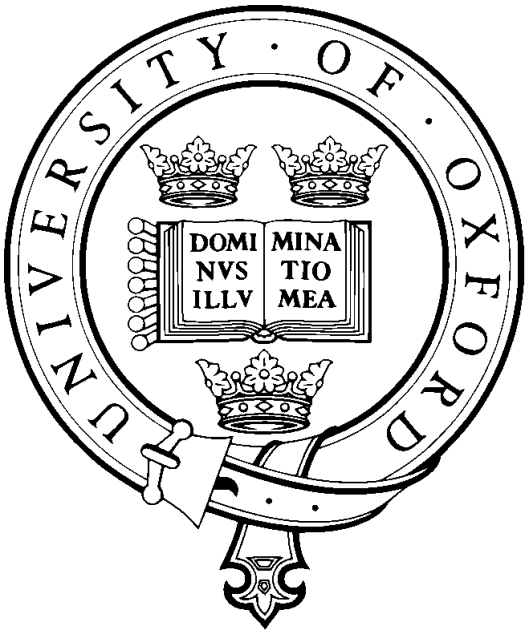
Features And Methods for Improving Large Scale Face Recognition



Omkar Moreshwar Parkhi

Brasenose College

University of Oxford

Supervised by
Dr. Andrea Vedaldi and Professor Andrew Zisserman
Submitted: Trinity Term 2015

Omkar M. Parkhi
Brasenose

Doctor of Philosophy
December 2015

Features And Methods for Improving Large Scale Face Recognition

Abstract

This thesis investigates vector representations for face recognition, and uses these representations for a number of tasks in image and video datasets.

First, we look at different representations for faces in images and videos. The objective is to learn compact yet effective representations for describing faces. We first investigate the use of "Fisher Vector" descriptors for this task. We show that these descriptors are perfectly suited for face representation tasks. We also investigate various approaches to effectively reduce their dimension while improving their performance further. These "Fisher Vector" features are also amenable to extreme compression and work equally well when compressed by over 2000 times as compared to their non compressed counterparts. These features achieved the state-of-the-art results on challenging public benchmarks until the re-introduction of Convolution Neural Networks (CNN) in the community. Second, we investigate the use of "Very Deep" architectures for face representation tasks. For training these networks, we collected one of the largest annotated public datasets of celebrity faces with minimum manual intervention. We bring out specific details of these network architectures and their training objective functions essential to their performance and achieve state-of-art result on challenging datasets.

Having developed these representation, we propose a method for labeling faces in the challenging environment of broadcast videos using their associated textual data, such as subtitles and transcripts. We show that our CNN representation is well suited for this task. We also propose a scheme to automatically differentiate the primary cast of a TV serial or movie from that of the background characters. We modify existing methods of collecting supervision from textual data and show that the careful alignment of video and textual data results in significant improvement in the amount of training data collected automatically, which has a direct positive impact on the performance of labeling mechanisms. We provide extensive evaluations on different benchmark datasets achieving, again, state-of-the-art results.

Further we show that both the shallow as well the deep methods have excellent capabilities in switching modalities from photos to paintings and vice-a-versa. We propose a system to retrieve paintings for similar looking people given a picture and investigate the use of facial attributes for this task. Finally, we show that an on-the-fly real time search system can be built to

search through thousands of hours of video data starting from a text query. We propose product quantization schemes for making face representations memory efficient. We also present the demo system based on this design for the British Broadcasting Corporation (BBC) to search through their archive.

All of these contributions have been designed with a keen eye on their application in the real world. As a result, most of chapters have an associated code release and a working online demonstration.

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Omkar M. Parkhi, Brasenose College

Acknowledgements

I would like to thank my supervisors, Professor Andrew Zisserman and Dr. Andrea Vedaldi for their guidance, support, advice and most importantly, encouragement. I would also like to thank my collaborators, Esa Rahtu, Karen Simonyan, Relja Arandjelović, Ken Chatfield, Elliot Crowley and Makarand Tapaswi for all their contributions. Special thanks to Rob Cooper from BBC R&D for providing quality data for building a search system. Many thanks to all project members of EU Project AXES and IARPA JANUS Program for intersting discussions on and off the topic. After spending nearly seven years at VGG it's difficult to name everyone in the lab. I also thank everyone in VGG, AVL, TVG and MRG for making it such a nice environment to work in. I would like to thank my parents for all their support and understanding. And finally, many many thanks to my wife Shruti and my daughter Ira for all the wonderful time throughout this journey.

Contents

Chapter 1: Introduction

1.1 Objective and Motivation

The objective of this thesis is to develop powerful features for face recognition, and algorithmic methods which exploit these features for various tasks. We focus on representing faces in images and videos and look at applications of these representations for various activities such as labeling faces in movies and TV series, retrieving paintings of celebrities given pictures of their look-a-likes and searching on very large scale un-annotated datasets. All these activities fall under the umbrella of face recognition and require specific attention and modification in underlying features and algorithms. In the following, we explain some face recognition related activities relevant in the context of this thesis.

Face Identification. Face identification is the most well known of the face recognition tasks. The goal here is to assign identities to the faces in given images and videos. The task is to build identity specific models for different people and evaluate them on a new image or a video. Variations in different facial attributes such as age, expressions, facial hair etc.need to be accounted for while building such models. It finds major applications in search and in annotating collections. Fig. 1.1 shows an example of this task.

Face Verification. In face verification, images or videos are presented in pairs and the task is to verify if they belong to the same or different persons. Applications of this task are in search and authentication domains. This task has recently gained lot of popularity owing to few public benchmark datasets being available. Fig. 1.2 shows example of such a task.

Figure 1.1: Identification: *The goal here is to label faces with the corresponding identity labels. This can be done by learning identity-specific models and applying them to incoming images or videos. An example of this can be seen above in the famous $30bn queue of Indian business personalities on their way to meet US president Barack Obama on his state visit.*

Figure 1.2: Verification: *Given a pair of images or videos, the goal here is to confirm whether the identity of person appearing in them is same (top row) or different (bottom row). Some of the challeges in matching images of the same identity can be changes in the age, fate or self enlightenment. For non matching identities the challenges include, natural similarity, similarity in view point and sentiments, though sometimes identities in the picture differ just naturally.*

Face Retrieval. Given a large collection of faces from images and videos, the goal here is to retrieve faces of a particular identity starting from a text query or example images of that person. The challenge is to make this work in real time. Along with identities, the collection can also be searched based on facial attributes. Fig. 1.3 shows a very successful application; "Google Image Search".

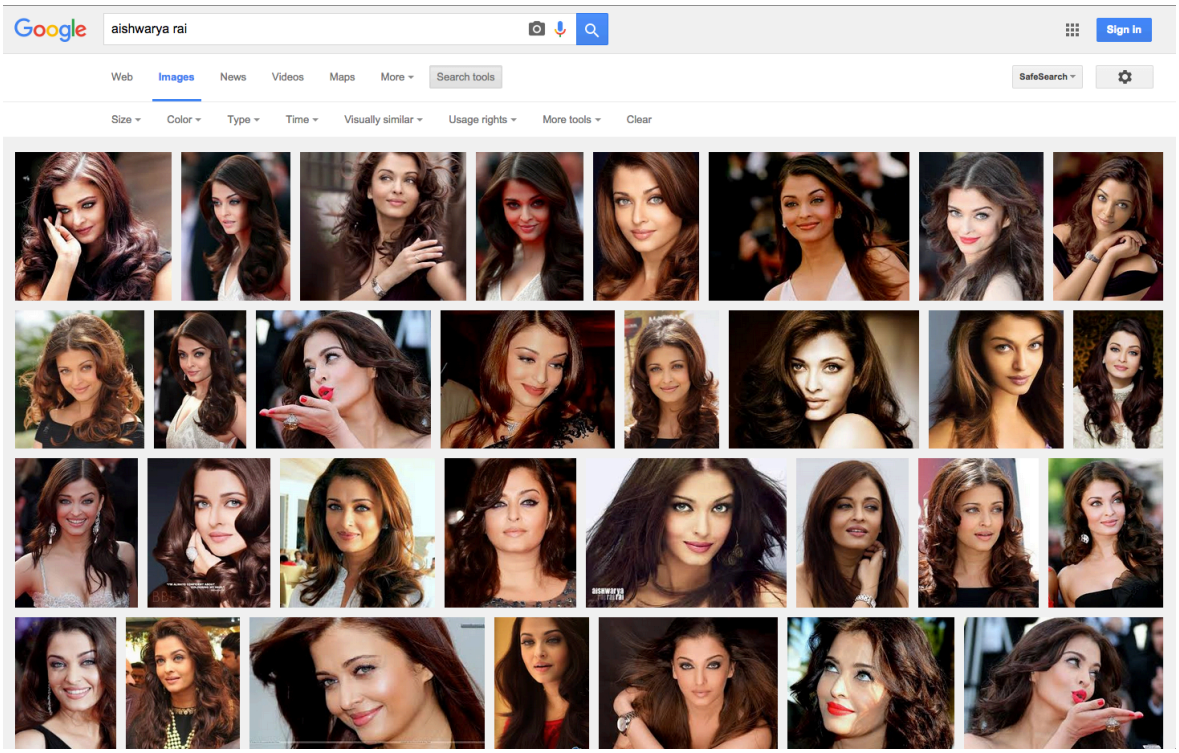


Figure 1.3: Search: *Search is a classic application of face recognition. The goal is to quickly and accurately retrieve images or videos of a person from a collection given a query text or an image. Collections to be searched can be as big as the world wide web like in this case.*

1.1.1 Challenges

There are many challenges in dealing with the applications listed above, here we list a few.

Appearance variations. While building models for identities or trying to verify the identities given a pair, there are several factors that a typical system needs to be invariant to. Human faces are naturally expressive. Variations in facial expressions can be a cause of confusion. Similar contributors are changes in pose, age variation, and lighting conditions. In addition to this eyeware, partial occlusions, facial decorations etc. cause changes in the appearance making it

harder to recognise faces. In the case of actors, changes due to character specific getup also contribute to poor performance if not accounted for. Fig. 1.4 shows some examples of these types of problems.

Lack of large scale supervised datasets. Building models for assigning the identities, one needs to use annotated images of people to train their models. Convolution Neural Networks (CNNs) have been instrumental in advancing the state-of-the-art in image and video classification. These models typically require millions of annotated training examples. While few such datasets exist for the general image/object classification domain, no such dataset exists for face recognition tasks. Naturally, most of the advances in face recognition using CNNs in the recent past have been coming from web giants having access to large collections of images.

Speed and scalability While working with the collections as big as BBC's digital archieve, both speed and scalability of the proposed solution attain paramount importance. While searching through thousands of hours of data and millions of frames, results are still expected in real-time. Also, representing faces in millions of images demands careful design of the feature representation. Training models for a specific identity requires sourcing images for training and processing them in real time.

Content processing Challenges here are twofold. Large datasets require modifications to the processing algorithms to be processed in reasonable time. As the size of such datasets increases, even the real time processing may not be adequate for processing the contents in a reasonable time. Additionally these videos are encoded using various different algorithms, have several different resolutions and aspect ratios and may or may not have interlaced encodings. The same is applicable to textual data. There are various formats available for subtitles and there is almost no specific format for transcripts which accompany videos. All these factors need to be accounted for while proposing any solutions.

Figure 1.4: Visual challenges: *First row: Queen Elizabeth II is Britian's longest serving monarch. Naturally, her photographs and videos available on the web and in the archieves cover similar span showing great age variation amongst them. Second Row: An actor plays various roles throughout his/her career. Each character is essentially a different personality resulting in appearance variation. Pictures show appearace variation for British actor Sir Ben Kingsley who has portraied variety of characters throughout his career. Bottom Rows: Somebody as important as the US President finds himself constantly pictured by photographers. The result is tremendous variation in facial expressions and pose. This is in contrast to the previous two examples. The age and the appearance are relatively stable as compared to the previous examples.*

1.2 Specific Applications

Having looked at objectives and the challenges, we look at some of the specific applications in the broad area of face recognition. We have worked on some of these as a part of this thesis.

Video search. As explained before institutions with large un-annotated corpus of videos often feel a need for an automated real time search engine. While Google and Microsoft provide this on the web, there is the need to build similar systems for private content. We are collaborating with the British Broadcasting Corporation (BBC) to provide them with video search functionality. This is particularly useful for their production and archive teams to retrieve old footage of a celebrity for reuse. We have built a system allowing them to search through over $10,000$ hours of broadcasts. Use of such a system helps them save hours of efforts to manually look through hours and hours of programming. Fig. 1.5 shows an example of our system's output for a textual query.

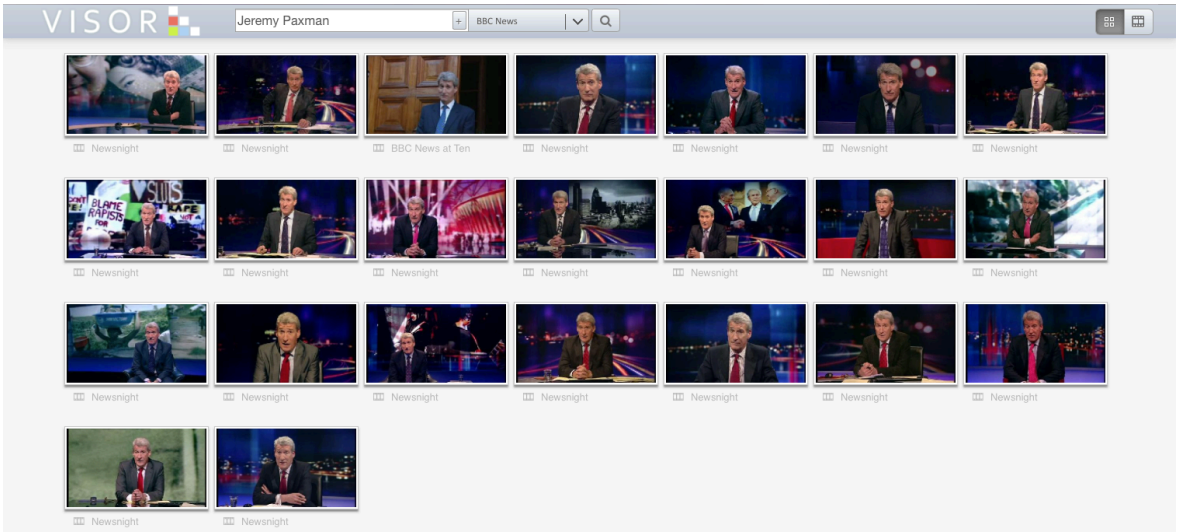


Figure 1.5: Searching BBC videos: *Top results for on-the-fly visual search to find videos of "Jeremy Paxman". The dataset contains over 10,000 hours of video broadcasting from BBC's prime time news programs across all channels.*

Actor identification. How many times have you had a moment that while watching your favorite movie or a TV series, you find a face familiar but can't remember where you have seen them before? Actor identification solves just this problem. Some of the web's popular

content streaming companies have already started providing these services to their customers (Fig. 1.6). We look at tackling this problem without any manual supervision and making use of subtitles and transcripts which typically accompany such content.

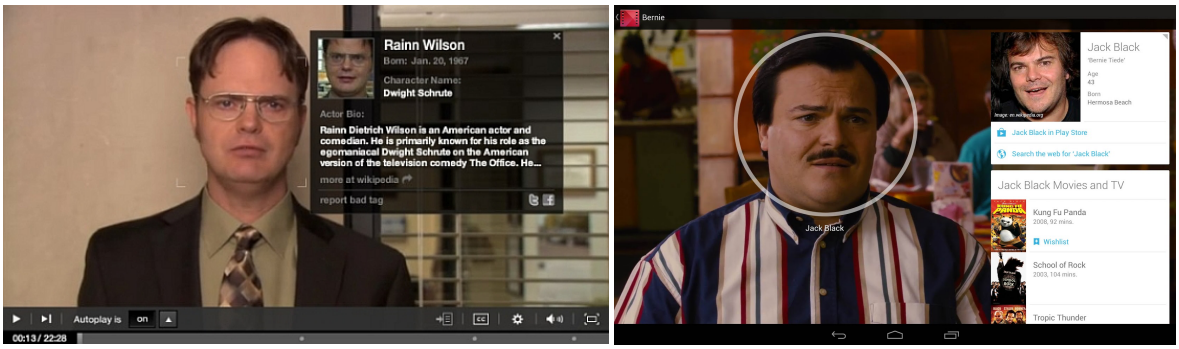


Figure 1.6: Actor Spotting: *Major content providers are using face recognition or related technologies to provide additional information to the viewers. Left: Hulu's face recognition based actor cards help viewers find out more information about actors appearing on the screen. Right: Google also has similar actor cards for some selected movies on their play store.*

Organizing personal photo collection. With the tremendous increase in the personal digital content thanks to the smart phones, egocentric cameras and drones there is a growing need for automatic organization of these pictures and videos. Recently, Google and Facebook released photo organizers equipped with a capability to cluster picture collections based on person identities.

Exploring portraits collections. There are a large number of portraits and paintings in Britian's galleries. In a similar spirit to that of our work with the BBC, we are working with the Public Catelogue Foundation (PCF) to enable searching for portraits of famous identities.

Visual anthropology. Another rather unexpected application is in assisting studies in anthropologies. Some of the tribes in Africa make specific markings on the face, some have specific styles for posing for photographs, while some migrate from place to place where they might be pictured. A system similar again to the one developed for the BBC can be used to assist in searching for all photographs of a particular trait.

1.3 Contributions and thesis outline

In this section we list some of the key contributions made in this thesis. First we present feature representations for describing faces in the images and videos and later build algorithms developed using these features for various application. Specifically, Chapter 3 is about use of Fisher Vector encodings for Face Representation. Chapter 4 discusses Deep Convolution Neural Network architectures for face recognition and construction of a large scale dataset required to train them. Chapter 5 investigates use of these features for labeling faces in broadcast videos while Chapter 6 investigates transfer learning capabilities of these features by looking at tasks of retrieving paintings given pictures and vice-a-versa. Finally, Chapter 7 discusses methods and architectures of a large scale on-the-fly search system. The contributions of each of these chapters are briefly discussed next.

1. Shallow features for face representation. (Chapter 3) We investigate use of Fisher Vectors [112] for representing faces in images and videos. We show that similar to the object recognition pipe line, these features are equally powerful in representing faces in images and videos. Additionally we present schemes to encode face tracks in videos into a single descriptor and investigate methods for incorporating dataset augmentation. Finally, we show that the rather high dimensionality of these descriptors can be reduced by orders of magnitude without loss in performance.

2. Deep features for face representation. (Chapter 4) We show that our CNN representation based on "Very Deep" architectures [135] achieves the state-of-the-art performance on public benchmark datasets. We also show the importance of the objective function in learning such networks. We present an elaborate method to collect large amount of training images required for training CNNs. More specifically, we create a dataset with 2.6 Million images with 2622 different identities.

3. Weakly supervised labeling of characters in broadcast video. (Chapter 5) We show that as the face detectors become more and more accurate, they generate more confusion for the labeling scheme by detecting tracks belonging to "background" characters. These are extra characters typically appearing in the background of a shot due to the location or the script demand. We present a method to separate main characters from these background characters and show that this approach significantly improves performance of labeling characters in a movie or television series given its transcript and subtitles. We also show that very accurate negative supervision can be obtained for these tasks from the union of subtitles and transcripts. We also show that both shallow and deep features are extremely effective for this task and we push labeling performance on some of the standard benchmark datasets to near saturation.

4. Investigating transfer characteristics of representations. (Chapter 6) We look at transfer capabilities of both shallow and deep features with an application to retrieval of paintings given pictures and vice-a-versa. We show that CNN features show exceptional transfer abilities while Fisher Vector features need additional learning. We also look at use of facial attributes to retrieve similar looking faces to a given query image.

5. Large scale face retrieval. (Chapter 7) We discuss methods for building a large scale face retrieval system. We describe architectural details, modifications to feature representations and a source of training data for making this search on-the-fly. We evaluate performance on large scale datasets and show examples from the system built for the BBC on 10,000 hours of programming.

1.4 Publications

The work of Chapter 3 on face verification in images was presented in BMVC 2013 [131]. The material on face verification in video was presented in CVPR 2014 [107], while the contributions of Chapter 4 were presented in BMVC 2015 [109]. Part of Chapter 5 was presented in CVPR 14 along with the work on video verification above. The major contributions of that

work is under review at ICCV 2015 workshop. The contributions of Chapter 6 were presented in BMVC 2015 [38]. the on-the-fly retrieval work described in Chapter 7 was presented at WIAMIS 2012 [108] as in IJMLR [28].

Additional publications. Other papers published during the course of this PhD include: (i) work on TRECVid project which is mainly based on Chapter 7 - TRECVid 2012 [7], TRECVid 2013 [6] and ICMR 2013 [99]. (ii) unsupervised clustering of face tracks in broadcast videos using Fisher Vector features was presented in ICVGIP 2014 [152]. (iii) slightly outside the central theme of the thesis, work on fine grain categorization of Cats and Dogs was presented in CVPR 2012 [110]. All these publication have been excluded from this thesis due to lack of space.

Chapter 2: Related Work

In this chapter we review the development of the field of face recognition. In the following chapters, we consider problems ranging from face representation and face labelling to efficient retrieval of faces in large scale datasets. Here, we look at previous work related to each of these contributions. We begin by discussing in Sect. 2.1 face detection and tracking as it is the most basic building block of our work. Then in Sect. 2.2 we look at different representations, both shallow and deep, that can be used to describe faces in images and in videos. In Sect. 2.3.1, we look at approaches to label faces in broadcast media given the representation and some additional cues. Finally we conclude the chapter with the review of prior work on face retrieval and domain transfer from pictures to paintings.

2.1 Faces, Landmarks detection and tracking

Detecting faces and tracking them in videos are the first steps in most face recognition pipelines. They therefore have a significant impact on the performance of the face representation or labelling algorithm that is applied on top. For example, in the case of face representation, if the underlying face detector can only extract frontal faces, it is highly likely that the learnt representation will not be invariant to the pose of the face. Similarly in Chapter 5, we show that as the face detectors become more and more capable of detecting all faces in a frame of a video, it creates serious challenges for the labelling task built on top of it. Landmark detection is essential for various tasks. First, it allows finding point correspondences to perform 2D and 3D alignment of faces. It is also helpful in the analysis of facial attributes; for example, the region near the eyebrows may not help in deciding whether a person sports a moustache

and the region near the mouth is unlikely to reveal whether a person is wearing glasses. Once faces are detected in a video, face tracking forms temporal face sequences. This is important for two reasons. Firstly, it provides an unsupervised clustering mechanism to group together faces that share the same identity. Secondly, it reduces the computational cost of the following representation and labelling steps by avoiding the need to consider each individual frame as a separate labelling problem. In the rest of the section, we look at face detection, followed by facial landmark detection, and finally face tracking.

2.1.1 Face Detection

Face detection is one of the classic topics in the field of computer vision and as a result thousands of paper have been written on the topic. In this section we look at some of the most notable contributions to the field. First, we look at some of the classic approaches to face detection using classifier cascades. Then we review notable approaches using CNN and we conclude by reviewing Deformable Parts Model (DPM) [55] based methods.

Cascade Methods. These methods apply a series of weak classifiers to achieve the goal of obtaining strong classification. Each weak classifier is trained to reject proposals from the previous weak classifier. Viola-Jones [158] (VJ) introduced a very popular implementation of this scheme for face detection which achieved real-time highly-accurate detection. The VJ detector used integral images to evaluate a large amount of detection hypotheses using boosted weak classifiers. Part of its popularity is due to its implementation in the OpenCV [25] package. The original implementation was designed for detecting frontal faces. Later, a profile detector was also included in OpenCV. Several works were built on top of this replacing Harr-like features with better features. We refer readers to an extensive review presented in [174] for details. The window-centric approach of Viola *et al.* [158] was modified to a feature-centric cascade detection approach by Schneiderman *et al.* [121]. This resulted in a very accurate, robust and fast approach to multi-view face detection. Unfortunately, no public implementation exists for this method.

