



A comparison of deep learning models for end-to-end face-based video retrieval in unconstrained videos

Gioele Ciaparrone¹ · Leonardo Chiariglione² · Roberto Tagliaferri¹

Received: 5 July 2021 / Accepted: 15 December 2021 / Published online: 5 January 2022
© The Author(s) 2022

Abstract

Face-based video retrieval (FBVR) is the task of retrieving videos that containing the same face shown in the query image. In this article, we present the first end-to-end FBVR pipeline that is able to operate on large datasets of unconstrained, multi-shot, multi-person videos. We adapt an existing audiovisual recognition dataset to the task of FBVR and use it to evaluate our proposed pipeline. We compare a number of deep learning models for shot detection, face detection, and face feature extraction as part of our pipeline on a validation dataset made of more than 4000 videos. We obtain 97.25% mean average precision on an independent test set, composed of more than 1000 videos. The pipeline is able to extract features from videos at ~ 7 times the real-time speed, and it is able to perform a query on thousands of videos in less than 0.5 s.

Keywords Video retrieval · Deep learning · Face recognition

1 Introduction

Video retrieval is the task of matching an input query with a set of videos, in order to select the videos that are relevant to the given query. Retrieval tasks often deal with one of two possible types of queries: text based or content based. Text-based queries look at metadata and data annotations such as file names, descriptions, or tags in order to determine the relevant items to retrieve. However, this relies on accurate and complete annotations, which are time-consuming to produce and are thus a rare occurrence in real-world scenarios. Content-based retrieval uses instead visual features extracted from images or videos in order to perform the matching. This makes content-based retrieval more desirable (and more accurate) in situations where accurate text-based annotations are not available.

In particular, this work focuses on the use of faces for the video retrieval task, i.e., the image of a face is used to retrieve all the videos in a given database that include the same person appearing in the query image. We refer to this as the task of *face-based video retrieval* (FBVR). We emphasize the fact that FBVR is different from the related task of *face video retrieval*: FBVR deals with “whole” unconstrained videos, potentially including multiple shots, people, and with variable length and quality; in face video retrieval, instead, the video dataset is composed of face tracks, i.e., small video clips that only portray a single face [1–6]. FBVR is more challenging, since in addition to the face recognition and matching steps, a FBVR algorithm also has to perform face detection and tracking in the database videos, in order to extract features for each person included in them. In this sense, we can say that FBVR is an end-to-end task: it starts from the “raw” videos and produces the features that are then used in the retrieval stage.

In addition to this, this work is specific to retrieval of television-like videos, i.e., videos that share similar visual characteristics with TV shows or programs, as opposed to, for example, surveillance videos, which are often gray scale, only contain small faces at lower resolutions and are often recorded by a static camera. We argue that different visual contexts require different tuning or even entirely different pipelines in order to have an effective and efficient FBVR system.

✉ Gioele Ciaparrone
gciaparrone@unisa.it

Leonardo Chiariglione
leonardo@chiariglione.org

Roberto Tagliaferri
robttag@unisa.it

¹ DISA-MIS, University of Salerno, 84084 Fisciano, SA, Italy

² CEDEO.net, 10040 Villar Dora, TO, Italy

Most algorithms in the literature, as we will see in more detail in the next section, deal with face tracks or with partially constrained videos, that do not contain multiple camera shots or multiple people and are thus not suitable to real-world applications. The use of completely unconstrained videos allows instead to appropriately evaluate an end-to-end pipeline for use in real-world systems.

For this reason, we propose the first end-to-end FBVR pipeline for efficient and effective retrieval of unconstrained television-like videos in large-scale datasets, containing thousands of videos. Moreover, our pipeline is also capable of providing the list of shots in which the person of interest appears in each matching video. Given the recent successes of deep learning in many visual tasks, we numerically evaluate and compare the use of different state-of-the-art deep architectures for effective shot boundary detection, face detection, and face feature extraction. We highlight some advantages and drawbacks of the selected models with regards to result quality and computational speed, as well as some of the interactions between the chosen algorithms.

Since no existing large-scale video retrieval dataset contains multi-shot, multi-person videos with variable resolution, length, and quality, we also built an appropriate dataset for a fair evaluation of the presented pipeline, by adapting the existing VoxCeleb2 audiovisual recognition dataset [7]. Our pipeline obtains 97.25% mean average precision (MAP) on the proposed dataset, while being able to analyze videos at $\sim 7\times$ real-time speed and perform a query on databases with thousands of videos in less than 0.5 s on consumer hardware.

We can thus summarize the article contributions as follows:

- we present the first end-to-end FBVR pipeline for retrieval of unconstrained television-like videos in large-scale datasets;
- we create the first large-scale video retrieval dataset for FBVR with fully unconstrained videos by adapting the existing VoxCeleb2 dataset;
- we evaluate our pipeline on the proposed dataset and numerically compare result quality and computational speed of various state-of-the-art deep learning models for the steps of shot boundary detection, face detection, and face feature extraction.

The rest of the paper is organized as follows. Section 2 includes a summary of related works in FBVR, face video retrieval, and face recognition, as well as the relevant datasets in the literature. In Sect. 3, we describe our pipeline, while in Sect. 4, we present the dataset construction process. Section 5 includes details about the experiments that we performed and the results obtained by the pipeline on the proposed dataset, with the related

discussion. Finally, Sect. 6 summarizes the findings and presents some future directions of research.

2 Related works

2.1 Face video retrieval and related tasks

A number of works in the literature tackled the problem of face video retrieval, FBVR, or similar tasks.

Early approaches Arandjelović and Zisserman proposed a shot retrieval system [8, 9] for movies, in which each shot was characterized by signature images of quasi-frontal faces, which were independent from pose and illumination changes. A distance measure between those signatures was devised in order to perform matching and retrieve shots. Sivic et al. [10] improved the previous system by using SIFT descriptors to represent faces, aggregated for each track, and χ^2 distance to perform matching. However, both algorithms worked on relatively small datasets, with a few hundred shots extracted from two movies and a situation comedy.

Herrmann et al. [1] proposed to encode face track features together with measurement information to perform face video retrieval. Features were aggregated using Bags of visual words [11] or Fisher Vectors [12].

Learning binary representations Other works focused on compressing face video features using binary codes.

Li et al. [2, 4, 13] proposed the hierarchical hybrid statistic-based video binary code (HHSVBC). They represented frames using Fisher Vectors computed on local visual features and used covariance matrices to aggregate features across tracks. These high-dimensionality features were compacted into a low-dimensionality vector in Hamming space by using support vector machines (SVM) with Riemannian kernels [14]. The same authors also presented a variation of the algorithm that can perform queries using images instead of videos [3].

Dong et al. [15] introduced one of the first uses of a deep neural network to learn the hash code to represent the video data. They trained an AlexNet convolutional neural network (CNN) [16] jointly with an SVM, using triple ranking loss, in order to learn the hash function. The algorithm obtained 94.12% and 32.61% MAP on *The Big Bang Theory* and *Prison Break* videos (from the ICT-TV dataset [4]), respectively, improving the results obtained by HHSVBC. The authors later refined the procedure [6], by training the network on more data and using hard negative mining.

Qiao et al. [17, 18] presented a deep video code (DVC) extracted using temporal feature pooling over features

computed by a 3-layer CNN. Some fully connected layers were used to compute the hash representation of each video. The system obtained 99.41% and 97.88% MAP in the retrieval task on the aforementioned TV shows of the ICT-TV dataset.

Among other methods employing deep learning to compute hashing functions, we can also mention deep heterogeneous hashing (DHH) [19], in which feature representation, video modeling, and heterogeneous hashing are learned jointly, and hybrid video and image hashing (HVIH) [20], that exploits both dense frame-level features and video-level aggregated features at the same time for the matching step.

Recent FBVR approaches While the previously described approaches mainly focused on face video retrieval, which involves cropped face videos, some works have instead tackled the FBVR task, which deals with full, unconstrained videos.

Mühling et al. [21] presented a system able to perform video search based on textual descriptions or face images, in addition to face identification and clustering. The authors used Faster R-CNN [22] for face detection and another CNN [23] for feature extraction. While their FBVR system used full unconstrained videos, like in our approach, their dataset was relatively small, with only 101 trailer videos, and they evaluated the algorithm using faces from only 6 different actors, with variable results (MAP ranging from 2 to 90%).

Fang et al. [24] proposed a model which used quality information, such as face detection confidence, in order to aggregate and denoise the face features provided in the IQIYI-VID 2019 video retrieval dataset [25]. The authors trained a multi-layer perceptron (MLP) to classify faces into the 10,034 identities of the dataset, obtaining 89.83% MAP in the retrieval task. However, the presented method has a major drawback: it can only work with a pre-defined set of identities. Adding new identities to the database requires changing the final layer of the network and retraining it. For this reason, the method is not suitable for FBVR tasks with unknown identities, as in our case.

In summary, to the best of our knowledge, existing works in the literature either focused on the task of retrieving face videos, as opposed to full unconstrained videos, or evaluated their FBVR system on relatively small datasets. Performing retrieval on face videos does not allow to evaluate different shot detectors, face detectors, and tracking algorithms, all necessary components in real-world situations where face tracks are not directly available. At the same time, evaluating a system on a larger and more varied dataset ensures that the pipeline is robust to variation in face poses and video characteristics. For this reason, we propose a FBVR pipeline that is able to perform

effective and efficient video retrieval on thousands of unconstrained videos, with variable resolution and quality, each containing multiple shots, multiple people, and different environments and visual features.

2.2 Face recognition

The term *face recognition* usually refers to the tasks of face verification or face identification. Face verification consists in determining if two faces belong to the same person, without necessarily knowing their identity. Face identification is instead the task of classifying a face image as belonging to a certain identity, sometimes done using a set of reference gallery templates, each representing a different identity.

