min} = 3$ initial detected frames is needed to begin a trajectory. Also, a trajectory cannot end in predicted bounding boxes, but has to end in a real detection.
5. Based on an external signal, trackers can be prematurely killed. This is useful in a non-continuous video stream with shot cuts, where objects should not be tracked past shot boundaries.

4.3.1 Tracking subtask

The MOT consists of many individual trackers that are used to form trajectories of bounding boxes. The tracking process is explained in this section.

To track objects, frames are processed sequentially, one by one. At the beginning of a film, there are no active trackers. When for the first time a bounding box $b_{i,j}$ of a face is detected, it is mapped to a state variable $\boldsymbol{x}_0$. If many boxes are detected in one frame, each is mapped to an independent state. The state vector at time step k has the shape

$$\boldsymbol{x}_k = [p_x, p_y, s, r, \dot{p}_x, \dot{p}_y, \dot{s}, \dot{r}]^T. \tag{4.3}$$

The point (p_x, p_y) is the center of the bounding box. The properties s and r denote the scale (area) and ratio (width-to-height) of the rectangle. The remaining four properties are the derivatives of the initial four. The derivatives are always initialized to 0 for any new state $\boldsymbol{x}_0$. Note that this format is slightly different from the bounding boxes that were used in Chapter 3, but the mapping is trivial both ways.

The Kalman filter [44] is used to update the state. A constant velocity model[1] is used here. This approximation is sufficient, since the face detector performs well and only short occlusions occur. Within this setting, a *tracker* is only the process by which the state is updated once every frame. The full lifecycle of a tracker is as follows:

1. Set initial state $\boldsymbol{x}_0 = [p_x, p_y, s, r, 0, 0, 0, 0]^T$ based on the initial bounding box b_0. Set the initial covariance matrix $\boldsymbol{P}_0$ as a diagonal matrix with small initial values. Initialize a new trajectory $t = \{b_0\}$.

2. Compute prior estimates for the next state $\boldsymbol{x}_{k|k-1}$ and $\boldsymbol{P}_{k|k-1}$ according to the Kalman filter prediction step. The corresponding bounding box prior is then $b_{k|k-1}$.

3. At every frame, new detected bounding boxes are matched to the $b_{k|k-1}$ prior for proper assignment. We do *not* assign the new bounding boxes based the actual previous bounding box, but based on this prior estimate for where the next one should appear.

   (a) If an assignment is made to this tracker, continue to the next step.

   (b) If an assignment is not made, add $b_{k|k-1}$ to the trajectory and go back to step 2. Here $b_{k|k-1}$ is only a prediction for where a bounding box should have appeared. Adding these prior estimates to the trajectories gives the tracker the ability to outlast short occlusions.

   (c) If a real update (step 4) has not been made $t_{max} = 5$ iterations, terminate the trajectory. Remove the last t_{max} bounding boxes from this trajectory, since they are merely predictions.

4. Using the assigned bounding box as the measurement, perform the update step of the Kalman filter to derive posteriors $\boldsymbol{x}_{k|k}$ and $\boldsymbol{P}_{k|k}$. Map $\boldsymbol{x}_{k|k}$ to a bounding box and append the bounding box b_k to the trajectory t. Go back to step 2.

[1] See Appendix B for full transition and covariance matrices $\boldsymbol{F}$, $\boldsymbol{H}$, $\boldsymbol{Q}$ and $\boldsymbol{R}$.

Using this process, trajectories are built by sequentially processing each frame of a film. Upon exhausting the full frame buffer, trajectories will be considered *completed* if they pass two criteria. First, the trajectory t has to contain at least t_{min} bounding boxes, where $t_{min} = 3$ by default. Second, the first t_{min} bounding boxes in a trajectory cannot be prior predictions, but rather have to correspond to a real, detected (step 4) bounding box. It is hypothesized that this will reduce the amount of detected false positive faces that then turn into noisy trajectories.

4.3.2 Assignment subtask

The previous section summarized how individual trajectories are formed whenever new detections are assigned to them. As such, new detections need to be assigned to an existing trajectory for it to be updated. This task is not trivial, since overlapping bounding boxes in busy scenes could potentially each be assigned to the same trajectory.

The specific assignment problem can be defined as follows. The current MOT has n individual active trackers or trajectories T_a. When the next frame is processed and m new bounding boxes B_i are detected, these should be assigned to a tracker or a new tracker should be initialized.

To assign faces optimally, the Hungarian algorithm [49] is used in conjunction with the IoU (see Section 3.2.2) kernel $J(b_1, b_2)$. To perform optimization, the $m \times n$ utility matrix is computed as

$$\boldsymbol{U}_{IoU} = \begin{bmatrix} J(b^1_{k|k-1}, a^1) & J(b^2_{k|k-1}, a^1) & \dots & J(b^n_{k|k-1}, a^1) \\ J(b^1_{k|k-1}, a^2) & J(b^2_{k|k-1}, a^2) & \dots & J(b^n_{k|k-1}, a^2) \\ \vdots & \vdots & \ddots & \vdots \\ J(b^1_{k|k-1}, a^m) & J(b^2_{k|k-1}, a^m) & \dots & J(b^n_{k|k-1}, a^m) \end{bmatrix}. \tag{4.4}$$

In this case, a variable $b^i_{k|k-1}$ is the bounding box prior from tracker i, and a^j is the jth bounding box detected in the new frame. Given the chosen kernel, the matrix will be sparse.

Given the utility matrix, the Hungarian algorithm is applied to find the optimal bijection $f : B'_i \to T'_a$, that maps a subset $B'_i \subseteq B_i$ of bounding boxes to a subset $T'_a \subseteq T_a$. If $|B_i| = |T_a|$, then $B'_i = B_i$ and $T'_a = T_a$. The subsets are the optimal subsets that maximize the total utility. As the last step, the mapping is pruned so that $J(b^i_{k|k-1}, a^j) \geq \tau_{box}$ for all matrix values. This is to say: the IoU has to be greater than $\tau_{box} = 0.5$ for an assignment to be possible at all. For any new bounding box that does not pass this constraint, a new tracker is initialized as according to step 1 in the tracking procedure,

as described in Section 3.2.2. This new tracker is added to the set of active trackers T_a. Similarly, if a tracker is terminated, it is removed from T_a and will not be considered for assignments in the following iterations.

4.4 Clustering methods

In order to cluster trajectories of detected bounding boxes, a set of special considerations is needed. In this section, a simple and effective method is shown.

Within the scope of this thesis, the clustering objective can be described in terms of two key problems. The proposed method needs to be able to:

1. Cluster trajectories of faces. This is problematic, since there is no obviously useful distance metric for trajectories. Individual bounding boxes however, correspond to feature vectors.
2. Set a maximum size c_{max} for clusters. When clusters are shown as a set of images in annotation, arbitrarily large collections cannot be shown.

In our method, a multifaceted approach is selected to deal with the problems individually. To be able to cluster trajectories, a representative vector is needed for each trajectory. This is similar to how Nguyen [64] uses representatives for classification. However, in this work a query synthesis (described in Section 2.4.1) approach is selected instead. Since the individual bounding boxes in each trajectory correspond to FaceNet feature vectors, a representative vector can be synthesized. It is reasonable to assume that the trajectory is the mean of all its components. Thus, from a given set of feature vectors V_j corresponding to a trajectory t_j, the representive vector $\bar{\boldsymbol{v}}_j$ is computed as

$$\bar{\boldsymbol{v}}_j = \frac{1}{|V_j|} \sum_{\boldsymbol{v}_i \in V_j} \boldsymbol{v}_i \,. \tag{4.5}$$

Using this approach, each trajectory can be represented with a Euclidean vector. This makes the clustering task trivial, and any classic clustering algorithm can be applied. Analysis in Chapter 5 suggests that Ward or k-means clustering have higher adjusted Rand index (ARI) [39] and purity scores than three other common algorithms. Thus, these two methods should be preferred. The reference implementation for this thesis uses k-means clustering.

The clustering algorithms come with the challenge that they require us to select the number of clusters in advance as a parameter. A naive choice could be based on the numbers of actors in the movie at hand. However, due to noise and imperfections in the data, this would likely lead to imperfect clusters that require even more manual work to fix later on. Instead, it is determined that an ideal size for a cluster in the labeling task is approximately $c_{size} = 22$. Using the annotation tool, described in Section 3.3, this many trajectories can be conveniently annotated in a single view. Based on this, and the total number of extracted trajectories n_t, the number of clusters is derived as

$$n_{clusters} = \frac{n_t}{c_{size}}. \tag{4.6}$$

Then, k-means with $n_{clusters}$ clusters and k-means++ [3] seeding is used to get a clustering. Finally, post-processing is performed to limit cluster sizes to better fit the annotation task. For this purpose, we heuristically set an upper bound on cluster sizes $c_{max} = 2 \cdot c_{size}$. Any clusters C_i produced in the initial k-means clustering that are greater than c_{max} in size $|C_i|$, are evenly split into smaller clusters. Thus, for the final clustering $\boldsymbol{C}$ we have

$$|C_i| \leq c_{max} \forall\, C_i \in \boldsymbol{C}. \tag{4.7}$$

The clustering now contains clusters of trajectories that can be conveniently shown in the annotation step.

4.5 Shot segmentation algorithm

In this section, a simple baseline shot segmentation method is presented. It is designed to detect abrupt shot cuts in videos. The system used is a modified version based on the work by Yi [92]. However, the modifications are minor. Thus, most of the method presented in this section is directly using the approach suggested by the original authors. The proposed technique uses feature extraction as a preprocessing step, followed by shot segmentation in the feature space.