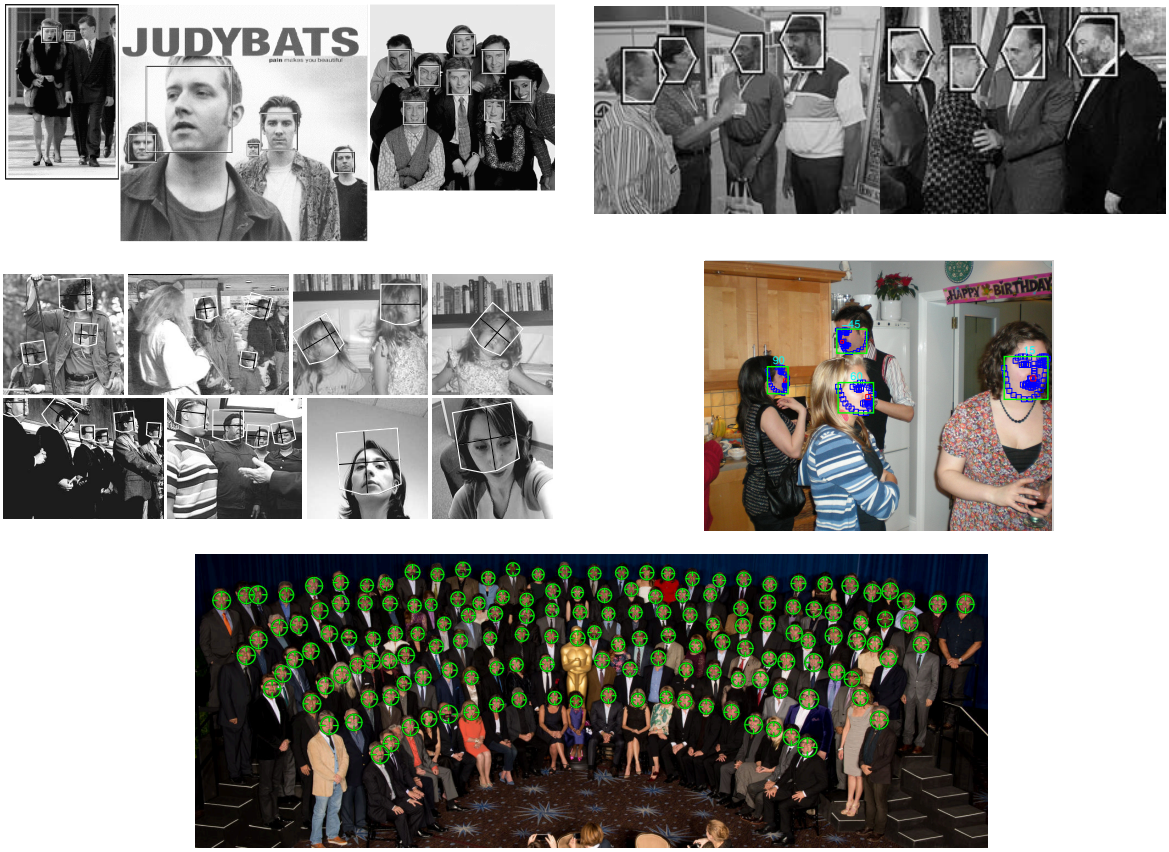


Figure 2.1: Face detection through time: *Top Left: The cascade detection approach of [158]. Their approach resulted in fast but mostly near-frontal detections. Top Right: The Pitpatt detector of [121] provided a method to accurately detect frontal as well as profile detections. Middle left: An early CNN based approach was presented in [104].Their method presented a unified solution to both frontal and non-frontal detections alongside the pose estimation. Middle right: A More recent tree structured DPM approach of [178]. Their detector is capable of detecting more extreme poses along with their facial landmarks. Bottom row: The head hunter model of [98]. Their method is capable of realtime detection of faces in various poses and at extreme scales.*

Although these cascaded methods were fast, they mainly focussed on detecting frontal faces. Their performance in detecting faces in low resolution images, non-frontal poses, and in challenging lighting conditions remained a challenge to be resolved.

CNN based approaches. Convolutional Neural Networks (CNNs) allow the learning of rich representations optimally tuned to a task. One of the early adaptations of CNNs for the task of multi-view face detection was presented in [104]. In addition to end-to-end learning, they trained a network to predict pose simultaneously with the detection. Very recently, [90] presented a cascade of CNNs approach for detecting faces. As the name suggests, they trained several networks to work in cascade fashion and achieved state of the art results on various standard benchmark tests. Recently, CNN-based object detection has seen tremendous improvements in performance both in terms of accuracy and speed. These approaches use region proposal algorithms to obtain a shortlist of image regions that may contain object occurrences; then a CNN trained for object classification is used to classify these windows as object occurrences or background. The fast R-CNN approach of [60] proposed a joint framework for region proposals as well as classification thereby providing a fast and accurate method for object detection. Whether such approaches result in improving face detections is something that remains to be seen.

Deformable Part Models. Felzenswalb *et al.* [55], proposed a Deformable Part (DPM) Based method for object detection. They modelled objects as collections of parts with associated deformation costs. Before the introduction of the latest generation of CNNs, DPMs were the most popular approach for object detection. Felzenswalb *et al.* also published a faster version of this method using a cascade strategy [52]. Zhu and Ramanan [178] used this technique for multi-pose face detection. Along with face detection, their method provided the location of facial landmarks. Although very robust in detecting faces with significant pose variations, this method had two major drawbacks. First, the model was trained on very high resolution images (HD) and as a result, its performance on low resolution (non-HD) images was poorer. Second, due to the use of multi-view DPMs with several parts for different face landmarks, the result-

ing implementation was extremely slow, requiring a few minutes per image on a single core CPU. Despite these shortcomings, it received large adaption owing to its excellent multi-view detection performance. Very recently, Matthias *et al.* [98] presented a very fast face detector based on their earlier work on fast pedestrian detection [18]. The proposed face detector was shown to be extremely fast and achieved state-of-the-art performance on many face detection benchmarks including the FDDB [71] dataset. The paper also presented another important comparison. Along with their fast method, a face detector based on the DPM [55] was also shown to achieve state-of-the art performance. This was an important finding since it not only made available a robust detector like that of [178] but one that was significantly faster, without the need of special GPU-based implementations. In our work, we use the retrained Cascaded DPM version of this model [52] which works at about 1 frame/sec on standard HD images.

In addition to these face detection approaches, researchers have looked at other approaches to face detection. Tapaswi *et al.* [151] used the Modified Census Transform based approach of [59]. Marin-Jimnez *et al.* [97] took a slightly different approach and used a DPM-based head detector instead of performing traditional face detection. An advantage of this was being able to detect the back of heads which made the subsequent tracking more robust. Hoai *et al.* [66] provided a method for detecting people in broadcast media based on configurations in which they appear. There are many approaches for upper-body detection which bring robustness to the face detection but reviewing them all here is beyond the scope of this review.

2.1.2 Facial landmark detection

Facial landmark detection is another important component of the face processing pipeline. There are multiple applications of this stage. Detected facial landmarks can be used as an additional confidence cues to the original face detector. Facial landmarks also provide salient regions of the face and their surrounding regions can be used to describe faces. The confidence score of these landmark detections can also be used for pruning detections. The main use of these landmark detections is in aligning the face images to a canonical pose. This application is so popular that facial landmark detection and facial alignments are used almost interchangi-

bly in recent publications. Here we review some of the classic methods for detection of these face landmarks and some methods for aligning them to a canonical pose. In one of the early methods, Everingham and Zisserman [51] presented a method to localise eyes given the face detection. They later proposed pictorial structure [53] based method for detecting nine facial landmarks [48] to assist with multiple tasks in face labelling. They used the landmark detections for face alignment, feature extraction and to detect whether a person is speaking or not based on their lip movement patterns. Fig. 2.2 shows example detections of this method. In other related methods Cootes *et al.* [35] suggested an Active Appearance Based Method for landmark localization and alignment. This popular method has been widely used in the community for facial alignment. In a relatively recent work, Dantone *et al.* [43] presented a Conditional Random Forest Regression approach for detecting landmarks. This method used Random Forests for the detection of landmarks in achieving real time and accurate detections. Xiong and De la Torre [168] presented another fast yet accurate method for landmark detection. During training time, their method learnt gradient steps to be taken given appearance features at current location estimates. The estimates were then modified using the gradient and the process was repeated. There are multiple applications of this as discussed above. Facebook showed that landmark detections can be used to frontalise faces and achieve better recognition performance [149]. Like face detection, frontalisation is also widely used as a pre-processing step in face understanding; however, reviewing all the methods for frontalisation is beyond the scope of this review.

(a) Face detections in original frames (b) Localized facial features

Figure 2.2: Facial landmark detection: *An example detection using the pictorial structure based method of Everingham* et al. *[48]. Nine landmark points were then used for various purposes such as Face Alignment, Face Representation and Speaker Detection.*

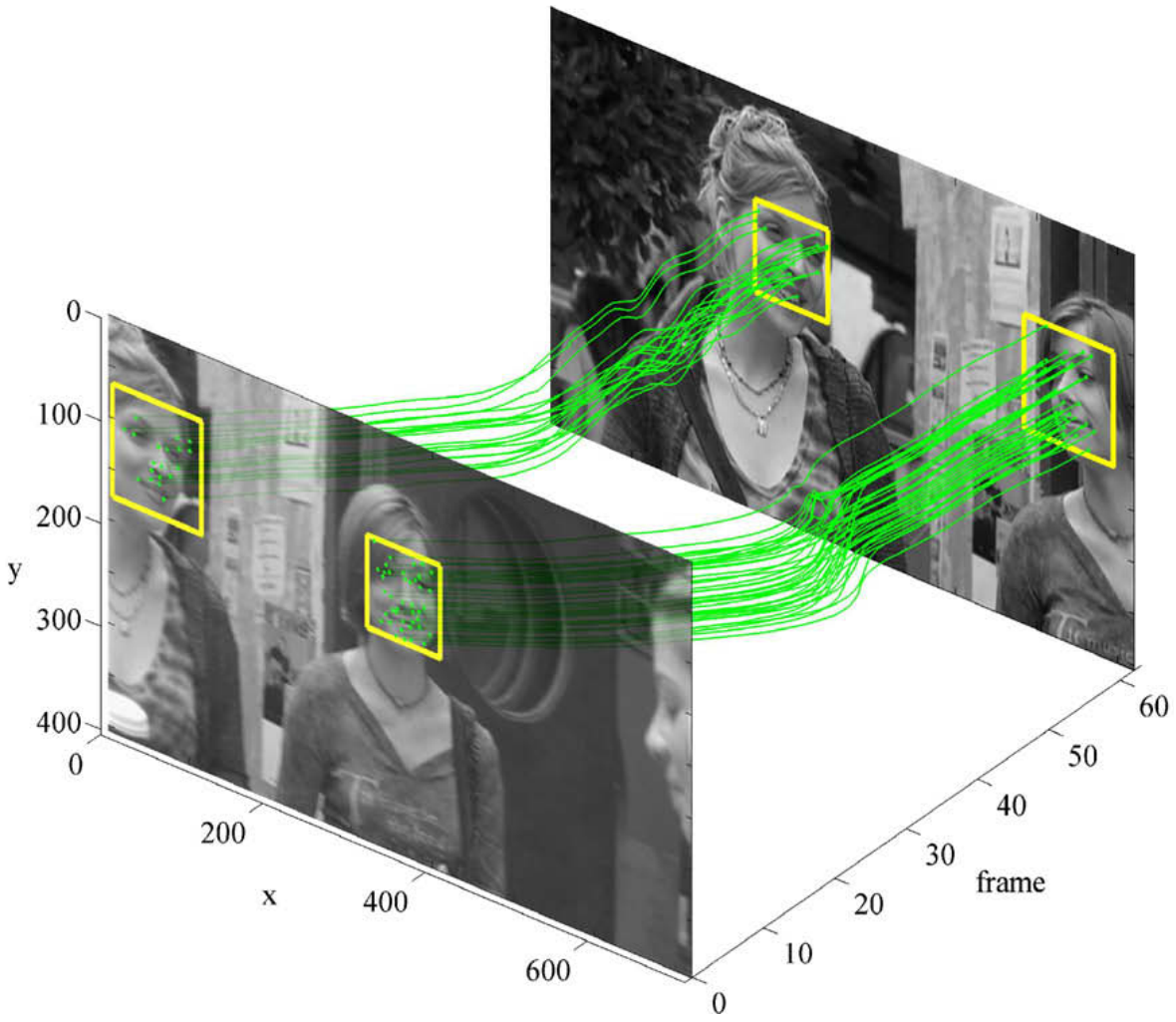


Figure 2.3: Face tracking: *The face tracking approach of [48]. KLT feature points are tracked in forward and backward directions in a shot and their intersection with a previously computed face detection is measured to obtain interlinked face detections providing face tracks.*

2.1.3 Face tracking

The goal of face tracking is to group face detections into face tracks corresponding to the same person in a video shot. In general, tracking is performed by first running a general-purpose region tracker and then associating face detections in different frames based on the region tracks connecting them. Sivic *et al.* [136] presented such an approach. It was further extended by [48]. The idea was to use the Kanade-Lucas-Tomassi (KLT) [129] feature point tracker on all frames in a shot. Detections intersecting with a set of features are assumed to be belonging to the same face track. To achieve robustness, feature point tracking is run in both directions in a shot, once from start to end and later in reverse. This helps in finding more features that overlap with detections providing robust tracking. Fig. 2.3 shows an overview of this method. At times, tracking introduces false positives largely due to false positive face detections. It becomes essential to remove them as they otherwise impact on the performance of weak labelling methods (Sect. 2.3.1) by introducing multiple candidates per label. Klaser *et al.* [77] and Tapaswi *et al.* [152] proposed an elaborate method for carrying out this post-processing by removing false positive tracks. Their method trains a classifier for false positive track removal using features built on track statistics. In our work, we use the tracks provided by the authors of the respective publications we will be comparing with. For the 'Scrubs' dataset described in Chapter 5, we use an implementation of Everingham *et al.* [48] described above. In the following sections we review previous work directly related to the contributions of this thesis.

2.2 Face Representation

Representing images has been an important topic in computer vision since its early days. The same has been true for face representation, where researchers have been investigating the use of general-purpose image representation as well as face-specific solutions. The diversity of methods is surprising. Some methods propose holistic representations of faces such as the eigenfaces of Turk *et al.* [155], while other methods build representations starting from detected

facial landmarks such as the pixel intensity based descriptor of [48]. Another distinction, which has been particularly important in recent years, is the use of shallow or deep models. The following section is organised around this: we first review methods using shallow representations and then look at deep ones. At the end of the section we will look at methods for representing collection of faces such as video face tracks.

2.2.1 Shallow features

Several papers investigated shallow features including LBP and its variants [31, 32, 67, 92, 101, 115, 148, 163, 164], SIFT [63, 92, 136], and learnt representations [115, 172]. Sharma *et al.* [126] used the Fisher vector encoding of local intensity differences as a face descriptor. Another interesting approach is to learn and extract semantic face attributes as facial features for identification and other tasks [19, 84]. A HOG based representation was used in [140]. Another standard method is that of [48] where features are computed around facial landmarks and concatenated to form a descriptor vector for representation. They computed normalised intensity patches around these regions for representation. Sivic *et al.* [136] used SIFT to represent the regions around the landmarks. A similar approach was later used in [63]. Statistical learning is generally used to map face representations to a final recognition result, with metric or similarity learning being the most popular approach, particularly for Face Verification tasks [32, 63, 67, 101, 173]. Some of these approaches will been discussed later in this section. Another popular approach is based on exemplar SVMs [148, 163, 164].

Dense features and their encodings for generic object recognition. Dense feature extraction is an essential component of many state-of-the-art image classification methods [87, 103, 119]. The idea is to compute features such as SIFT densely on an image, rather than on a sparse and potentially unreliable set of points obtained from an interest point detector. Dense features are then encoded into a single feature vector, summarising the image content in a form suitable for learning and recognition. The best known encoding is probably the Bag-of-Visual-Words (BoVW) model [41, 138], which builds a histogram of occurrences of vector-quantised

descriptors. More recent encodings include VLAD [73], Fisher Vectors (FVs) [112], and Super Vector Coding [177]. A common aim of these encodings is to reduce the loss of information introduced by the vector quantisation step in BoVW. In [27] it was shown that FVs outperform the other encodings on a number of image recognition benchmarks, so we adopt them in this thesis for face description.

2.2.2 Deep Features

Despite the fact that deep features only recently demonstrated their potential, there is a considerable amount of work that uses deep CNN architectures for representing faces. The defining characteristic of such methods is the use of a CNN feature extractor, a learnable function obtained by concatenating several linear and non-linear operators. A representative system of this class of methods is *DeepFace* [149]. This method uses a deep CNN trained to classify faces using a dataset of 4 million examples spanning 4,000 unique identities. It also uses a *siamese network* architecture, where the same CNN is applied to pairs of faces to obtain descriptors that are then compared using Euclidean distance. The goal of training is to minimise the distance between congruous pairs of faces (*i.e.*portraying the same identity) and maximise the distance between incongruous pairs, a form of *metric learning*. In addition to using a very large amount of training data, DeepFace uses an ensemble of CNNs, as well as a pre-processing phase in which face images are aligned to a canonical pose using a 3D model. When introduced, DeepFace achieved the best performance on the Labelled Faces in the Wild (LFW; [68]) benchmark as well as the Youtube Faces in the Wild (YTF; [162]) benchmark (Sect. 3.4). The authors later extended this work in [150], by increasing the size of the dataset by two orders of magnitude, including 10 million identities and 50 images per identity. They proposed a bootstrapping strategy to select identities to train the network and showed that the generalisation of the network can be improved by controlling the dimensionality of the fully connected layer.

The DeepFace work was extended by the DeepId series of papers by Sun *et al.* [142–145], each of which incrementally but steadily increased the performance on LFW and YTF. A number of new ideas were incorporated over this series of papers, including: using multiple

CNNs [144], a Bayesian learning framework [31] to train a metric, multi-task learning over classification and verification [142], different CNN architectures which branch a fully connected layer after each convolution layer [145], and very deep networks inspired by [135, 146] in [143]. Compared to DeepFace, DeepID does not use 3D face alignment, but a simpler 2D affine alignment and trains on a combination of CelebFaces [144] and Chen*et al.* [31]. However, the final model in [143] is quite complicated, involving around 200 CNNs.

Very recently, researchers from Google [123] used a massive dataset of 200 million face identities and 800 million image face pairs to train a CNN similar to [146] and [125]. A point of difference is in their use of a "triplet-based" loss, where a pair of two congruous (a, b) and a third incongruous face c are compared. The goal is to make a closer to b than c; in other words, unlike other metric learning approaches, comparisons are always relative to a "pivot" face. This matches more closely how the metric is used in applications, where a query face is compared to a database of other faces to find the matches. In training, this loss is applied at multiple layers, not just the final one. This method currently achieves the best performance on LFW and YTF.

2.2.3 Discriminative dimensionality reduction.

The aim of discriminative dimensionality reduction is to obtain smaller image descriptors, while preserving or even improving their ability to discriminate images based on their content. This is often formalised as the problem of finding a low-rank linear projection W of the descriptors that minimises the distances between images with the same content (*e.g.*same face) and maximises it otherwise. "Fisherfaces" [17] is one of the early examples of discriminative learning for dimensionality reduction, applied to face recognition. A closely related formulation is that of learning a Mahalanobis matrix $M = W^\top W$, a problem that has convex formulations [160], even in the case of low-rank constraints [132]. However, learning the matrix M is practical only if the starting dimensionality of the descriptor is moderate (*e.g.*less than 1,000 dimensions), so different approaches are required otherwise. One approach is to first reduce the dimensionality using PCA, and then perform metric learning in a low-dimensional

space [32, 63], but this is suboptimal as the first step may lose important discriminative information. Another approach, which we use later in Chapter 3, is to directly optimise the projection matrix W, as its size depends on the reduced dimensionality, although this results in a non-convex formulation [64, 153]. Sharma *et al.* [127] presented an interesting approach with "Expanded Parts based Metric Learning" to select discriminative regions that improve verification accuracy. They allowed the spatial bins used to pool local features to adapt their shapes to be more robust to occlusions. Fig. 2.4 shows an illustration explaining their approach.

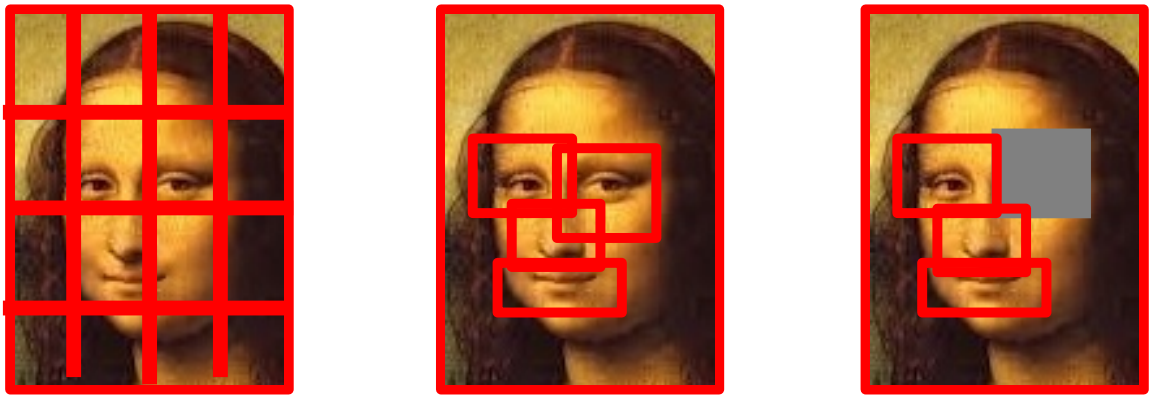


Figure 2.4: EPML: *An Expanded Parts based Metric Learning (EPML) model can optimally mine out the spatial bins required for the task (middle) leading to the discriminative full representation capable of handling occlusions.*

2.2.4 Representing faces in videos

Another important goal our work is to learn representations for video face recognition "in the wild". For the most part authors have focused on face tracks that contain frontal faces only, but a small number of papers have also included profile faces [50, 137, 151, 162]. Irrespective of pose, there are two approaches that have been followed in the previous search for representations. The first approach is based on representing faces in a track individually and comparing two face tracks as sets of images. The second, is to represent the track as a whole, for example by a single vector or as a 3D model or other manifold, with no explicit reference to the individual face detections. A good survey of these methods can be found in [15]. An early approach was suggested in [48] where features are computed around facial landmarks and concatenated to

form a descriptor vector for each frame of the track. Tracks are subsequently classified using the min-min distance on the set of descriptors (either as a nearest neighbor classifier or as a kernel). Others have varied the landmark features, *e.g.*multi-scale SIFT or HOG [9,34,136,137], as well as the type of the classifier, *e.g.*random forests, MKL, or SVM-minus scoring [9,137,165], and also the weight given to each frame [141]. Sharma *et al.* [128] presented a latent max-margin scheme for comparing face tracks, which effectively mitigates the effects of pose, expression, and illumination variations in comparing face tracks.

The second approach has used both generative and statistical representations. One method is to build 3D models from the data [81,176] and use these to explain new data, for example by view synthesis [106]. Often, the face track is represented as a linear or non-linear manifold [10, 65]. Others have learnt local features across the track and represented them using a BoW model [136], or a GMM [89], or by sparse coding [42]. If any linear inference mechanism is used on these features, then averaging features over frames of the track and then multiplying them with a classifier vector turns out to be equal to running the classifier on every frame and then averaging the scores. The former approach results in faster speeds and low storage requirements. A similar approach was used in [151]. An alternative representation is to simply use the affine hull of the face sets [26,171].

2.3 Applications

Having explained the representation, in the following sections we review various methods related to the application part of this thesis.