Face recognition is strictly related to FBVR, since a step of face verification is usually needed in order to determine if the identities in a video match with the query identity. In a sense, we can consider the task of face identification in videos as the “inverse” task of FBVR: while in face identification the goal is to find the specific identity contained in each input video (which usually contains a single person), in FBVR the identity is given (represented by the query face), and the objective is to find *all* the videos containing that identity.

In recent years, deep learning algorithms have obtained superior performance in face recognition [26, 27].

Face recognition in images DeepFace [28] was one of the first methods to show that deep neural networks had the potential to reach human performance in face verification on images. With an ensemble of CNNs, the authors obtained 97.35% accuracy in face verification on the Labeled Faces in the Wild (LFW) dataset [29], compared to 97.53% obtained by humans [30].

Around the same time, Sun et al. proposed the DeepID algorithm [31], that used 60 CNNs to extract features from 60 different patches in a face image. They used a Joint Bayesian technique [32] to perform face verification, obtaining results comparable to DeepFace. The same authors later proposed improved versions of this method, DeepID2 [33], and DeepID3 [34], that used different CNNs, such as AlexNet or Inception layers [35], with the latter obtaining 99.53% accuracy on LFW.

FaceNet [36] later reached 99.63% accuracy on LFW by using a GoogleNet-like CNN trained with a triplet loss function and online semi-hard negative mining. The network was also tested for face verification on videos of the YouTube Faces dataset [37], by averaging the distances between all pairs of faces in the first 100 frames of each clip, and obtained 95.12% accuracy.

Parkhi et al. proposed the use of a VGG-16 network to learn an Euclidean embedding using triplet loss on their

newly proposed VGGFace dataset [38], where it reached 98.95% verification accuracy. The network also obtained state-of-the-art accuracy on YTF, 97.3%. A larger version of the VGGFace dataset, VGGFace2, was proposed in 2018 [39], with a greater variety in pose, ethnicity, and age of the faces. The authors trained a ResNet-50 and a SENet-50 to classify identities on their new dataset using standard cross-entropy loss. Including VGGFace2 in the training procedure allowed the networks to reach state-of-the-art performance on the IJB-A dataset [40].

Recently, a lot of focus has been put on the design of better loss functions to train face recognition networks. Using the right loss function can help to better separate faces belonging to different identities.

A notable example is the center loss [41], later improved in the contrastive-center loss [42], which introduced the maximization of the distance between each feature vector and the centers of different classes. We can also mention the large-margin Softmax (L-Softmax) loss [43], the congenerous cosine (COCO) loss [44], the Angular margin Softmax (A-Softmax) loss employed in SphereFace [45], and the Large Margin Cosine Loss (LMCL) used in CosFace [46].

Building on the previous works, Deng et al. presented ArcFace [47], which is a further improvement over CosFace and SphereFace. The CNN is trained with an additive angular margin loss that encourages the network to learn a mapping that keeps intra-class feature vectors close to their class centers and forces an angular margin between vectors belonging to different classes. ArcFace obtained state-of-the-art accuracy on a number of face recognition datasets, including YTF (98.02%), Megaface [48] (96.98% verification accuracy and 81.03% Rank-1 identification accuracy), LFW (99.83%), IJB-B [49] (94.2% verification accuracy), and IJB-C [50] (95.6% verification accuracy). The network was also tested on the iQIYI-VID 2018 dataset [51] where it reached 79.80% mean average precision (MAP) in the video retrieval task.

Face recognition in videos While the methods described above mostly tackled the problem of face recognition in images, few works have also focused specifically on the task of face recognition in videos.

In particular, some works explicitly exploited the temporal structure of the videos using deep neural networks. Rao et al. [52] proposed a Generative Adversarial Network, called Discriminative Aggregation Network (DAN), to generate an “aggregated” face image starting from a series of consecutive input images taken from a video. Instead of extracting discriminative features from each frame, the features were extracted directly on the aggregated face image.

The same authors also presented an attention-aware deep reinforcement learning (ADRL) method [53], in order to selectively choose the most important frames to consider when comparing two face videos. They used both CNNs and LSTMs to extract spatiotemporal features, used by a Frame Evaluation Network to iteratively select the most representative frames.

Ding et al. proposed the Trunk-Branch Ensemble Convolutional Neural Network (TBE-CNN) [54] to tackle the pose variability in videos by adding side branches to the network, trained to extract features from specific image patches, which are less sensitive to face pose.

Recently, Zheng et al. [55] proposed an end-to-end pipeline to perform face recognition in unconstrained multi-shot videos. Faces are detected in the probe videos using the SSD CNN [56]; facial landmarks are computed using the All-in-One Face CNN [57], in order to align the faces; features are extracted using one of three possible CNNs described in [58]. Intra-shot tracking is performed using the Kernelized Correlation Filter (KCF), and a one-shot SVM is then used to further associate faces across multiple shots. The authors proposed various subspace learning techniques to map features into an appropriate space for face matching, and the system obtained top recognition performance on the IJB-B and IJB-S [59] datasets. While their end-to-end pipeline is in some aspects similar to our proposed method, their system was designed and evaluated for face recognition and not for video retrieval.

2.3 Large-scale video datasets for face recognition and retrieval

Various large-scale datasets for face recognition and retrieval in videos have been published throughout the years. As we will see, none of them is suited for an appropriate end-to-end evaluation of our proposed pipeline, which is the reason why we decided to build a different dataset.

One of the earliest large-scale face video datasets is YouTube Faces (YTF) [37], presented by Wolf et al. in 2011. It is composed of 3,425 videos of 1,595 celebrities, with an average of 2.1 videos per subject. It was built from YouTube videos using the Viola–Jones face detector [60] to extract clips containing each subject. While it has the advantage of containing many identities and of including complete face track annotations, it has some significant drawbacks which make it unsuitable to test an end-to-end FBVR pipeline effectively: it does not include multi-shot videos, and many of the videos only focus on a single person, without other people in the background. In addition to that, since it was collected in 2011, it is limited to relatively low-resolution videos. Finally, since it was not built

for video retrieval, the dataset does not contain still images to use as queries.

UMDFaces [61, 62] includes instead 3,735,476 annotated video frames extracted from 22,075 videos with 3,107 subjects. The dataset focuses on face verification. At the time of writing, the dataset is in maintenance and not available for download [63].

The IARPA Janus Benchmark-C (IJB-C) [50] dataset, published in 2017, extension of the IARPA Janus Benchmark-A (IJB-A) [40] and IARPA Janus Benchmark-B (IJB-B) [49] datasets, is one of the most comprehensive face detection and recognition datasets in videos that are currently publicly available. It provides 31,334 images, and frames extracted from 11,779 videos containing 3531 subjects. The dataset includes person-centric videos, such as interviews, collected from YouTube, and thus presents a large variety of face poses, illumination conditions, and other visual properties. The challenge includes various evaluation protocols, including face detection, 1:1 face verification, and 1:N face identification. In addition to those, the benchmark includes end-to-end protocols, which require the algorithm to first detect the faces in the input image/video, and then match them against the gallery templates. However, there is no video retrieval protocol available for this dataset.

The iQIYI-VID-2019 dataset [64] is a large-scale video retrieval dataset presented by iQIYI for the 2019 iQIYI Celebrity Video Identification Challenge. It is an extension of the iQIYI-VID-2018 dataset [51]. The 2019 version contains 211,490 video clips from 10,034 identities. The goal of the challenge is to search for videos in the test set for each of the identities included in the training set. The models can be trained to recognize each identity by using the training videos, including audio, but external data can also be used. The test set also includes distractor videos. While this dataset was conceived for video retrieval, it still presents some issues that make it unsuitable to evaluate our pipeline: the clips are of short duration (4 s on average), do not contain shot transitions, and most of them only include a single person each. Since our pipeline includes components such as shot detectors and face tracking algorithms, that are suited for long, multi-shot, multi-person videos, this dataset is not appropriate for a fair evaluation of our pipeline.

Recently, the VoxCeleb datasets have been published, the first large-scale datasets for audiovisual person recognition in unconstrained videos. They contain clips of people speaking, extracted mainly from interview videos on YouTube. It was published in two versions: VoxCeleb1 [65] and VoxCeleb2 [7]. The main goal of the dataset is to help develop and evaluate audiovisual models for speaker recognition. They both provide URLs to the original videos, face detections and tracks, the audio files with the

utterances of each clip, cropped face videos, and speaker metadata. The main difference between VoxCeleb2 and VoxCeleb1 is in the size of the dataset and the heterogeneity of the identities, both improved in the second version of the dataset. VoxCeleb2 contains 1,128,246 utterances extracted from 150,480 videos and 6112 speakers.

Some smaller datasets for recognition and retrieval in movies and TV shows have also been used in the literature. We can mention the *Buffy the Vampire Slayer* [66], *The Big Bang Theory* [67] and *Sherlock* [68] datasets, or the Cast Search in Movies (CSM) dataset [69]. However, they are either of small size or only contain cropped tracklets, posing the same problems of some of the datasets described earlier.

Since, to the best of our knowledge, no existing dataset in the literature was built for large-scale unconstrained video retrieval by faces, we built a custom dataset for FBVR by adapting the VoxCeleb2 dataset. The dataset construction process is detailed in Sect. 4.

3 The proposed pipeline

We present a pipeline for efficient and effective FBVR on large-scale sets of unconstrained videos, representing television-like contents.