Within scene and shot segmentation, a common feature used is mean absolute frame difference (MAFD) [97]. The feature value is computed using the pixel values of two consequtive frames in a video

$$MAFD_i = \frac{1}{w \cdot h} \sum_{x=1}^{w} \sum_{y=1}^{h} |i(x,y)_i - i(x,y)_{i-1}|. \tag{4.8}$$

In the above equation, w and h refer the width and height of the image in terms of pixels. The computation is based on luminance (intensity, i) values of pixels only. The range of intensity values for pixels is $[0, 255]$. The raw grayscale values are used for black-and-white videos, while a conversion is applied for RGB-colorized input data. In accounting for differences in color sensitivity of the human eye, the following RGB-to-luminance conversion formula is used

$$i(x,y) = 0.299 \cdot r(x,y) + 0.587 \cdot g(x,y) + 0.114 \cdot b(x,y). \tag{4.9}$$

In general, a large MAFD value might indicate that a shot transition has occurred. Figure 4.5 illustrates this with the feature values of frame transitions within the first 15 minutes of a film. The authors argue that while almost all shot transitions are associated with a large MAFD value, the opposite is not always true. Because of this, a detector built using only the

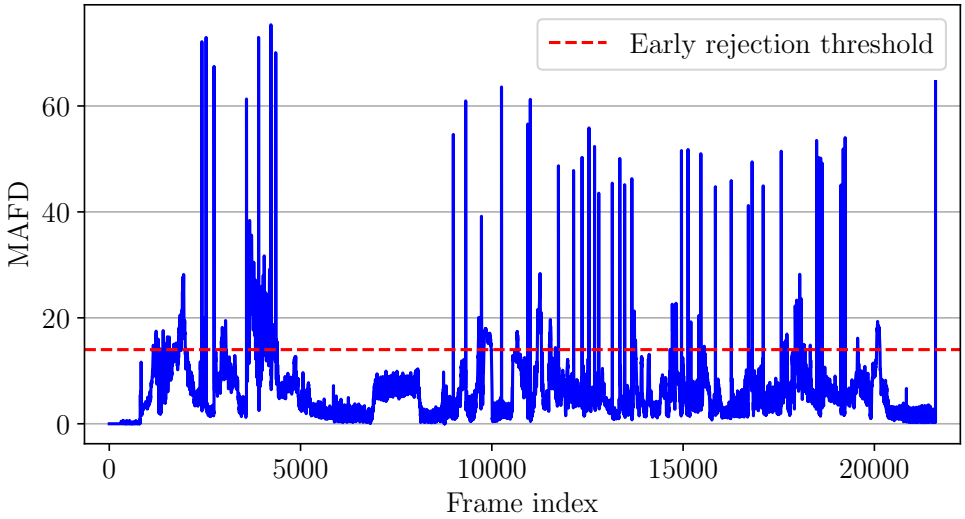

Figure 4.5: MAFD values for the first 15 minutes of the film Tuntematon sotilas (1955). A threshold value of 14 is used for early rejections, which can speed up the computation.

MAFD feature could have good recall, but would suffer from poor precision. Therefore, two other features are computed to allow for a more accurate detector. These features are the signed difference of MAFD (SDMAFD), and absolute difference of frame variance (ADFV). SDMAFD is derived simply by subtracting the previous MAFD value from the current one,

$$SDMAFD_i = MAFD_i - MAFD_{i-1}\,. \tag{4.10}$$

To compute ADFV, a relaxed definition of "frame variance" (FV) is introduced. Thus, the full calculation proposed is

$$\begin{aligned} ADFV_i &= |FV_i - FV_{i-1}| \\ FV_i &= \frac{1}{w \cdot h} \sum_{x=1}^{w} \sum_{y=1}^{h} |i(x,y)_i - MAFD_i|. \end{aligned} \tag{4.11}$$

To normalize the input image and to account for various illumination scenarios in input frames, histogram equalization is applied. The purpose of this method is to enhance the contrast of an image. Histogram equalization is an effective method of normalizing an image, which can be applied to various tasks within computer vision. [67]

Pseudo code for the final classifier is presented in Algorithm 4.1. The classifier returns *true* if an abrupt shot cut was detected between a frame and the frame before. Otherwise, it returns *false*. The algorithm uses both the original luminance-valued image and its histogram-equalized version. In the code, a variable with the suffix *eq* will denote that the feature was computed from the histogram-equalized images. If no suffix is present, the original luminance-valued pixels have been used.

This classification scheme is otherwise identical to the original work [92], except for an additional inequality added on line 13. The added inequality is `mafd < 100`. The decision to tweak the original algorithm is based on experiments conducted using a manually labeled dataset. Ultimately, by analysis shown in Section 5.5, it is determined that the algorithm performs adequately.

```
# Early rejection 1.
mafd = get_mafd(frame_buffer)
if mafd < 14:
    return False

# Early rejection 2.
mafd_eq = get_mafd(frame_buffer_eq)
if mafd_eq < 40:
    return False

adfv_eq = get_adfv(frame_buffer_eq)
sdmafd_eq = get_sdmafd(frame_buffer_eq)
if mafd_eq < 100 and mafd_eq > 58 and mafd < 100 and adfv_eq > 23:
    return True
if mafd_eq < 85 and mafd > 170:
    return True
if adfv_eq < 2 or sdmafd_eq < 5:
    return False
if mafd_eq > 50 and mafd > 35 and sdmafd_eq > 50 and adfv_eq > 50:
    return True

return False
```

Algorithm 4.1: Pseudo code the shot segmentation algorithm.

Chapter 5

Evaluation

So far a novel video extraction and labeling system has been presented and explained in Chapters 3 and 4. Here, a quantitative evaluation of the system performance is given. The key problems studied in this chapter relate to the correctness and cost associated with the labeling system. Additionally some numerical analysis is performed, with respect to individual methods used in this work. In Section 5.1 the dataset of Finnish feature films is introduced.

5.1 The MoMaF-2020 dataset

The overall system architecture defined in Chapter 3 is generic and can be applied to any video dataset. However, in practice the MoMaF-2020 (Movie Making Finland) dataset of Finnish films[1] is used throughout the work in this thesis. The original data source for this data is the Finnish National Audiovisual Institute, KAVI.

This dataset is not an ordinary machine learning dataset, since it does not contain any labeled data to begin with. That is, there are no predefined detected faces or labeled actors in the dataset. As such, the data on its own partly motivates this research, since proper labels will be required for us to be able to reason about actor appearances.

The MoMaF-2020 dataset consists of 256 movie files along with metadata about the movies themselves and the actors. The earliest movie in the dataset was released in 1921 and the newest release is from the year 2017. Figure 5.1 shows the distribution over release years in the dataset. Indeed, the collection of movies covers a wide range of Finnish film history with a majority of the movies being shot before the age of color film in Finland. The included video

[1]Appendix A lists all movies included in the dataset.

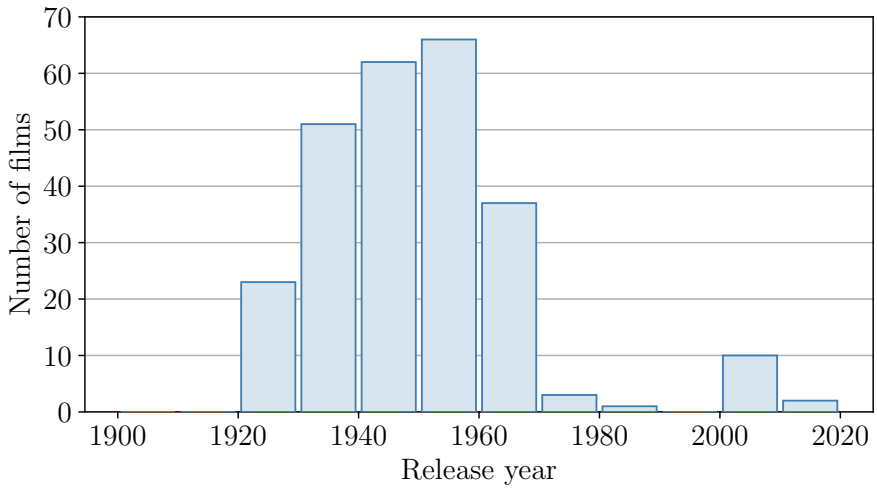

Figure 5.1: The distribution of movie release years in the MoMaF-2020 dataset. Many of the included productions are older black-and-white films.

files from KAVI also cover a range of video codecs and compression schemes, so algorithms that rely on certain compression artifacts [59] can not be used. This is the reason all methods used in this thesis work on raw pixel data, even if the general architecture does not require this.

Apart from raw movie data, the dataset also contains various metadata about movies and actors. Table 5.1 shows minimal metadata for the movie Tuntematon sotilas ("The Unknown Soldier"). The corresponding metadata about the top three (by billing order) actors in the same movie is shown in Table 5.2. For the purposes of the presented system architecture in the previous chapters, only the video annotation software makes use of movie metadata. This information is shown in the user interface, in order to help the human labeler.

Movie id	**Title**	**Release year**
113528	Tuntematon sotilas	1955

Table 5.1: Example of a movie from the dataset. "The Unknown Soldier" is a classic, so it deserves its own table.

Actor ID	Name	Born	Birthday	Role	Order
229234	Kosti Klemelä	Metsämaa	29.02.1920	Koskela	1
161566	Heikki Savolainen	Pori	09.04.1922	Hietanen	2
229235	Reino Tolvanen		1920	Rokka	3

Table 5.2: Metadata about the three main actors in The Unknown Soldier from 1955. The data is sometimes irregular and missing information.

Another key component of the movie dataset is the set of archived actor images from KAVI. In this work they are used for two purposes. First, the images are used in initial classification, as described in Section 3.2.7. The images are also used as hints for human annotators in the labeling process. In total, 16551 images are included in the collection. Each image is tagged with the unique movie ID of the film which it relates to, as well as IDs of all actors shown in the image. When only one actor is tagged as being present in the image, it can be used in supervised classification. However, 1657 unique actors appear in the MoMaF-2020 dataset as a whole, but only 543 have one or more archived images that can be uniquely tied to them in this way. That is to say: this data is helpful for classification, but not plentiful or varied enough to be used alone. Table 5.3 shows an overview of the most important statistics related to the MoMaF-2020 dataset as a whole.

A subset of the MoMaF-2020 dataset was run through the full extraction pipeline and annotated manually. In the following section, we survey some general statistics of this annotated subset.