2.3.1 Weakly supervised learning

One of the earliest works to tackle the weakly supervised learning problem in face recognition is that of Everingham *et al.* [47]. This paper introduced three ideas that have been adopted by most of the follow-up work: (i) associating frontal faces in a shot using tracking by detection (*i.e.* detections before tracking), so that a face-track is the "unit" to be labelled; (ii) the use of aligned transcripts with subtitles to provide supervisory information for character labels; and

(iii) visual speaker detection to strengthen the supervision (if a person is speaking then their identity is known from the aligned transcript). In their subsequent work [48, 137], the authors extended their framework by adding profile face tracks and investigating other features and classifiers, but did not change the supervisory regime.

A significant extension in the use of supervisory information was introduced by Cour *et al.* [36] who cast the problem as one of ambiguous labelling. The key innovations were: (i) that supervisory information could be used from every shot (not just where a person is speaking); and (ii) a convex learning formulation for multi-class labelling under these partially supervised conditions. The idea of formulating the problem as one of multiple labels [37] is very appropriate for this scenario, given the ambiguous supervision available. An alternative approach to ambiguous supervision is to use Multiple Instance Learning (MIL), as used by [24,78,161,170]. In particular the work of Bojanowski *et al.* [24] proposed an elegant convex relaxation of the formulation of the problem. Further important improvements have been contributed to the visual descriptors (e.g. by unsupervised and partially-supervised metric learning) [34, 64], and to obtaining an episode wide consistent labelling [151] (by using a graph formulation and other visual cues).

2.3.2 Transfer learning

Work on the domain adaptation problem of learning from photos and retrieving paintings has come into being in recent years. Shrivastava *et al.* [130] use an Exemplar SVM [96] to retrieve paintings of specific buildings. Aubry *et al.* [14] improve on this by utilising mid-level discriminative patches, the patches in question demonstrating remarkable invariance between photos and paintings. Subsequently, Crowley *et al.* [40] showed that this patch-based method can be extended to object categories in paintings beyond the instance matching of [14]. Others [166,167] have considered the wider problem of generalising across many visual styles (e.g. photo, cartoon, painting) by building a depiction invariant graph model.

Recently Crowley *et al.* [39] showed that classifiers using CNN features learnt on photos are able to accurately recognise object categories in paintings. We examine here whether this

can be extended to facial identities.

2.3.3 On the fly retrieval

Computer vision researchers saw the potential of image search engines as soon as they were introduced [21, 56, 57, 91, 93, 122]. Early papers were concerned with improving the quality of the returned images, for example by reranking based on visual consistency to promote the target class. However, due to click-through crowd sourcing the quality of the images is now extremely high over a vast variation of queries, to the extent that for most queries the first 100 or so top ranking images are for the most part free of non-class images. The problem of visual polysemy still remains [122], (e.g. "Jaguar" the car versus "jaguar" the animal), but to an extent this can be avoided by more specific search queries ("jaguar car" or "jaguar cat") or by employing the clusters automatically provided by the search engines.

Learning on-the-fly from a reservoir of annotated images (the web or proprietary datasets) has been investigated by a number of groups, including [11, 22, 29, 30, 58, 94, 139, 154]. Such learning is in contrast to the more conventional approach of using hand-curated collections of positive and negative training images, such as PASCAL VOC [49] or ImageNet [45], where the set of categories is preset. On-the-fly learning offers a way to overcome the 'closed world' problem in computer vision, where object category recognition systems are restricted to only these pre-defined categories. There are applications to searching video archives, such as those of the BBC, and to searching personal image and video collections [85, 86], since both archives and personal collections have only sparse textual annotations at best. Specific to face retrieval, [23] presented local hierarchical projection scheme for efficient face retrieval.

2.4 Large Scale Datasets

In this section we look at efforts of the community to build large scale datasets for generic object categorisation and face recognition. Collecting annotated datasets for the task of object classification has been in trend for while. Griffin *et al.* [62] presented the Caltech-256 dataset catered for object categorisation. The PASCAL VOC dataset [49] set an extraordinary benchmark for

dataset collection and benchmarking of different algorithms on it. The dataset consists of images belonging to 20 object categories and defines challenges in image classification, object detection and image segmentation tasks. Very recently, the ImageNet challenge dataset [117] has been carrying forward the task for similar challenges on a large scale. The dataset consists of 1 million images and is annotated with $1,000$ different categories. The challenges run in a similar spirit to that of PASCAL VOC. Besides these generic image classification datasets, researchers have presented several datasets for specific applications. Designed to bring out subtle difference within typically different species of a single object category, they are typically referred to as the "Fine-Grained" datasets. Nielsback *et al.* [102] presented such a dataset for flower classification, researchers at Caltech and UCSD presented a popular dataset for this fine-grained classification of bird species [159]. There have been many such contributions and reviewing them all will be well beyond our scope here.

On the the facial front, there has been constant stream of datasets as well. Phillips *et al.* [114] proposed the "FERET" dataset and evaluation challenge. The dataset consists of $14,126$ images of $1,199$ identities. This dataset inspired some of the early works in facial recognition technology [175]. Some of the shortcomings of this dataset were monochrome images and the rather controlled environment in which the pictures were taken. Overcoming these, the "Labelled Faces in the Wild" (LFW) dataset was introduced as a benchmark for face verification tasks. This dataset originated from the 'Names and Faces in the News' work of Berg *et al.* [20] and consists of color images of celebrities taken in uncontrolled settings. We discuss this dataset in detail in Sect. 3.4.1. Kumar *et al.* [83] collected the "FaceTracer " dataset of face images from the web and manually annotated them with facial landmarks and attributes such as facial hair, age, race, facial expressions etc.In addition, there are pose parameters (yaw, pitch and roll) associated with each face. This dataset contains $15,000$ faces in total. The authors also presented a method to search through the collection of images based on the attributes using classifiers learnt on this data. In Sect. 7.6 we show that such a system can be developed to run on-the-fly without needing to train classifiers beforehand. The only downside is that this dataset was released as URLs due to copyright issues which meant that

the amount of images available for download decreased over time as the links expired. The authors published another dataset (PubFig [84]) of celebrity images. The dataset consists of $58,797$ images of 200 celebrities collected in a similar manner to that of the FaceTracer. This again was released as URLs and hence is almost extinct now.

The LFW dataset is the de-facto standard for evaluating face verification in images. The YouTube Faces in the Wild dataset introduced by Wolf *et al.* [162] fills the same space in the video face verification domain. Similar to the LFW dataset, the majority of the work presented in this thesis is evaluated on this dataset. We discuss this dataset in detail in Sect. 3.4.2. There has been some work very recently on introducing new datasets for these tasks as the performances on these datasets is reaching its saturation point. IARPA introduced a new dataset under its JANUS program to overcome the shortcomings of the YTF dataset, while the researchers at the University of Washington introduced a dataset of a million distractor images to make verification tasks on the LFW dataset more difficult. On the topic of face tracking and labeling in videos, Everingham *et al.* [48] presented a dataset of labelled face tracks from popular television series "Buffy the vampire slayer". Cinbis *et al.* [34] also presented a dataset for face verification tasks on the same series. For the task of labelling characters in the movies, Bojanowaski *et al.* [24] made labelled face tracks from the movie "Casablanca" available. All of these video datasets are labelled manually after faces are tracked automatically. Recently, Ozerov *et al.* [105] presented a manually labelled dataset of face tracks in a movie. This dataset consists of labelled faces, tracks and identity labels from the popular movie "Hannah and her sisters". This dataset provides a unique opportunity to evaluate the performance of tracking methods as well as labelling methods with varying degree of tracking supervision.

Chapter 3: Shallow features for face recognition

In this chapter we investigate use of the Fisher vector features for representing faces in images and videos. To this end, this chapter makes two contributions: first, and somewhat surprisingly, we show that the Fisher vectors computed on densely sampled SIFT features, *i.e.* an off-the-shelf object recognition representation, are suitable for representing faces in images and videos showing competitive performance on several standard benchmark datasets. Second, since Fisher Vectors are very high dimensional, we show that a compact representation can be learnt from them using discriminative dimensionality reduction. This compact representation has a better recognition accuracy and is very well suited to large scale identification tasks. The organization of this chapter is as follows, first we describe local features and their encoding using Fisher Vectors for faces in images and videos. We then look at discriminative dimensionality reduction techniques to obtain low dimensional projections of these high dimensional descriptors. We also test extreme compression techniques to obtain a very low dimensional yet very functional representation. Finally, we present the performance of these features on several public benchmark datasets.

3.1 Dense features and Fisher vector construction

Fisher Vectors (FV) encode first and second order statistics of several local features computed densely on a given image. The typical extraction workflow consists of dense feature computation followed by an encoding step based on a precomputed vocabulary (Gaussian Mixture Models) of these dense features. Speed of encoding can be improved using an approximation of the formulation which is helpful in the task of encoding faces in videos having several hun-

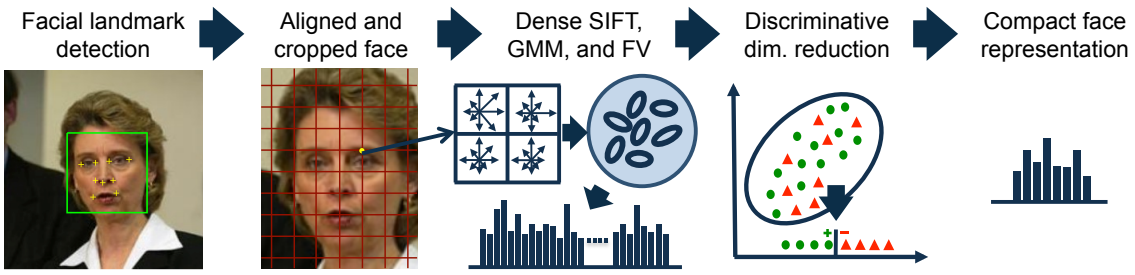


Figure 3.1: Method overview: a face is encoded in a discriminative compact representation

dred frames. Data augmentation for additional invariances and performance improvements can be done either at the encoding stage or can be performed as a redundant operation producing one feature per augmentation. Each of these aspects are discussed at length in the following sections.

3.1.1 Dense features

The FV construction starts by extracting patch features such as SIFT [95] from the image. The traditional approach for describing faces has been to extract appearance features from few sampled locations. (*e.g.* facial landmarks). Rather than sampling locations and scales sparsely by running a carefully tuned face landmark detector, our approach extracts features densely in scale and space. Specifically, 24×24 pixels patches are sampled with a stride of one pixel and for each patch the root-SIFT representation of Arandjelovic *et al.* [12] (referred simply as "SIFT" in the following) is computed. The process is repeated at five scales, with a scaling factors of $\sqrt{2}$. The procedure results in about 25K 128-dimensional descriptors per face.

3.1.2 Spatial information

Although the Fisher Vector is an effective encoding of the feature space structure, it does not capture the distribution of features in the spatial domain. Several ways of incorporating the spatial information have been proposed in the literature. Perronnin *et al.* [112], presented a spatial pyramid coding [87], which consists of dividing an image into a number of cells and then stacking the FVs computed for each of these cells. The disadvantage of such an approach is that the dimensionality of the final image descriptor increases linearly with the number of

cells. Krapac *et al.* [79] proposed a generative model (*e.g.*GMM) for the spatial location of each visual word, and FV encoding was used to encode both feature appearance and location. In this chapter, we employ a approach similar to that of Sanchez *et al.* [120], which consists of augmenting the visual features with their spatial coordinates, and then using the FV encoding of the augmented features as the image descriptor. In more detail, our dense features have the following form: $\left[\mathcal{S}_{xy}; \frac{x}{w} - \frac{1}{2}; \frac{y}{h} - \frac{1}{2}\right]$, where $\mathcal{S}_{xy}$ is the (SIFT) descriptor of a patch centred at (x, y), and w and h are the width and height of the face image. Sect. 3.5.3 and Fig. 3.4 and 3.5 illustrate how Gaussian mixture components are spatially distributed over a face when learnt for a face verification task.

3.1.3 Fisher vectors

The FV encoding aggregates a large set of vectors (*e.g.* the dense SIFT features just extracted) into a high-dimensional vector representation. In general, this is done by fitting a parametric generative model, *e.g.* the Gaussian Mixture Model (GMM), to the features and then encoding the derivatives of the log-likelihood of the model with respect to its parameters [70]. Following [112], we train a GMM with diagonal covariances, and only consider the derivatives with respect to the Gaussian means and variances. This leads to the representation which captures the average first and second order differences between the (dense) features and each of the GMM centers:

$$\Phi_k^{(1)} = \frac{1}{N\sqrt{w_k}} \sum_{p=1}^{N} \alpha_k(\mathbf{x}_p) \left(\frac{x_p - \mu_k}{\sigma_k} \right), \quad \Phi_k^{(2)} = \frac{1}{N\sqrt{2w_k}} \sum_{p=1}^{N} \alpha_k(\mathbf{x}_p) \left(\frac{(x_p - \mu_k)^2}{\sigma_k^2} - 1 \right) \tag{3.1}$$

Here, $\{w_k, \mu_k, \sigma_k\}_k$ are the mixture weights, means, and diagonal covariances of the GMM, which is computed on the training set and used for the description of all face images; $\alpha_k(\mathbf{x}_p)$ is the soft assignment weight of the p-th feature x_p to the k-th Gaussian. An FV ϕ is obtained by stacking the differences: $\phi = \left[\Phi_1^{(1)}, \Phi_1^{(2)}, \ldots, \Phi_K^{(1)}, \Phi_K^{(2)}\right]$. The encoding describes how the

distribution of features of a particular image differs from the distribution fitted to the features of all training images. To make the dense patch features amenable to the FV description based on the diagonal-covariance GMM, they are first decorrelated by the PCA. In our experiments, we applied PCA to SIFT features, reducing their dimensionality from 128 to 64. The FV dimensionality is $2Kd$, where K is the number of Gaussians in the GMM, and d is the dimensionality of the patch feature vector. We note that even though FV dimensionality is high (65536 for $K = 512$ and $d = 64$), it is still significantly lower than the dimensionality of the vector obtained by stacking all dense features (1.7M in our case). Following [112], the performance of a FV is further improved by passing it through signed square-root and L_2 normalization.

3.1.4 Efficient computation using hard-assignment Fisher Vectors

In the traditional FV formulation of Eqn. 3.1 the soft assignment coefficients $\alpha_k(\mathbf{x}_p)$ assign local features $\mathbf{x}_p$ to all K components of the GMM. When the number of Gaussians is large, averaging the statistics over N features and K Gaussian components becomes a computational bottleneck. This problem is exacerbated in dealing with multiple frames forming a face track. To accelerate this computation we employ an FV encoding variant, called hard-FV [133], that replaces the soft assignment of features to GMM components with the hard assignment to a single Gaussian, which corresponds to the largest likelihood. Typically, the hard assignment delivers a speed up in computation by a factor of six, at the cost of a small (1%) drop in performance.

3.1.5 FV encoding of face tracks

Another aspect to discuss is how to merge information across multiple frames while computing a descriptor for a face in a video track. We experimented with two alternatives: (i) *image pooling* – computing Fisher Vectors for each of the face track frames individually, followed by averaging the Fisher Vectors across frames; and (ii) *video pooling* – computing a single Fisher Vector over the whole face track by pooling together SIFT features from all the faces in a track. This approach helps in significantly reducing the memory footprint of the descriptor by yielding

one descriptor per track instead of several hundred individual descriptors per frame in the track. As shown in Sect. 3.3, video pooling outperforms image pooling. It should be noted that the two methods are not equivalent, since in the image pooling method each frame FV is individually normalized prior to averaging, while in the video pooling the features from the whole track are combined together and normalized only once. Fig. 3.2 shows the pictorial representation of our approach. More approaches for representing video face tracks are described in Chapter 7.

3.1.6 Data augmentation

Data jittering, also known as virtual sampling, is a common technique to improve the invariance of learnt descriptors. The idea is to enrich the training set with transformed variants of the data to simulate distortions that are expected to occur at test time (*e.g.* small face rotations).We evaluate two strategies for the data augmentation, the first one produces one feature per augmentation which is used at the test time (test time flip). While the second one takes inspiration from the previous section of pooling features together and produces a single feature vector despite the augmentations (jittered pooling).

Test time flip. Following the work of Huang *et al.* [67], we considered the augmentation of the feature set by taking the horizontal reflections of the face images. Since this increases the number of features, we primarily use this during the test time only.

Jittered pooling. Rather than increasing the size of the training set like in the case of the test time flip, here we use the simple idea of extending video pooling of Sect. 3.1.5 to include jittered version of the data. This is computationally quite cheap, results in descriptors of the same dimensionality and indeed achieves the best results in our tests (Sect. 3.5.2). In the current implementation we considered only the horizontal flips of individual images as jitters to be added to the feature, but it is trivial to extend this to further variations such as rotations and crops. Note that this efficient augmentation is applied at both training and test time. Fig. 3.2 shows the depiction of the scheme.

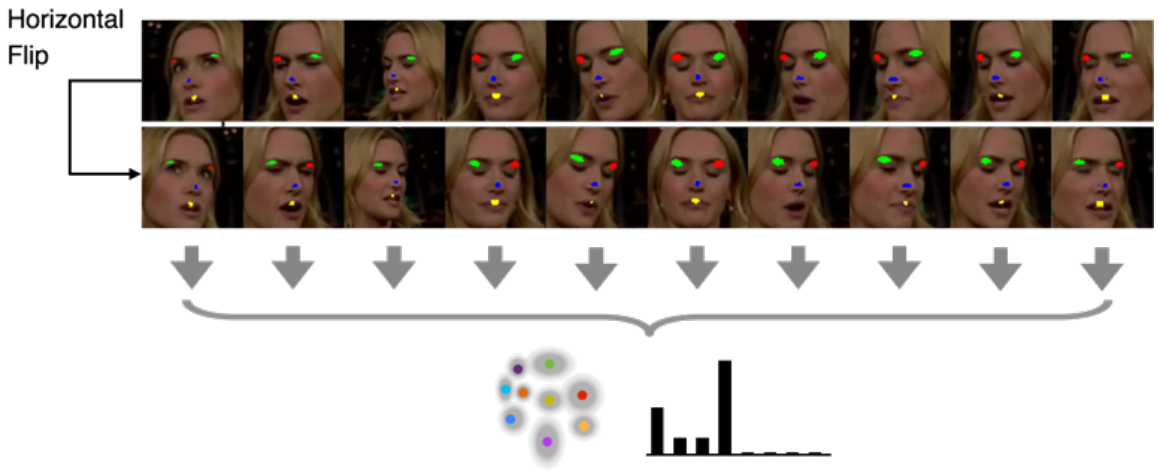


Figure 3.2: Video pooling: Local features from all frames of a face track along with their augmentations are pooled together into a single descriptor.

3.2 Large-margin dimensionality reduction

In this section we explain how a high-dimensional FV encoding (Sect. 3.1.3) is compressed to a small discriminative representation. The compression is carried out using a linear projection, which serves two purposes: (i) it dramatically reduces the dimensionality of the face descriptors, making them applicable to large-scale datasets; and (ii) it improves the recognition performance by projection onto a subspace with a discriminative Euclidean distance.

3.2.1 Discriminative dimensionality reduction

The aim here is to learn a linear projection $W \in \mathbb{R}^{p \times d}$, $p \ll d$, which projects high-dimensional Fisher Vectors $\phi \in \mathbb{R}^d$ to low-dimensional representation $W\phi \in \mathbb{R}^p$, such that the squared Euclidean distance $d^2_W(\phi_i, \phi_j) = \|W\phi_i - W\phi_j\|^2_2$ between images i and j is smaller than a learnt threshold $b \in \mathbb{R}$ if i and j are the same person, and larger otherwise. We further impose that these conditions are satisfied with a margin of at least one, resulting in the constraints:

$$y_{ij}\left(b - d^2_W(\phi_i, \phi_j)\right) > 1 \tag{3.2}$$

where $y_{ij} = 1$ iff images i and j contain the faces of the same person, and $y_{ij} = -1$ otherwise.

Note that the Euclidean distance in the p-dimensional projected space can be seen as a

low-rank Mahalanobis metric in the original d-dimensional space:

$$d_W^2(\phi_i, \phi_j) = \|W\phi_i - W\phi_j\|_2^2 = (\phi_i - \phi_j)^T W^T W(\phi_i - \phi_j), \tag{3.3}$$

where $W^T W \in \mathbb{R}^{d \times d}$ is the Mahalanobis matrix defining the metric. Due to the factorisation, the Mahalanobis matrix $W^T W$ has rank equal to p, *i.e.* much smaller than the full rank d. As a consequence, learning the projection matrix W is the same as learning a low-rank metric $W^T W$. Direct optimisation of the Mahalanobis matrix is however quite difficult, as the latter has over 2 billion parameters for the $d = 67$K dimensional FVs. On the contrary, W has $p \times d = 8.5$M parameters for $p = 128$, which can be learnt in the large scale learning scenario.

Learning W optimises the following objective function, incorporating the constraints (Eqn. 3.2) in a hinge-loss formulation:

$$\arg\min_{W,b} \sum_{i,j} \max\left[1 - y_{ij}\left(b - (\phi_i - \phi_j)^T W^T W(\phi_i - \phi_j)\right), 0\right] \tag{3.4}$$

The minimiser of $Eqn.\ 3.4$ is found using a stochastic sub-gradient method. At each iteration t, the algorithm samples a single pair of face images (i, j) (sampling with equal frequency positive and negative labels y_{ij}) and performs the following update of the projection matrix:

$$W_{t+1} = \begin{cases} W_t & \text{if } y_{ij}\left(b - d_W^2(\phi_i, \phi_j)\right) > 1 \\ W_t - \gamma y_{ij} W_t \psi_{ij} & \text{otherwise} \end{cases} \tag{3.5}$$

where $\psi_{ij} = (\phi_i - \phi_j)(\phi_i - \phi_j)^T$ is the outer product of the difference vectors, and γ is a constant learning rate, determined on the validation set. Note that the projection matrix W_t is left unchanged if the constraint (Eqn. 3.2) is not violated, which speed-ups learning (due to the large size of W, performing matrix operations at each iteration is costly). We choose not to regularize W explicitly; rather, the algorithm stops after a fixed number of learning iterations (1M in our case).

Finally, note that the objective in $Eqn.\ 3.4$ is not convex in W, so initialization is important. In practice, we initialize W to extract the p largest PCA dimensions of the training

FVs. Furthermore, differently from standard PCA, we equalize the magnitude of the dominant eigenvalues (whitening) as the less frequent modes of variation tend to be amongst the most discriminative. It is important to note that PCA-whitening is only used to *initialize* the learning process, and the learnt metric substantially improves over its initialization (Sect. 3.5.1). In particular, this is *not* the same as learning a metric on the low-dimensional PCA-whitened data (p^2 parameters); instead, a projection W on the *original* descriptors is learnt ($pd \gg p^2$ parameters), which allows us to fully exploit the available supervision.

Learning the metric threshold. The objective function for learning the projection W and the threshold (bias) b is presented in Eqn. 3.4. It is optimized using the stochastic sub-gradient method, in which at each iteration we update both W and b. The update equation for the projection is given in Eqn. 3.5. Similarly, the update equation for the bias takes the following form:

$$b_{t+1} = \begin{cases} b_t & \text{if } y_{ij}\left(b - d^2_W(\phi_i, \phi_j)\right) > 1 \\ b_t + \gamma y_{ij} b_t & \text{otherwise} \end{cases} \tag{3.6}$$

We found that the verification accuracy can be further improved after learning is finished by re-estimating the bias on the validation set (a held-out part of the training set). This is done by an exhaustive search over the values of the learnt distance $d^2_W(\phi_i, \phi_j)$ between the validation pairs, and setting b to the value which leads to the highest verification accuracy on the validation set.