The pipeline includes two distinct, but related, processes: the extraction of facial features from the videos that

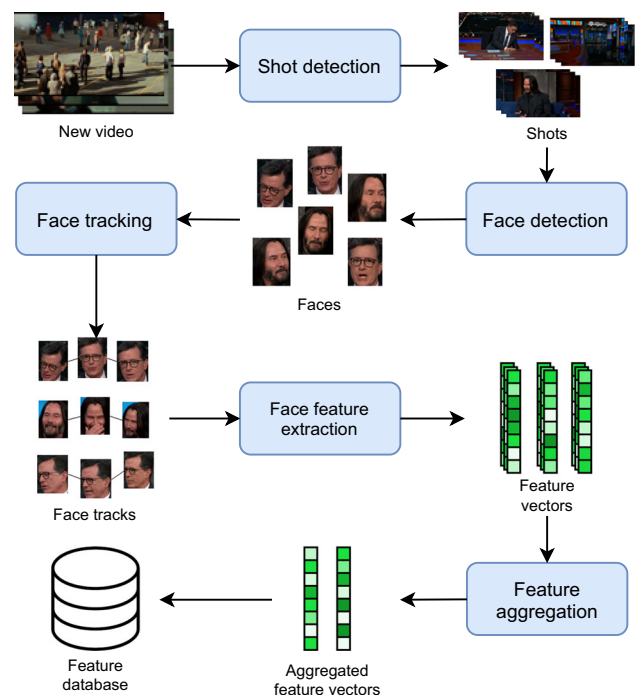


Fig. 1 The feature extraction stage of the proposed pipeline

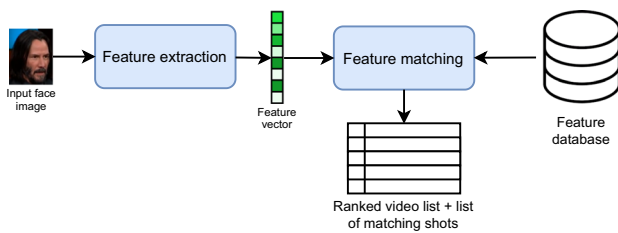


Fig. 2 The query stage of the proposed pipeline

constitute the searchable archive, and the query execution process. The two tasks are summarized in Figs. 1 and 2, respectively.

3.1 Feature extraction pipeline

For every video in the database, the pipeline extracts the facial features from it and saves them into a *feature database*, that is then queried in order to retrieve the videos of interest.

The feature extraction process consists of the following steps:

- first, the video is segmented into coherent shots using a shot boundary detection algorithm;
- a face detector is used to extract faces at specific intervals inside each shot;
- fast, intra-shot face tracking is performed in order to group the faces in different frames that belong to the same person;
- for each track, features are extracted from a subset of its faces;
- finally, the obtained features are aggregated, first across each track, then across the entire video by merging tracks from different shots. A single feature vector per person is obtained, and it is stored into the feature database.

Note that the presented pipeline is modular and can in principle work with any shot detection, face detection, face tracking, and feature extraction algorithm. However, as we will see in Sect. 5, the use of a particular algorithm in an earlier step can in some cases influence the quality of the results produced by later steps of the pipeline.

Shot detection A shot is a series of interrelated consecutive frames, recorded by a single camera, which represents a continuous action in time and space [70]. Shots in a video are separated by transitions that can be cut or gradual: cut transitions are instantaneous changes of shots, while gradual transitions span a number of frames during which both shots are partially visible.

We employed a shot detection algorithm in our pipeline for three main reasons:

- the use of a shot detector can reduce the complexity of the face tracker. Most multiple object tracking algorithms work by assuming the absence of sudden changes in the scene [71], whose presence would instead require the use of computationally expensive person re-identification techniques. Thus, the use of a fast shot detector can have a net positive impact on the pipeline performance;
- detecting and discarding frames belonging to gradual shot transitions helps prevent the extraction of face features from visually unstable faces that can be blended with other contents during the transition;
- in addition to retrieving videos, our pipeline can also return a list of shots in which the person of interest appears. This allows users of the system to quickly jump to the desired parts of each retrieved video, which can be especially useful for long videos.

We evaluated the use of both deep and non-deep shot detection algorithms, as explained in Sect. 5.

Face detection Since consecutive frames in a video are likely to contain the same faces, with similar poses, we only detect faces in a subset of the frames of each video shot, at regular intervals. This ensures that the pipeline does not spend time extracting redundant information and remains computationally efficient, since the face detection step is one of the most expensive ones in the pipeline. While skipping frames can in general have a negative impact on tracking algorithms, faces rarely move significantly in consecutive frames in most situations, especially in our dataset, that is mainly composed of interviews. In addition to that, our pipeline is robust to tracking errors, thanks to the clustering performed during the feature aggregation stage, as we will see. We tested a number of different CNN-based face detectors, as explained in Sect. 5.

Face tracking After extracting faces, we perform intra-shot tracking in order to group faces belonging to the same person. While our pipeline can technically work without a tracking step, we decided to include a fast multiple object tracker for two main reasons. First, grouping faces leads to a more efficient face feature extraction process: we can in fact only extract facial features from a subset of the faces of a person, assuming that features extracted for the same person are similar enough in closely related frames. Second, performing a preliminary intra-shot feature aggregation step can stabilize the facial features used in the inter-shot clustering stage, producing a more accurate result. In Sect. 5, we will see how the inclusion of a tracking step has a positive effect on the pipeline performance, even when using a fast, relatively simple tracker. In order to reduce the number of false tracks, we remove any track that is shorter

than a certain threshold, which we selected using our validation set.

Face feature extraction Our system runs queries by performing face verification. For this reason, in order to perform the video search, the pipeline needs to extract facial features that are then compared with the ones extracted from the query image.

For each face track computed in the previous stage, a fixed number of faces is selected for feature extraction, since we do not want to extract highly redundant information. This again helps with the efficiency of the pipeline. We avoid selecting faces that are right at the start or at the end of a face track in order to reduce possible sources of noise, such as occlusions or frame instabilities that are more likely to happen at (or near) shot boundaries.

Since face detectors are trained to detect tight boxes around faces, while face recognition CNNs are usually trained to include some visual context around the faces, we extract image patches from slightly expanded boxes (we determined the optimal enlargement ratio using our validation set) before feeding them to the feature extraction network.

The output of this stage is thus a set of feature vectors for each person appearing in each video shot.

We evaluated a number of existing CNNs used in face recognition tasks, as described in Sect. 5.

Feature aggregation While it is in theory possible to just save into the feature database all the feature vectors extracted from the previous step, this would lead to a number of issues. First, we would store a large amount of redundant information, increasing disk usage and memory footprint of the feature database. Second, having an unnecessarily large number of feature vectors in the database can degrade the performance of the query step, since the query features would need to be compared to a larger number of database features. Third, since some noise can still be present in the features at the single-face level, this might also have an impact on the accuracy of the system.

For these reasons, we include a feature aggregation stage, that is composed of the following sub-steps:

- first, we average the feature vectors obtained for each face track, in order to produce a single vector representing the person in that shot. Feature averaging has already been employed in some face recognition tasks, for example with the VGGFace2 features [39]. We normalize feature vectors to have unitary norm both before and after the averaging process;
- after that, we perform video-level inter-shot clustering of the features. In this way, we group together tracks from different shots that represent the same person;

- we average the feature vectors in each obtained cluster, and we re-normalize it to have unitary norm.

The output of this step is thus a set of feature vectors, each representing a different person appearing in the video, without redundancy. These features are then saved into the feature database, ready to be used to perform queries. In addition to the facial features, the list of shots in which each person appears is also stored.

We use the well-known hierarchical agglomerative clustering (HAC) algorithm for this step, with a fixed maximum distance threshold (chosen using our validation dataset) in order to determine the cutting point of the clustering tree. The use of a fixed threshold is facilitated by the fact that we employed distance measures that are bounded in the interval $[0, 2]$. In fact, depending on the network employed for feature extraction, we either used the Euclidean distance or the cosine distance¹, both computed on vectors with unitary norm, which guarantees that the distance values are bounded. We also evaluated different linkage methods, as detailed in the supplementary materials.

The clustering algorithm does not force any temporal constraint on the track clusters. In other words, the algorithm can merge tracks that appear in the same shot or that overlap in some frames. This is intentional and aims to make the algorithm robust to face detection and tracking errors, such as duplicate detections (and tracks) or tracks that were mistakenly split due to occlusions or tracking errors. Note that it is very unlikely that two different subjects appearing in the same frame produce appearance features that are similar enough that they get merged by mistake.

3.2 Query pipeline

The query process is simpler and consists of just two steps. First, the query image is fed to the same feature extraction network used in the video feature extraction process. The resulting feature vector is then compared to all the vectors stored in the feature database using the appropriate distance measure (cosine distance or Euclidean distance, depending on the network employed).

For each video, the person that matches with the query image with the lowest distance score is considered the matching identity for that video, and its matching score becomes the video's matching score. A ranked list of videos is then returned, sorted by matching score. For each retrieved video, the pipeline also returns a list of the shots in which the matching person appears, as previously mentioned.

¹ The cosine distance is defined as one minus the cosine similarity.

Note that with this system it is also possible to perform queries using face videos or a set of images as input, by simply averaging the feature vectors computed for each frame/image in the input. However, this is not explored in this paper.

4 Dataset construction

As discussed previously, in order to appropriately evaluate our FBVR pipeline in an end-to-end fashion, including shot detection, face detection, tracking, and recognition, a large dataset of unconstrained videos is needed, with appropriate ground truth annotations for retrieval. The videos must have similar characteristics to television content and must have variable length, quality, number of shots and people, in order to evaluate the robustness of our pipeline to the variety of content that can be found in television videos.

However, building and annotating a large-scale dataset from scratch is challenging and time-consuming. Instead, we decided to adapt an existing face recognition dataset to the task of face-based video retrieval. We chose the VoxCeleb2 audiovisual recognition dataset [7] as a starting point for the two following reasons.

First, the videos in VoxCeleb2, while mainly focusing on interviews, are extremely varied in setting, lighting conditions, length, quality, shot transitions, number of people, face poses, and sizes. Some of the videos in the dataset were even captured from old VHS tapes or recorded in grayscale.