Metric	Value	Explanation
films	256	Total number of films.
actors	1657	Number of credited actors, total.
actors / film	16.9	Average number of actors per film.
one-actor images	5735	Number of archive images with only one actor.
one-image actors	543	Number of actors that had at least one image where only they appeared.

Table 5.3: Statistics about the films in the MoMaF-2020 dataset and the related archive metadata. Note that many actors appear in multiple movies. This is why the “actors per film”-metric of 16.9 cannot be derived simply by dividing the number of films by the number of actors.

5.2 Labeled data

In Chapter 3, reducing the labeling effort is chosen to be one of the key targets of the architecture design. The main idea is to perform tracking in order to form trajectories of facial images. These can then be labeled by showing merely one or two representative images, thus providing an advantage in labeling. By performing extraction on a large subset of films within the dataset, the potential advantage can be quantified.

The key metrics of the extraction process are shown in Table 5.4 as mean values over the films. From annotation point of view, certain numbers are worth paying attention to. The average number of clusters on the movie-level is 91.8. The full labeling task for a single film is compressed into less than 100 batches or clusters, where every trajectory has been shown to the oracle. As films on average produce over 20000 valid images faces across frames, this reduction in potential annotation effort is substantial.

On average, the trajectory length is around four seconds. This falls within our expectations, as actors tend to appear in short shots and not in very long continuous sequences. However, by manual inspection it shows that the object tracker still fails to fully track some longer trajectories. While the baseline tracker used in this work is suitable for the task, this could still be improved.

In the following sections, a correctness and cost analysis will be provided. This analysis is based on labels assigned to a small subset of films. The actor labels were collected using the annotation tool described in Section 3.3. To be able to easily refer to this labeled dataset throughout the rest of this

Metric	**Value**	**Explanation**
clusters / movie	91.8	Avg. number of clusters in a movie.
trajectories / movie	1075.3	Avg. number of trajectories in a movie.
trajectories / cluster	11.7	Avg. number of trajectories per cluster.
images / movie	21 476	Total valid face detections in a movie on average, if every 5th frame is used.
images / trajectory	20.7	When filtering images to use every 5th detection as a saved image or embedding.
trajectory length	4.1	Avg. trajectory length, as measured in seconds instead of nr. of frames.

Table 5.4: Key metrics as produced by the extraction process, applied on the MoMaF-2020 dataset. An image, here, is a detected face that is part of a complete trajectory.

Metric	Value	Explanation
films	18	Number of movies with some amount of labels.
clusters	784	Total labeled clusters across movies.
trajectories	10 479	Total labeled trajectories.
images	252 695	Total labeled face images.
actors	162	Number of actors with some amount of labels.
annotators	6	Number of people that labeled the data.

Table 5.5: Annotated data produced by labeling a small subset of the film data. In the labeling process, exactly one actor label is assigned to each cluster.

	Full labeled dataset	Noisy dataset
clusters	784	332
trajectories	10 479	4730
images	252 695	82 514

Table 5.6: Evaluation subsets of the labeled data along with related core statistics. These subsets are referred to throughout Chapter 5. "Images" here refer to the total number of facial images, when every fifth detection is saved.

chapter, we will name it the *full labeled dataset*. The extent of this subset is shown in Table 5.5.

Additionally, the full labeled dataset will be compared against a "noisier" subset of itself. Let us call this subset the *noisy dataset*. In the noisy dataset, only clusters where the image status was changed for at least one image, are included. As such, every cluster in the noisy dataset will have at least one image with a status of "wrong actor" or "discarded". Table 5.6 compares the key metrics of the noisy subset against the full labeled dataset.

Comparing all the labeled data against a noisier subset offers some advantages in the analysis. If the status of some image was changed, the cluster is likelier to be "noisy", and not only have images of one actor. Probably, the noisy clusters also contain more invalid trajectories. In any event, the extraction process made an error since these clusters were not pure. Thus, we can use this noisy data to eliminate possible bias in the selections of clusters that the annotators chose to label. In the following analyses, the results of these labeled datasets will be compared to an extent, and the noisy dataset used as a lower bound of acceptable results.

5.3 Face detector

A key part of the presented active learning system is its ability to reduce labeling effort. To a large extent, this is achieved by presenting data in batches, but the quality of the data presented is important too. After all, it is not desirable to burden the annotator by showing noisy batches of images that could have been filtered automatically.

In this section we attempt to evaluate the noise levels in the face detections of the extracted data. In the baseline extraction pipeline, the RetinaFace face detector was run with a probability threshold at 0.95. Thus, we should expect to see at most around 5% of false positives in the batch of images presented to people for annotation. Additionally, an attempt was made to reduce this noise further in the extraction pipeline. That is, by requiring multiple sequential detections to be seen before a complete trajectory can be formed. In the baseline solution, $t_{min} = 3$ sequential detections are required to begin a complete trajectory. This should reduce the amount of false positives further, despite the high detector threshold.

It should be noted that the annotated data does not cointain negative samples at all. That is, samples that would have been classified as *not* being a human face. Naturally, we do not want to bother annotators with showing negative examples, so this can almost be seen as a requirement of the whole system. Nevertheless, we cannot directly determine the number of false negatives. Especially given the high threshold of the detector, there are certainly a number of faces that were falsely predicted as negative. However, the purpose of this thesis work is to efficiently extract labeled data from video by using active learning. To produce a good set of labeled data, we do not need all of the faces that appear in the movies. Rather, if a good portion of extracted faces can be given a label, our goal of annotating many faces is already achieved.

To evaluate face detection performance along with proper measures of uncertainty, a Bayesian approach is adopted. The variable to be modeled is the ratio of true positives to the number all labeled images. As such, an image in the labeled data is an actual face (true positive) with a probability p_{face}. The distribution corresponding to this random variable is thus a Bernoulli distribution, and

$$p_{face} = \frac{|true\ positive\ images|}{|all\ images|}. \tag{5.1}$$

In Bayesian terms, the likelihood is Bernoulli-distributed. To model p_{face}, a uniform Beta distribution $Beta(1, 1)$ is chosen as the prior. Then, by con-

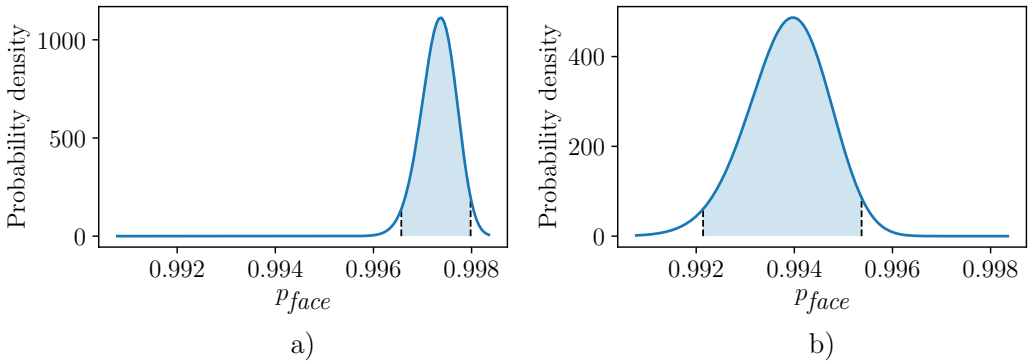

Figure 5.2: Posterior distribution of p_{face} for a) the full labeled dataset and b) the noisy dataset. The variance is low for both cases, so only a part of the support $[0, 1]$ is shown. The shaded area highlights the 95% credible interval.

jugacy, the posterior is $Beta(1 + x, 1 + n - x)$, where x is the number of true positives in the dataset, and n is the total number of labeled images. The resulting posteriors are plotted and presented in Figure 5.2. For the full labeled dataset and the noisy dataset, the distributions $Beta(20462, 55)$ and $Beta(8903, 55)$, respectively.

In analysing the posterior distribution for each labeled dataset, it is evident that the ratio of false positives to all samples, is very low. This is an encouraging result, meaning that only a small minority of clusters in the labeling process were noisy, with respect to detector performance. With high confidence, it can be concluded that the rate of false positives is at around 1% or less. Even for the noisy dataset, the Bayesian 95% credible interval is $0.992 \leq p_{face} \leq 0.995$. The variance is very small which yields strong confidence in the result. As expected, the variance is even smaller for the posterior of the full labeled dataset.

5.4 Trajectories

Using a similar method as in the previous section, the performance of the multi-object tracker is evaluated. In order to look at the quality of complete trajectories, the fact that two images were shown for each trajectory in the annotation task, is utilized. If one of the images in a trajectory has been assigned a different status, the trajectory is considered invalid. For example, an annotated trajectory might have one regular image showing the cluster's

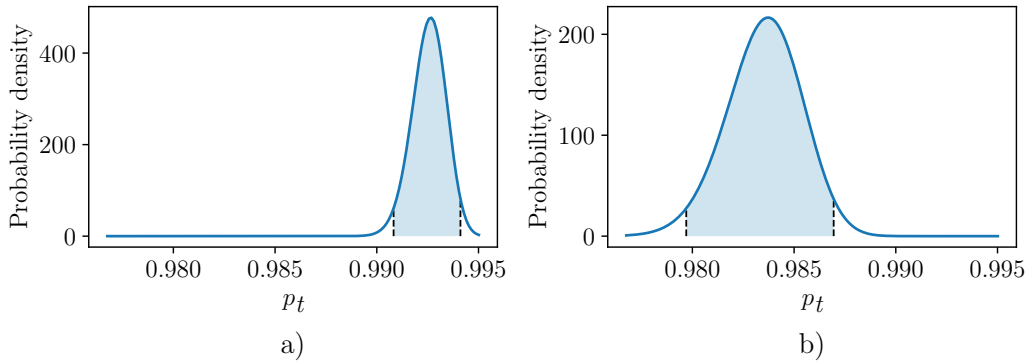

Figure 5.3: Posterior distribution of p_t for a) the full labeled dataset and b) the noisy dataset. The shaded area highlights the 95% credible interval.

main actor, while the second image is marked as "wrong actor". In this case, the trajectory is invalid since it contained faces of two different people. Using this definition, we are able to model the probability p_t of a trajectory being valid as

$$p_t = \frac{|valid\ trajectories|}{|all\ trajectories|}. \tag{5.2}$$

Again, using a uniform Beta-prior and the Bernoulli likelihood, the posterior is computed analytically. For the full labeled dataset and the noisy dataset, the posteriors are $Beta(10403, 78)$ and $Beta(4654, 78)$. These results are plotted in Figure 5.3.