3.2.2 Joint metric-similarity learning

Recently, a "joint Bayesian" approach to face similarity learning has been employed in [31, 32]. It effectively corresponds to joint learning of a low-rank Mahalanobis distance $(\phi_i - \phi_j)^T W^T W(\phi_i - \phi_j)$ *and* a low-rank kernel (inner product) $\phi_i^T V^T V \phi_j$ between face descriptors ϕ_i, ϕ_j. Then, the difference between the distance and the inner product can be used as a score function for face verification. We consider it as another option for comparing face descriptors (apart from the low-rank metric learning and diagonal metric learning), and incorporate joint

metric-similarity learning into our large-margin learning formulation (Eqn. 3.4). In this case, we perform stochastic updates of Eqn. 3.5 on both low-dimensional projections W and V.

3.2.3 Binary compression

While the low-rank metric learning method described in the previous sections is already capable of achieving a very good compression factor, for large scale applications the goal is to further decrease the number of bits required to encode each face track. This section describes a method to map a low-dimensional real valued descriptor $\psi \in \mathbb{R}^m$ to a binary code $\beta \in \{0,1\}^q$ with the bit length q (where $q \geq m$). While there are several alternative hashing methods that could be used for this purpose, Jegou *et al.* [74] suggested a very competitive and efficient method based on computing a Parseval tight frame, followed by thresholding. Recently, this binarization method has been successfully applied by [134] in the local feature descriptor domain.

In more detail, a *frame* is a matrix $U \in \mathbb{R}^{q \times m}$ whose row-span is the space of vectors $\psi \in \mathbb{R}^m = \operatorname{span} U^\top$ to be encoded. A *tight frame* has the additional property that $U^\top U = I$. Multiplying the vector ψ by the tight frame produces an over-complete representation $U\psi \in \mathbb{R}^q$; while the latter has an increased dimensionality, it "spreads the information" among many redundant dimensions. The result is that thresholding this vector by the sign function (defined as $\operatorname{sign}(a) = 1$ iff $a > 0$ and 0 otherwise) yields a binary vector

$$\beta = \operatorname{sign}(U\psi) \tag{3.7}$$

that is an accurate representation (in terms of the Euclidean distances) of the vector ψ. In practice, as suggested by [74], U is computed by keeping the first m columns of an orthogonal matrix obtained from a QR-decomposition of a random $q \times q$ matrix. For thresholding the $sign$ function to produce a meaningful binary string, the vector ψ needs to be zero centered. This is achieved by subtracting a mean vector computed over a large number of feature vectors.

Note that, while the dimensionality q of the binary code is not smaller than the dimensionality of the data m, the representation is significantly more compact provided that $q/m \ll 32$;

since encoding each binary dimension requires a single bit, whereas encoding a floating point number usually requires 32 or 64 bits. In this manner, changing q allows us to generate the binary descriptors with any desired bitrate. Additionally, the Euclidean distance between the binary vectors reduces to the Hamming distance which can be computed very quickly using the XOR and POPCNT (population count) instructions in recent CPUs.

3.3 Implementation details and extensions

In this section we describe some details of the feature computation and learning algorithms.

Face alignment and extraction. Images in the LFW dataset Sect. 3.4.1 are preprocessed in the following manner. Given an image, we first run the Viola Jones detector [158] to obtain the face detection. Using this detection, we then detect nine facial landmark positions using the publicly available code of [48]. Similar to them, we then apply similarity transformation using all these points to transform a face to a canonical frame. In the aligned image, we extract a 160×125 face region around the landmarks for further processing. For every other dataset used for the evaluation, we use the given detection as is for computing the descriptor.

Face descriptor computation. For dense SIFT computation and Fisher Vector encoding, we utilized publicly available packages [27, 156]. Dimensionality reduction learning is implemented in MATLAB and takes a few hours to compute on a single core (for each split). Given an aligned and cropped face image, our mexified MATLAB implementation takes 0.6s to compute a descriptor on a single CPU core (in the case of 2 pixel SIFT density). While computing FVs for a video, in order to reduce memory consumption, we incrementally encode SIFTs from each frame of the track into a FV but do not perform the normalization. Normalization is only performed at the end when all frames are encoded. This is possible due to the additive construction of FVs (Eqn. 3.1) and saves us from having to store SIFTs for all frames in a track into memory for pooling.

Training pairs sampling strategy. For the discriminative dimensionality reduction, training pairs were formed based on the person identities. Considering that different people have different number of occurrences,we sampled the pairs uniformly with respect to the identity (names). Namely, to construct a negative pair, we uniformly sampled two identities, and then uniformly sampled an image/video of each of them. To construct a positive pair, we uniformly sampled a single identity (which has at least two images/videos) and then we uniformly sampled two instances of the person.

Diagonal "metric" learning. Apart from the low-rank Mahalanobis metric learning (Sect. 3.2), we also consider diagonal metric learning on the full-dimensional Fisher Vectors. It is carried out using a conventional linear SVM formulation, where features are the vectors of squared differences between the corresponding components of the two compared FVs. We did not observe any improvement by enforcing the positivity of the learnt weights, so it was omitted in practice (*i.e.* the learnt function is not strictly a metric).

3.4 Dataset and evaluation protocol

3.4.1 Labeled Faces in the Wild dataset

Our framework is evaluated on the popular "Labeled Faces in the Wild dataset" (LFW) [68]. The dataset contains 13,233 images of 5,749 people downloaded from the Web and is considered the *de-facto* standard benchmark for automatic face verification. For evaluation, the data is divided into 10 disjoint splits, which contain different identities and come with a list of 600 pre-defined image pairs for evaluation (as well as training as explained below). Of these, 300 are "positive" pairs portraying the same person and the remaining 300 are "negative" pairs portraying different people.

We follow the recommended evaluation procedure [68] and measure the performance of our method by performing a 10 fold cross validation, training the model on 9 splits, and testing it on the remaining split. All aspects of our method that involve learning, including PCA projections for SIFT, Gaussian mixture models, and the discriminative Fisher Vector projections, were

trained independently for each fold.

Two evaluation measures are considered. The first one is the *Receiver Operating Characteristic-Equal Error Rate* (ROC-EER), which is the accuracy at the ROC operating point where the false positive and false negative rates are equal [63]. This measure reflects the quality of the *ranking* obtained by scoring image pairs and, as such, is independent on the bias learnt in Eqn. 3.2. ROC-EER measure is used to compare the different stages of the proposed framework. In order to allow a direct comparison with published results, however, our final classification performance is also reported in terms of the classification accuracy (percentage of image pairs correctly classified) – in this case the bias is important.

LFW specifies a number of evaluation protocols, two of which are considered here. In the "restricted setting", only the pre-defined image pairs for each of the splits (fixed by the LFW creators) can be used for training. Instead, in the "unrestricted setting" one is given the identities of the people within each split and is allowed to form an arbitrary number, in practice much larger, of positive and negative pairs for training.

3.4.2 YouTube Faces Dataset

The YouTube Faces dataset [162] is a popular benchmark for face verification in video data captured in uncontrolled conditions. It contains 3,425 videos of 1,595 celebrities collected from YouTube, with an average of 2 videos per identity, and the average clip length of 181 frames. The dataset includes pre-extracted face tracks, the alignment of faces to a canonical frame (normalization), as well as a specification of 6,000 track pairs, divided in 10 splits of 600 pairs. Each of the 10 splits contains 300 positive pairs (same identity) and 300 negative pairs. Similar to the LFW dataset, two evaluation settings are defined. In the *restricted setting*, 9 splits (5,400 pairs) are used for training and the remaining split – for testing. The final performance measure is obtained by averaging the results over the 10 folds. In the *unrestricted setting*, one is allowed to combine the training tracks in an arbitrary number of training pairs. Face verification results are reported using three metrics: Area under the Receiver Operating Characteristic curve (AUC), *Receiver Operating Characteristic Equal Error Rate* (ROC-EER,

or simply EER) and verification accuracy [162].

3.4.3 INRIA-Buffy dataset

This dataset consists of 639 face tracks, automatically extracted from episodes 9, 21, and 45 of the television series "Buffy the Vampire Slayer" [34]. Tracks are assigned one of nine labels: eight for the main characters in the series and the ninth denoting "others". The training set consists of 312 tracks, and the test set of 327 tracks. Note that there are no track-level face identity labels available in the training set. Therefore we will use this dataset mainly to evaluate transferring representations learned on YTF.

3.5 Experiments

This section evaluates the performance of the proposed methods on still images first, followed by an extensive evaluation on video datasets.

3.5.1 Image verification performance

We evaluate the performance of our descriptor on very a popular image verification benchmark dataset LFW dataset described in Sect. 3.4.1.

3.5.1.1 Framework parameters

First, we explore how the different parameters of the feature representation and metric learning affect its performance. The experiments were carried out in the unrestricted setting starting from unaligned LFW images and aligned using a simple alignment procedure described in Sect. 3.3. We explore the following settings: SIFT density (the step between the centers of two consecutive descriptors), the number of Gaussians in the GMM, the effect of spatial augmentation, dimensionality reduction, distance function and horizontal flipping. The results of the comparison are given in Table 3.1. As can be seen, the performance increases with denser sampling and more clusters in the GMM. Spatial augmentation boosts the performance with only a moderate increase in dimensionality (caused by the addition of the (x, y) coordinates

to 64-D PCA- SIFT). Our dimensionality reduction to 128-D achieves 528-fold compression and further improves the performance. We found that using projection to higher-dimensional spaces (e.g. 256-D) does not improve the performance, which can be caused due to overfitting.

As far as the choice of the FV distance function is concerned, a low-rank Mahalanobis metric outperforms both the full-rank diagonal metric and unsupervised PCA-whitening, but is somewhat worse than the function obtained by the joint large-margin learning of the Mahalanobis metric and inner product. It should be noted that the latter comes at the cost of slower learning and the necessity to keep two projection matrices instead of one. Finally, using horizontal flipping consistently improves the performance. In terms of the ROC-EER measure, our best result is 93.13%.

SIFT density	GMM Size	Spatial Aug.	Desc. Dim.	Distance Function	Hor. Flip.	100%-EER
2 pix	256		32768	diag. metric		89.0
2 pix	256	✓	33792	diag. metric		89.8
2 pix	512	✓	67584	diag. metric		90.6
1 pix	512	✓	67584	diag. metric		90.9
1 pix	512	✓	128	low-rank PCA-whitening		78.6
1 pix	512	✓	128	low-rank Mah. metric		91.4
1 pix	512	✓	256	low-rank Mah. metric		91.0
1 pix	512	✓	128	low-rank Mah. metric	✓	92.0
1 pix	512	✓	2×128	low-rank joint metric-sim.		92.2
1 pix	512	✓	2×128	low-rank joint metric-sim.	✓	93.1

Table 3.1: Framework parameters*: The effect of different FV computation parameters and distance functions on ROC-EER. All experiments done in the unrestricted setting.*

3.5.1.2 LFW dataset: comparison with the state of the art

Unrestricted setting. In this scenario, we compare against the best published results obtained using both single (Table 3.2, left-bottom) and multi-descriptor representations (Table 3.2, left-top). Similar to the previous section, the experiments were carried out using unaligned LFW images, processed as described in Sect. 3.3. This means that the outside training data is only utilized in the form of a simple landmark detector, trained by [48].

Our method achieves 93.03% face verification accuracy, closely matching to that of [32],

Unrestricted setting

Method	Mean Acc.
LDML-MkNN [63]	0.8750 ± 0.0040
Combined multishot [148]	0.8950 ± 0.0051
Combined PLDA [92]	0.9007 ± 0.0051
face.com [147]	0.9130 ± 0.0030
CMD + SLBP [67]	0.9258 ± 0.0136
LBP multishot [148]	0.8517 ± 0.0061
LBP PLDA [92]	0.8733 ± 0.0055
SLBP [67]	0.9000 ± 0.0133
CMD [67]	0.9170 ± 0.0110
High-dim SIFT [32]	$0.9177 \pm$ N/A
High-dim LBP [32]	0.9318 ± 0.0107
Our Method	$\mathbf{0.9303 \pm 0.0105}$

Restricted setting

Method	Mean Acc.
V1-like/MKL [116]	0.7935 ± 0.0055
PEM SIFT [89]	0.8138 ± 0.0098
APEM Fusion [89]	0.8408 ± 0.0120
Our Method	$\mathbf{0.8747 \pm 0.0149}$

Table 3.2: Left: Face verification accuracy in the unrestricted setting. *Using a single type of local features (dense SIFT), our method outperforms a number of methods, based on multiple feature types, and closely matches results of [32].* **Right: Face verification accuracy in the restricted setting (no outside training data)**. *Our method is one of the top performing methods on this dataset setting.*

which achieves 93.18% using LBP features sampled around 27 landmarks. It should be noted that (i) the best result of [32] using SIFT descriptors is 91.77%; (ii) we do not rely on multiple landmark detection, but sample the features densely. The ROC curves of our method as well as the other methods are shown in Fig. 3.3.

Restricted setting. In the restricted setting, no outside training data is used, even for the landmark detection. Following [89], we used centered 150×150 crops of "LFW-funneled" images, provided as a part of the LFW dataset. We found that the limited amount of training data, available in this setting, is insufficient for dimensionality reduction learning. Therefore, we learnt a diagonal "metric" function using an SVM as described in Sect. 3.3. Achieving the verification accuracy of 87.47% , our descriptor achieves comparable to the state of the art performance in the restricted setting (Table 3.2, right), outperforming the recently published result of [89] by 3.4%. It should be noted that while [89] also use GMMs for dense feature clustering, they do not utilize the compressed Fisher Vector encoding, but keep all extracted features for matching, which imposes a limitation on the number of features that can be extracted and stored. In our

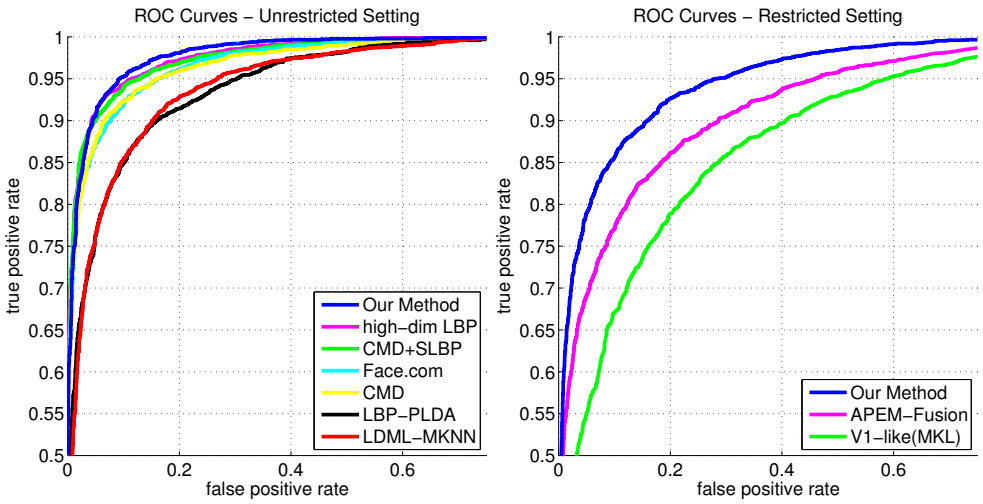


Figure 3.3: Comparison with the state of the art: *ROC curves of our method and the state-of-the-art techniques in LFW-unrestricted (left) and LFW-restricted (right) settings.*

case, we are free from this limitation, since the dimensionality of a FV does not depend on the number of features it encodes. The best result of [89] was obtained using two types of features and GMM adaptation ("APEM Fusion"). When using non-adapted GMMs (as we do) and SIFT descriptors ("PEM SIFT"), their result is 6% worse than ours.

Our results in both unrestricted and restricted settings confirm that the proposed face descriptor can be used in both small-scale and large-scale learning scenarios and is robust with respect to the face alignment and cropping techniques.

3.5.2 Video verification performance

We show the effectiveness of the descriptor for describing faces in a video track by evaluating performance of our descriptor on two public datasets described before, Youtube Faces In the Wild (Sect. 3.4.2) and INRIA Buffy dataset (Sect. 3.4.3).

3.5.2.1 YouTube Faces dataset: comparison with the state of the art

Face track representation is computed on the aligned face images, provided by the YTF organizers. From these images, we extract a 150×150 patch around the center of the face, and use it for feature computation. The resulting descriptor is compressed to a low-dimensional vector using discriminative dimensionality reduction (Sect. 3.2), followed by optional binarization

(Sect. 3.2.3).

In Table 3.3, we evaluate different settings of our discriminatively projected VF^2 in terms of the EER performance measure. The experiments are performed in the unrestricted setting. First, we compare FV pooling techniques (Sect. 3.1.5): image pooling (row 1) and video pooling (row 2). As can be seen, the latter performs better, which indicates that it is beneficial to aggregate the visual statistics across the whole face track prior to normalization. Soft quantized Fisher vectors improve the results by 1.2% (row 3) but increases the processing time up to 6 times. Jittered pooling (row 4) yields a better improvement at a smaller cost by pooling more features together. Next, we asses how the error rate changes depending on the projected FV dimension (rows 5–7) and conclude that, on this benchmark, there is no improvement beyond dimension 128 due to model overfitting. In rows 8 and 9 we report the verification performance of the 128-D projected FV descriptor (row 2), binarized to 1024 and 2048 bits respectively using the method of Sect. 3.2.3. As a result of binarization, not only is the face descriptor memory footprint decreased from 512 bytes (128×4-byte single-precision values) to 128 bytes (1024 bits), but also the EER decreases from 16.2% (row 2) to 15.0% (row 8) due to the binary representation being more robust. A similar improvement is observed for the jittered pooling variant: 14.2% (row 4) to 13.4% (row 10).

Finally, rows 11–13 show the results of the joint similarity-distance learning formulation [31, 131] (Sect. 3.2), reducing the error rate to 14.4% compared to learning only a distance metric (rows 2 and 11) and to 12.3% for the jittered pooled descriptor (rows 4 and 13). Although using test time flips improves results (row 12), dataset augmentation using jittered pooling achieves the best results of 12.3% (row 13).

The comparison with the current state-of-the-art methods is given in Table 3.4. In the restricted setting, similar to the LFW case explained previously, we could not train a full projection matrix W due to the small number of training pairs available in this case, leading to overfitting. Instead, we used diagonal metric learning (Sect. 3.3). This combination outperforms the previous best results by $\sim 4\%$ and achieves an EER of 16.1% (row 5). This is improved further to 14.9% with test-time flips (row 6).

In the large-scale unrestricted setting (Table 3.4, rows 7-8), we can fully leverage the available amount of annotation and learn a full projection matrix W. To the best of our knowledge, we are the first to report the performance in the unrestricted setting. Compared to the restricted case, the EER in the unrestricted scenario is smaller by 2.6% (12.3% vs 14.9%).

Setting	descriptor variant	Proj. Dim.	100% - EER
1	image pool.	128	82.7
2	video pool.	128	83.8
3	video pool. + soft quantization	128	85.0
4	video pool. + jittered pool.	128	85.8
5	video pool.	256	83.1
6	video pool.	512	83.0
7	video pool.	1024	83.0
8	video pool. + binar. 1024 bit	128	85.0
9	video pool. + binar. 2048 bit	128	85.0
10	video pool. + binar. 1024 bit + jitt. pool.	128	86.6
11	video pool. + joint sim.	128×2	85.6
12	video pool. + joint sim. + flip	128×2	87.0
13	video pool. + joint sim. + jitt. pool.	128×2	**87.7**

Table 3.3: The performance of different descriptor settings on YouTube face verification (unrestricted setting). *Our joint similarity metric learning approach achieves the best performance Notice that the binary version of the descriptor also achieves comparable performance despite its compact size.*

	Method	Accuracy	AUC	100% - EER
1	MGBS & SVM- [165]	78.9 ± 1.9	86.9	78.8
2	APEM FUSION [89]	79.1 ± 1.5	86.6	78.6
3	STFRD & PMML [42]	79.5 ± 2.5	88.6	80.1
4	VSOF & OSS (Adaboost) [100]	79.7 ± 1.8	89.4	80.0
5	Ours (restricted)	83.5 ± 2.3	92.0	83.9
6	Ours (restricted & flip)	84.7 ± 1.4	93.0	85.1
7	Ours (unrestricted & flip)	83.5 ± 2.1	94.0	87.0
8	Ours (unrestricted & jitt. pool.)	83.8 ± 1.6	95.0	**87.7**

Table 3.4: Comparison with the state of the art on YouTube verification (restricted and unrestricted). *Our Fisher Vector face track descriptors significantly outperform the existing methods using shallow representations similar to ours. All rows except the last two rows show results on the restricted training setting.*

3.5.2.2 INRIA-Buffy dataset

Similar to the YouTube face verification task, this task involves determining whether a given pair of tracks portrays the same person or not. We compare our descriptor with the state-of-the-art method of [34] in two settings: unsupervised and supervised.

In the unsupervised setting, the Euclidean distance between the uncompressed descriptors computed on the 80×80 face patches provided with the dataset is used for comparison (Table 3.5). We compare training the GMM used to compute the representation on YouTube Faces and INRIA-Buffy. Even when the GMM is trained on a different dataset (YTF), we significantly (7.2%) outperform the descriptor of [34] on INRIA-Buffy (row 2). The results are further improved by computing the GMM on INRIA-Buffy directly, leading to a substantial 12% improvement over the state of the art (rows 3 and 4).

Having shown the superiority of the descriptor in the unsupervised case, we now couple it with the discriminatively trained dimensionality reduction, which simultaneously leads to dimensionality reduction. In order to learn the metric, we require positive and negative face tracks. Unfortunately, forming positive track pairs is not possible in INRIA-Buffy due to a limitation of the dataset (one only knows that frames within a track correspond to the same person)[1]. Hence we learn the metric on YTF and evaluate the resulting descriptor on INRIA-Buffy, verifying in this manner the ability of the method to transfer from one dataset to another. Note that this requires training the GMM on YTF as well. As shown in Table 3.6, despite learning on a completely different dataset, we achieve an EER of 69.76% (row 3), 6% better than [34] (row 1). Using joint metric learning and test time flips improves the EER to 74.23% (row 5). Binarization improves the EER further to 78.1%, 8% better than the state of the art (row 2).

[1]The method of [34] does learn a metric from INRIA-Buffy, but uses individual track frames rather than tracks as a whole as we require.

Setting	Method	100% - EER
1	Cinbis *et al.* [34]	57.50
2	Ours (GMM trained on YTF)	64.73
3	Ours (GMM trained on Buffy)	69.52
4	Ours (GMM trained on Buffy) & flip	**69.89**

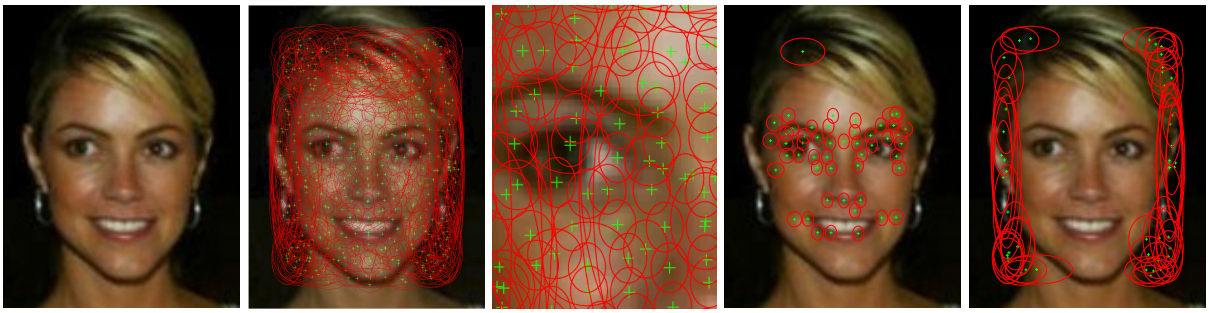
Table 3.5: Comparison with the state of the art on INRIA-Buffy verification (using the unsupervised Euclidean distance). *Our method outperforms the state of the art by* 12%.