The second reason is related to the query construction process. In order to evaluate our pipeline on the dataset, we need a set of queries with their related ground truth. Since VoxCeleb2 provides face tracks of the person of interest in each video, we are able to extract face images for each identity and use them as the query input.

4.1 Identity selection

We built two video sets for the evaluation of our pipeline: a validation set, used to select the best pipeline hyperparameters, and a test set, for a final independent evaluation. Note that we did not need a FBVR training set, since each model in the pipeline was already trained on the specific task that it was designed for.

Since 116 out of the 118 identities in the VoxCeleb2 test set were also part of the training sets of the face recognition networks employed in the feature extraction stage, we used videos from the VoxCeleb2 test set as our validation set for the FBVR pipeline.

For the test set, we chose instead a subset of the identities from the VoxCeleb2 development set that did not overlap with the training sets of the VGGFace2 and MS-

Celeb-1M [72] datasets. These are the two datasets used to train the VGGFace2 and ArcFace face recognition CNNs, which we used in our final independent tests. This ensures that the results obtained on the test set are not subject to bias.

After removing the shared identities from the VoxCeleb2 development set, 78 identities were left. Note these are just the 78 annotated identities; the videos contain a large number of background identities, that act as distractors. Figure 3 shows example frames from videos in the test set, highlighting the variety of content in the dataset.

Once the identities were selected, we downloaded the highest resolution version of each related video, with a maximum of 720p. Since some videos of the VoxCeleb2 dataset were not available anymore, we obtained 4260 of the 4910 videos for the validation set, and 1375 of the 1536 videos containing the 78 non-overlapping identities of the test set.

On average, the validation set contains 36 videos for each identity, while the test set contains about 18. The validation videos contain a total of 40,153,695 frames, with 396.4 hours of footage and an average of 28.1 FPS. The test videos contain a total of 13,862,357 frames, with 146 hours of footage and an average of 26.4 FPS. Videos are also quite varied in resolution, ranging from 160×120 pixels up to 1280×720 , with an average of 1007×585 pixels for the validation videos and 956×563 for the test videos.

4.2 Query creation process

VoxCeleb2 includes cropped clips of each face track annotated in the dataset. However, these clips were resized and underwent an additional video encoding step, leading to lower quality images. For this reason, we instead decided to extract the face images directly from the original videos².

For each face track provided with VoxCeleb2, we extracted three face images, one from the frame at 25% of the track length, one from the middle frame of the track, and one from the frame at 75% of the track length. We avoided starting and ending frames in order to reduce possible errors from initial and final frames in the tracks, similar to the sampling process we used for feature extraction (see Sect. 3).

We obtained 94,233 query faces for the validation set, and 29,652 query faces for the test set. The high number of faces per identity allows a thorough evaluation of the robustness of the pipeline to pose and illumination changes.

² We performed preliminary tests on the validation set using query images extracted from the cropped clips and from the original videos, and the pipeline obtained slightly better results using the latter.



Fig. 3 Example frames extracted from videos in the test dataset. Note the variety of locations and image quality

For each query image, we defined the corresponding ground truth as all the videos that contain the same person as the query image, according to the VoxCeleb2 annotations. The video from which each face was extracted was excluded from the ground truth and ignored at evaluation time, since it is trivial for the face recognition network to match identical faces.

Figure 4 shows examples of query faces from the test set.

4.3 Test set cleaning procedure

While performing some preliminary experiments on the test set, we discovered that the test set annotations were significantly noisier than the ones in the validation set. A number of videos that were supposed to contain the queried identities, did not actually contain them, and some extracted query faces did not portray the right person or any person at all. This was due to the automated annotation process used by the VoxCeleb2 authors that led to a number of annotation mistakes.

While some annotation errors can be accepted in our validation set, since we are mostly interested in the relative accuracy differences when selecting different models or hyperparameters of our pipeline, having a more accurate score on the test set is important, in order to have a more precise idea of the performance that the pipeline might have in real-world systems.

For this reason, we performed a manual cleaning process of the test set annotations, while we left the validation set intact. This was also due to the larger size of the validation set, which would make a manual cleaning more time-consuming.

Every video that did not include the expected person was removed from the ground truth of all the queries of that identity. 96 out of the 1375 test videos were thus removed from the ground truth. Note that those videos were still kept in the dataset, so that their features could act as additional distractors.

In addition to that, we also removed all the query faces extracted from videos that contained more than 15% of annotation mistakes. We did not remove the wrong faces one by one, since it was not practical given the size of the dataset.

After the dataset cleaning, the test set included 23,208 queries from 1077 videos and 74 identities. Four identities were removed because less than two valid videos were left with them in the dataset after the cleaning procedure. Please note again that the full set of 1375 videos was still used in the feature extraction phase for every experiment, as previously noted.

5 Results and discussion

We performed several experiments in order to select the best configuration of the pipeline and to validate it. Note that since, to the best of our knowledge, this is the first FBVR pipeline designed to work on a large dataset of unconstrained videos without prior knowledge of the involved identities, a fair comparison with previous works in the literature is not possible.

In Sect. 5.1, we describe the software and hardware configurations used for the experiments. In Sect. 5.2, we describe the metric used to evaluate the pipeline. In Sect. 5.3, we detail some of the experiments performed on the validation set for the hyperparameter selection, while the full list of experiments is presented in the supplementary materials. Finally, in Sect. 5.4 we describe and discuss the results of the experiments on the test set.

5.1 Experimental setup

The pipeline was implemented using Python 3.7 on Ubuntu 16.04. We ported the original Windows implementation of the ImageLab Shot Detector [73], one of the shot detectors tested in our pipeline, to Linux. The code was kept in C++, but we added a Python interface on top. We used a TensorFlow version of the TransNetV2 shot detector [74]. Other than those two models, we used PyTorch implementations for every other deep learning model tested in



Fig. 4 Examples of query face images from the test dataset. Images were resized to fit the grid in this figure, but originally ranged from 108×108 pixels to 394×394 pixels

the pipeline, including the original TransNet shot detector [75], all the face detectors (MTCNN [76], SSH [77], RetinaFace [78]) and all the face recognition networks (VGGFace [38], VGGFace2 [39], ArcFace [47]). We used PyTorch 1.4 and TensorFlow 2.3, both backed by CUDA 10 and cuDNN 7.

We ran the experiments on a machine with an Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20 GHz, NVIDIA GeForce GTX 1080 (8 GB VRAM), and 32 GB RAM.

5.2 Evaluation metric

We report results for every experiment using the mean average precision (MAP) metric, commonly used in information retrieval tasks. It is defined as follows:

$$MAP = \frac{1}{n_q} \sum_{i=1}^{n_q} AP_i,$$

where n_q is the total number of queries performed, and AP_i is the average precision (AP) of the i -th query. The average precision approximates the area under the precision–recall curve, and various formulations exist. We used the scikit-learn implementation [79], which uses the following formulation:

$$AP = \sum_{k=1}^N (R_k - R_{k-1}) P_k,$$

where N is the total number of items, P_k and R_k are the precision and recall, respectively, computed considering

only the top k retrieved items (videos in our case), sorted by descending matching score. The AP metric implicitly assigns a higher weight to errors in the top results with respect to errors ranked lower in the list (i.e., with lower matching scores). This is a desirable property, since the end user of a retrieval system is mainly interested in the top retrieved results and expects the first results to match better to the query with respect to the ones that are ranked lower.

5.3 Hyperparameter search on the validation set

Since the pipeline has a large number of hyperparameters, and each different configuration requires between 1 and 3.5 days of computation on our hardware to extract the features from all the videos and run the queries on the validation set, testing every possible combination of hyperparameters (e.g., by using a grid search) was not feasible. For this reason, we started from a base configuration and gradually evaluated changes in one hyperparameter at a time in order to find the locally best configuration.

For the sake of brevity, we only include here the experiments and comparisons involving the different deep models used in the various steps of the pipeline. The supplementary materials provide a more thorough description of the hyperparameter tuning process.

Using the validation set, we found that the pipeline worked the best with the following configuration:

- minimum face size s_{\min} set to 32 pixels per side, smaller faces are discarded;
- frame sampling interval for face detection I_{fd} set to 1 second;
- minimum track length t_{\min} set to 4 detections (equivalent to a 3-s interval, if $I_{fd} = 1s$), shorter tracks are discarded;
- 3 faces are sampled from each track for face feature extraction;
- element-wise average as the feature aggregation function;
- maximum clustering distance threshold σ_{cl} set to 0.4.

Regarding the models used in each step, we tested the following algorithms:

- shot detectors: ILSD [73], TransNet [75], and TransNetV2 [74];
- face detectors: MTCNN [76], SSH [77], and RetinaFace [78];
- intra-shot tracking: we only tested the simple IOU tracker [80]. This is because its simplicity makes it extremely fast, while still performing well on videos with limited subject motion, such as interviews;
- face feature extractors: VGGFace [38], VGGFace2 [39], and ArcFace [47];

- inter-shot clustering: hierarchical agglomerative clustering with average or complete linkage and either Euclidean or cosine distance metrics, depending on the feature extraction network employed.

5.3.1 Shot detector comparison

We compared the three shot detectors in conjunction with the MTCNN face detector and the VGGFace2 feature extractor with SENet-50 backbone. The other pipeline hyperparameters were set as explained in the previous section. We selected the three mentioned shot detectors (TransNet, TransNetV2, and ILSD) because they obtained the best results on the RAI dataset [73], and the respective authors provided a public implementation. We included a non-deep shot detector in our tests, ILSD, since “classical” shot detectors are still able to compete in accuracy with more modern deep learning-based ones.