Here, each dataset contains a total of 77 invalid trajectories. Even for the noisy dataset the posterior variance is very low, yielding an accurate estimate of p_t with the given data. The distribution mean is at $p_t = 0.984$, indicating that the overall number of invalid trajectories is very low, below 2%. This validates the performance of the baseline multi-object tracker in the extraction pipeline. The tracker is sufficient in producing valid trajectories to be shown in annotation.

5.5 Shot segmentation metrics

The shot segmentation algorithm used as a baseline is validated in this section. At its core, shot segmentation of abrupt shot cuts can be considered a binary classification task. At each frame, the classifier predicts whether or

Mean precision	Mean recall	Mean F1-score
0.994	0.767	0.862

Table 5.7: Classification performance for the baseline shot segmentation algorithm used in the extraction pipeline.

not a shot transition has occurred. Thus, to evaluate the performance of the shot segmentation, the precision, recall and F1-score metrics are used.

To perform evaluation, a small labeled dataset is prepared. The dataset contains frame indices of frames at which an abrupt shot transition occurred, as according to a human annotator. In total, 214 shot transitions are labeled for two different films within the MoMaF-2020 dataset. The computed metrics are shown in Table 5.7.

These overall performance metrics are satisfactory given the task at hand. Due to the tight tolerances of the classifier, the false positive numbers are kept very low. This yields a very high precision score. On the other hand, many true shot transitions are missed and are falsely predicted as negative, lowering the recall.

The shot segmentation system is only used as a supporting mechanism to prevent the multi-object tracker from forming trajectories across shot boundaries, so this is still acceptable. Improving the shot segmenter is viable, but other tasks should be prioritized in future work. Some examples of such tasks are provided in the conclusions of this thesis work, in Chapter 6.

5.6 Classification results

In performing annotation, it is helpful for the human labeler if predetermined guesses for the correct label are accurate. In the proposed labeling tool in Chapter 3, these predictions of the correct actor are highlighted and shown at the top of the actor list. In this section the performance of the k-NN classifier is evaluated against assigned labels.

In order to calculate the results, the predicted class label is considered to be the mode of the cluster-level categorical distribution over actors. Thus, the predicted class is

$$\hat{c} = \underset{c_i \in A}{\operatorname{argmax}}\ p(X_C = c_i). \tag{5.3}$$

The corresponding ground truth is the class label assigned by the human annotator.

Precision	Recall	F1-score
0.560	0.582	0.546

Table 5.8: Average precision, recall and F1-scores for all movies in the annotated subset of clusters.

To summarize the results of this multi-class labeling problem, results are averaged and calculated as:

1. For each film individually, calculate the confusion matrix.
2. For each film, calculate the class-wise precision, recall and F1-scores.
3. Compute weighted average of precision, recall and F1 across all films. The weights correspond to the number of clusters in each film.

The results, shown in Table 5.8, are somewhat surprising. For one, the dataset used for predictions is sparse and lacks images for many actors altogether. The dataset was not filtered to contain only images of an actor in one movie, but instead shows mixed appearances of a person along their career. At the pre-classification stage, this is a necessity due to the overall low number of usable reference images. Considering these factors, the results are good. This suggests that using a larger dataset of actor images, outside of the MoMaF-2020 dataset, would help the annotation task by allowing for better machine-generated guesses.
Nevertheless, it is hard to determine whether the results are unbiased. A plausible hypothesis is that human annotators prefer labeling clusters that are easy to label. In this scenario, participants would choose to label the same, "easy" clusters that were already accurately predicted in pre-classification, thus causing an artificially high F1-score. When asked about this, participants involved in the study denied that they would ignore hard cases, while preferring easy ones. In Section 5.9, this is further investigated by examining clusters based on labeling cost.

5.7 Clustering performance

For the active learning system presented in this thesis to be effective, clustering is an essential subtask. In using a batch learning approach, the system relies on batches being coherent and consisting of images with one actor only. This section presents a short analysis of clustering quality within the MoMaF-2020 database of known actor images. Furthermore, the actual clustering quality within the labeled dataset is assessed.

To measure clustering and batch quality, three widely used clustering quality metrics are utilized. These are the adjusted Rand index (ARI), cluster purity and cluster entropy. Each of the metrics compare a predicted clustering $\boldsymbol{D}$ to ground-truth clustering $\boldsymbol{M}$, in order to get a scalar score. In computing ARI, the original definition by Hubert [39] is used

$$\begin{aligned}
\mathrm{ARI}(\boldsymbol{D};\boldsymbol{M}) &= \frac{\sum_{i,j}\binom{n_{i,j}}{2} - \left[\sum_i \binom{n_i}{2}\sum_j \binom{n_j}{2}\right]\Big/\binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_i}{2}\sum_j \binom{n_j}{2}\right] - \left[\sum_i \binom{n_i}{2}\sum_j \binom{n_j}{2}\right]\Big/\binom{n}{2}}, \text{ where} \\
n &= \sum_i |D_i| = \sum_j |M_j|, \\
n_{i,j} &= |D_i \cap M_j|, \\
n_i &= |D_i| \text{ and} \\
n_j &= |M_j|.
\end{aligned} \tag{5.4}$$

This measure is closely related to the formal concept of accuracy within classification, but further adjusts for the randomness of some clustering algorithms, such as k-means. In the following definitions of purity and entropy, the notation for n, n_i, n_j and $n_{i,j}$ are used as in Equation (5.4). D_i and M_j refer to individual clusters within their respective clusterings. For purity, the following definition is used:

$$\mathrm{purity}(\boldsymbol{D};\boldsymbol{M}) = \frac{1}{n}\sum_j \max_i n_{i,j}. \tag{5.5}$$

Intuitively, the purity is high for clusterings where most clusters only have items belonging to a single real cluster. Within the use case of this thesis purity is an important metric. After all, it is highly preferred that the clusters of actors only have images of one actor and not multiple; even at the cost of having images of a main actor be spread across multiple clusters. However, it should be noted that purity on its own is not a satisfactory metric. For instance, putting all items into their own cluster within $\boldsymbol{D}$ will result in the maximum purity score of 1.0.

The clustering entropy, which is closely related to entropy within probability theory, is defined as

$$\begin{aligned}
H(\boldsymbol{D};\boldsymbol{M}) &= -\sum_j \frac{n_j}{n}\sum_i P(D_i;M_j)\log P(D_i;M_j) \\
P(D_i;M_j) &\approx \frac{n_{i,j}}{n_i}.
\end{aligned} \tag{5.6}$$

Since proper probabilities $P(D_i; M_j)$ cannot be derived, the value is approximated when calculating the clustering entropy values.

To inform the decision on which clustering algorithm to use in the extraction pipeline, a comparison of five common algorithms is conducted. In this analysis, a set of images of 30 different actors is used. This is a subset of the same set of archive images that were used to perform k-NN classification in the extraction pipeline. The results of the analysis are presented in Table 5.9. Notably, the authors of FaceNet [74] use agglomerative clustering, which is why this method was also used for evaluation of clustering quality later in this section. However, the authors of FaceNet do not further motivate why agglomerative clustering should perform better than other methods, in the space of facial feature vectors. In our analysis however, k-means clustering outperforms all other methods, as seen in Table 5.9. This suggests that it could be a viable alternative to agglomerative clustering, even if this simple analysis is a bit different from our use case of clustering trajectories. DBSCAN could otherwise be a desirable algorithm, since it does not require a preset number of clusters as an input parameter. With poor performance in this preliminary analysis though, its use cannot be condoned within this domain. With various hyperparameters, DBSCAN tended to produce a lot of very small clusters, contributing to its high entropy and low ARI scores.

The same feature vectors are projected into a 2D-space using Uniform Manifold Approximation and Projection (UMAP) [58]. This is visualized in Figure 5.4, where many clear, small clusters show even in the projected space. A central region with significant overlap between clusters exists, making the items in that region difficult to cluster correctly. This illustrates the central problem in the extraction pipeline: noisy feature vectors that cannot be easily put into pure clusters.

Algorithm	**ARI**	**Purity**	**Entropy**	**Notes**
agglomerative (e.g., [71])	0.529	0.637	1.261	Max distance, Euclidean distance.
Ward [88]	0.603	0.775	0.976	–
k-means [54]	**0.723**	**0.842**	**0.752**	30 clusters, k-means++.
DBSCAN [22]	0.167	0.465	2.259	Min. 2 neighbors, 0.8 max dist.
spectral (e.g., [63])	0.219	0.257	2.795	–

Table 5.9: Comparison of clustering algorithms using MoMaF-2020 archive actor images. The best result on each metric is highlighted in bold text.