Setting	descriptor variant	Proj. Dim.	100% - EER
1	Cinbis *et al.* [34] (trained on LFW)	N.A.	63.8
2	Cinbis *et al.* [34] (trained on Buffy)	N.A.	70.00
3	video pool.	128	69.76
4	video pool. + flip	128	71.84
5	video pool. + joint sim. + flip	128×2	74.23
6	video pool. + binar. 1024 bit + flip	128	77.79
7	video pool. + binar. 2048 bit + flip	128	**78.1**

Table 3.6: Comparison with the state of the art on INRIA-Buffy verification (using the learnt distance). *proj-n – projection dimensionality. bin-n – dimensionality after binarization.*

3.5.3 Learnt projection model visualization

Having shown the superiority of our features for representing the faces, in this section we demonstrate that the learnt model can indeed capture face-specific features. To visualize the projection matrix W, we make use of the fact that each GMM component corresponds to a part of the Fisher Vector and, in turn, to a group of columns in W. This makes it possible to evaluate how important certain Gaussians are for comparing human face images by computing the energy (Euclidean norm) of the corresponding column group. In Fig. 3.4 we show the GMM components which correspond to the groups of columns with the highest and lowest energy. Each Gaussian captures joint appearance-location statistics (Sect. 3.1.2), but here we only visualize the location as an ellipse with the center and radii set to the mean and variances of the spatial components. As can be seen from Fig. 3.4-d, the 50 Gaussians corresponding to the columns with the highest energy match the facial features without being explicitly trained to do so. They have small spatial variances and are finely localized on the image plane. On the contrary, Fig. 3.4-e shows how the 50 Gaussians corresponding to the columns with the lowest



(a) (b) (c) (d) (e)

Figure 3.4: Coupled with discriminative dimensionality reduction, a Fisher vector can automatically capture the discriminative parts of the face. *(a): an aligned face image; (b): unsupervised GMM clusters densely span the face; (c): a close-up of a face part covered by the Gaussians; (d): 50 Gaussians corresponding to the learnt projection matrix columns with the highest energy; (e): 50 Gaussians corresponding to the learnt projection matrix columns with the lowest energy.*

energy cover the background areas. These clusters are deemed as the least meaningful by our projection learning; note that their spatial variances are large.

Videos. Our descriptor representation for videos is remarkably simple as it does not account *explicitly* for invariance properties of faces in a track such as head pose, location of landmarks etc.and yet achieves state-of-the-art results in a number of datasets. Fig. 3.5 shows that, in fact, this representation is achieving invariance *implicitly* by visualizing the pixels in a face image, whose descriptors are hard- assigned to selected GMM components. The key observation is that GMM components fire in a *co-variant* manner, such that different components can be seen as capturing different parts of the face (*e.g.* eyes, mouth, and nose). Furthermore, the same components fire on analogous parts in different faces, establishing semantically meaningful correspondences. By stacking x, y coordinates to the SIFT descriptor, moreover, these act as "spring terms" encouraging given components to fire around the mean part location, with a regularization effect. Note that no explicit selection, learning or detection of facial landmarks is required.

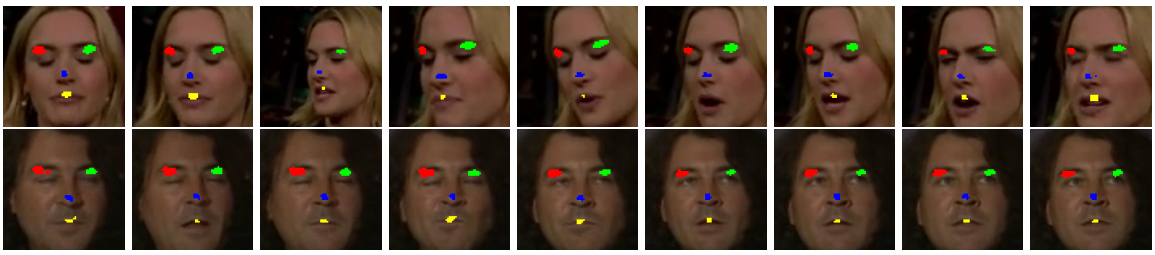


Figure 3.5: ***Unsupervised GMM implicitly performs robust face part tracking.*** *On the left frame of the track in the* top row*, we manually selected 4 points on eyelids, nose and mouth. These points correspond to 4 SIFT patches, which are assigned to particular Gaussians from the GMM (colour-coded with red, green, blue, and yellow). We visualize the pixels assigned to these Gaussians on 9 frames of the track. As can be seen, the same face parts get consistently assigned to the same Gaussians, which indicates the robustness of the representation with respect to the face position variation. In the* bottom *row, we visualize the face pixels assigned to exactly the same Gaussians as in the top row, but for a different person. Again, they correspond to the same face parts, which means that the assignment is consistent across different identities. The comparison of Fisher vectors thus can be seen as the implicit and robust comparison of face parts.*

3.5.4 Summary

We have shown that an off-the-shelf image representation based on dense SIFT features and Fisher Vector encoding are indeed suitable for representing human faces and achieves comparable to the state-of-the-art performance on challenging image and video benchmark datasets. The use of dense features allowed us to avoid applying a large number of sophisticated face landmark detectors. Also, we have presented a large-margin dimensionality reduction framework, well suited for high-dimensional Fisher Vector representations. In the next chapter, we look at learning representations for faces in a more end-to-end manner using Convolutional Neural Network (CNN) based approch.

3.6 Source code and data release.

The source code, reproducing the results presented on the LFW dataset in this chapter, is publicly available from `http://www.robots.ox.ac.uk/~vgg/software/face_desc/` We also released the data packages with the learnt models and the pre-computed descriptors and verification scores.

Chapter 4: Deep features for face recognition

Convolutional Neural Networks (CNNs) have taken the community by storm in recent days. They have resulted in collossal improvement in the state of the art in the computer vision community. These networks operate on images by convolving them with banks of filters and passing the output of the convolution through the nonlinear projection. This process is serially repeated several times giving them their popular "Deep" identity. Fig. 4.1 shows an example architecture and overview of the process. In this chapter we investigate use of these CNNs for representing faces in images and videos. To this end, this chapter makes two contributions: first, we show that a very large scale dataset can be built efficiently with minimal manual effort from the information available on the Web. Next we show that, a popular CNN architecture shown to be successful for image classification task is indeed useful for face recognition tasks as well. Next, we investigate impact of strategies like face alignment, popularly used in the community in the context of this proposed CNN architecture. Finally, we present the performance of these features on several public benchmark datasets.

4.1 Dataset Collection

In this section we propose a multi-stage strategy to collect a large face dataset containing hundreds of example images for thousands of unique identities (Table 4.2). The different stages of this process and corresponding statistics are summarised in Table 4.1. Individual stages involved in this process of data collection are discussed in detail in the following paragraphs.

Stage 1. Bootstrapping a list of candidate identity names. The first stage in building the dataset is to obtain a list of names of candidate identities for obtaining faces. The idea is to

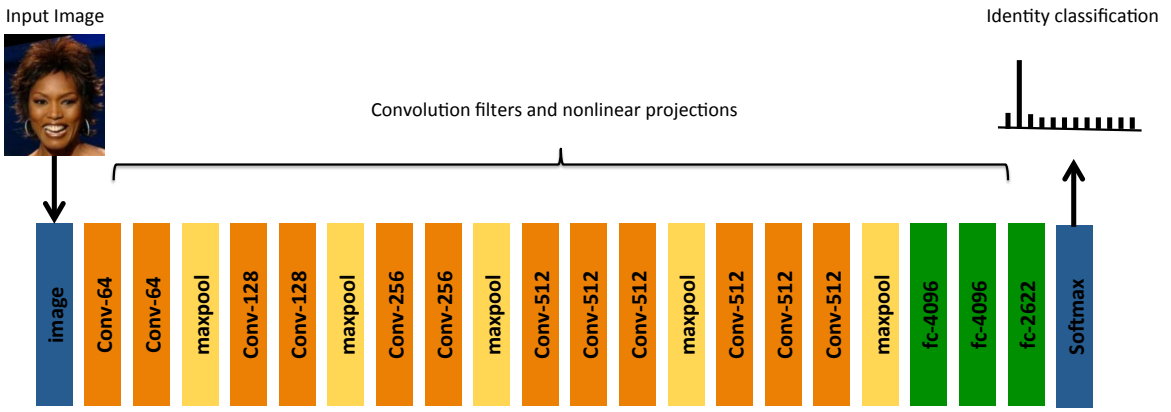


Figure 4.1: Overview: *Input image is passed through a series of convolution filters and non-linear projections to obtain the identity classification. Weights of the filters are learnt to minimise the loss in the identity classification.*

focus on celebrities and public figures, such as actors or politicians, so that a sufficient number of distinct images are likely to be found on the web, and also to avoid any privacy issue in downloading their images. An initial list of public figures is obtained by extracting males and females, ranked by popularity, from the *Internet Movie Data Base* (IMDB) celebrity list. This list, which contains mostly actors, is intersected with all the people in the *Freebase knowledge graph* [4], which has information on about 500K different identities, resulting in a ranked lists of 2.5K males and 2.5K females. This forms a candidate list of 5K names which are known to be popular (from IMDB), and for which we have attribute information such as ethnicity, age, kinship etc. (from the knowledge graph). The total of 5K images was chosen to make the subsequent annotation process manageable for a small annotation team. 200 images are then downloaded for each name of the candidate list using Google Image Search.

Stage 2. Filtering a list of candidate identity names. The candidate list is then filtered to remove identities for which there are not enough distinct images, and to eliminate any overlap with standard benchmark datasets. The 200 images per candidate are then presented to human annotators (sequentially in four pages of 50) to determine which identities result in sufficient image purity. Specifically, annotators are asked to retain an identity only if the corresponding set of 200 images is roughly 90% pure. Fig. 4.2 and 4.3 show examples of this process. The lack of purity could be due to homonymy or image scarcity. This filtering step reduces the

candidate list to 3,250 identities. Next, any names appearing in the LFW and YTF datasets are removed in order to make it possible to train on the new dataset and still evaluate fairly on those benchmarks. In this manner, a final list of 2,622 celebrity names is obtained.

Figure 4.2: Stage 2. Candidate list filtering: *We select the names where the Google Image Search results represent unimodal nature.*

Stage 3. Collecting more images for each identity and automatic filtering. Each of the 2,622 celebrity names is queried in both Google and Bing Image Search, and then again after appending the keyword "actor" to the names. This results in four queries per name and 500

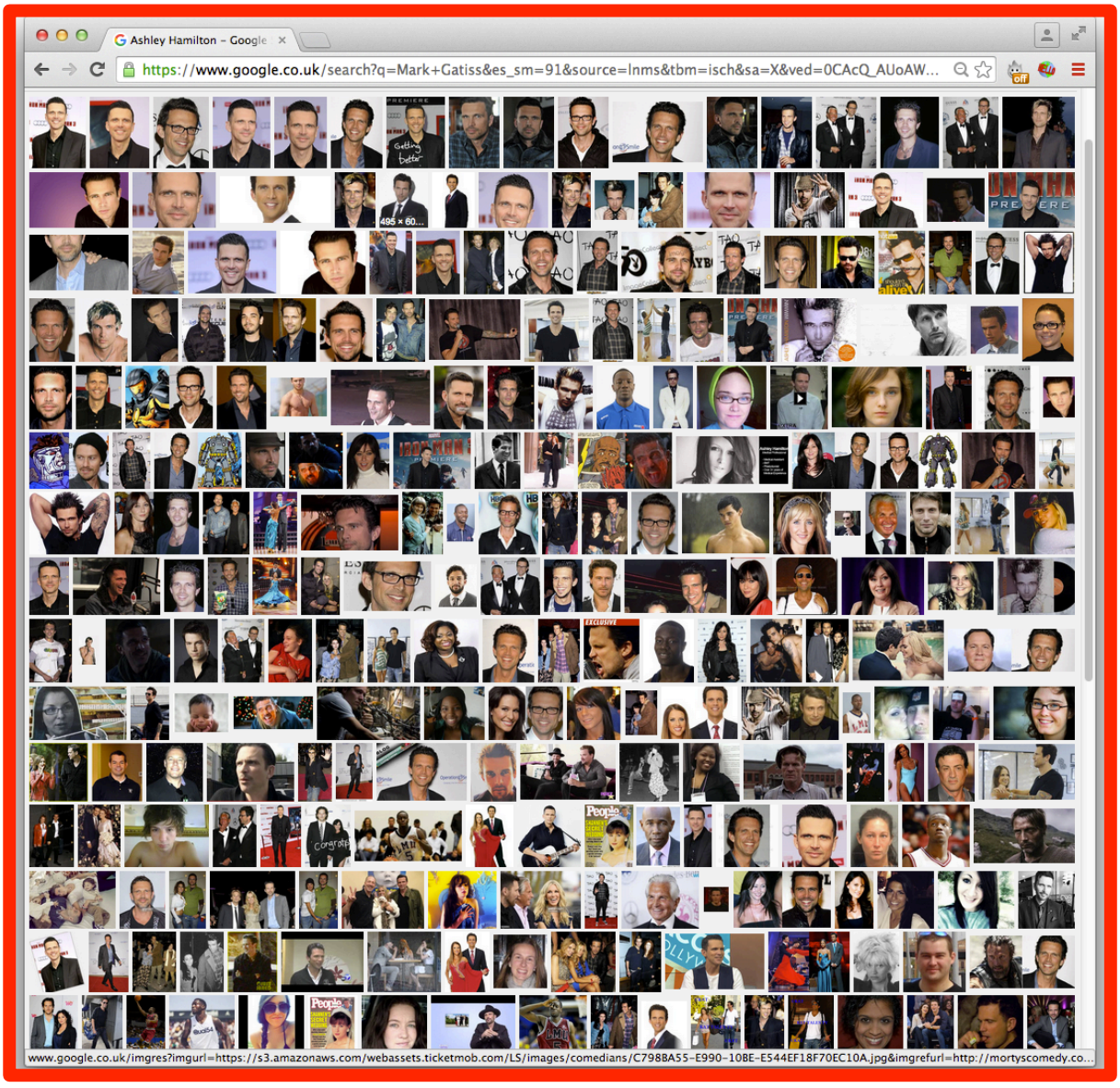


Figure 4.3: Stage 2. Candidate list filtering: *We reject the names where the Google Image Search returns incoherent results.*

results for each, obtaining 2,000 images for each identity. The next step is to remove any erroneous faces in each set automatically using a classifier. To achieve this the top 50 images (based on Google search rank in the downloaded set) for each identity are used as positive training samples, and the top 50 images of all other identities are used as negative training samples. A one-vs-rest linear SVM is trained for each identity using the Fisher Vector Faces descriptor described in the previous chapter (Chapter 3). The linear SVM for each identity is then used to rank the 2,000 downloaded images for that identity, and the top 1,000 are retained (the threshold number of 1,000 was chosen to favour high precision in the positive predictions).

Stage 4. Near duplicate removal. Exact duplicate images arising from the same image being found by two different search engines, or by copies of the same image being found at two different Internet locations, are removed. Near duplicates (e.g. images differing only in colour balance, or with text superimposed) are also removed. This is done by computing the VLAD descriptor [13, 75] for each image, clustering such descriptors within the 1,000 images for each identity using a very tight threshold, and retaining a single element per cluster.

Stage 5. Final manual filtering. At this point there are 2,622 identities and up to 1,000 images per identity. The aim of this final stage is to increase the purity (precision) of the data using human annotations. However, in order to make the annotation task less burdensome, and hence avoid high annotation costs, annotators are aided by using automatic ranking once more. This time, however, a multi-way CNN is trained to discriminate between the 2,622 face identities using the AlexNet architecture of Krizhevsky *et al.* [80]; then the softmax scores are used to rank images within each identity set by decreasing likelihood of being an inlier. In order to accelerate the work of the annotators, the ranked images of each identity are displayed in blocks of 200 and annotators are asked to validate blocks as a whole. In particular, a block is declared *good* if approximate purity is greater than 95%. The final number of good images is 982,803, of which approximately 95% are frontal and 5% profile.

Figure 4.4: Dataset examples: *Example images from our dataset. Images show intra class variations wrt expressions, pose, lighting conditions and various other facial attributes.*

Stage	Aim	Type	# of persons	# of images per person	total # images	Anno. effort (person-days)	100%-EER
1	Candidate list generation	A	5,000	200	1,000,000	–	–
2	Candidate list filtering	M	2,622	-	-	4	–
3	Rank image sets	A	2,622	1,000	2,622,000	–	96.90
4	Near dup. removal	A	2,622	623	1,635,159	–	–
5	Final manual filtering	M	2,622	375	982,803	10	92.83

Table 4.1: Dataset statistics after each stage of processing. *A considerable part of the acquisition process is automatically carried out. Type A and M specify whether the processing stage was carried out automatically or manually. The EER values are the performance on LFW for CNN configuration A trained on that stage of the dataset and compared using $l2$ distance.*

Discussion. Overall, this combination of using Internet search engines, filtering data using existing face recognition methods, and limited manual curation is able to produce an accurate large-scale dataset of faces labelled with their identities. The human annotation cost is quite small – the total amount of manual effort involved is only around 14 person-days, and only four person-days up to stage 4. Fig. 4.4 shows example images from our dataset and Table 4.2 compares our dataset to several existing ones.

A number of design choices have been made in the process above. Here we suggest some alternatives and extensions. The Freebase source can be replaced by other similar sources such as DBPedia (Structured WikiPedia) and Google Knowledge Graph. In fact, Freebase will be shut down and replaced by Google Knowledge Graph very soon. On the image collection front, additional images can be collected from sources like Wikimedia Commons, IMDB and also from search engines like Baidu and Yandex.

The order of the stages could be changed to remove near duplicates before stage 3. In terms of extensions, the first stage of annotation can be automated by looking at the distribution of pairwise distances between the downloaded images. An image class with high purity should exhibit a fairly unimodal distribution.

4.2 Network architecture and training

This section describes the CNNs used in our experiments and their training. Inspired by the architecture proposed by Simonyan *et al.* [135], the networks are "very deep", in the

Dataset	Identities	Images
LFW	5,749	13,233
Chen *et al.* [31]	2,995	99,773
CelebFaces [144]	10,177	202,599

Dataset	Identities	Images
VGG Face (ours)	2,622	2.6M
FaceBook [149]	4,030	4.4M
Google [123]	8M	200M

Table 4.2: Dataset comparisons: *Our dataset has the largest collection of face images outside industrial datasets by Goole, Facebook, or Baidu, which are not publicly available.*

sense that they comprise a long sequence of convolutional layers. Such CNNs have recently achieved state-of-the-art performance in some of the tasks of the ImageNet ILSVRC 2014 challenge [117], as well as in many other tasks [61, 135, 146].

4.2.1 Learning a face classifier

Initially, the deep architectures ϕ are bootstrapped by learning to recognize the $N = 2,622$ unique identities of the dataset of Sect. 4.1, by formulating a N-way classification problem. The CNN associates to each training image $\ell_t, t = 1, \ldots, T$ a score vector $\mathbf{x}_t = W\phi(\ell_t) + b \in \mathbb{R}^N$ by means of a final fully-connected layer containing N linear predictors $W \in \mathbb{R}^{N\times D}$, $b \in \mathbb{R}^N$, one per identity. These scores are compared to the ground-truth class identity $c_t \in \{1, \ldots, N\}$ by computing the empirical *softmax log-loss*

$$E(\phi) = -\sum_t \log\left(e^{\langle \mathbf{e}_{c_t}, \mathbf{x}_t\rangle} / \sum_{q=1,\ldots,N} e^{\langle \mathbf{e}_q, \mathbf{x}_t\rangle}\right) \tag{4.1}$$

where $\mathbf{x}_t = \phi(\ell_t) \in \mathbb{R}^D$, and $\mathbf{e}_c$ denotes the one-hot vector of class c.

After learning, the classifier layer (W, b) can be removed and the vectors $\phi(\ell_t)$ used to represent a face. Face identity verification can then be carried out using Euclidean distance to compare the corresponding vectors. However, the performance can be significantly improved by tuning for verification in the Euclidean space using a "triplet loss" training scheme, described in the next section. While triplet loss training is essential to obtain a good overall performance, bootstrapping the network as a classifier, as explained in this section, was found to make triplet training significantly easier and faster.

4.2.2 Learning a face embedding using a triplet loss

Triplet-loss training aims to learn score vectors that perform well in the final application, *i.e.*identity verification by comparing face descriptors in Euclidean space. This is similar in spirit to "metric learning" described in Sect. 3.2 and, similar to other metric learning approaches, is used to learn a projection that is at the same time distinctive and compact, achieving dimensionality reduction at the same time.

Our triplet-loss training scheme is similar in spirit to that of Schroff *et al.* [123]. The output $\phi(\ell_t) \in \mathbb{R}^D$ of the CNN, pre-trained as explained in Sect. 4.2.1, is l^2-normalised and projected to a $L \ll D$ dimensional space using a linear projection $\mathbf{x}_t = W'\phi(\ell_t)/\|\phi(\ell_t)\|_2$, $W' \in \mathbb{R}^{L\times D}$. While this formula is similar to the linear predictor learned above, there are two key differences. The first one is that $L \neq D$ is not equal to the number of class identities, but it is the (arbitrary) size of the descriptor embedding (we set $L = 1,024$). The second one is that the projection W' is trained to minimise the empirical *triplet loss*

$$E(W') = \sum_{(a,p,n)\in T} \max\{0, \alpha - \|\mathbf{x}_a - \mathbf{x}_n\|_2^2 + \|\mathbf{x}_a - \mathbf{x}_p\|_2^2\} \tag{4.2}$$

Note that, differently from the previous section, there is no bias being learned here as the differences in Eqn. 4.2 would cancel it. Here $\alpha \geq 0$ is a fixed scalar representing a *learning margin* and T is a collection of *training triplets*. A triplet (a, p, n) contains an *anchor* face image a as well as positive $p \neq a$ and negative n examples of the anchor's identity. The projection W' is learned on target datasets such as LFW and YTF honouring their guidelines. The construction of the triplet training T set is discussed in Sect. 4.2.4.

4.2.3 Architecture

We consider three architectures based on the A, B, and D architectures of [135]. The CNN architecture A is given in full detail in Table 4.3. It comprises 11 blocks, each containing a linear operator followed by one or more non-linearities such as ReLU and max pooling. The first eight such blocks are said to be convolutional as the linear operator is a bank of linear

layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
type	input	conv	relu	conv	relu	mpool	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv
name	–	conv1_1	relu1_1	conv1_2	relu1_2	pool1	conv2_1	relu2_1	conv2_2	relu2_2	pool2	conv3_1	relu3_1	conv3_2	relu3_2	conv3_3	relu3_3	pool3	conv4_1
support	–	3	1	3	1	2	3	1	3	1	2	3	1	3	1	3	1	2	3
filt dim	–	3	–	64	–	–	64	–	128	–	–	128	–	256	–	256	–	–	256
num filts	–	64	–	64	–	–	128	–	128	–	–	256	–	256	–	256	–	–	512
stride	–	1	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	2	1
pad	–	1	0	1	0	0	1	0	1	0	0	1	0	1	0	1	0	0	1

layer	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
type	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	softmx
name	relu4_1	conv4_2	relu4_2	conv4_3	relu4_3	pool4	conv5_1	relu5_1	conv5_2	relu5_2	conv5_3	relu5_3	pool5	fc6	relu6	fc7	relu7	fc8	prob
support	1	3	1	3	1	2	3	1	3	1	3	1	2	7	1	1	1	1	1
filt dim	–	512	–	512	–	–	512	–	512	–	512	–	–	512	–	4096	–	4096	–
num filts	–	512	–	512	–	–	512	–	512	–	512	–	–	4096	–	4096	–	2622	–
stride	1	1	1	1	1	2	1	1	1	1	1	1	2	1	1	1	1	1	1
pad	0	1	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0

Table 4.3: Network configuration. *Details of the face CNN configuration A. The FC layers are listed as "convolution" as they are a special case of convolution (see Sect. 4.2.3). For each convolution layer, the filter size, number of filters, stride and padding are indicated.*

filters (linear convolution). The last three blocks are instead called Fully Connected (FC); they are the same as a convolutional layer, but the size of the filters matches the size of the input data, such that each filter "senses" data from the entire image. All the convolution layers are followed by a rectification layer (ReLU) as in [80]; however, differently from [80] and similarly to [135], they do not include the Local Response Normalization operator. The first two FC layers output are 4,096 dimensional and the last FC layer has either $N = 2{,}622$ or $L = 1{,}024$ dimensions, depending upon the loss functions used for optimization, either N-way class prediction (Sect. 4.2.1) or L-dimensional embedding (Sect. 4.2.2). In the first case, the resulting vector is passed to a softmax layer to compute the class posterior probabilities. Networks B and D are similar to A but include 2 and 5 additional convolution layers respectively.