The results are shown in Table 1. Changing shot detector did not significantly affect the pipeline accuracy. However, the time/memory efficiency of the three algorithms is different. TransNet is much faster than ILSD: extracting features on the 4260 videos of the validation set took 3189 min (2.21 days) with TransNet, compared to 3773 min (2.62 days) using ILSD. That means that changing the shot detector alone made the pipeline ~16% faster. Note that this is after we optimized the running time performance of the original implementation of ILSD by implementing parallel video decoding, so that the shot detection process was not bottlenecked by the frame decoding process.

TransNetV2 had a similar running time to TransNet in our tests. However, due to the TensorFlow implementation, it used more GPU memory than TransNet. For this reason, we chose to use the original TransNet for the subsequent experiments.

5.3.2 Face feature extractor comparison

After fixing the shot detector, we evaluated the three mentioned feature extraction networks. In particular, we chose to evaluate VGGFace1, VGGFace2, and ArcFace since they are state-of-the-art face recognition models with

public implementations. Comparing VGGFace1 and VGGFace2 allows to highlight the importance of larger training datasets. ArcFace is instead a representative of the family of face recognition networks that focus on designing an appropriate training loss, as explained in Sect. 2.

We tested different bounding box scaling factors s_{det} (see Sect. 3.1). For each network, we used a different clustering distance metric, depending on the distance measure used in the original papers.

Since ArcFace requires face alignment before extracting the features, we used the facial landmarks extracted using MTCNN in order to align the faces according to the original ArcFace implementation [81].

Since VGGFace1 with Euclidean distance obtained much worse results than VGGFace2, we also tested the network with a different distance metric and by extracting the features from a different layer. In the original paper, the features from the last fully connected layer were used; we also tested the use of features from the last convolutional layer, similar to the other networks.

As we can see in Table 2 though, VGGFace1 did not perform particularly well. This shows the importance of training face recognition networks on larger, more diverse face datasets, such as VGGFace2. Increasing the scaling factor improved the performance of VGGFace2, suggesting that VGGFace2 was trained on faces with more context around them than what the face detectors were trained to

Table 1 MAP (percentage) obtained by the pipeline on the validation set using different shot detectors

Shot detector	MAP
ILSD	93.71
TransNet	93.71
TransNetV2	93.61

Table 2 MAP (percentage) on the validation set using different face feature extraction networks

Feature extr. net	Distance metric	s_{det}	MAP
VGGFace2 (SN-50)	Cosine	1.2	93.74
VGGFace2 (SN-50)	Cosine	1.5	94.34
VGGFace1 (conv)	Cosine	1.2	55.71
VGGFace1 (conv)	Euclidean	1.2	67.32
VGGFace1 (FC)	Cosine	1.2	74.69
VGGFace1 (FC)	Euclidean	1.2	75.25
ArcFace (RN-18)	Euclidean	1.2	90.38
ArcFace (RN-18)	Euclidean	1.5	90.38
ArcFace (RN-50)	Euclidean	1.2	92.74
ArcFace (RN-50)	Euclidean	1.5	92.75
ArcFace (RN-101)	Euclidean	1.2	94.59
ArcFace (RN-101)	Euclidean	1.5	94.58

Feature extr. net indicates the feature extraction network employed, where VGGFace1 is the CNN trained on the original VGGFace, (-conv) indicates that the pooled convolutional features were used, while (FC) indicates that the fully connected features were used. RN-18, RN-50, SN-50, and RN-101 indicate that a ResNet-18, ResNet-50, SENet-50, or ResNet-101 backbone was used, respectively. s_{det} is the face bounding box scaling factor. The two best results are highlighted in bold

extract. ArcFace was instead not impacted by the box rescaling, likely because of the face alignment step, and it obtained in general the best performance on the validation set. While ArcFace with ResNet-101 obtained the best results overall, it is important to note that when using a backbone network with the same depth (SENet-50 for VGGFace2 vs. ResNet-50 for ArcFace), VGGFace2 still seems to perform better when coupled with the MTCNN face detector. This will change with a different face detector, as we will see in the next set of experiments.

Regarding speed, VGGFace1 was the slowest network of the three, since it used the VGG-16 backbone. Extracting features from the entire dataset took around 2.4 times longer than VGGFace2. Paired with its worse performance, we excluded this network from all the subsequent experiments. VGGFace2 and ArcFace had instead similar computational times. More details are provided in the Supplementary materials.

5.3.3 Face detector comparison

Finally, we compared three different face detectors with the two best face feature extractors. All three face detector networks were chosen because of their public implementation and good performance on various face detection datasets. In particular, the multi-task cascaded convolutional network (MTCNN) [76] uses a series of three small CNNs that progressively refine face boxes and landmark predictions. The Single-Shot Headless (SSH) face detector [77] uses instead a fully convolutional VGG-16 CNN with a series of top branches that look at faces of different scales. These branches include context modules that allow the network to look at the context around faces in order to refine the predictions. Note that SSH is not able to predict face landmarks. RetinaFace [78] is instead a lightweight face detector that reaches state-of-the-art performance in face detection by exploiting a multi-task training strategy, which involved training the network for face landmark prediction, self-supervised 3D mesh prediction, and camera pose prediction. RetinaFace also used context modules similar to the ones in SSH, as well as a Feature Pyramid Network for multi-scale face detection.

Results are shown in Table 3. Note that for RetinaFace we employed the faster MobileNet-0.25 backbone.

In general, RetinaFace obtained the best results with every face recognition network, although the performance gain was much more substantial in combination with ArcFace. This might be due to one of two possible reasons. The first one is that RetinaFace produces more accurate face landmarks, used for face alignment, which we found to be fundamental to obtain a good performance with ArcFace. The second possibility is that RetinaFace returned a lower number of faces, possibly excluding more

Table 3 MAP (percentage) on the validation set by varying the face detector used

Face detector	Feature extr. net	s_{det}	MAP
MTCNN	VGGFace2	1.2	93.74
MTCNN	VGGFace2	1.5	94.34
MTCNN	ArcFace (RN-50)	1.2	92.74
MTCNN	ArcFace (RN-101)	1.2	94.59
RetinaFace	VGGFace2	1.2	93.75
RetinaFace	VGGFace2	1.5	94.44
RetinaFace	ArcFace (RN-50)	1.2	94.59
RetinaFace	ArcFace (RN-101)	1.2	95.48
SSH	VGGFace2	1.2	93.57
SSH	VGGFace2	1.5	94.16
SSH	ArcFace (RN-101)	1.2	2.07

uncertain/occluded faces. More details about these two observations can be found in the Supplementary materials.

SSH obtained worse performance with VGGFace2 and was also slower, due to the more computationally expensive backbone. SSH does not produce face landmarks, thus the faces were not aligned when using SSH. As we can see, ArcFace is not able to produce reliable features without face alignment.

Section 1.1.7 of the supplementary materials contains additional information and observations about these experiments.

5.3.4 Running time considerations

While not the main focus of this paper, the pipeline was still designed with efficiency in mind.

Running the pipeline in the second configuration³ shown in Table 1 in single-process mode, it was able to extract the features from the 396.4 hours of footage of the validation set in 3189 min. This means that on average the pipeline ran shot detection, face detection, tracking, feature extraction, aggregation, and clustering at 7.5 times the video real-time speed. We found that face detection took roughly 65% of the total running time, while shot detection took about 30%. Face recognition, tracking, and feature aggregation were almost negligible in terms of computational time. While the time taken by the shot detector might look significant, it was only slightly slower (1.4 times) than the FFmpeg video decoding speed, that constitutes a hard limit on the shot detection speed.

Regarding query speed, it grows linearly with the amount of feature vectors in the database. However, even

³ The majority of the other good performing configurations ran at similar speed.

with more than 100,000 feature vectors, 6 parallel processes were able to perform about 15 queries per second. This results in a lower bound on single process of about 2.5 queries per second, or 0.4 s per query. The performance in single process can be better since GPU access acts as a bottleneck for the 6 concurrent processes.

5.3.5 Ablation studies

We performed three ablation experiments in order to verify if the inclusion of tracking and feature aggregation actually benefits the pipeline. In fact, differently from shot detection, face detection, and feature extraction, the pipeline can still function without those two components.

So, we first tested the effects of removing the tracking step by feeding the single faces to the clustering algorithm; second, we evaluated the removal of inter-shot clustering by directly storing the aggregated features from the tracking step in the feature database; third, we tested the removal of any feature aggregation step by extracting a single face from each track and without any clustering. Results are shown in Table 4. The first configuration, used as a reference, is the same as the first configuration in Table 3.

Removing tracking resulted in a loss of more than 1% MAP, indicating that pre-associating faces by their location in adjacent frames can lead to better feature vectors and helps to obtain a more stable clustering.

Removing inter-shot clustering had no significant effect on the MAP, but produced a very high number of feature vectors, which is expensive both in terms of disk/memory usage and in terms of query execution time. The experiment in fact produced a 2.5 GB feature database, as opposed to 362 MB for the original configuration, and queries took 293 min to run, as opposed to the original 57 min. The advantage of inter-shot clustering is thus apparent, considering that it does not significantly impact on the overall pipeline computational time.

Finally, removing feature aggregation resulted in a 0.74% lower MAP, with all the same time/space complexity drawbacks as the previous experiment. This also highlights the importance of selecting multiple faces per track in order to smooth out pose, expression, and illumination differences, as well as any other sources of noise that may be present in a track.

5.4 Experiments on the test set

As already discussed, in order to ensure a fair, unbiased evaluation of the pipeline, we collected a test set with no identity in common with the training sets of both VGGFace2 and ArcFace. On this dataset, we evaluated the best configurations for VGGFace2, ArcFace with ResNet-

50, and ArcFace with ResNet-101, since the choice of the feature extraction network is the one that impacts the most on the pipeline accuracy. Since VGGFace1 obtained significantly worse results on the validation set, we did not evaluate it on the test set. Results are shown in Table 5.