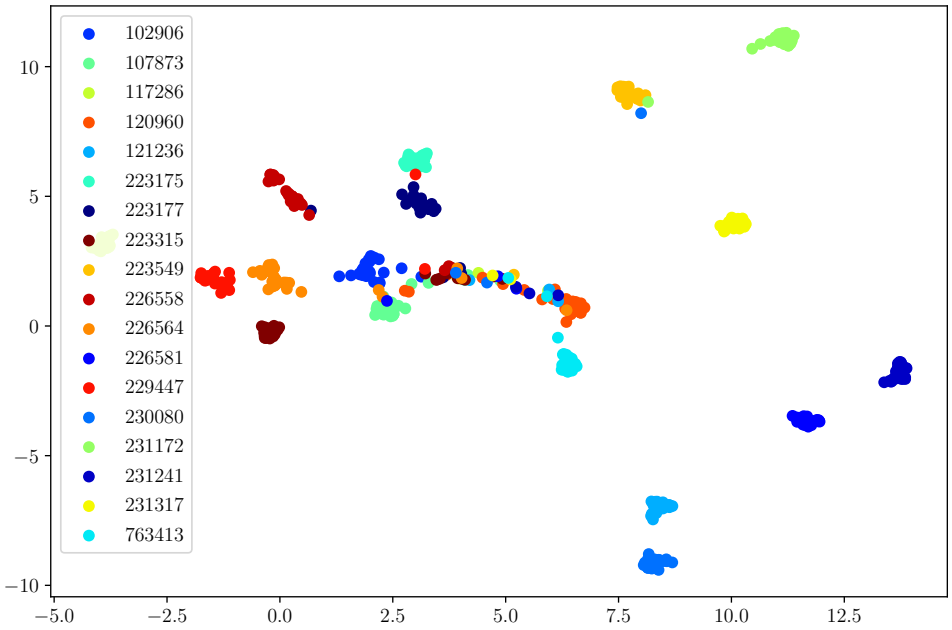

Figure 5.4: UMAP projections of feature vectors of images for 18 actors of the MoMaF-2020 dataset. Each actor has a unique ID number.

Finally, the same methodology is applied to the labeled data in an attempt to assess the quality of the actual clusters created in extraction. Reusing the same methods here is necessarily an approximation, since the annotated data does not perfectly separate noisy clusters into their correct subclusters. Note that the data used in this analysis comes from the extraction pipeline, and agglomerative clustering was used in this study. To approximate the true clustering, the image statuses are used to separate noisy clusters. Thus, within a predicted cluster that was annotated, images marked “regular”, “wrong actor” and “discarded” are put into their own subclusters. As such, a maximum of three subclusters is formed from each annotated cluster. The combined clustering formed from all computed subclusters is then used as the ground truth in comparing against the predicted clusters. To account for bias in the labeled data again, the same two labaled data subsets are used, as defined in Section 5.2.

The computed clustering metrics, shown in Table 5.10, indicate that the quality of labeled clusters is very high. In this analysis, an ARI or purity score of 1.0 is the maximum possible value, while the entropy is 0.0 for a perfect clustering. There are two possible explanations to this. First, the

Labeled subset	ARI	Purity	Entropy
Full labeled dataset	0.929	0.902	0.301
Noisy dataset	0.821	0.776	0.688

Table 5.10: Quality assessment of predicted clusters, as compared to data labeled by human annotators.

clusterings produced might indeed be high in quality; this is corroborated by some annotators used in the study. On the other hand, the data itself might be biased. However, even in using the noisy dataset where every predicted clustering is guaranteed to have at least two subclusters, the clustering quality metrics turn out to be acceptably good. This also lends credence to the argument that the quality of clusters in general is decent. As such, even noisy clusters are by and large pure, with a single main actor being dominant. To conclude, the results are satisfactory, confirming that pre-clustering can be successfully applied to assist an active learning approach of classifying film actors.

5.8 Quantitative analysis of labeling cost

A key indicator that will determine the viability of the presented active learning pipeline, is the labeling cost. To enable the measurement of labeling cost, the processing time is recorded during annotation. More specifically, the time measured for annotation is the screen time of a given cluster. That is, the timer starts when a user is first shown the batch of faces and stops when they switch to a different cluster.

This section will analyse the annotation performance of the system with time as the labeling cost metric. Thus, the overall performance is assessed and questions are posed about what affects the labeling cost. Once again, the two subsets of labeled data that were presented in Section 5.2 are used: the full labeled dataset and the noisy dataset.

The distribution of screen times for the full labeled dataset is presented in Figure 5.5. The distribution is Zipfian [103], and roughly 60% of clusters are labeled in less than 30 seconds overall. The median, 10th and 90th percentiles of the distribution are at 23.7, 7.0 and 106 seconds, respectively. This translates to an average labeling speed of around 0.99 visible images per second for the dataset at a whole. Using statistics from Section 5.2, it is equivalent to 10.2 saved face images per second. This is somewhat remarkable, and is achieved by the efficient clustering and bundling of images into trajectories.

The annotation times are contrasted with a similar plot from the noisy dataset, whose labeling time distribution is shown in Figure 5.6. Recall that

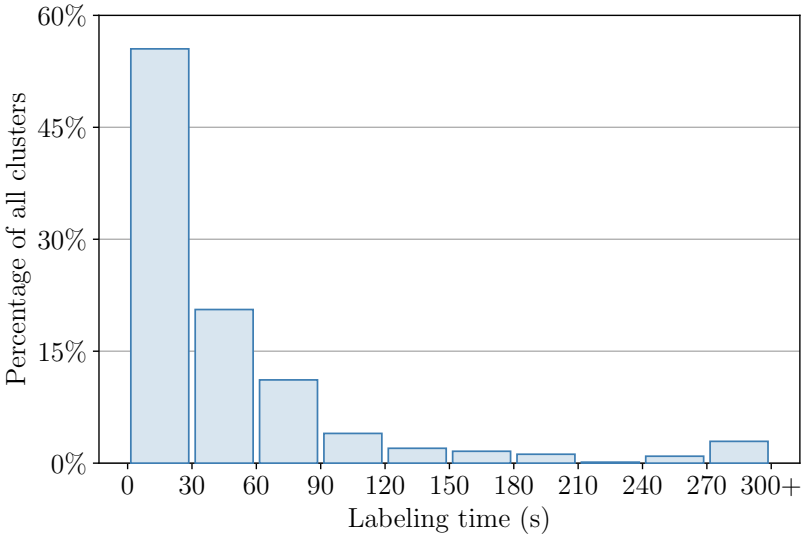

Figure 5.5: Normalized histogram of user screen time during the annotation process, for the full labeled dataset. The structure of this filtered and labeled dataset is defined in Section 5.2.

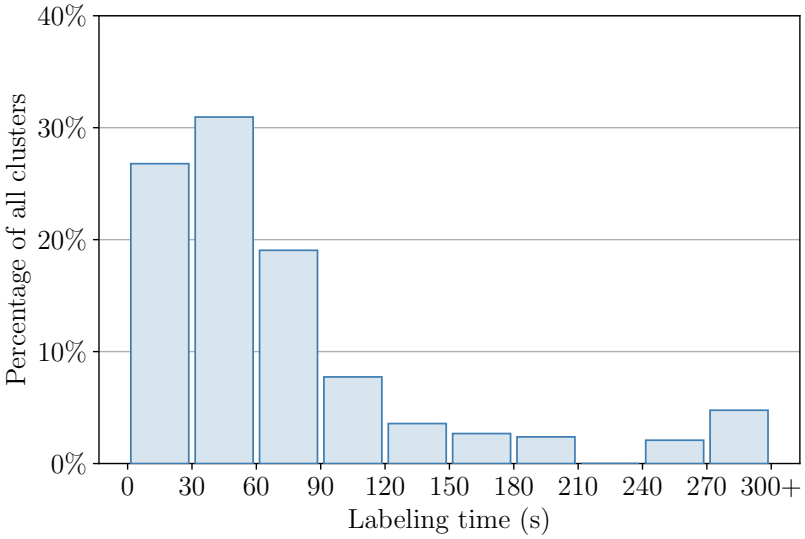

Figure 5.6: The noisy dataset equivalent to Figure 5.5.

this dataset is a subset of the full labeled dataset used for Figure 5.5. It is clear that any noise in a cluster that has to be marked by the annotator, contributes to a significant increase in labeling cost. As such, the median labeling time here is more than double the previous, at 50.8 seconds.

Apart from cluster noise, one might attempt to find other factors that increase labeling time. One such factor is cluster size, in terms of visible images shown to the user. The cluster size is plotted in Figure 5.7 against the labeling time, and the trend line is determined using linear regression (ordinary least squares, OLS).

The analysis shows that there is a correlation between labeling time and cluster size. A majority of small clusters, with less than 20 visible images, were labeled in less than 50 seconds. On the opposite end, all clusters with a labeling time of over 100 seconds are also larger than 20 visible images. As such, it turns out that all batches associated with a high cost are larger in size. Even so, this does not give a complete picture, since a sizable portion of large clusters are still labeled fast; in one minute or less.

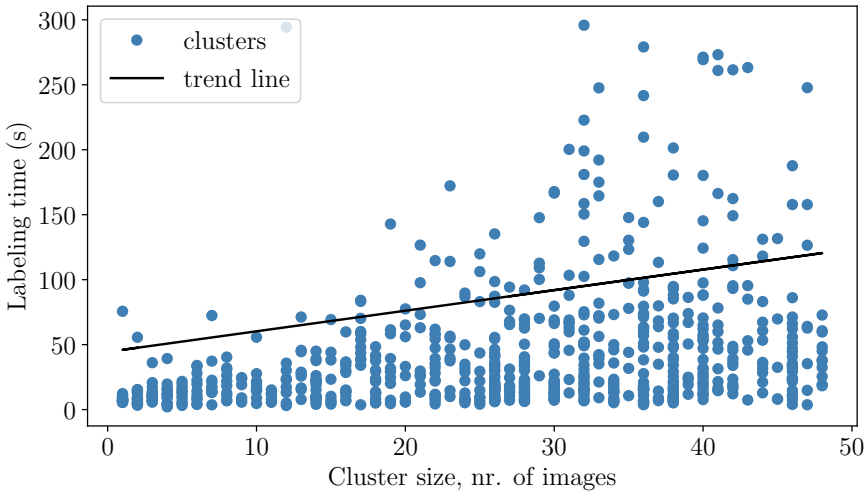

Figure 5.7: Labeling time as a function of cluster size. Clusters whose labeling times are above 300 seconds are not plotted.

5.9 Qualitative analysis of labeling cost

To further examine what makes the labeling task difficult, the images are qualitatively evaluated. In this experiment, three groups of images are considered. The groups are named and defined as follows:

1. *Fast* group: images in a cluster that was labeled within a time less than the 10th percentile (7 seconds).
2. *Slow* group: images in a cluster that was labeled within a time greater than the 90th percentile (106 seconds).
3. *Noisy* group: images in a cluster where one or more images had a status of "wrong actor" or "discarded".