The input to all networks is a face image of size 224×224 with the average face image (computed from the training set) subtracted – this is critical for the stability of the optimization algorithm.

4.2.4 Training

Learning the N-way face classifier (Sect. 4.2.1) follows the steps of [80] with the modifications suggested by [135]. The goal is to find the parameters of the network that minimize the average prediction log-loss after the softmax layer.

We describe the procedure for the CNN A configuration first, and then the variants for the B and D configurations. Optimization is done by the stochastic gradient descent method using

mini-batches of 64 samples and momentum coefficient of 0.9 [88]. The model is regularized using dropout and weight decay; the coefficient of the latter was set to 5×10^{-4}, whereas dropout was applied after the two FC layers with a rate of 0.5. The learning rate was initially set to 10^{-2} and then decreased by factor of 10 when the validation set accuracy stopped increasing. Overall, the model was trained using three decreasing learning rates.

The weights of the filters in the CNN were initialized by random sampling from a Gaussian distribution with zero mean and 10^{-2} standard deviation. Biases were initialized to zero. The training images were rescaled such that the smaller of width and height was equal to 256. During training, the network is fed with random 224×224 pixel patches cropped from these images (where crops change every time an image is sampled). The data was further augmented by flipping the image left to right with 50% probability; however, we did not perform any color channel augmentation as described in [135] and [80].

The CNN configuration A is trained from scratch, whereas configurations B and D are trained by starting from the trained A. This is obtained by appending additional fully connected layers to A, randomly initialized, and then training the latter as well as fine-tuning (train with a lower learning rate) the network again.

For learning the embeddings using the triplet loss (Sect. 4.2.2), the network is frozen except the last fully-connected layer implementing the discriminative projection. This layer is then learnt for 10 epochs using SGD with a fixed learning rate of 0.25. An epoch here contains all the possible positive pairs (a, p), where image a is considered the anchor and p its paired positive example. Choosing good triplets is crucial and should strike a balance between selecting informative (*i.e.*challenging) examples and swamping training with examples that are too hard. This is achieved by extending each pair (a, p) to a triplet (a, p, n) by sampling the image n at random, but only between the ones that violate the triplet loss margin. The latter is a form of hard-negative mining, but it is not as aggressive as (and much cheaper than) choosing the *maximally violating* example, as often done in structured output learning. The negatives are always searched in a mini-batch for two reasons. First to save time on exhaustive search and secondly, to make sure that labeling noise doesn't affect the performance. *E.g.*, it might

be possible that some images of George Bush has Tony Blair in them and these will be labeled incorrectly. So while finding negatives for the true Tony Blair class, we risk finding these false negatives if we exhaustively search through the entire dataset.

At the test time, the embedded descriptors $W'\phi(\ell_t)$ are compared in Euclidean distance for the purpose of face verification. In verification the goal is to tell whether two face images ℓ_1 and ℓ_2 have the same identity or not and this is obtained by testing whether the distance $\|W'\phi(\ell_1) - W'\phi(\ell_2)\|_2$ between embedded descriptors is smaller than a threshold τ. This threshold is not provided by the training procedure outlined above and is instead learned separately to maximize the *verification accuracy* (Acc.– or rate of correctly classified pairs) on suitable validation data.

4.3 Datasets and evaluation protocols

In order to allow for a direct comparison to previous work, evaluation is performed on existing benchmark datasets (which, by construction, contain different identities from our dataset).

The first dataset is the the Labeled Faces in the Wild (LFW) dataset [68]. The dataset is described in detail in Sect. 3.4.1. We follow the standard evaluation protocol defined for the "unrestricted setting" using data external to LFW for training, which in our case is the new face dataset. In addition to the verification accuracy Acc., we use the *Equal Error Rate* (EER) as an evaluation metric for testing the performance of different approaches.

The second dataset is the YouTube Faces (YTF) dataset [162]. The details of this dataset are discussed in Sect. 3.4.2. Again, we follow the standard evaluation protocol defined for the "unrestricted setting", and report EER.

4.4 Experiments and results

In the following section we first test the performance of several variations of the proposed models and training data using the LFW dataset. We then compare the performance of the best setups with state-of-the-art methods on LFW and YTF.

Implementation details. Our implementation is based on the MATLAB toolbox MatConvNet [157] linked with the NVIDIA CuDNN libraries to accelerate training. All our experiments were carried on NVIDIA Titan Black GPUs with 6GB of onboard memory, using four GPUs together. This is important due to the very significant memory footprint (and high complexity) of the very deep networks.

The CNN $\phi(\ell_t)$ contains all but the linear class predictor and softmax layers, outputting $D = 4{,}096$ descriptor vectors. Given a face image ℓ, four 224×224 pixel patches are cropped from the four corners and the center with horizontal flip (i.e. 10 in total, as in [28]), and the feature vectors from these are averaged. To enable multi-scale testing, the face is first scaled to three different sizes of 256, 384 and 512 pixels, and the cropping procedure repeated for each [135]. The resulting descriptor for the face is an average over all of these feature vectors.

Faces are detected using the method described in [98]. If face alignment is used, then facial landmarks are computed using the method of [48] and a 2D similarity transformation is applied to map the face to a canonical position.

For the YTF videos, K face descriptors are obtained for each video by ordering the faces by their facial landmark confidence score, and selecting the top K. Frontal faces are 2D aligned, but no alignment is used for profiles. Finally, the video is represented by the average of the K face descriptors.

4.4.1 Component analysis

This section evaluates the effect of different components of the system when training a network for face verification as in Sect. 4.2.4 and evaluating it on the LFW data. Table 4.4 summarises the results.

Dataset curation: First we analyze the curation effort for its effect on the performance of the network. We use dataset snapshots at stages 3 and 5 (Sect. 4.1), i.e. before and after curation, and test them using configuration A. The reason for selecting this configurations is that the networks can be trained from scratch. As can be seen, performance before curation is better

No.	Config	Data	Train Align.	Test Align.	Embedding	100% - EER
1	A	C	No	No	No	92.83
2	A	F	No	No	No	95.80
3	A	F	No	Yes	No	96.70
4	B	F	No	Yes	No	97.27
5	B	F	Yes	Yes	No	96.17
6	D	F	No	Yes	No	96.73
7	B	F	No	Yes	Yes	99.13

Table 4.4: Performance evaluation on LFW, unrestricted setting. *Training on the full dataset (F, stage 3), leads to a better performance than training on the curated dataset (C, stage 5). 2D alignment at the test time slightly improves the performance. Learning embedding for verification significantly boosts the performance. All results are obtained using $l2$ distance measure between the test samples.*

(Table 4.4 rows 1 and 2). There are probably two reasons for this: first, it is better to have more data, even with label noise; and second, a more subtle point, some of the hard positives present in the stage 3 data get removed as a side effect of the curation process, and so the stage 5 data training does not benefit from these.

Alignment: As can be seen from Table 4.4 rows 2 and 3, using 2D alignment on test images does improve the performance, but performing 2D alignment on the training data does not provide an additional boost – see Table 4.4 rows 4 and 5.

Architecture: Next we vary the architecture of the network. We observe a slight boost in performance from configuration A to B (Table 4.4 rows 3,4) while configuration D fails to improve the results over configuration B (Table 4.4 rows 4,6). There are several possible reasons for this: the number of parameters in config. D is much more than B, due to the greater number of convolution layers. Also since network D is trained using fine-tuning of the new layers, setting parameters like learning rate and momentum becomes critical. Training the network from scratch is also an option which needs to be investigated in the future.

Triplet-loss embedding: Learning a discriminative metric by minimising the triplet loss (Sect. 4.2.2) further improves performance by 1.8% (Table 4.4 row 7 vs. row 4). Note that this amounts to reducing the error rate by 68%.

No.	Method	Images	Networks	Acc.
1	Fisher Vector Faces (Chapter 3, [134])	-	-	93.10
2	DeepFace [149]	4M	3	97.35
3	Fusion [150]	500M	5	98.37
4	DeepID-2,3		200	99.47
5	FaceNet [123]	200M	1	98.87
6	FaceNet [123] + Alignment	200M	1	99.63
7	VGG Face (ours)	2.6M	1	98.95

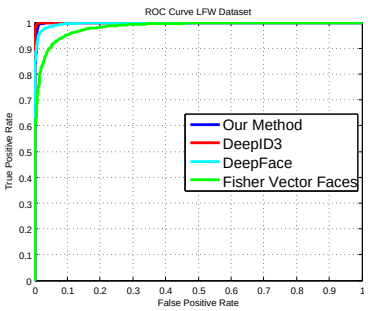


Table 4.5: LFW unrestricted setting. *Left: we achieve comparable results to the state of the art using much less training data (than DeepFace and FaceNet) and using a simpler network architecture (than DeepID-2,3). NB DeepID3 used a test set with label errors corrected. The other methods used uncorrected test sets. Right: ROC curves.*

No.	Method	Images	Networks	100%- EER	Acc.
1	Video Fisher Vector Faces [107]	-	-	87.7	83.8
2	DeepFace [149]	4M	1	91.4	91.4
3	DeepID-2,2+,3		200	-	93.2
4	FaceNet [123] + Alignment	200M	1	-	95.1
5	VGG Face (ours)	2.6M	1	92.8	91.6
6	VGG Face - Embedding Learning (ours)	2.6M	1	97.4	97.3

Table 4.6: Results on the Youtube Faces Dataset, unrestricted setting. *We used* 100 *faces to represent each video.*

4.4.2 Comparison with the state-of-the-art

LFW: Table 4.5 compares our results with the best results on LFW dataset, and also shows these as ROC curves. It can be observed that we achieve comparable results to the state of the art using much less training data and a much simpler network architecture.

YTF: Table 4.6 shows the performance on the YTF dataset. We achieve the state of the art performance using a triplet loss embedding method.

4.5 Summary

In this work we have made two contributions. First, we have designed a procedure that is able to assemble a large scale dataset with low label noise, while minimizing the amount of manual annotation required. The second contribution was to show that a deep CNN, without any embellishments but with appropriate training, can achieve results comparable to the state of

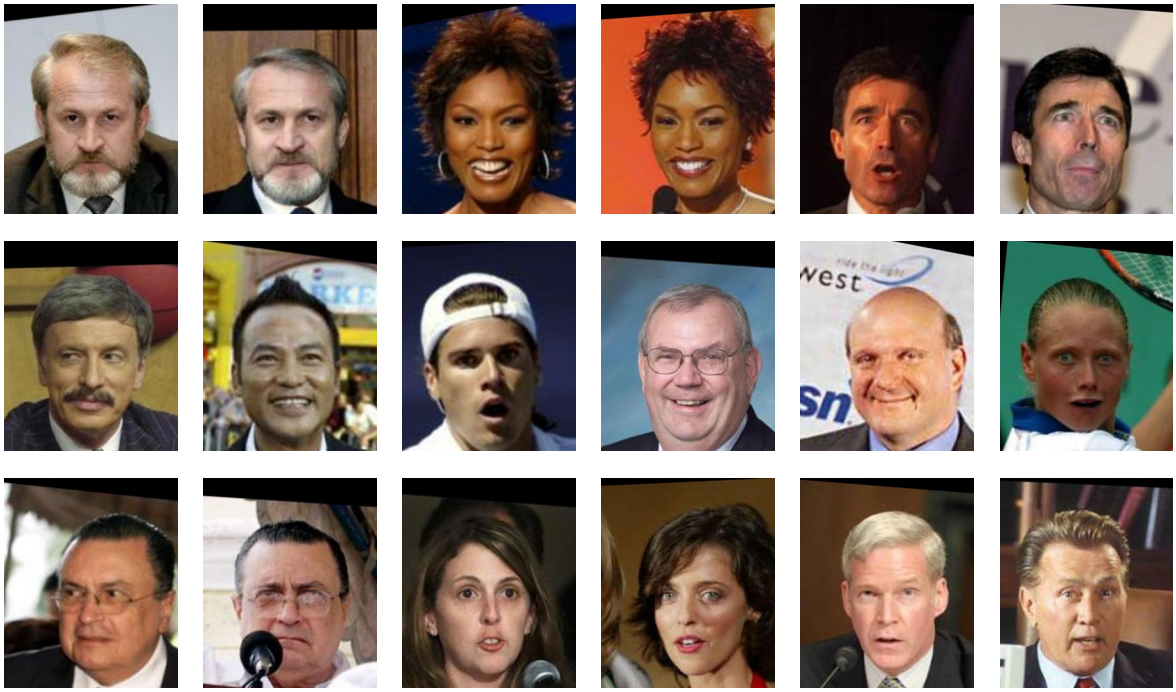


Figure 4.5: Qualitative Analysis: *Top row: Correctly classified positive pairs from the LFW dataset. Middle Row: Correctly classified negative pairs from the LFW dataset. Faces are classified correctly regardless of pose, lighting, hairstyle variations. Bottom row: Misclassifications from the LFW dataset. Main reasons for confusion appear to be occlusions and facial expressions.*

the art. In the previous and current chapter we have proposed different methods for describing faces in images and videos. In the following chapters, we will investigate their effectiveness for various applications such as labeling faces in broadcast videos and retrieving paintings of people given their pictures.

Source code and data release. The MATLAB source code for and the network model described in this chapter is available from `http://www.robots.ox.ac.uk/~vgg/software/vgg_face/`. We have also released the network model in Caffe and Torch frameworks.

Chapter 5: Weakly supervised labeling

The objective of this chapter is the automatic labeling of characters in TV video and movies, given weak supervisory information provided by an aligned transcript (Fig. 5.1). We make the following four novel contributions: **first**, Sect. 5.2, we propose a way of obtaining stronger supervisory information for the primary cast of a movie or a tv series (principal characters) than the previously proposed approaches. The change in obtaining the positive training data is quite subtle – it involves tight alignment with the subtitles rather than the shot or scene which was used previously. For negative training however, we are able to obtain somewhat surprisingly, far more than has been used in previous work; **second**, Sect. 5.3, we introduce an explicit classifier for *background characters* by using their face-track information. These are the actors that have non-speaking parts, and are usually in the background (for example, in an audience or busy street scene). These background characters are a rich source of negative training data when identifying the principal characters and positive training data for identifying the non-principal characters; **third**, we evaluate the suitability of two face-track feature vectors described previously for this task. The first based on the Fisher Vector representation described in Chapter 3, and the second based on training a ConvNet architecture described in Chapter 4; **Finally**, we investigate the generalization of the classifiers by labeling new episodes *without* the use of supervisory information from the transcript, and compare this performance to using weak and strong supervision.

To show the improvement in performance brought by each contribution, in Sect. 5.7 we compare to the previous methods using the face-tracks of the previous work [24, 47, 137, 151], so that the performance is assessed on exactly the same training and test data and face feature, and the improvements due to each contribution can be isolated. For example, the stronger su-

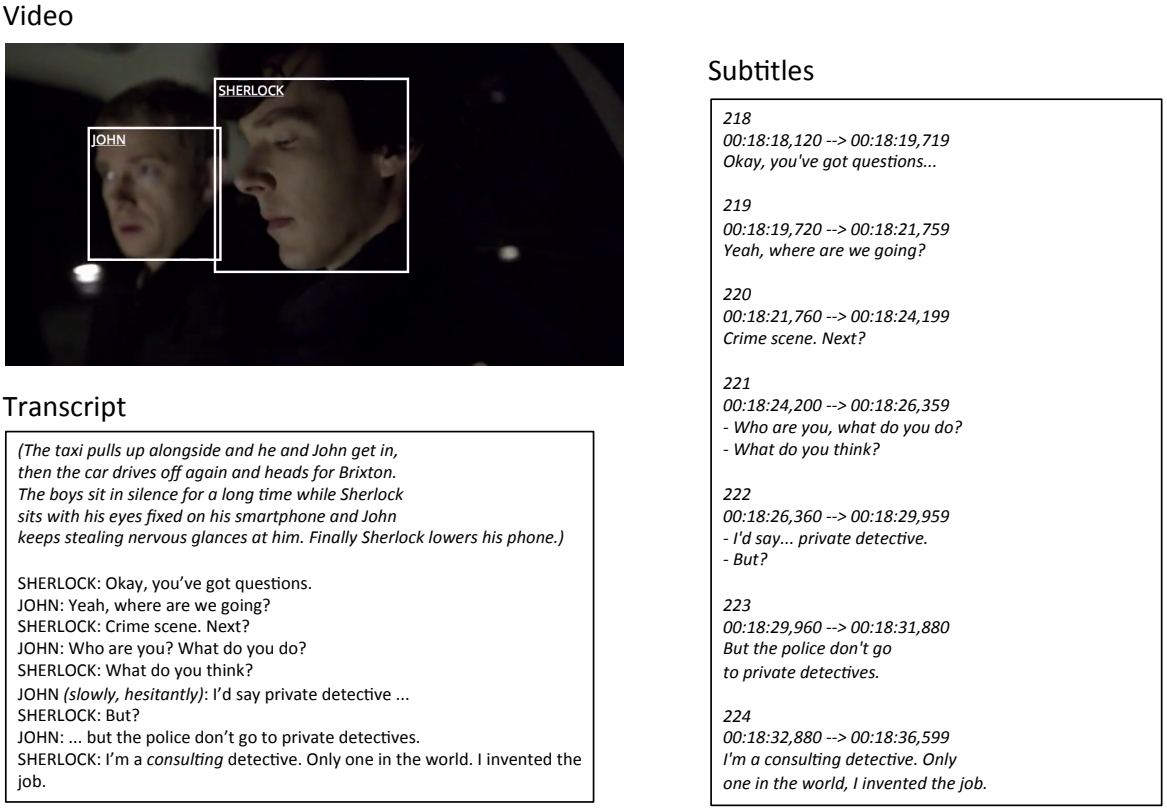


Figure 5.1: Problem overview: *Transcripts provide important clues about identities of characters appearing in a video. They lack all important timing information which is provided by the subtitles. Together, they provide information about which characters are speaking at what time. This week supervisory information is then utilized to bootstrap stronger models for labeling characters on screen. One of the problems is that the speaking character may not be appearing on the screen* i.e. *camera can be focused on one character while the other character is speaking. This presents a challenge for the learning methods.*

pervisory information alone brings a significant improvement in performance (see Fig. 5.6). We implemented previous methods where code was not available [36, 161] so that the strengths and weaknesses of previous approaches can be compared. The learning and inference formulation is described in Sect. 5.4, and the implementation details are given in Sect. 5.5. Overall we achieve a dramatic improvement over the state of the art on both TV series and film datasets, and almost saturate performance on some benchmarks.

5.1 Problem Specification

The problem we are considering here is the following: given a set of face tracks and a label set $\mathcal{Y}$ of characters, the objective is to predict a label $\hat{y}_i \in \mathcal{Y}$ for each face track indicating that this

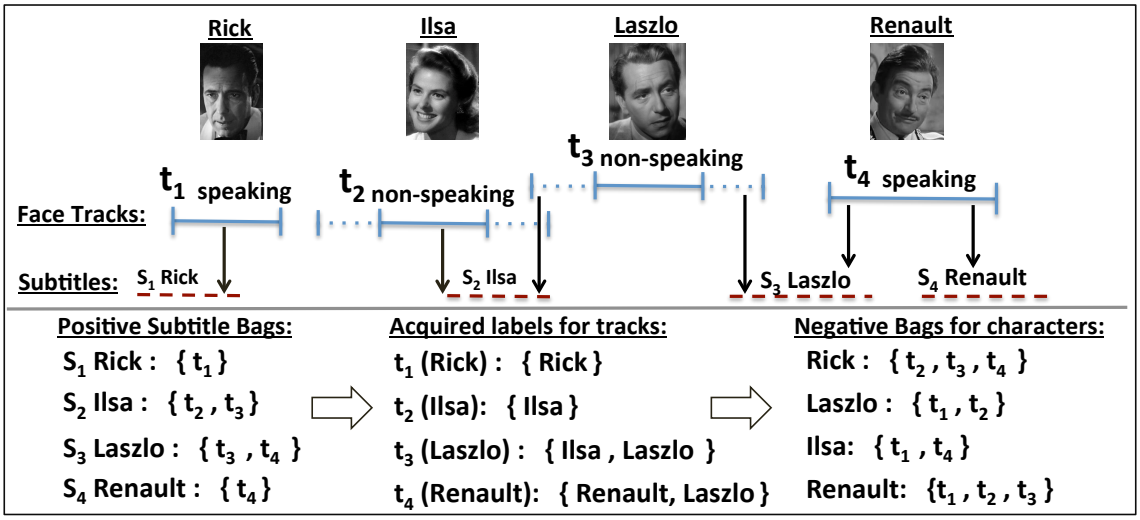


Figure 5.2: Label bag formation for the principal characters: *Top part: face tracks (t_i, Blue Lines) and their ground truth (Names and Pictures) and associated speaking/non-speaking label. The boundaries of non-speaking tracks are extended to provide better coverage. The aligned subtitles (S_i, Dotted Red Lines) are used to form the bags as follows: (a) each subtitle forms a bag containing all overlapping tracks ("Positive Subtitle Bags"). (b) These bags determine the possible labels for each track ("Acquired labels"). (c) Track with given possible labels forms negative example for all other track not containing those labels in their bags ("Negative bags").*

track depicts the target character $\hat{y}_i$.

We formulate the character naming problem as Multiple-Instance Learning [8]. This means that for each character we group the face tracks into sets (referred to as bags) and label them as positive if they contain at least one sample of the target character and negative otherwise. Then we learn a classification function that forces a maximum margin between the best scoring track in the positive bags and all the tracks in the negative bags. The initial bags are formed using text-based information obtained from the subtitles and transcripts.

In more detail, every bag, $\mathcal{B}$, is associated with one label for each target character y. The label l^y can take one of the values $\{1, -1, 0\}$, where 1 indicates that at least one track in $\mathcal{B}$ depicts characters y, -1 indicates that none of the tracks in $\mathcal{B}$ belong to the character y, and 0 indicates that we have no information if the tracks in $\mathcal{B}$ depict the character y or not. These cases are referred to as positive, negative, and ignore bags, respectively. Note, a face track can appear in multiple bags (as it can be positive for the target character, but negative for others).

An optimal positive bag would be small, but still contains at least one track of the target character. Conversely, the negative bags should contain as many tracks as possible, but ideally

not a single track of the target character. In practice, it is not possible to construct perfect bags and it turns out that the quality of these bags has a significant impact on the final classification performance (as discussed in Sect. 5.7).

In this chapter, we apply very different approaches for generating the bags for the principal characters and background characters. For the principal characters (Sect. 5.2), the supervisory information is obtained from the aligned subtitles and transcripts [47], which provides a speaker label $sp_k \in \mathcal{Y}$ and the corresponding time interval for each subtitle k, and the bags are based on this. For the background characters (Sect. 5.3), the supervisory information is obtained from a background character classifier.

5.2 Automatic supervision for the principal characters

A positive bag is constructed for each *subtitle* k as follows: consider all tracks that overlap with the corresponding frame interval by more than v_p frames (we use $v_p = 5$). If any such tracks exist, collect them into a bag $\mathcal{B}$ and set the label $l^y = 1$ for $y = sp_k$ and zero for all the others. Furthermore, if any of these tracks are detected to be speaking, remove all other tracks from the bag. This is done only if the speaker detection is used. Note, using the subtitle interval provides a very 'tight' bag, as opposed to using bags defined by multiple shots or a scene level grouping. We return to this point in Sect. 5.5.2. Clearly, this bag formation may contain errors, for example in a reaction shot where the person speaking the line is not shown. However, the MIL-SVM learning framework is tolerant to such noisy supervision.