The pipeline using VGGFace2 achieved the best performance on the test set, with 97.25% MAP. It obtained almost 1.3% MAP higher than the slower ArcFace with ResNet-101, despite obtaining a worse performance on the validation set. This suggests that VGGFace2 can more easily generalize on unseen identities. In general, the pipeline obtained higher scores on the test set for two possible reasons: first, the test set underwent an annotation cleaning process; second, the test set is smaller than the validation set, making the video retrieval problem slightly easier. To give an idea of the impact of the dataset cleaning process, the first configuration in Table 5 obtained 88.15% MAP on the raw test set. This means that more than 9% MAP was lost due to annotation errors and not because of model mistakes.

Figure 5 shows a comparison between the AP distribution over the 23,208 queries of the test set for the three considered configurations. As we can see, the vast majority of queries obtains near-perfect score: for example, for the configuration with VGGFace2, 21,555 queries (92.9%) obtain an AP higher than 90%, and 13,890 queries (almost 60%) obtain an AP higher than 99.99%, which is basically a perfect score, considering numerical errors in computing the AP. Remember that an AP score of 1 means that the k relevant videos for a query are returned exactly in the top k positions in the ranked video list, which is the ideal result. The ResNet-101 configuration achieves a similar number of queries (21,611) with AP higher than 90%, but also obtains a very low AP, less than 10%, for 451 queries ($\sim 2\%$ of the total), which is the main cause for the overall lower MAP. This is also evident in Fig. 5c.

We explored the results in order to understand the difference in accuracy between VGGFace2 and ArcFace on the test set. We discovered that a significant fraction of the errors produced by ArcFace was due to face detection failures on low-resolution images by RetinaFace. This in turn led to skipping the face alignment step, which probably affected the ArcFace performance significantly, as previously seen on the validation set. This highlights a weakness of ArcFace: VGGFace2 does not need face alignment and does not suffer from face detection errors like ArcFace. Besides, not requiring a face detection step at query phase also speeds up computation, making this a further reason to prefer models like VGGFace2 in a FBVR system.

In order to check if changing the face detector could result in less errors and thus a better performance for ArcFace on the test set, we also ran the pipeline with

Table 4 MAP (percentage) on the validation set when removing tracking, clustering, or any feature aggregation

Tracking	Samples/track	Clustering	MAP
Yes	3	Yes	93.74
No	3	Yes	92.58
Yes	3	No	93.76
Yes	1	No	93.00

Table 5 MAP (percentage) on the test set for the best pipeline configurations involving VGGFace2 and ArcFace

Feature extraction net	MAP (val.)	MAP (test)
VGGFace2 (SENet-50)	94.44	97.25
ArcFace (ResNet-50)	94.59	95.48
ArcFace (ResNet-101)	95.48	95.96

These configurations are previously described in Table 3. Validation MAP is also shown for reference

MTCNN instead of RetinaFace. However, like in the validation set, the scores obtained by using MTCNN with ArcFace were worse than using RetinaFace (94.61% MAP and 95.79% MAP for the ResNet-50 and 101 versions, respectively). However, this time the errors were not due to the faces not being detected, but mostly to landmark prediction errors. This reinforces the idea that ArcFace is particularly susceptible to face alignment issues.

We also examined the errors produced by VGGFace2 and discovered that they were either annotation mistakes that were not filtered in our cleaning procedure, or were due to hard cases like low-resolution images. Note that manual error inspection did not show that face pose or expression was a significant factor in determining lower AP

score. This confirms the validity of using models trained on datasets with large face pose variations, such as VGGFace2 [39]. We show in Fig. 6 the 16 query images that resulted in the lowest AP for VGGFace2 on the test set.

Finally, we checked if some identities were particularly hard to recognize. We found out that the pipeline obtained a mean AP higher than 90% for most of the identities, and only one (for VGGFace2) or two (for ArcFace) led to a MAP lower than 70%. The worst-recognized identity for VGGFace2 contains many low-resolution query images and the identity sometimes wears sunglasses; moreover, there seems to be some notable age difference between the various videos of this person, which might contribute to his lower score. Example faces from this identity are shown in Fig. 7a. This is also one of the two worst identities for ArcFace; the other one is shown instead in Fig. 7b. We can see that he often wears sunglasses, which might again be a factor of its worse performance.

In conclusion, we can see that the pipeline performs reliably even on unseen identities, with MAP reaching over 97% for the best performing configuration.

6 Conclusion

We presented a novel pipeline for the task of face-based video retrieval on large-scale unconstrained multi-shot video datasets. The pipeline includes shot detection, face detection, tracking and recognition, and feature aggregation, and it presents a query protocol to perform video retrieval. We also built a large-scale video dataset for the proper evaluation of face-based video retrieval, derived from the VoxCeleb2 dataset, since existing datasets were not large and varied enough, or were not appropriate for an end-to-end evaluation of the proposed pipeline.

The pipeline was able to efficiently analyze challenging videos and perform fast queries on databases containing

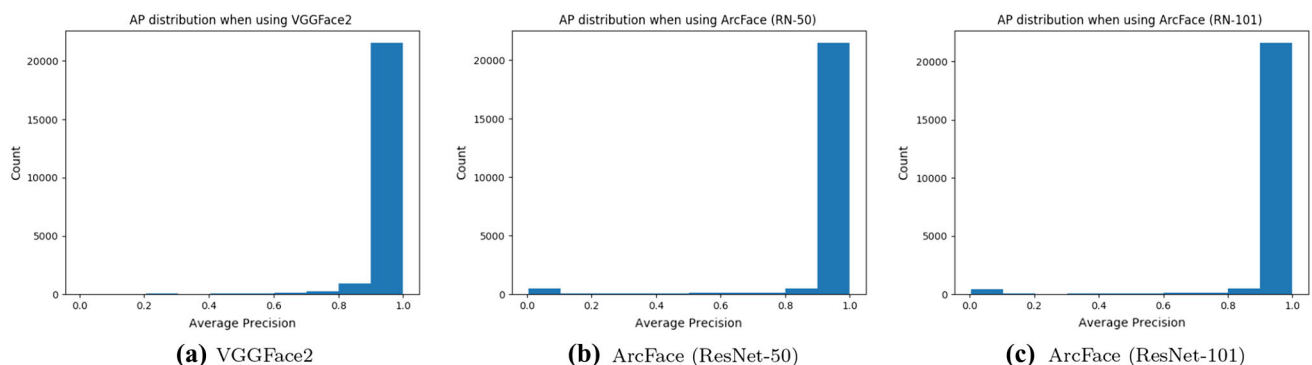


Fig. 5 Distribution of the AP for the three configurations evaluated on the test set. The vast majority of queries obtained an AP > 90%. The pipeline configurations with ArcFace produce more queries with AP < 10%

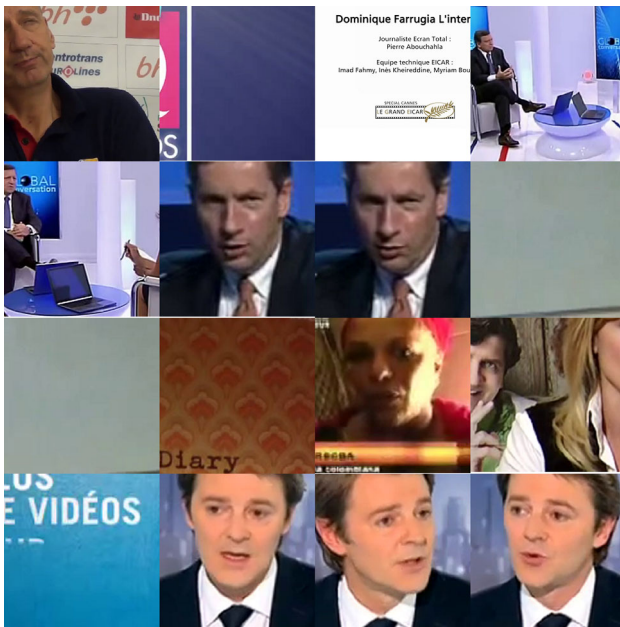


Fig. 6 The 16 query image faces which gave the lowest AP for VGGFace2. Images have been resized to have the same width. Most of the images that actually contain people are annotation errors, such as the ones in rows 2 and 4

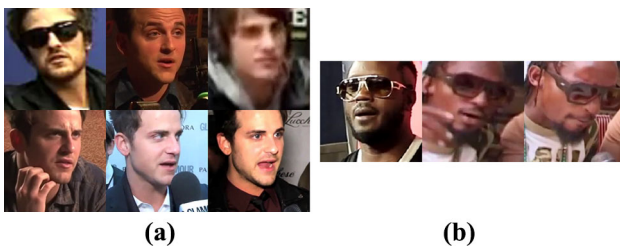


Fig. 7 Sample images from the two worst performing identities in the test set. VGGFace2 obtains less than 70% MAP only on identity (a), while ArcFace (both ResNet-50 and ResNet-101 versions) obtains less than 70% MAP on both identities (a) and (b)

features extracted from thousands of videos. By using a combination of the TransNet shot boundary detector, the RetinaFace face detector, and the VGGFace2 face feature extractor, the pipeline reached a mean average precision of 97.25% over more than 23,000 queries on videos with unseen identities. We compared a variety of approaches and network models for each step of the pipeline, and described the pros and cons of each model, highlighting the trade-offs in terms of accuracy and time/memory efficiency. In particular, we found that deep shot detectors can compete in terms of accuracy with classical algorithms, while also running faster. We also showed that the use of a particular face detection algorithm can have a significant effect on the quality of the features extracted by a face recognition network. For example, the use of CNNs that

rely on accurate face alignment, such as ArcFace, requires the use of face detection algorithms that can predict face landmarks in order to extract meaningful face features. For this reason, face recognition networks that do not require face alignment are more robust to face box inaccuracies and are thus preferable in most cases for a FBVR pipeline.