These groups were sampled uniformly for images, which were then manually examined. A subset of these samples is shown in Figure 5.8. In the remaining part of this section, the findings of the examination are summarized.

Images in the fast group are generally well-behaved. Not all images here are bright and clear, but most feature front face views of famous actors that appeared in many shots. Surprisingly, it is not simple to determine which factors differentiate the images in the slow group from the fast group. Perhaps this group has a larger share of blurry images and difficult poses, which would make the task of labeling slower overall. Regardless, the differences, qualitatively, are small. It is likely that external factors, such as the annotator's knowledge of the particular film, might play an important role here.

Finally, it is easy to assess some of the factors that contribute to cluster noise. In the noisy group, many images are clearly blurry or otherwise invalid. This group includes actual paintings of people, and faces in poses that make it hard to produce quality feature vectors using FaceNet. Faces occluded by objects or very dark images are also common here. With images such as these, it is not possible to reliably determine the actor by the face alone.

The difficulty of reliably classifying the noisy images is one that is not easy to overcome. Since classification and clustering is done in the embedded vector space, the quality of embeddings becomes important. The authors of FaceNet claim pose invariance for their model [74], but even this has limitations. Clearly, the model cannot produce good embeddings if the input image is a mostly occluded face from a weird angle — a common sight in the movie dataset. Such a scenario would be difficult to classify correctly even for a person with specialized knowledge of the film in question. Similarly,

Figure 5.8: Random sample from image groups used in the qualitative evaluation. The groups are a) fast, b) slow and c) noisy. Some images appear blue due to encoding issues in digitized versions of black-and-white films.

embeddings for very blurry images suffer in quality, sometimes making it hard to correctly cluster such cases.

In the system proposed in this thesis, the correct clustering of difficult cases is alleviated by the formation of trajectories. This allows for filtering out feature vector noise by calculating the arithmetic mean of all embeddings in a trajectory. Future work could improve upon this by finding even more representative features. One possible heuristic here would be to only use feature vectors from images that are sharp, and where the face is fully visible.

Chapter 6

Conclusion

This thesis work presented a novel annotation system that applies active learning techniques to video data. The pipeline has two major components: a data extraction pipeline and a complete architecture for annotation of actors in feature films. These components together form a system that is effective at reducing the face data aquisition costs for videos. This is done by combining many different methods. A pre-clustering approach, together with object tracking, enables annotation in large batches of images. Additionally, an ad-hoc annotation tool assists the process by providing hints during labeling.

Overall, the presented system is generic and modular. As such, individual components, such as the face detector or object tracker, can be swapped out. Still, a good set of baselines was chosen and used for evaluation. Improvements to some baseline methods were also made. For instance, it was shown how generic clustering algorithms can be applied to trajectories of detected faces. In evaluation of the overall approach, it was shown that the system performs very well in terms of labeling effectiveness. On average, the labeling effort is reduced to less than 100 clusters per movie, as multiple trajectories can be shown to the annotator at once. Similarly, the extracted clusters are low in noise that would inhibit the work of assigning labels.

Nevertheless, certain parts of the system should receive additional attention in future work. One area of focus could be the rigid structure of the clusters, which perhaps should be made more flexible. Many participants that used the labeling tool expressed the will to assign multiple labels in one cluster; this is one way of receiving additional labels with little added overhead. As such, it would make the clusters less rigid by dividing them into smaller, labeled parts at the time of annotation. The evaluation also showed that pre-classification can yield acceptable results at the actor classification task. Thus, a larger set of archive images would further improve initial predictions, making the labeling task even faster.

Bibliography

[1] Aggarwal, C. C., Kong, X., Gu, Q., Han, J., and Philip, S. Y. Active learning: A survey. In *Data Classification: Algorithms and Applications*. CRC Press, 2014, pp. 571–605.

[2] Angluin, D. Queries and concept learning. *Machine Learning 2*, 4 (1988), 319–342.

[3] Arthur, D., and Vassilvitskii, S. k-means++: The advantages of careful seeding. Tech. rep., Stanford, 2006.

[4] Bartlett, M. S., Movellan, J. R., and Sejnowski, T. J. Face recognition by independent component analysis. *IEEE Transactions on Neural Networks 13*, 6 (2002), 1450–1464.

[5] Baum, E. B., and Lang, K. Query learning can work poorly when a human oracle is used. In *Proceedings of the International Joint Conference on Neural Networks* (1992), vol. 8, p. 8.

[6] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. Simple online and realtime tracking. In *Proceedings of the IEEE International Conference on Image Processing* (2016), pp. 3464–3468.

[7] Bledsoe, W. W. A Man-Machine Facial Recognition System —— Some Preliminary Results. *Technical report PRI*, 19A (1966).

[8] Bledsoe, W. W., and Chan, H. A man-machine facial recognition system — some preliminary results. *Panoramic Research, Inc, Palo Alto, California., Technical Report PRI A 19* (1965).

[9] Bochinski, E., Eiselein, V., and Sikora, T. High-speed tracking-by-detection without using image information. In *Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance* (2017), pp. 1–6.

[10] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics 5* (2017).

[11] Cao, Q., Shen, L., Xie, W., Parkhi, O. M., and Zisserman, A. VGGFace2: A dataset for recognising faces across pose and age. *arXiv*, 1710.08092v1 [cs.CV] (2018).

[12] Cevikalp, H., Neamtu, M., Wilkes, M., and Barkana, A. Discriminative common vectors for face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 1 (2005), 4–13.

[13] Chasanis, V., Likas, A., and Galatsanos, N. Simultaneous detection of abrupt cuts and dissolves in videos using support vector machines. *Pattern Recognition Letters 30*, 1 (2009), 55–65.

[14] Cohn, D. Neural network exploration using optimal experiment design. *Advances in Neural Information Processing Systems 6* (1993), 679–686.

[15] Cohn, D., Atlas, L., and Ladner, R. Improving generalization with active learning. *Machine Learning 15*, 2 (1994), 201–221.

[16] Condorelli, F., Rinaudo, F., Salvadore, F., and Tagliaventi, S. Heritage recognition in historical film footage using neural networks. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (2019), vol. XLII-2/W15.

[17] Cortes, C., and Vapnik, V. Support-vector networks. *Machine Learning 20*, 3 (1995), 273–297.

[18] Cox, I. J., Ghosn, J., and Yianilos, P. N. Feature-based face recognition using mixture-distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (1996), pp. 209–216.

[19] Crow, F. C. Summed-area tables for texture mapping. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques* (1984), pp. 207–212.

[20] Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. Deformable convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 764–773.

[21] Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I., and Zafeiriou, S. RetinaFace: Single-stage Dense Face Localisation in the Wild. *arXiv*, 1905.00641v1 [cs.CV] (2019).

[22] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *International Conference on Knowledge Discovery and Data Mining* (1996), vol. 96, pp. 226–231.

[23] Fischler, M. A., and Elschlager, R. A. The representation and matching of pictorial structures. *IEEE Transactions on Computers 100*, 1 (1973), 67–92.

[24] Fix, E., and Hodges, J. *Discriminatory Analysis-Nonparametric Discrimination: Consistency Properties*. Report, USAF school of Aviation Medicine, 1951.

[25] Freund, Y., and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences 55*, 1 (1997), 119–139.

[26] Ghiasi, G., Lin, T.-Y., and Le, Q. V. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 7036–7045.

[27] Girshick, R. Fast r-cnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1440–1448.

[28] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems* (2014), pp. 2672–2680.

[29] Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. Maxout networks. *arXiv*, 1302.4389v1 [stat.ML] (2013).

[30] Graf, H., Cosatto, E., Gibbon, D., Kocheisen, M., and Petajan, E. Multi-modal system for locating heads and faces. *Proceedings of the Conference on Automatic Face and Gesture Recognition* (1996), 88–93.

[31] Guo, Y., and Schuurmans, D. Discriminative batch mode active learning. *Advances in Neural Information Processing Systems 20* (2007), 593–600.

[32] Guo, Y., Zhang, L., Hu, Y., He, X., and Gao, J. MS-Celeb-1M: A Dataset and Benchmark for Large-Scale Face Recognition. In *ECCV* (2016).

[33] He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask R-CNN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 2961–2969.

[34] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.

[35] Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. Exploiting the circulant structure of tracking-by-detection with kernels. In *Proceedings of the European Conference on Computer Vision* (2012), pp. 702–715.

[36] Honka-Hallila, A., Laine, K., and Pantti, M. K. *Markan tähden: yli sata vuotta suomalaista elokuvahistoriaa*. Turun Yliopiston täydennyskoulutuskeskus, 1995.

[37] Huang, G. B., Mattar, M., Berg, T., and Learned-Miller, E. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in "Real-Life" Images: detection, alignment, and recognition* (2008).

[38] Huang, J., Child, R., Rao, V., Liu, H., Satheesh, S., and Coates, A. Active Learning for Speech Recognition: the Power of Gradients. *arXiv*, 1612.03226v1 [cs.CL] (2016).

[39] Hubert, L., and Arabie, P. Comparing partitions. *Journal of Classification 2*, 1 (1985), 193–218.

[40] Jafri, R., and Arabnia, H. R. A survey of face recognition techniques. *Journal of Information Processing Systems 5*, 2 (2009), 41–68.

[41] Joshi, A. J., Porikli, F., and Papanikolopoulos, N. Multi-class active learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 2372–2379.

[42] Kagan, D., Chesney, T., and Fire, M. Using data science to understand the film industry's gender gap. *Palgrave Communications 6* (2020).

[43] Kalal, Z., Mikolajczyk, K., and Matas, J. Tracking-Learning-Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*, 7 (2011), 1409–1422.

[44] Kalman, R. E. A New Approach to Linear Filtering And Prediction Problems. *ASME Journal of Basic Engineering* (1960).

[45] Kanade, T. Picture Processing System by Computer Complex and Recognition of Human Faces, 1973.

[46] Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 4401–4410.

[47] Kirby, M., and Sirovich, L. Application of the karhunen-loeve procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12*, 1 (1990), 103–108.