A negative bag is constructed for each *face-track* i by finding all the subtitles k that overlap with the extended face track interval $\{f_i^{start} - v_n, f_i^{end} + v_n\}$, where the frame indices $f_i = \{f_i^{start}, f_i^{end}\}$ refer to the first and last frame of the track. The magnitude v_n of the extension is 200 frames if the track i is not detected to be speaking, and zero otherwise. If overlapping subtitles are found, take the corresponding speaker names and label the track i as negative sample for all the other characters. For example, if the overlapping subtitles for the track t_4 in Fig. 5.2 indicate that characters "Laszlo" and "Renault" are speaking, use t_1 as a negative sample for all other characters but "Laszlo" and "Renault". After going though all tracks, we

pool all face tracks that are marked as negative for a particular character into a single negative bag.

5.3 Learning a classifier for the background characters

As described previously, movies and TV-series usually contain two types of characters, the principal characters and the background characters. The principal characters appear and speak often during the film, whereas, the background characters may appear just once and they often don't speak at all. A sample of background characters are shown in Fig. 5.3.

The number of background characters depends very much on the material. For example, in the TV sitcom 'The Big Bang Theory' which mainly centers on the principal characters, there are very few scenes containing background actors. However, in other sitcoms like 'Friends', there are many background characters in the cafe scenes; and in a feature film like Casablanca, where there are many scenes in a bar or outside on the streets, then there can be tens to hundreds of background actors. Table 5.2 reports the number of principal and background characters for the datasets used in this chapter. In the "Casablanca" dataset, for example, as many as 30% of the total tracks belong to the background category. Interestingly, there are whole websites devoted to background characters, for example spotting bizarre behavior or bad acting.

Prior approaches to background characters: In previous work [24, 48, 137, 151, 161] the task of labeling background characters has hardly been discussed, though a distinction has been made between principal characters and 'others', with 'others' generally being considered as a single class. Cour *et al.* [37] pruned all face tracks from the data that did not contain one of the target (principal) characters. In [47], [137], [161] and [151] the background characters were not pruned, but no classifier was learned for this category. In [151] a character was labeled as background class if none of the principal characters obtained a high score. For [47], [137], [161], any track depicting a background character was guaranteed to be misclassified. Since the test material in these works consisted of TV-series, where only a few background characters appear, this problem was not clearly visible in the results.

To our knowledge, Bojanowski *et al.* [24], is the only work that attempts to explicitly classify the background characters. However, their use of the term background refers more to an 'other' character (i.e. not one of the principal characters) rather than the role of a background character introduced in our work. They use 300 random samples from the "Labeled Faces in the Wild" dataset as positive samples of the background class, but does not make use of face-track information (e.g. size, position) as we do here.

As a result of this, their prediction accuracy for the background class is significantly lower than for any of the principal characters and also over 20% less than our performance (see Fig. 5.7 in Sect. 5.7).

In most works, including ours, the aim is to learn a separate classifier for each principal character and a single classifier for the remainder. The issue is that it is very difficult to obtain supervisory information for background characters as they do not appear in the transcripts (as they don't speak) and this can lead to labeling confusions with the principal characters. To this end, we propose next a novel classification scheme to automatically classify tracks as background or not, using features formed from the face-tracks, and generate bags for them in a manner that takes proper account of the available information.

5.3.1 Background/non-background track classifier

The classifier is based on simple features obtained from track level statistics together with a linear SVM classifier. It is inspired by the false positive track classifier of [77]. We observe that background characters appear at particular locations, often away from the central action in a frame. This makes the location of the detections of a track, their sizes and their motion in a shot, key elements in making a decision. More specifically, a feature vector consists of: the track length; mean and standard deviation of three properties (i) detection sizes, (ii) location of the face detection center, and (iii) facial landmark detection scores; giving a seven dimensional feature vector per track. A linear SVM classifier is trained using ground truth obtained from the television series "Scrubs". This learnt classifier is then used to rank the tracks in a given video. This simple scheme achieves an Average Precision (AP) of 75% (Fig. 5.4). Fig. 5.5 shows some

Figure 5.3: ***Examples of background characters:*** *Each row shows three frames of a track. One can observe that background characters have typical characteristics: the scale of the face, the motion through the shot, and their location. These characteristics assist in identifying them.*

example tracks classified as background characters by our method.

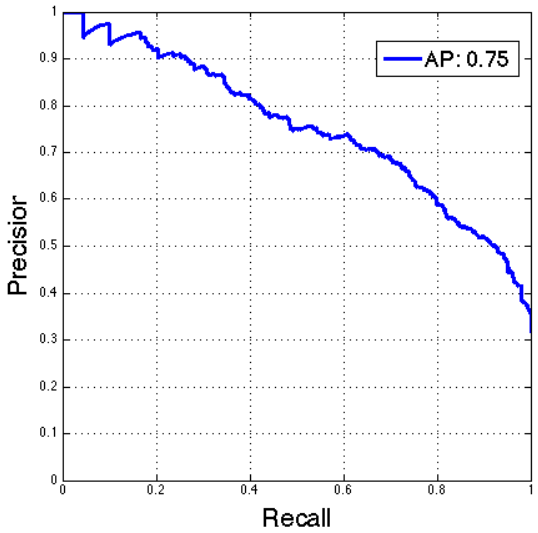
5.3.2 Bag construction

The background-classifier is applied to all face tracks, and tracks are then ranked according to their classification score. This results in an ordering $\{t_1, t_2, t_3, \ldots, t_n\}$, where the face track t_1 has the highest likelihood to depict a background character. Then we start constructing positive bags for the background characters as follows:

A bag is formed from the face tracks t_1 and t_2, and labeled as positive for the background character (and ignore for the principal characters). Next, a bag is formed from the tracks t_2 and t_3, and labeled in the same way. We continue this manner until the background-classification score falls below zero. The motivation for this procedure is that since the background-classifier makes mistakes, the positive bags are formed by taking two consecutive tracks, instead of a single track. This increases the probability that any positive bag actually contains at least one track depicting a background character. In fact, we could take even more then two consecutive tracks, but two was found to be a good trade-off between the bag size and the accuracy.

The scores given by the background classifier are also used to construct a negative bag

for the background characters. In this bag, we take the N_n lowest scoring tracks (i.e. those corresponding to the principal characters), where N_n is set to be 40% of the total number of face tracks. Here we assume that at least 40 % of the tracks in a video depict the principal characters. Although the precision of our classifier is good (see below), there may be a few background characters included, but our MIL framework does not require all bags to be extremely pure.



(a)

Figure 5.4: ***Background classifier performance:*** *The precision-recall curve obtained by training on the Scrubs dataset and testing on the Casablanca dataset.*

5.4 Learning formulation and optimization

The definition of our learning objective roughly follows the standard MIL-SVM formulation [8], where the idea is to force a maximum margin between the best scoring samples in positive bags and all the samples in the negative bags. The difference to [8] is that we use a one-versus-all multi-class setup and have separate slack variable for each negative face track sample. This is equivalent to using a separate bag for every negative sample in the framework of [8].

We learn a linear classifier $\{w_y, b_y\}$ for each character y by minimizing the following ob-

(a)

Figure 5.5: ***Background classifier results:*** *Examples of high scoring classifications from the test set.*

jective:

$$\min_{w_y, b_y, \xi} \quad \frac{1}{2}||w_y||^2 + C\left(\sum_I \xi_I + \sum_{I,i} \xi_{Ii}\right) \tag{5.1}$$

$$\text{s.t.} \quad \langle w_y, x_{s(I)} \rangle + b_y \geq 1 - \xi_I \quad \forall I : l_I^y = 1,$$

$$-\langle w_y, x_i \rangle - b_y \geq 1 - \xi_{Ii} \quad \forall I, i : i \in \mathcal{B}_I, \text{ and } l_I^y = -1,$$

$$\xi_I \geq 0, \text{ and } \xi_{Ii} \geq 0 \quad \forall I, i \tag{5.2}$$

where bags are indexed by I and tracks by i, $\langle \cdot \rangle$ denotes the inner product, C is a constant (set to $C = 0.6$ in all experiments), and

$$s(I) = \underset{i \in \mathcal{B}_I}{\text{argmax}} \langle w_y, x_i \rangle + b_y \tag{5.3}$$

is a selector variable denoting the highest scoring sample in the bag $\mathcal{B}_I$. The slack variables ξ are introduced to explicitly model the noise in the bag labels. That is to say that the classifier is allowed to misclassify training samples with a certain additional cost. This is important if e.g. a bag $\mathcal{B}_I$ with label $l_I^y = 1$ actually does not contain any track of the character y.

The minimization of (5.1) is a non-convex problem and we solve it by alternating between the estimation of the selector variables (5.3) with fixed $\{w_y, b_y\}$, and the minimization of (5.1) with respect $\{w_y, b_y, \xi\}$ with fixed selectors $s(I)$. Once the selector variables are fixed, the optimization of (5.1) turns into a convex problem and it can be solved using standard techniques developed for SVMs.

The alternating optimization needs to be initialized by providing either the initial classifiers $\{w_y, b_y\}$ or the indices for the best scoring track in each positive bag. Such initialization can be obtained using e.g. [137], [37], or [24]. In the experiments we show that the proposed approach is not sensitive to the initialization.

Once the classifiers have been inferred, we compute the classification scores $s_i^y = w_y^T x_i + b_y$ for *all* face tracks i. The obtained values are further normalized by applying a sigmoid function $p_i^y = (1 + \exp(A_y s_i^y + B_y))^{-1}$ to map the original scores to corresponding posterior probabilities p_i^y. The parameters A_y and B_y are learned using 25% of the most confident positive and negative samples in the class y. If there are classes with less than 10 positive bags, we estimate shared normalization parameters for all of them. Finally, the identities of all face tracks i are predicted as $\hat{y}_i = \text{argmax}_y\ p_i^y$, and $\max_y p_i^y$ is used as the final prediction reliability.

5.5 Implementation details

5.5.1 Face track descriptors

We implement and compare two face-track descriptors, one based on Fisher Vectors (FV), the other based on ConvNets.

Fisher Vectors. This representation follows the approach described in Chapter 3. More specifically, dimensionality reduced SIFTs (64 dim.) are computed on a 2 pixel spaced grid on every face detection of the track, and aggregated into a single Fisher Vector for that track. With 512 GMM clusters, this generates a $67,584$ dimensional descriptor. Discriminative dimensionality dimension reduction is then employed to reduce its dimensions to 1024.

ConvNet. This representation follows the approach described in Chapter 4. We use VGG Face 16 layer model trained using MatConvNet [157]. A multi-scale feature vector is obtained for each face in the track as follows: the face detection is rescaled so that the lower of the height or width of the face matches three different sizes (256, 384 and 512 pixels). A 4096 dimensional descriptor (from the penultimate convolutional layer of the ConvNet) is then computed for each of 10 crops (corners, center with horizontal flips) at each scale of the face and sum-pooled. To obtain the track-level descriptor, face feature vectors are computed on 10 equally-spaced frames from the track, sum-pooled and $L2$ normalized. The final descriptor is therefore 4096 dimensional.

5.5.2 Supervisory information for prior methods

We describe here, briefly, how the supervisory information (e.g. the positive/negative bags) is generated from aligned transcripts in prior work [24, 37, 47, 78, 137, 161], and our implementation of it. We do this so that we can compare methods on the same data (tracks and features, and train/test splits) and with the same classifier. In the past methods have often been evaluated on different datasets with different features and different classifiers, so that it has not been possible to know which was superior.

Method (A). Everingham *et al.* [47] and [137]: Here a standard multi-class SVM formulation is used, which requires training samples with a unique class label. Tracks with a unique label are obtained by only selecting those speaking tracks where exactly one speaker name appears in the temporally overlapping aligned transcripts and subtitles. Such tracks generated a training sample which is labeled as positive for the speaking character and negative for all the others. Note that although this results in strongly supervised training material, it utilizes only a small part of the available information as non-speaking tracks are completely ignored in the supervisory information.

Method (B). Wohlhart *et al.* [161] and [78] : This uses a MIL approach for learning. Positive and negative bags are constructed as in Method (A) above, and then further negative in-

formation is added as follows: (i) make a bag from every non-speaking track and label it as negative for all characters appearing in temporally overlapping subtitles, (ii) make a bag from every track that overlaps temporally with positive bags and label it as negative to corresponding characters. Furthermore, a bag is made out of all face tracks that appear in one scene (a *scene* bag). The scene bag is labeled as positive for all characters that speak at least once during the scene according to subtitles and negative for all the others.

This method obtains more training data than (A), but the approach becomes heavily dependent on the speaker detection since every misclassified speaking track generates an incorrect negative sample (e.g. t_2 in Fig. 5.2). Also, the scene bags are often large and do not provide much additional supervisory information. In contrast, our approach is highly tolerant to speaker detection errors (and indeed works even without speaker detection) and we utilise also non-speaking tracks without overlapping subtitles (e.g. t_3 in Fig. 5.2).

Method (C). Bojanowski *et al.* **[24]:** Positive bags are formed by (i) assigning all speakers names from the aligned transcript to shots, and (ii) making one bag for each speaker name in each shot by collecting all face tracks corresponding to that shot and its neighboring shots (a *shot* bag). This bag formation strategy succeeds in gathering many more positive training samples than (A) and (B), but there are no negative bags.

Method (D). Cour *et al.* **[37]:** This method formulates the character naming task as an ambiguous label learning problem where the training material consists of face tracks associated with multiple character labels (i.e. a *track* bag, that contains a single track and one of the multiple labels is correct, rather than shot or scene bags with multiple tracks but only one label). These labels are obtained by assigning all speaker names that appear in a scene to all tracks in the same scene. Only scenes containing up to three speaking characters are included. Additionally, if the face track is detected to be speaking then the labels are restricted to those obtained from the overlapping subtitles. There is an assumption that all characters appearing in the video are named in the transcript. In our implementation we form "bags" for each face track using temporally overlapping subtitles (if any exist) and assign all corresponding

Dataset	Episodes					
	E1	E2	E3	E4	E5	E6
BBT	622 (8)	565(2)	613 (87)	581 (41)	558 (82)	820 (195)
Buffy	592 (2)	756 (3)	822 (9)	652 (32)	604 (31)	
Scrubs	608 (104)	518 (101)	449 (68)	430 (55)	454 (49)	
Casablanca	1278 (404)					

Table 5.1: ***Dataset Statistics:*** *The total number of tracks in each video (The number of background character tracks shown in brackets). Note that Casablanca has a high proportion of background characters.*

character names to this track. In addition, we make similar scene bags to Method (B), but use only those that contain up to three characters.

5.6 Datasets

The method is entirely evaluated on publicly available benchmark datasets. We evaluate on two television series, “Buffy" and “Big Bang Theory", and on the feature film “Casablanca". Additionally, we use 6 episodes of the TV-series “Scrubs" for parameter tuning and training of the background classifier. For all of these datasets, we make use of face tracks and ground truth annotations provided by the authors of the original publications. Shot boundary detection and subtitle processing was done using methods described in [140]. For “Scrubs", face detection was done using the head detector described in [97], while face tracking, facial landmark detection and speaker detection were done using methods similar to that of [24, 137]. Alignment of subtitles and transcript are obtained using the method of [48].

Buffy: This dataset, based on the popular supernatural drama “Buffy – The Vampire Slayer" was first introduced in [47]. This was later extended in [137] to include face tracks using both frontal and profile face tracks. We used the first 5 episodes from Season 5 of the series. It has a good number of primary characters thus providing reliable training data. Also, due to the nature of the show, this dataset has a large variation in lighting conditions. We use face tracks obtained from the author’s webpage [137] (with both frontal and profile detections) for fair comparison. Speaker identification output is also provided with the dataset. Supervisory

information was obtained by aligning subtitles with transcripts available from fan website.

Big Bang Theory: This dataset consists of episodes of the popular American Sitcom “Big Bang Theory". It was first used for person labeling in [151]. It consists of 6 episodes from Season 1 of the series. This is a more recent series than Buffy or Scrubs and, consequently, the picture quality of this series is much better. However, this series has only a limited number of principal characters and they appear in very restrictive environments. Even the clothing styles of the characters are very distinctive making it one of the easiest datasets to work on. We use the face tracks obtained from the website of the authors [151] and perform textual processing ourselves.

Casablanca: This movie was used in [24] for joint face and action labeling. It is a black and white film, a feature which is different from the other datasets used here. We use data entirely obtained from author’s website [24] for the evaluation.

Scrubs: This dataset, formed of 6 episodes from Season 1 of the series “Scrubs" is introduced by us and is used solely for parameter tuning. Its characteristics are very similar to that of the Buffy dataset. One of the key differences from Buffy is the number of background characters appearing in this series. Since the storyline is focused on the life of employees of a hospital, there are many patients and staff appearing in the background. This provides us with the necessary data for tracking background characters and learning their track classifier.

5.7 Experiments

5.7.1 Results

Effect of bag formation: The bag formation strategy has a significant effect on the resulting classification performance. In Sect. 5.2 we described our approach, and in Sect. 5.5.2 compared this to several previously proposed methods. In this experiment, we assess all these techniques using the same tracks and features from the Buffy dataset. The corresponding results, in Ta-

Dataset	Episodes					
	E1	E2	E3	E4	E5	E6
BBT	622 (8)	565(2)	613 (87)	581 (41)	558 (82)	820 (195)
Buffy	592 (2)	756 (3)	822 (9)	652 (32)	604 (31)	
Scrubs	608 (104)	518 (101)	449 (68)	430 (55)	454 (49)	
Casablanca	1278 (404)					

Table 5.2: ***Dataset Statistics:*** *The total number of tracks in each video (The number of background character tracks shown in brackets). Note that Casablanca has a high proportion of background characters.*

ble 5.3, clearly indicate that our bag formation strategy obtains the best overall performance. It is also notable that our results did not change much even if the speaker detection was not applied. Note that the results in Table 5.3 are not comparable with those in [161] since they used older and smaller dataset from [47] that contains only frontal face detections.

To enable further analysis, we calculated the bag characteristics for the first episode. In Table 5.4, we report the total number of positive bags, their average size, and the proportion that actually contains the corresponding character.

The approach of Sivic *et al.* [137] (row 1) generates the bags using the speaking character with unique speaker label. This strategy results in bags that contain exactly one sample, but it obtains a very limited number of them. Wohlhart *et al.* [161] (row 2) enhanced the previous method by adding scene bags. This increases the number of samples and their accuracy, but at the cost of substantially enlarging the average bag size (and the AP decreases). However, their strategy clearly improved the negative bags by almost tripling the number of tracks without compromising their accuracy. Bojanowski *et al.* [24] (row 3) generate the bags by considering a set of shots around each subtitle. This strategy demonstrates a clear improvement in the number of positive bags, without exploding the average bag size or losing their accuracy. This approach was only intended for generating positive bags.

Our approach constructs the bags using tracks around each subtitle time frame. In this way we obtain slightly fewer bags than [24], but the average bag size drops from 5.1 to 2.1, and further to 1.4 face tracks if the speaker detection is applied. This is a substantial improvement in reducing the ambiguity of the supervision. Moreover, our approach results in a considerable

Method	Spk	Buffy						
		1	2	3	4	5	Mean	Median
[137]	✓	0.85	0.63	0.65	0.77	0.78	0.74	0.77
[137]+Our neg	✓	0.84	0.76	0.73	0.85	0.79	0.79	0.79
[161]	✓	0.79	0.53	0.52	0.68	0.78	0.66	0.68
[24]	-	0.88	0.71	0.65	0.82	0.81	0.77	0.81
[37]	✓	0.83	0.61	0.75	0.79	0.78	0.75	0.78
Ours	✓	**0.94**	**0.86**	**0.87**	**0.93**	**0.93**	**0.91**	**0.93**
Ours	-	0.94	0.85	0.82	0.91	0.89	0.88	0.89

Table 5.3: ***Comparison of different bag formation strategies:*** *The average precision values obtained using different bag formation strategies, on the same set of tracks and with the same Fisher Vector face features in each case. The Spk column indicates if speaker detection is used. "Our neg" refers to using our negative bags with [137]. These results are computed using our implementations of the corresponding methods.*

Method	Spk	Positive Bags			Negative Bags		AP
		Num	Size	Corr.	Num smpl	Corr.	
[137]	✓	58	1.0	87.9%	522	98.7%	0.85
[161]	✓	90	53.6	92.2%	1928	94.5%	0.79
[24]	-	673	5.1	90.6%	-	-	0.88
Ours	-	581	2.1	86.9%	4157	97.4%	0.94
Ours	✓	581	1.4	79.0%	4426	97.3%	0.94

Table 5.4: ***Evaluation of bag properties with different formation strategies:*** *For the positive bags, we show the total number of bags generated over all characters, the average size of an individual bag, and the proportion of bags that actually contain the indicated character. For negative bags, we report the total number of tracks in all negative bags (in this case the number of bags is irrelevant) and the proportion of those tracks that do not depict the corresponding target character. These results are using Fisher Vector face features, and correspond to the first episode of the Buffy dataset. The Spk column indicates if speaker detection is used.*

increase in the number of negative samples with almost no loss in their accuracy. For completeness, we also tested if our negative bags can benefit the baseline method [137]. The results in Table 5.3 indicate 5% improvement in the mean AP.

Effect of initialization on learning algorithm: As discussed in Sect. 5.4, the alternating minimization of our learning algorithm requires an initialization. This can be provided in the form of initial classifiers $\{w_y, b_y\}$ or the best scoring track indices $s(I)$. Such initialization can be obtained using the method of [137], [37], [24], or by randomly picking the initial selector variable $s(I)$ for each positive bag.

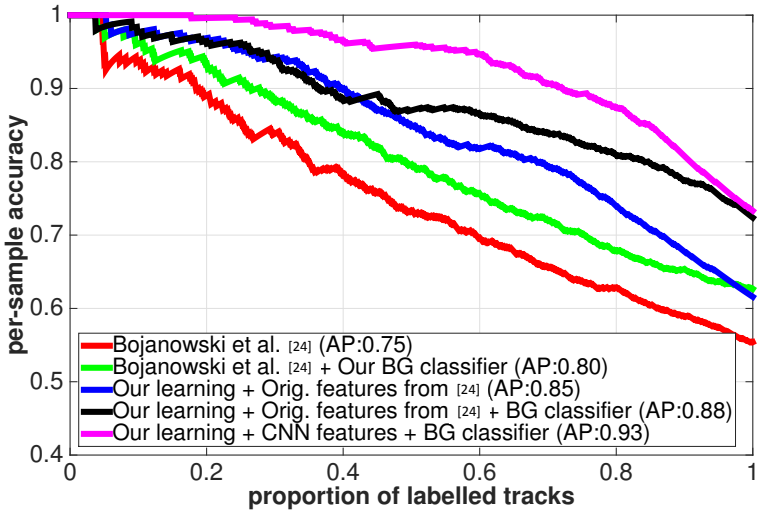


Figure 5.6: *Results of the automatic character naming using the "Casablanca" dataset provided by Bojanowski* et al. *[24]. In addition to the baseline curve of [24] the other curves are: (i) baseline [24] with our explicit background character modelling; (ii) using features of [24] with our learning framework and bag formation; (iii) adding explicit modelling of background characters, and (iv) changing the features to our ConvNet (CNN) based face descriptors.*

In order to demonstrate the robustness to the initialization, we evaluated our algorithm using the first five episodes, one at a time, of the Buffy dataset with all four different initialization methods mentioned above. This resulted in mean average precision 0.9 for initializing using [24], and 0.89 for all other initializations. It is notable that the random initialization resulted in equal performance compared to the other methods. This is probably partially due to the tight bags obtained using our approach.