However, the pipeline has still room for improvement. First of all, the presented dataset focuses on interviews. While this can be representative of some TV programs, like talk shows, it would be interesting to evaluate the pipeline on more dynamic videos, like movies or, in general, videos with higher camera and content motion. Another possible future line of research is the investigation of an efficient implementation of feature quantization, which can be useful for systems with a large number of users, for which space usage optimization is fundamental, and query execution time is even more important. Another unexplored route in this paper is the use of hashing strategies in order to reduce the query time complexity from linear (in the number of database feature vectors) to constant (on average); as we have seen, many hashing strategies exist in the literature, each of which may work differently depending of the type of features and the network used to extract them.

In addition to that, different tracking algorithms could be explored in order to help the feature aggregation process to produce more stable descriptors. Different aggregation strategies might also be investigated, by changing both the clustering algorithm and the feature merge process itself, with a smarter weighting of the feature vectors in the averaging process, in order to exclude outliers. The use of a fast online clustering algorithm might also be investigated, with the aim to implement an efficient inter-video clustering process. This would help reduce feature variability across videos by merging features of the same subject across different visual contexts. This could in turn help with both accuracy and query running time.

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1007/s00521-021-06875-x>.

Funding This work was partially supported by Ministero dell'Università e della Ricerca, PON Ricerca e Innovazione 2014-2020, grant number D48G18000150006.

Declaration

Conflict of interest The authors declare that they have no conflicts of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the

source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Herrmann C, Beyerer J (2015) Face retrieval on large-scale video data. In: 2015 12th conference on computer and robot vision, pp 192–199. IEEE . <https://doi.org/10.1109/CRV.2015.32>
- Li Y, Wang R, Shan S, Chen X (2015) Hierarchical hybrid statistic based video binary code and its application to face retrieval in TV-series. In: 2015 11th IEEE international conference and workshops on automatic face and gesture recognition (FG), vol. 1, pp 1–8. IEEE . <https://doi.org/10.1109/FG.2015.7163089>
- Li Y, Wang R, Huang Z, Shan S, Chen X (2015) Face video retrieval with image query via hashing across Euclidean space and Riemannian manifold. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4758–4767. <https://doi.org/10.1109/CVPR.2015.7299108>
- Li Y, Wang R, Cui Z, Shan S, Chen X (2016) Spatial pyramid covariance-based compact video code for robust face retrieval in TV-series. IEEE Trans Image Process 25(12):5905–5919. <https://doi.org/10.1109/TIP.2016.2616297>
- Jing C, Dong Z, Pei M, Jia Y (2017) Fusing appearance features and correlation features for face video retrieval. In: Pacific rim conference on multimedia, pp 150–160. Springer . https://doi.org/10.1007/978-3-319-77383-4_15
- Dong Z, Jing C, Pei M, Jia Y (2018) Deep CNN based binary hash video representations for face retrieval. Pattern Recognit 81:357–369. <https://doi.org/10.1016/j.patcog.2018.04.014>
- Chung JS, Nagrani A, (2018) Isserman A VoxCeleb2: deep speaker recognition. In: Proceedings of the 19th annual conference of the international speech communication association, vol 1, pp 1086–1090 . <https://doi.org/10.21437/Interspeech.2018-1929>
- Arandjelović, O., Zisserman, A.: Automatic face recognition for film character retrieval in feature-length films. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), vol 1, pp 860–867. IEEE (2005). <https://doi.org/10.1109/CVPR.2005.81>
- Arandjelović O, Zisserman A On film character retrieval in feature-length films. In: Interactive video, pp 89–105. Springer (2006). https://doi.org/10.1007/978-3-540-33215-2_5
- Sivic J, Everingham M, Zisserman A (2005) Person spotting: video shot retrieval for face sets. In: International conference on image and video retrieval, pp 226–236. Springer . https://doi.org/10.1007/11526346_26
- Sivic J, Zisserman A (2003) Video Google: A text retrieval approach to object matching in videos. In: Proceedings ninth IEEE international conference on computer vision, p 1470. IEEE . <https://doi.org/10.1109/ICCV.2003.1238663>
- Perronnin F, Sánchez J, Mensink T (2010) Improving the fisher kernel for large-scale image classification. In: European conference on computer vision, pp 143–156. Springer . https://doi.org/10.1007/978-3-642-15561-1_11
- Li Y, Wang R, Cui Z, Shan S, Chen X (2014) Compact video code and its application to robust face retrieval in TV-series. In: Proceedings of the British machine vision conference, pp 1–12. BMVA Press . <https://doi.org/10.5244/C.28.93>
- Wang R, Guo H, Davis LS, Dai Q (2012) Ovariance discriminative learning: A natural and efficient approach to image set classification. In: 2012 IEEE conference on computer vision and pattern recognition, pp 2496–2503. IEEE . <https://doi.org/10.1109/CVPR.2012.6247965>
- Dong Z, Jia S, Wu T, Pei M (2016) Face video retrieval via deep learning of binary hash representations. In: Thirtieth AAAI conference on artificial intelligence, pp 3471–3477
- Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 25:1097–1105
- Qiao S, Wang R, Shan S, Chen X (2016) ep video code for efficient face video retrieval. In: Asian conference on computer vision, pp 296–312. Springer . https://doi.org/10.1007/978-3-319-54187-7_20
- Qiao S, Wang R, Shan S, Chen X (2020) eep video code for efficient face video retrieval. Pattern Recognit. <https://doi.org/10.1016/j.patcog.2020.107754>
- Qiao S, Wang R, Shan S, Chen X (2019) Deep heterogeneous hashing for face video retrieval. IEEE Trans Image Process 29:1299–1312. <https://doi.org/10.1109/TIP.2019.2940683>
- Wang R, Qiao S, Shan S, Chen X (2020) Hybrid video and image hashing for robust face retrieval. In: 2020 15th IEEE international conference on automatic face and gesture recognition, pp 186–193 . <https://doi.org/10.1109/FG47880.2020.00028>
- Mühling M, Korfhage N, Müller E, Otto C, Springstein M, Langelage T, Veith U, Ewerth R, Freisleben B (2017) Deep learning for content-based video retrieval in film and television production. Multimed Tools Appl 76(21):22169–22194. <https://doi.org/10.1007/s11042-017-4962-9>
- Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: Towards real-time object detection with region proposal networks. Adv Neural Inf Process Syst 28:91–99
- Yi D, Lei Z, Liao S, Li SZ (2014) Earning face representation from scratch. arXiv preprint [arXiv:1411.7923](https://arxiv.org/abs/1411.7923)
- Fang X, Zou Y (2019) Ake the best of face clues in iQIYI celebrity video identification challenge 2019. In: Proceedings of the 27th ACM international conference on multimedia, pp. 2526–2530 . <https://doi.org/10.1145/3343031.3356056>
- 2019 iQIYI celebrity video identification challenge. <http://challenge.ai.iqiyi.com/detail?raceId=5c767dc41a6fa0ccf53922e6>. Accessed: 20 Oct 2020
- Taskiran M, Kahraman N, Erdem CE (2020) ace recognition: Past, present and future (a review). Digit Signal Process. <https://doi.org/10.1016/j.dsp.2020.102809>
- Guo G, Zhang N (2019) A survey on deep learning based face recognition. Comput Vis Image Underst 189:102805. <https://doi.org/10.1016/j.cviu.2019.102805>
- Taigman Y, Yang M, Ranzato M, Wolf, L (2014) Deepface: closing the gap to human-level performance in face verification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1701–1708
- Huang GB, Mattar M, Berg T, Learned-Miller E (2008) Labeled faces in the wild: a database for studying face recognition in unconstrained environments. Workshop on faces in “real-life” images: detection, alignment, and recognition. Erik Learned-Miller and Andras Ferencz and Frédéric Jurie, Marseille, France, pp 7–49
- Kumar N, Berg AC, Belhumeur PN, Nayar SK (2009) Attribute and simile classifiers for face verification. In: 2009 IEEE 12th international conference on computer vision, pp 365–372. IEEE . <https://doi.org/10.1109/ICCV.2009.5459250>
- Sun Y, Wang X, Tang X (2014) Deep learning face representation from predicting 10,000 classes. In: Proceedings of the IEEE