[48] Kohonen, T., and Lehtio, P. Storage and processing of information in distributed associative memory systems. *Parallel Models of Associative Memory* (1981), 129–167.

[49] Kuhn, H. W. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly 2*, 1–2 (1955), 83–97.

[50] Lewis, D. D., and Gale, W. A. A sequential algorithm for training text classifiers. In *Proceedings of the Conference on Research and Development in Information Retrieval* (1994), pp. 3–12.

[51] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 2117–2125.

[52] Liu, S., Huang, D., and Wang, Y. Learning spatial fusion for single-shot object detection. *arXiv*, arXiv:1911.09516 (2019).

[53] Lu, J., Plataniotis, K. N., and Venetsanopoulos, A. N. Face recognition using lda-based algorithms. *IEEE Transactions on Neural Networks 14*, 1 (2003), 195–200.

[54] MacQueen, J., et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (1967), vol. 1, pp. 281–297.

[55] Masi, I., Wu, Y., Hassner, T., and Natarajan, P. Deep face recognition: A survey. In *Proceedings of the Conference on Graphics, Patterns and Images* (2018), pp. 471–478.

[56] McCallumzy, A. K., and Nigamy, K. Employing EM and Pool-Based Active Learning for Text Classification. In *Proceedings of the International Conference on Machine Learning* (1998), pp. 359–367.

[57] McConnell, R. K. Method of and apparatus for pattern recognition. *Google Patents*, US Patent 4, 567, 610 (1986).

[58] McInnes, L., Healy, J., and Melville, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv*, 1802.03426v1 [stat.ML] (2020).

[59] Meng, J., Juan, Y., and Chang, S.-F. Scene change detection in an MPEG-compressed video sequence. In *Digital Video Compression: Algorithms and Technologies* (1995), vol. 2419, pp. 14–25.

[60] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv*, 1301.3781v1 [cs.CL] (2013).

[61] Nair, V., and Hinton, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. In *International Conference on Machine Learning* (2010).

[62] Nakajima, Y., Ujihara, K., and Yoneyama, A. Universal scene change detection on MPEG-coded data domain. In *Visual Communications and Image Processing* (1997), vol. 3024, pp. 992–1003.

[63] Ng, A. Y., Jordan, M. I., Weiss, Y., et al. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems 2* (2002), 849–856.

[64] Nguyen, H. T., and Smeulders, A. Active learning using pre-clustering. In *Proceedings of the International Conference on Machine Learning* (2004), p. 79.

[65] Nixon, M. Eye spacing measurement for facial recognition. In *Applications of Digital Image Processing VIII* (1985), vol. 575, pp. 279–285.

[66] Pennington, J., Socher, R., and Manning, C. D. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing* (2014), pp. 1532–1543.

[67] Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., ter Haar Romeny, B., Zimmerman, J. B., and Zuiderveld, K. Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing 39*, 3 (1987), 355–368.

[68] Pratap, V., Hannun, A., Xu, Q., Cai, J., Kahn, J., Synnaeve, G., Liptchinsky, V., and Collobert, R. Wav2Letter++: A Fast Open-source Speech Recognition System. *IEEE International Conference on Acoustics, Speech and Signal Processing* (2019).

[69] Qi, G.-J., Hua, X.-S., Rui, Y., Tang, J., and Zhang, H.-J. Two-dimensional active learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2008), pp. 1–8.

[70] Ren, S., He, K., Girshick, R., and Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems* (2015), pp. 91–99.

[71] Rokach, L., and Maimon, O. Clustering methods. In *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 321–352.

[72] Rowley, H. A., Baluja, S., and Kanade, T. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 1 (1998), 23–38.

[73] Salmi, H., Laaksonen, J., Ginter, F., and Kurimo, M. Movie Making Finland: Finnish fiction films as audiovisual big data, 1907–2017. `https://momaf.github.io/`, 2017. Online; accessed 2021-02-16.

[74] Schroff, F., Kalenichenko, D., and Philbin, J. FaceNet: A Unified Embedding for Face Recognition and Clustering. *IEEE Conference on Computer Vision and Pattern Recognition* (2015).

[75] Schultz, M., and Joachims, T. Learning a distance metric from relative comparisons. *Advances in Neural Information Processing Systems 16* (2003), 41–48.

[76] Settles, B. Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[77] Settles, B. From theories to queries: Active learning in practice. In *Active Learning and Experimental Design workshop* (2011), pp. 1–18.

[78] Smailović, J., Grčar, M., Lavrač, N., and Žnidaršič, M. Stream-based active learning for sentiment analysis in the financial domain. *Information Sciences 285* (2014), 181–203.

[79] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1701–1708.

[80] Tan, M., Pang, R., and Le, Q. V. EfficientDet: Scalable and Efficient Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2020), pp. 10781–10790.

[81] Tong, S., and Chang, E. Support vector machine active learning for image retrieval. In *Proceedings of the ACM International Conference on Multimedia* (2001), pp. 107–118.

[82] Turk, M. A., and Pentland, A. P. Face recognition using eigenfaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (1991), pp. 586–587.

[83] Uusitalo, K. *Suomen kansallisfilmografia, 1: 1907–1935.* Helsinki: Edita, 1996.

[84] Viola, P., and Jones, M. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2001), vol. 1.

[85] Viola, P., and Jones, M. J. Robust real-time face detection. *International Journal of Computer Vision 57*, 2 (2004), 137–154.

[86] Von Bagh, P. *Suomalaisen elokuvan uusi kultainen kirja.* Otava, 2005.

[87] Wang, M., and Hua, X.-S. Active learning in multimedia annotation and retrieval: A survey. *ACM Transactions on Intelligent Systems and Technology 2* (2011), 1–21.

[88] Ward Jr, J. H. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association 58*, 301 (1963), 236–244.

[89] Weinberger, K. Q., Blitzer, J., and Saul, L. Distance metric learning for large margin nearest neighbor classification. *Advances in Neural Information Processing Systems 18* (2005), 1473–1480.

[90] Wevers, M., and Smits, T. The visual digital turn: Using neural networks to study historical images. *Digital Scholarship in the Humanities 35*, 1 (2019), 194–207.

[91] Yang, M.-H., Kriegman, D. J., and Ahuja, N. Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 1 (2002), 34–58.

[92] Yi, X., and Ling, N. Fast pixel-based video scene change detection. In *IEEE International Symposium on Circuits and Systems* (2005), pp. 3443–3446 Vol. 4.

[93] Zafeiriou, S., Zhang, C., and Zhang, Z. A survey on face detection in the wild: Past, present and future. *Computer Vision and Image Understanding 138* (2015), 1–24.

[94] Zhang, B., Li, J., Wang, Y., Tai, Y., Wang, C., Li, J., Huang, F., Xia, Y., Pei, W., and Ji, R. Asfd: Automatic and scalable face detector. *arXiv*, arXiv:2003.11228 (2020).

[95] Zhang, C., and Zhang, Z. A survey of recent advances in face detection. *Microsoft Research* (2010).

[96] Zhang, F., Fan, X., Ai, G., Song, J., Qin, Y., and Wu, J. Accurate face detection for high performance. *arXiv*, 1905.01585v1 (2019).

[97] Zhang, H., Kankanhalli, A., and Smoliar, S. W. Automatic partitioning of full-motion video. *Multimedia Systems 1*, 1 (1993), 10–28.

[98] Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks. *IEEE Signal Processing Letters 23*, 10 (2016), 1499–1503.

[99] Zhang, S., Chi, C., Lei, Z., and Li, S. Z. RefineFace: Refinement Neural Network for High Performance Face Detection. *arXiv*, 1909.04376v1 [cs.CV] (2019).

[100] Zhao, Q., Sheng, T., Wang, Y., Tang, Z., Chen, Y., Cai, L., and Ling, H. M2Det: A single-shot object detector based on multi-level feature pyramid network. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), vol. 33, pp. 9259–9266.

[101] Zhao, W., Chellappa, R., Phillips, P. J., and Rosenfeld, A. Face recognition: A literature survey. *ACM Computing Surveys 35*, 4 (2003), 399–458.

[102] Zhu, X., Zhang, P., Lin, X., and Shi, Y. Active learning from data streams. In *Proceedings of the IEEE International Conference on Data Mining* (2007), pp. 757–762.

[103] Zipf, G. K. Human behaviour and the principle of least effort. *Addison-Wesley Press* (1949).

Appendix A

List of films

Full list of films in the MoMaF-2020 dataset.

Original title	**Release year**
Se parhaiten nauraa, joka viimeksi nauraa	1921
Finlandia	1922
Kihlaus	1922
Anna-Liisa	1922
Rakkauden kaikkivalta - Amor omnia	1922
Nummisuutarit	1923
Koskenlaskijan morsian	1923
Rautakylän vanha parooni	1923
Suursalon häät	1924
Myrskyluodon kalastaja	1924
Polyteekkarifilmi	1924
Ruotsin kuningasparin vierailu	1925
Suvinen satu	1925
Pohjalaisia	1925
Murtovarkaus	1926
Nuori luotsi	1927
Muurmanin pakolaiset	1927
Noidan kirot	1927
Vaihdokas	1927
Tukkijoella	1928
Kuningaspäivät Helsingissä toukok. 15-17 p. 1928	1928
H.M. Norjan kuninkaan Haakon VII vierailu Suomessa	1928
Meidän poikamme	1929
Korkein voitto	1929
Kahden tanssin välillä	1930
Kajastus	1930
Aatamin puvussa ja vähän Eevankin	1931
Rovastin häämatkat	1931
Tukkipojan morsian	1931
Olenko minä tullut haaremiin!	1932
Päikese lapsed	1932
Voi meitä! Anoppi tulee.	1933
Herrat täysihoidossa	1933