- conference on computer vision and pattern recognition, pp 1891–1898. <https://doi.org/10.1109/CVPR.2014.244>
32. Chen D, Cao X, Wang L, Wen F, Sun J (2012) Bayesian face revisited: a joint formulation. In: European conference on computer vision, pp 566–579. Springer. https://doi.org/10.1007/978-3-642-33712-3_41
 33. Sun Y, Chen Y, Wang X, Tang X (2014) Deep learning face representation by joint identification-verification. In: Advances in neural information processing systems, pp 1988–1996
 34. Sun Y, Liang D, Wang X, Tang X (2015) eepID3: face recognition with very deep neural networks. arXiv preprint [arXiv:1502.00873](https://arxiv.org/abs/1502.00873)
 35. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
 36. Schroff F, Kalenichenko D, Philbin J (2015) FaceNet: a unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 815–823 (2015)
 37. Wolf L, Hassner T, Maoz I (2011) Face recognition in unconstrained videos with matched background similarity. In: CVPR 2011, pp 529–534. IEEE. <https://doi.org/10.1109/CVPR.2011.5995566>
 38. Parkhi OM, Vedaldi A, Zisserman A (2015) Deep face recognition. In: Proceedings of the British machine vision conference, p 41.1–41.12
 39. Cao Q, Shen L, Xie W, Parkhi OM, Zisserman A (2018) VggFace2: a dataset for recognising faces across pose and age. In: 2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018), pp 67–74. IEEE. <https://doi.org/10.1109/FG.2018.00020>
 40. Klare BF, Klein B, Taborsky E, Blanton A, Cheney J, Allen K, Grother P, Mah A, Jain AK (2015) Pushing the frontiers of unconstrained face detection and recognition: Iarpa Janus Benchmark A. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1931–1939. <https://doi.org/10.1109/CVPR.2015.7298803>
 41. Wen Y, Zhang K, Li Z, Qiao Y (2016) A discriminative feature learning approach for deep face recognition. In: European conference on computer vision, pp 499–515. Springer
 42. Qi C, Su F (2017) Contrastive-center loss for deep neural networks. In: 2017 IEEE international conference on image processing (ICIP), pp 2851–2855. IEEE
 43. Liu W, Wen Y, Yu Z, Yang M (2016) Large-margin Softmax loss for convolutional neural networks. In: Proceedings of The 33rd international conference on machine learning, proceedings of machine learning research
 44. Liu Y, Li H, Wang X (2017) Rethinking feature discrimination and polymerization for large-scale recognition. arXiv preprint [arXiv:1710.00870](https://arxiv.org/abs/1710.00870)
 45. Liu W, Wen Y, Yu Z, Li M, Raj B, Song, L (2017) SphereFace: Deep hypersphere embedding for face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 212–220
 46. Wang H, Wang Y, Zhou Z, Ji X, Gong D, Zhou J, Li Z, Liu W (2018) CosFace: large margin cosine loss for deep face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5265–5274 (2018)
 47. Deng J, Guo J, Xue N, Zafeiriou S (2019) ArcFace: additive angular margin loss for deep face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4690–4699. <https://doi.org/10.1109/CVPR.2019.00482>
 48. Kemelmacher-Shlizerman I, Seitz SM, Miller D, Brossard E (2016) The MegaFace benchmark: 1 million faces for recognition at scale. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4873–4882. <https://doi.org/10.1109/CVPR.2016.527>
 49. Whitelam C, Taborsky E, Blanton A, Maze B, Adams J, Miller T, Kalka N, Jain AK, Duncan JA, Allen K, et al. (2017) Iarpa Janus Benchmark-B face dataset. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 90–98. <https://doi.org/10.1109/CVPRW.2017.87>
 50. Maze B, Adams J, Duncan JA, Kalka N, Miller T, Otto C, Jain AK, Niggel WT, Anderson J, Cheney J et al. (2018) Iarpa Janus Benchmark-C: Face dataset and protocol. In: 2018 international conference on biometrics (ICB), pp 158–165. IEEE. <https://doi.org/10.1109/ICB2018.2018.00033>
 51. Liu Y, Peng B, Shi P, Yan H, Zhou Y, Han B, Zheng Y, Lin C, Jiang J, Fan Y et al. (2018) iQIYI-VID: A large dataset for multimodal person identification. arXiv preprint [arXiv:1811.07548](https://arxiv.org/abs/1811.07548)
 52. Rao Y, Lin J, Lu J, Zhou J (2017) Learning discriminative aggregation network for video-based face recognition. In: Proceedings of the IEEE international conference on computer vision, pp 3781–3790 (2017). <https://doi.org/10.1109/ICCV.2017.408>
 53. Rao Y, Lu J, Zhou J (2017) Attention-aware deep reinforcement learning for video face recognition. In: Proceedings of the IEEE international conference on computer vision, pp 3931–3940. <https://doi.org/10.1109/ICCV.2017.424>
 54. Ding C, Tao D (2017) Trunk-branch ensemble convolutional neural networks for video-based face recognition. IEEE Trans Pattern Anal Mach Intell 40(4):1002–1014. <https://doi.org/10.1109/TPAMI.2017.2700390>
 55. Zheng J, Ranjan R, Chen CH, Chen JC, Castillo CD, Chellappa R (2020) An automatic system for unconstrained video-based face recognition. IEEE Trans Biom Behav Identity Sci 2(3):194–209. <https://doi.org/10.1109/TBIOM.2020.2973504>
 56. Chen JC, Lin WA, Zheng J, Chellappa R (2018) A real-time multi-task single shot face detector. In: 2018 25th IEEE international conference on image processing (ICIP), pp 176–180. IEEE. <https://doi.org/10.1109/ICIP.2018.8451649>
 57. Ranjan R, Sankaranarayanan S, Castillo CD, Chellappa R (2017) An all-in-one convolutional neural network for face analysis. In: 2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017), pp 17–24. IEEE. <https://doi.org/10.1109/FG.2017.137>
 58. Ranjan R, Sankaranarayanan S, Bansal A, Bodla N, Chen JC, Patel VM, Castillo CD, Chellappa R (2018) Deep learning for understanding faces: machines may be just as good, or better, than humans. IEEE Signal Process Mag 35(1):66–83. <https://doi.org/10.1109/MSP.2017.2764116>
 59. Kalka ND, Maze B, Duncan JA, O'Connor K, Elliott S, Hebert K, Bryan J, Jain AK (2018) IJB-S: IARPA Janus surveillance video benchmark. In: 2018 IEEE 9th international conference on biometrics theory, applications and systems (BTAS), pp 1–9. IEEE (2018). <https://doi.org/10.1109/BTAS.2018.8698584>
 60. Viola P, Jones M et al (2001) Robust real-time object detection. Int J Comput Vis 4(34–47):4
 61. Bansal A, Nanduri A, Castillo CD, Ranjan R, Chellappa R (2017) Umdfaces: an annotated face dataset for training deep networks. In: 2017 IEEE international joint conference on biometrics (IJB), pp 464–473. IEEE (2017). <https://doi.org/10.1109/BTAS.2017.8272731>
 62. Bansal A, Castillo C, Ranjan R, Chellappa R (2017) The do's and don'ts for CNN-based face verification. In: Proceedings of the IEEE international conference on computer vision workshops, pp 2545–2554. <https://doi.org/10.1109/ICCVW.2017.299>
 63. UMDFaces. <http://umdfaces.io/>. Accessed 19 Nov 2020
 64. Liu Y, Shi P, Peng B, Yan H, Zhou Y, Han B, Zheng Y, Lin C, Jiang J, Fan Y et al (2019) iQIYI celebrity video identification challenge. In: Proceedings of the 27th ACM international

- conference on multimedia, pp 2516–2520. <https://doi.org/10.1145/3343031.3356081>
65. Nagrani A, Chung JS, Zisserman A (2017) VoxCeleb: a large-scale speaker identification dataset. In: Proceedings of the 18th annual conference of the international speech communication association, pp 2616–2620 (2017). <https://doi.org/10.21437/InterSpeech.2017-950>
 66. Sivic J, Everingham M, Zisserman A (2009) ‘Who are you?’—learning person specific classifiers from video. In: 2009 IEEE conference on computer vision and pattern recognition, pp 1145–1152. IEEE. <https://doi.org/10.1109/CVPR.2009.5206513>
 67. Bauml M, Tapaswi M, Stiefelhagen R (2013) Semi-supervised learning with constraints for person identification in multimedia data. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3602–3609. <https://doi.org/10.1109/CVPR.2013.462>
 68. Nagrani A, Zisserman A (2018) From benedict cumberbatch to sherlock holmes: character identification in TV series without a script. arXiv preprint [arXiv:1801.10442](https://arxiv.org/abs/1801.10442)
 69. Huang Q, Liu W, Lin D (2018) Person search in videos with one portrait through visual and temporal links. In: Proceedings of the European conference on computer vision (ECCV), pp 425–441 (2018). https://doi.org/10.1007/978-3-030-01261-8_26
 70. Teng S, Tan W, Zhang W (2007) Cooperative shot boundary detection for video. In: International conference on computer supported cooperative work in design, pp 99–110. Springer. https://doi.org/10.1007/978-3-540-92719-8_10
 71. Ciaparrone G, Sánchez FL, Tabik S, Troiano L, Tagliaferri R, Herrera F (2020) Deep learning in video multi-object tracking: a survey. *Neurocomputing* 381:61–88. <https://doi.org/10.1016/j.neucom.2019.11.023>
 72. Guo Y, Zhang L, Hu Y, He X, Gao J (2016) MS-Celeb-1M: a dataset and benchmark for large-scale face recognition. In: European conference on computer vision, pp 87–102. Springer. https://doi.org/10.1007/978-3-319-46487-9_6
 73. Baraldi L, Grana C, Cucchiara R (2015) Shot and scene detection via hierarchical clustering for re-using broadcast video. In: International conference on computer analysis of images and patterns, pp 801–811. Springer. https://doi.org/10.1007/978-3-319-23192-1_67
 74. Souček T, Lokoč J (2020) TransNet V2: an effective deep network architecture for fast shot transition detection. arXiv preprint [arXiv:2008.04838](https://arxiv.org/abs/2008.04838)
 75. Souček T, Moravec J, Lokoč J (2019) TransNet: a deep network for fast detection of common shot transitions. arXiv preprint [arXiv:1906.03363](https://arxiv.org/abs/1906.03363)
 76. Zhang K, Zhang Z, Li Z, Qiao Y (2016) Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Process Lett* 23(10):1499–1503. <https://doi.org/10.1109/LSP.2016.2603342>
 77. Najibi M, Samangouei P, Chellappa R, Davis LS (2017) SSH: single stage headless face detector. In: Proceedings of the IEEE international conference on computer vision, pp 4875–4884
 78. Deng J, Guo J, Ververas E, Kotsia I, Zafeiriou S (2020) RetinaFace: single-shot multi-level face localisation in the wild. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 5203–5212
 79. sklearn.metrics.average_precision_score – scikit-learn 0.23.2 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html. Accessed 09 Dec 2020
 80. Bochinski E, Eiselein V, Sikora T (2017) High-speed tracking-by-detection without using image information. In: 2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS), pp 1–6. IEEE. <https://doi.org/10.1109/AVSS.2017.8078516>
 81. deepinsight/insightface: face analysis project on MXNet. <https://github.com/deepinsight/insightface>. Accessed: 15 Dec 2020

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.