Ne 45000	1933
Minä ja ministeri	1934
Siltalan pehtoori	1934
Meidän poikamme ilmassa — me maassa	1934
Syntipukki	1935
Roinilan talossa	1935
Kaikki rakastavat	1935
VMV 6	1936
Onnenpotku	1936
Vaimoke	1936
Kaikenlaisia vieraita	1936
Mieheke	1936
Lapatossu	1937
Asessorin naishuolet	1937
Ja alla oli tulinen järvi	1937
Kuin uni ja varjo	1937
Juurakon Hulda	1937
Nuorena nukkunut	1937
Miehen kylkiluu	1937
Kuriton sukupolvi	1937
Koskenlaskijan morsian	1937
Tulitikkuja lainaamassa	1938
Niskavuoren naiset	1938
Poikamiesten holhokki	1938
Markan tähden	1938
Rykmentin murheenkryyni	1938
Sysmäläinen	1938
Vieras mies tuli taloon	1938
Laulu tulipunaisesta kukasta	1938
Olenko minä tullut haaremiin	1938
Jääkärin morsian	1938
Helmikuun manifesti	1939
Aktivistit	1939
Seitsemän veljestä	1939
Hätävara	1939
Eteenpäin — elämään	1939
Avoveteen	1939
Lapatossu ja Vinski olympia-kuumeessa	1939
Rikas tyttö	1939
Vihreä kulta	1939
Pikku pelimanni	1939
Punahousut	1939
Taistelun tie	1940
SF-Paraati	1940
Tottisalmen perillinen	1940
Kyökin puolella	1940
Anu ja Mikko	1940
Kersantilleko Emma nauroi?	1940
Tavaratalo Lapatossu & Vinski	1940
Jumalan myrsky	1940
Yövartija vain...	1940
Oi, kallis Suomenmaa	1940
Poikani pääkonsuli	1940

Kulkurin valssi	1941
Suomisen perhe	1941
Antreas ja syntinen Jolanda	1941
Poretta eli Keisarin uudet pisteet	1941
Viimeinen vieras	1941
Kaivopuiston kaunis Regina	1941
Morsian yllättää	1941
Ryhmy ja Romppainen	1941
Onnellinen ministeri	1941
Neljä naista	1942
Varaventtiili	1942
Kuollut mies rakastuu	1942
Synnin puumerkki	1942
Yli rajan	1942
Suomisen Ollin tempaus	1942
Hopeakihlajaiset	1942
Keinumorsian	1942
Jees ja just	1943
Syntynyt terve tyttö	1943
Neiti Tuittupää	1943
Valkoiset ruusut	1943
Tositarkoituksella	1943
Miehen kunnia	1943
Kirkastettu sydän	1943
Suomisen taiteilijat	1943
Herra ja ylhäisyys	1944
Kuollut mies vihastuu	1944
Dynamiittityttö	1944
Kartanon naiset	1944
Suomisen Olli rakastuu	1944
Linnaisten vihreä kamari	1945
Kolmastoista koputus	1945
Suomisen Olli yllättää	1945
Vuokrasulhanen	1945
Valkoisen neilikan velho	1945
Rakkauden risti	1946
Viikon tyttö	1946
Minä elän	1946
Loviisa — Niskavuoren nuori emäntä	1946
Pimeänpirtin hävitys	1947
Koskenkylän laulu	1947
Maaret — tunturien tyttö	1947
Kilroy sen teki	1948
Hormoonit valloillaan	1948
Ihmiset suviyössä	1948
Ruusu ja kulkuri	1948
Prinsessa Ruusunen	1949
Kalle-Kustaa Korkin seikkailut	1949
Kanavan laidalla	1949
Sinut minä tahdon	1949
Jossain on railo	1949
Härmästä poikia kymmenen	1950
Amor hoi!	1950

Radio tekee murron	1951
Rovaniemen markkinoilla	1951
Sadan miekan mies	1951
Gabriel, tule takaisin	1951
Kesäillan valssi	1951
Kuisma ja Helinä	1951
Niskavuoren Heta	1952
On lautalla pienoinen kahvila	1952
Kulkurin tyttö	1952
Jees, olympialaiset", sanoi Ryhmy	1952
Mitäs me taiteilijat	1952
Hän tuli ikkunasta	1952
Tervetuloa aamukahville eli tottako toinenkin puoli?	1952
Kipparikvartetti	1952
Rengasmatka	1952
Omena putoaa...	1952
Pekka Puupää	1953
Huhtikuu tulee	1953
Sillankorvan emäntä	1953
Pekka Puupää kesälaitumilla	1953
Siltalan pehtoori	1953
Hilmanpäivät	1954
Kaksi vanhaa tukkijätkää	1954
Morsiusseppele	1954
Sininen viikko	1954
Pekka ja Pätkä lumimiehen jäljillä	1954
Oi, muistatkos...	1954
Herrojen Eeva	1954
Minäkö isä!	1954
Työn ja kisan kädenlyönti	1954
Onnelliset	1954
Tuntematon sotilas	1955
Pekka ja Pätkä puistotäteinä	1955
Minä ja mieheni morsian	1955
Kiinni on ja pysyy	1955
Lähellä syntiä	1955
Pekka ja Pätkä pahassa pulassa	1955
Ratkaisun päivät	1956
Yhteinen vaimomme	1956
Riihalan valtias	1956
Muuan sulhasmies	1956
Pikku Ilona ja hänen karitsansa	1957
Nummisuutarit	1957
Pekka ja Pätkä ketjukolarissa	1957
Kuriton sukupolvi	1957
Pekka ja Pätkä salapoliiseina	1957
Miriam	1957
Vieras mies	1957
Pekka ja Pätkä sammakkomiehinä	1957
Niskavuoren naiset	1958
Nuori mylläri	1958
Pekka ja Pätkä Suez'illa	1958
Pekka ja Pätkä miljonääreinä	1958

Verta käsissämme	1958
Tirlittan	1958
Patarouva	1959
Kovaa peliä Pohjolassa	1959
Ei ruumiita makuuhuoneeseen	1959
Vatsa sisään, rinta ulos!	1959
Yks' tavallinen Virtanen	1959
Pekka ja Pätkä mestarimaalareina	1959
Sampo	1959
Kohtalo tekee siirron	1959
Taas tapaamme Suomisen perheen	1959
Komisario Palmun erehdys	1960
Autotytöt	1960
Oho, sanoi Eemeli	1960
Nina ja Erik	1960
Isaskar Keturin ihmeelliset seikkailut	1960
Pekka ja Pätkä neekereinä	1960
Skandaali tyttökoulussa	1960
Taistelujen tie	1960
Kaks' tavallista Lahtista	1960
Kankkulan kaivolla	1960
Kaasua, komisario Palmu!	1961
Rakas..	1961
Nuoruus vauhdissa	1961
Toivelauluja	1961
Olin nahjuksen vaimo	1961
Pikku Pietarin piha	1961
Pojat	1962
Yksityisalue	1962
Kolmen kaupungin kasvot	1962
Mäki Moore maailmanmestaruus	1962
Villin Pohjolan kulta	1963
Lauantaileikit	1963
Totuus on armoton	1963
Jengi	1963
Teerenpeliä	1963
Luonnon kätköissä	1963
Naiset	1964
Raportti eli balladi laivatytöistä	1964
Kielletty kirja	1965
Vaaksa vaaraa	1965
Tunteita	1965
Tänään olet täällä	1966
Johan nyt on markkinat!	1966
Pähkähullu Suomi	1967
Miljoonaliiga	1968
Mannerheim Suomen marsalkka	1968
Rottasota	1968
Päämaja	1970
Maunu Kurkvaaran Kujanjuoksu	1971
Viimeinen savotta	1977
Tulitikkuja lainaamassa	1980
Levottomat	2000

Lomalla	2000
Pahat pojat	2003
Vares – yksityisetsivä	2004
Eläville ja kuolleille	2005
Matti	2006
Valkoinen kaupunki	2006
V2 — Jäätynyt enkeli	2006
Kummeli Alivuokralainen	2008
Rööperi	2009
Syysprinssi	2016
Saattokeikka	2017

Appendix B

Kalman filter parameters

$$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0001 \end{bmatrix}$$

$$\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$$

Appendix C

File formats

File formats used by the extraction pipeline's baseline implementation. The reference implementation uses JSON. The schema examples are shown here in a YAML-like format for readability purposes.

```yaml
# trajectories.jsonl - all trajectories for movie 12345
— movie_id: 12345
  index: 0  # Trajectory index
  start: 123  # Start frame index
  len: 100  # Trajectory length
  # Trajectory bounding boxes
  bbs: [[[191,110,249,168],[185,107,244,168],...,[182,108,239,168]]]
  # Real detection or prior?
  detected: [true,false,...true]
...
— movie_id: 12345
  index: 1
  start: 1050
  len: 50
  bbs: [[[191,110,249,168],[185,107,244,168],...,[182,108,239,168]]]
  detected: [true,false,...true]
```

```yaml
# clusters.json - clustering for movie 12345
movie_id: 12345
clusters:
  - 58  # Cluster index of trajectory 0
  - 0   # Cluster index of trajectory 1
  - 0   # Cluster index of trajectory 2, etc.
  - 3
  ...
  - 86
```

```yaml
# features.jsonl - embeddings for all detected faces, separately.
— movie_id: 12345
  frame: 1800  # Frame index of the bounding box
  box: [185,107,244,168]  # Bounding box dimensions
  tag: 12345:1800:185_107_244_168  # Tag
  # 128-dimensional FaceNet embedding for the face in this bounding box
  embedding: [−0.07521, 0.01891, ..., −0.01309]
...
— movie_id: 12345
  frame: 1801
  box: [668,44,737,162]
  tag: 12345:1800:185_107_244_168
  embedding: [−0.001279, −0.02205, ..., 0.11240]
```

```yaml
# predictions.json - probability distributions. One for each
    cluster.
movie_id: 12345
predictions:
   "0": # cluster id
      "actor-00123": 0.003 # probability of actor with id 00123
      "actor-00124": 0.053
      ...
      "actor-00125": 0.033
...
   "99":
      "actor-00123": 0.003
      "actor-00124": 0.053
      ...
      "actor-00125": 0.033
```

```yaml
# scene_changes.json
movie_id: 12345
frame_indices:
  # Frame index, the frame after an abrupt shot cut occurred.
  - 100
  - 130
  - 2050
  ...
  - 200314
```