Effect of algorithm components: In Sections 5.2–5.4, we describe different components of our algorithm. Fig. 5.6 and Table 5.5 show the contributions of each of these components. We compare our results with the strong baseline of [24] on the "Casablanca" dataset. By changing the learning algorithm and bag formation, we demonstrate a considerable boost in performance. This is largely due to the use of negative bags. Furthermore, by introducing the new approach for handling the background characters, we improve our results by an additional 3% and finally, by changing the track features, we obtain further 4% (with Fisher Vectors) or 5% (with ConvNet) improvement, taking the result up to 0.93 (compared to 0.75 in [24]). The relatively small performance difference between the feature representations, is most likely due

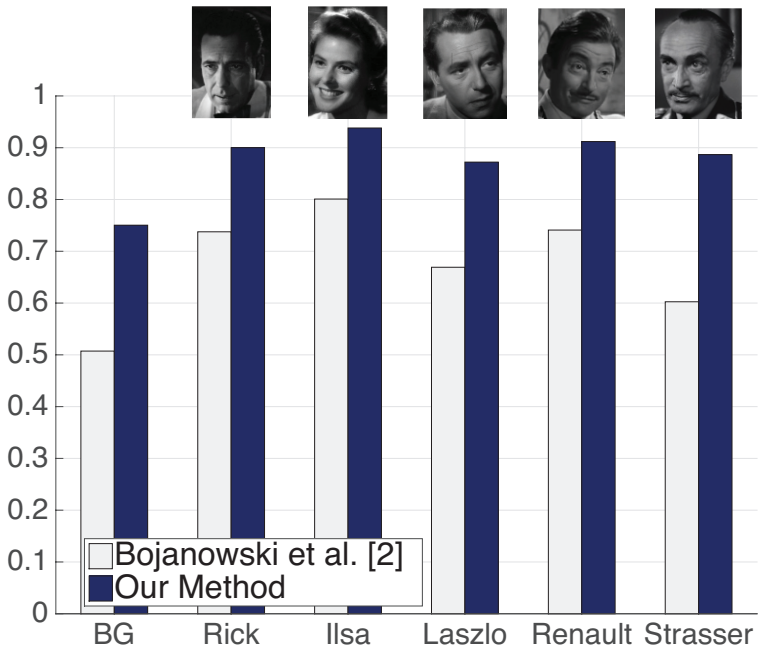


Figure 5.7: *Per character AP results on Casablanca using our best method and [24]. Characters are ordered according to their frequency of occurrence. Note the significant improvement for all categories, including "BG" (background).*

to fact that Casablanca is a black and white movie and the ConvNet representation was trained using color images. The benefit of the ConvNet features is clearly visible in Table 5.6, which shows our results for two TV-series. Additionally, we show that the proposed background character classification also improves the results of the baseline method [24].

5.7.2 Comparison with the state of the art

In Table 5.6, we compare our approach to the previously known state of the art methods on two different TV datasets. Our method using linear SVMs clearly outperforms the complex 162 Non-linear MKL method of [137] on the Buffy dataset. While on the Big Bang Theory (BBT) dataset we obtain 0.96 mean AP value. The baseline [151] reports the results in terms of accuracy and they train using manually labeled examples which are overlapping with the test data. Therefore our results are not directly comparable to theirs.

In Table 5.7, we compare our method on the "Casablanca" dataset. Here we make use of our best working setting from the components described earlier and compare against results published in [24]. Our method also significantly outperforms all other methods on this

BG-classifier	Learning Algorithm	Bags	Features	AP
-	A	A	A	0.75
✓	A	A	A	0.80
-	O	O	A	0.85
✓	O	O	A	0.88
✓	O	O	O-FV	0.92
✓	O	O	O-CNN	**0.93**

Table 5.5: ***Contributions of the different components of the algorithm on Casablanca Dataset*** *The average precision results for the Casablanca dataset using different versions of our method and the baseline [24] with and without our background character modeling. The columns correspond to background character classifier (BG-classifier), learning algorithm, bags, and features, indicating the source of these components.* ***A*** *denotes that these were used as available from authors of [24] while* ***O*** *indicates the use of our described method for obtaining them.* ***O-FV*** *refers to the Fisher Vector features and* ***O-CNN*** *to the ConvNet features.*

dataset. Fig. 5.7 illustrates the results per character for our method and the best performing baseline [24].

5.7.3 In the raw experiments

On TV data, we introduce a new measure of evaluating performance of the algorithms. Here we train our method using all other episodes of a series and test it using one episode disjoint from the training set. This is different from the usual practice of training and testing on the same episode. We evaluate in this way to assess the strength of the learned classifiers and to check if they are overfitting to the training data. As can be seen from the second last column of Table 5.6, our method indeed generalizes well, sometimes exceeding the performance achieved by training and testing on an episode – presumably because more training data is available.

For comparison, we also performed the same experiment using the ground truth character labels in the training. The results are shown in the last two column of Table 5.6, and it is evident that for some episodes the raw results match the ground truth training, which is quite impressive. For other episodes there is room for improvement which requires further investigation.

Data	Episode	[137]	[151] Acc	Our FV	Our CNN	Raw Our CNN	Raw GT CNN
Buffy	1	0.90	-	0.94	**0.99**	0.92	0.98
	2	0.83	-	0.86	**0.90**	0.95	0.96
	3	0.70	-	0.87	**0.94**	0.91	0.98
	4	0.86	-	0.93	**0.96**	0.92	0.97
	5	0.85	-	0.93	**0.97**	0.93	0.96
	Avg	0.83	-	0.91	**0.95**	0.93	0.97
	Med	0.85	-	0.93	**0.96**	0.92	0.97
BBT	1	-	0.89	0.98	**0.98**	1.00	1.00
	2	-	0.85	0.98	**0.99**	0.99	0.99
	3	-	0.80	0.94	**0.95**	0.96	1.00
	4	-	0.87	0.95	**0.96**	0.92	0.98
	5	-	0.73	0.95	**0.95**	0.91	0.99
	6	-	0.84	0.89	**0.92**	0.87	0.97
	Avg	-	0.83	0.95	**0.96**	0.94	0.99
	Med	-	0.85	0.95	**0.95**	0.94	0.99

Table 5.6: ***Comparison with state of the art (TV Series):*** *The average precision (AP) values for each of the tested episode in the Buffy and Big Bang Theory datasets. FV and CNN indicate if we are using Fisher Vector or ConvNet features. In addition, the two right most columns show the average precision values obtained in the "raw" experiment. For these, each row corresponds to the case where the indicated episode is used as a test data and all others for training. The baseline results are obtained from the corresponding original paper. Note that, results in [151] are provided in terms of accuracy, not AP.*

5.8 Summary

In this chapter, we have shown that stronger supervision can be obtained from an aligned transcript than that of previous methods. Furthermore, we have shown that explicit modeling and classification of background characters, also leads to a substantial improvement in classification of the principal characters.

Sr. No.	Dataset	Method	AP
1	Casablanca	**Sivic** *et al.* **[137] results by [24]**	0.63
2		**Cour** *et al.* **[37] results by [24]**	0.63
3		**Bojanowski** *et al.* **[24]**	0.75
4		**Our Method**	**0.93**

Table 5.7: ***Comparison with state of the art (Movies):*** *The average precision values for different methods on the Casablanca dataset.*

Chapter 6: Domain transfer: Recognising faces in paintings

Is there a painting of you out there? Probably not. But there may be one which looks just like you, as one man found out to his astonishment [1]. This raises the question of how to find such a painting (in a very large corpus) given a photo of a person's face. Of course, the extent to which a person in a photo resembles a different person in a painting is subjective, and very difficult to quantify. So, instead, we consider the question: given *photographs* of a person, can we retrieve *paintings* of that same person (in a large corpus)? The advantage of this question is that it is quantifiable, and so we can define a loss function and use tools from machine learning to learn a model that addresses this task. Given this model, we should then be able to find different, but similar looking people in paintings, starting from a photo.

One might be skeptical over whether retrieving paintings of a person starting from a photo is achievable. Photographs and paintings have very different low level statistics. To make matters worse, painted portraits are prone to large variations in style *e.g.* politicians are often highly caricatured and Hollywood icons of the past frequently get the Andy Warhol treatment and are transformed into pop art. This problem is essentially one of domain adaptation [44, 82, 118] from faces in photos to those in paintings; the challenge is to learn how to overcome both the low-level and stylistic differences.

To investigate how successfully we can use face photos to retrieve paintings, we require a large corpus for which there are both photos and paintings of the same person. To this end we use photos of celebrities and public figures to retrieve paintings from two distinct datasets: (i) paintings from the National Portrait Gallery, which are largely photo-realistic in nature, and (ii) paintings produced by the public crawled from the DeviantArt website [3], which are much more varied in style. The contrast between these datasets allows us to observe what effect

large variations in style have on retrieval. Fig. 6.1 shows samples from the datasets, which are described in more detail in Sect. 6.2.

Figure 6.1: *Top row – Paintings from the National Portrait Gallery: (a) Helen Mirren, (b) Harold Wilson, (c) Alan Bennett, (d) Joan Collins, (e) Dylan Thomas, (f) Anna Wintour, (g) Elton John, (h) Marco Pierre-White. Bottom row – Paintings crawled from DeviantArt: (i) Alan Rickman, (j) Cara Delevingne, (k) Cameron Diaz, (l) Michael Caine, (m) Alice Cooper, (n) Karl Pilkington, (o) John Lennon, (p) Lady Gaga, (q) Uma Thurman.*

We compare using shallow (Chapter 3) vs. deep (Chapter 4) features for this domain adaptation problem. Furthermore, we study whether the retrieval performance achieved with the raw features can be improved upon, by learning either (i) a linear projection on the features using discriminative dimensionality reduction (embedding learning), or (ii) face-specific classifiers. The rest of the chapter is organised as follows, Sect. 6.1 describes the learning methods, Sect. 6.3 outlines the implementation details, and Sect. 6.4 assesses the performance. Sect. 6.5 considers the inverse problem of retrieving photos of a person given their painting. Lastly, in Sect. 6.6, to return to our original question, we query a large dataset of oil paintings with photos of famous faces to find out if they have any previously unknown doppelgängers. This is further combined with attribute classifiers to retrieve paintings with specific facial attributes such as 'frowning'.

6.1 Learning to retrieve paintings using photos

In this section, we look at using photos to retrieve paintings. Assume we have a dataset $\mathcal{D}$ containing paintings of many different people where each painting is represented by a feature vector, y_j. Given a person, the dataset $\mathcal{D}$ is queried using n photos of that person, represented by feature vectors $x_1, x_2, ..., x_n$.

Three methods are considered: (i) using L2 distance on the pre-trained face features, (ii) learning an embedding, or (iii) learning classifiers. Both (ii) and (iii) utilize a training set of photos and paintings to learn how to transfer statistics between the two. The details of the features and how the learning methods are implemented are given in Sect. 6.3.

6.1.1 L2 Distance

For a given person, each painting in $\mathcal{D}$ is scored according to the mean Euclidean distance between its feature and those of the photos used to query. More formally, given photos $x_1, x_2, ..., x_n$, the score for each painting y_j is given by $\frac{1}{n}\sum_{i=1}^{n}||x_i - y_j||_2^2$. The paintings are ranked according to this score, from lowest to highest.

6.1.2 Embedding Learning

Here we learn a discriminative linear projection embedding W such that the L2 distance between projected features of a photo and painting, given by $||Wx - Wy||_2^2$, is small if the photo and painting are of the same person and larger by a margin if they are not(Sect. 4.2.2). There are three reasons for learning this embedding: firstly, it removes redundancy in the feature vectors, allowing them to become smaller, thus more suitable for large-scale retrieval; secondly, it tailors the features to specifically distinguish between faces, which would otherwise be lacking in the case of Fisher Vectors; thirdly, it specifically learns embeddings for the domain adaptation between photos and paintings.

The embedding is learnt using ranking loss on triplets in the similar spirit as described in Sect. 4.2.2: given a photo of a person x, a painting of the same person y_+ and a painting of a different person y_-, the projected distance between x and y_+ should be less than that between x and y_- by some margin:

$$||Wx - Wy_+||_2^2 + \alpha < ||Wx - Wy_-||_2^2 \tag{6.1}$$

Given sets of triplets, W can be learnt by optimizing the following cost function which incor-

porates the constraint Eqn. 6.1 softly in a hinge-loss:

$$\underset{W}{\operatorname{argmin}} \sum_{triplets} \max[0, \alpha - (||Wx - Wy_-||_2^2 - ||Wx - Wy_+||_2^2)] \tag{6.2}$$

This optimization is carried out using stochastic gradient descent: at each iteration t a triplet (x, y_+, y_-) is considered, and if the constraint ((6.1)) is violated, W_t is updated by subtracting the sub-gradient, as:

$$W_{t+1} = W_t - \gamma W_t(x - y_+)(x - y_+)^T + \gamma W_t(x - y_-)(x - y_-)^T \tag{6.3}$$

where γ is the learning rate. For retrieval, all features are projected by W before L2 distance is calculated, and then paintings are ranked on the mean distance, as above in Sect. 6.1.1.

6.1.3 Learning Classifiers

Instead of considering distances, it is possible to learn classifiers using the query photos of each person. As we query with a small number of photos, this is very similar to an Exemplar SVM formulation [96]. Given photos for a person, a linear SVM is learnt that discriminates these query photos from both paintings and photos not containing that person.

6.2 Obtaining datasets

As described in the beginning of this chapter, retrieval is performed on two distinct datasets containing portraits of people with known identities. Photos of these people are required to query these datasets. In addition to this, the learning methods of Sect. 6.1 require a training set of both photos and paintings. In this section we describe how the images are sourced (Sect. 6.2.1), and then how these are used to form the required datasets (Sect. 6.2.2). A summary of these datasets is provided in Table 6.1.

Dataset	Contents	No. People	Total Images
DEVret	1088 known paintings, 2000 distractor paintings	1,088	3,088
DEVquery	1088 sets of 5 photos to query DEVret	1,088	5,440
NPGret	188 known paintings, 2000 distractor paintings	188	2,188
NPGquery	188 sets of 5 photos to query NPGret	188	940
Train	248,000 photos and 9,000 paintings for learning	496	257,000

Table 6.1: *The statistics for the datasets used in this paper. 'No. People' refers to the number of known identities among the people present in the dataset. The datasets are described in Sect. 6.2.2.*

6.2.1 Image Sources

DeviantArt. The website DeviantArt [3] showcases art produced by the public, sorted by various categories (e.g. photography, traditional art, manga, cartoons). Among this work there are many portraits of well known figures, particularly in popular culture. To obtain these portraits, we used a compiled list of thousands of famous men and women (people who appeared frequently on IMDB [5]) from Chapter 6 and crawled DeviantArt using their names as queries. The paintings obtained from DeviantArt are highly prone to variation. Some have been painted and others sketched, many are caricaturistic in nature and lack a photo-realistic quality. The extent of this variety is made clear in the sample paintings provided in Fig. 6.1.

National Portrait Gallery. Many of the paintings in London's National Portrait Gallery are publicly available as a subset of the 'Your Paintings' [2] dataset. Some example portraits are shown in Fig. 6.1. The portraits are typically quite photo-realistic in nature and are predominantly painted using oil.

6.2.2 Dataset organization

We require three types of dataset: (i) query sets, that contain multiple photos of each person, (ii) retrieval sets, that contain paintings of the same persons, and (iii) a training set containing both photos and paintings of people, where the matrix W and classifiers will be learnt from. The query set is used to issue queries for each person and the performance is measured on the retrieval set. There should be no people in common between the training set and other

remaining sets. Furthermore, none of the retrieval identities are used to learn the network that produces CNN features.

Retrieval Set – DEVret. A single painting for each of 1,088 people obtained from DeviantArt form a retrieval set. To make the retrieval task more difficult this set is supplemented with 2,000 random portraits from DeviantArt that do not contain any of the people's names in the title.

Retrieval Set – NPGret. A painting for each of 188 people in the National Portrait Gallery is taken to form a retrieval set. The reason this number is not higher is because many people depicted in the National Portrait Gallery lived before the age of photography. These 188 portraits are supplemented with 2,000 random portraits from 'Your Paintings'.

Training Set – TRAIN. The training set consists of multiple paintings per person for each of 496 people from DeviantArt coupled with 500 photos per person from Google Image Search. Some examples of photo-painting pairs with the same identity are given in Fig. 6.2. There are 9,000 paintings in total. The distribution of paintings per person is a long tail, this is illustrated in Table 6.2, along with the names of the most prevalent people.

Query Sets. The sets of photos used for querying the retrieval sets, denoted as DEVquery and NPGquery each contain five photos from Google Image Search per person in their respective sets. The photos have been manually filtered to ensure that they have the correct identities.

Figure 6.2: *Photo-painting pairs that share an identity in the* **TRAIN** *set. In each case the photo is on the left and the painting is on the right. (a) Tim Minchin, (b) Carey Mulligan, (c) Zooey Deschanel, (d) Dita Von Teese, (e) Hugh Grant.*

Name	No. Paintings	Name	No. Paintings
Jared Leto	220	Taylor Swift	184
Clint Eastwood	167	Katy Perry	183
Robert Pattinson	144	Megan Fox	122
Tom Hiddleston	122	Dita Von Teese	89
David Tennant	103	Kate Moss	89
Billie Joe Armstrong	96	Keira Knightly	83
Ian Somerhalder	95	Adriana Lima	76

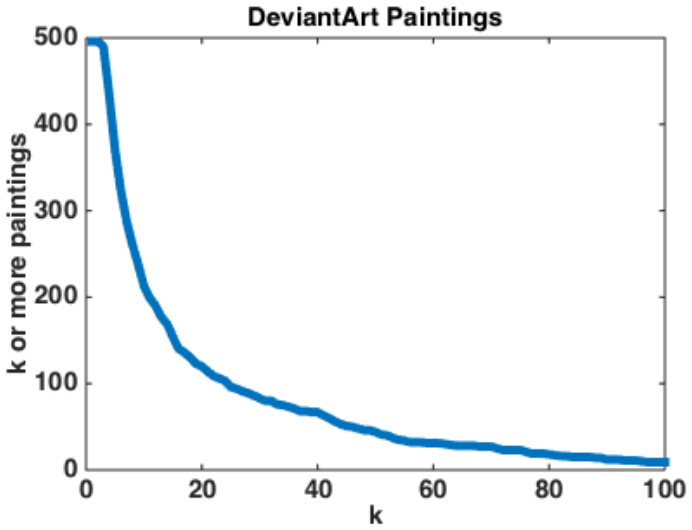


Table 6.2: *Top: the table shows the men and women for which there are the most paintings in* **TRAIN**. *Bottom: A plot that shows the number of people for which there are k or more paintings.*

6.3 Implementation Details

Here we describe in detail the feature representations used for the faces as well as the implementation of the methods of Sect. 6.1.

Face Detection. For a given image, face detections are obtained by the Cascaded Deformable Part Model [54, 55] based on the method of [98] (Sect. 2.1.1). The detection box is then expanded by 10% in each direction and cropped from the image. The cropped face is then used to compute either a Fisher Vector or CNN feature.

Fisher Vector Representation. This representation follows the approach described in Chapter 3. To generate these features the cropped face is first resized to 150×150 pixels and Root-

SIFT [12] features are extracted at multiple scales from each image. These are decorrelated and reduced using PCA to 64-D and augmented with the (x,y) co-ordinates of the extraction location. For each image, the mean and covariance of the distances between its features and each of the 512 centres of a pre-computed Gaussian Mixture Model are recorded and stacked resulting in a 67584-D Fisher Vector. Finally this vector is L2-normalised.

CNN Representation. This representation follows the approach described in Chapter 4. To obtain a feature vector, the cropped face is resized to 224×224 pixels before being passed into the VGG Face 16 network. The 4096-D output of the penultimate layer (the last fully-connected layer) is then extracted and L2-normalized.

Embedding Learning. A PCA projection to 128-D is learnt using the training data. This is used as the W to initialize the optimization. Triplets are either (i) generated at random offline, or (ii) semi-hard negative triplets [124] are formed online. In the latter case, at each iteration a positive photo-painting pair (x, y_+) is considered with each of n random negative paintings y_- as candidate triplets (we set $n = 100$). Only the candidate for which ($||Wx - Wy_-||_2^2 - ||Wx - Wy_+||_2^2$) has the lowest positive value is then used. The optimization is run for 1 million iterations.

Learning Classifiers. For each query, the photos are used as positive examples in an SVM. The negative examples are taken to be all the paintings and photos in the training data. The regularization parameter C is learnt on a held-out validation set as $C = 1$.

6.4 Experiments

In this section, the retrieval task is assessed on the two datasets. For each person in the query set (DEVquery or NPGquery), the photos of that person are used to rank all the paintings in the retrieval set (DEVret or NPGret) using a given method. The rank held by the correct painting (the one that has the same identity as the query photos) is recorded. Across all people queried, the recall at k for all k is recorded and averaged – this average recall is denoted as Re@k.

The Re@k for various k are given in Table 6.3 for different methods. The corresponding curves are shown in Fig. 6.3. A selection of successful retrievals are illustrated in Fig. 6.4.

Experiment	Dataset	Re@1	Re@5	Re@10	Re@50	Re@100	Re@1000
FV L2 distance	DEV	4.4	7.4	10.1	17.7	23.2	57.5
FV embedding learning (i)	DEV	5.9	13.8	18.3	37.0	46.4	91.5
FV embedding learning (ii)	DEV	7.4	16.3	21.8	40.6	49.4	92.4
FV Classifier	DEV	16.8	25.9	30.1	39.8	46.0	81.3
CNN L2 distance	DEV	26.0	42.2	47.3	63.3	71.4	93.7
FV L2 distance	NPG	4.3	13.3	17.6	25.5	33.0	72.3
FV embedding learning (i)	NPG	8.5	18.6	23.9	47.3	57.4	95.7
FV embedding learning (ii)	NPG	7.4	26.6	33.5	54.2	66.0	97.3
FV Classifier	NPG	15.6	24.5	28.7	42.0	49.4	87.2
CNN L2 distance	NPG	36.2	58.5	66.0	80.9	83.0	94.7

Table 6.3: *Percentage Re@k on the retrieval sets for assorted methods and features. embedding learning refers to Discriminative Dimensionality Reduction, (i) and (ii) refer to the methods of triplet selection given in Sect. 6.3.*

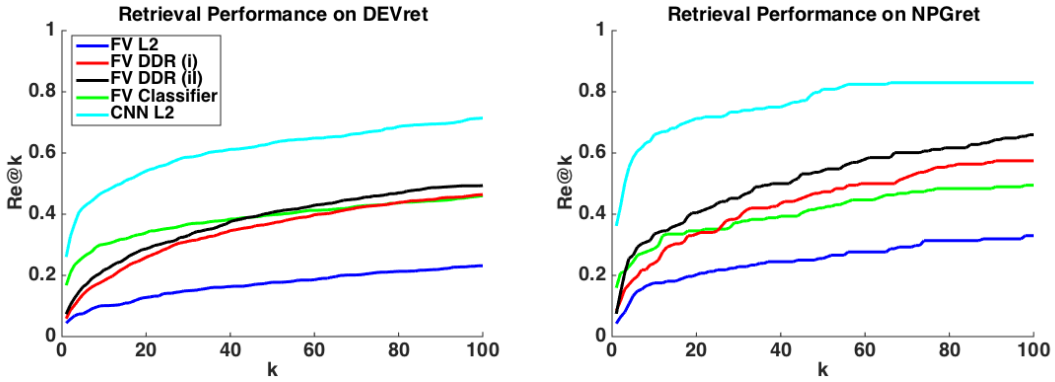


Figure 6.3: *Re@k vs. k plots for DEVret (left) and NPGret (right). The legend on the left plot also applies to the right plot.*

Fisher Vector Learning Results. Both embedding learning and classification boost the Re@k performance over raw L2 distances for a range of k. This shows that the domain adaptation learning is successful in overcoming the low level statistical and stylistic differences between the photos and paintings. For embedding learning, Re@k is generally higher for method (ii) (i.e. when semi-hard negative triplets are generated online) as it forces the learning to cope with the most difficult borderline cases, allowing it to distinguish between very similar looking people. Using a classifier (which can learn discriminatively what differentiates a particular identity

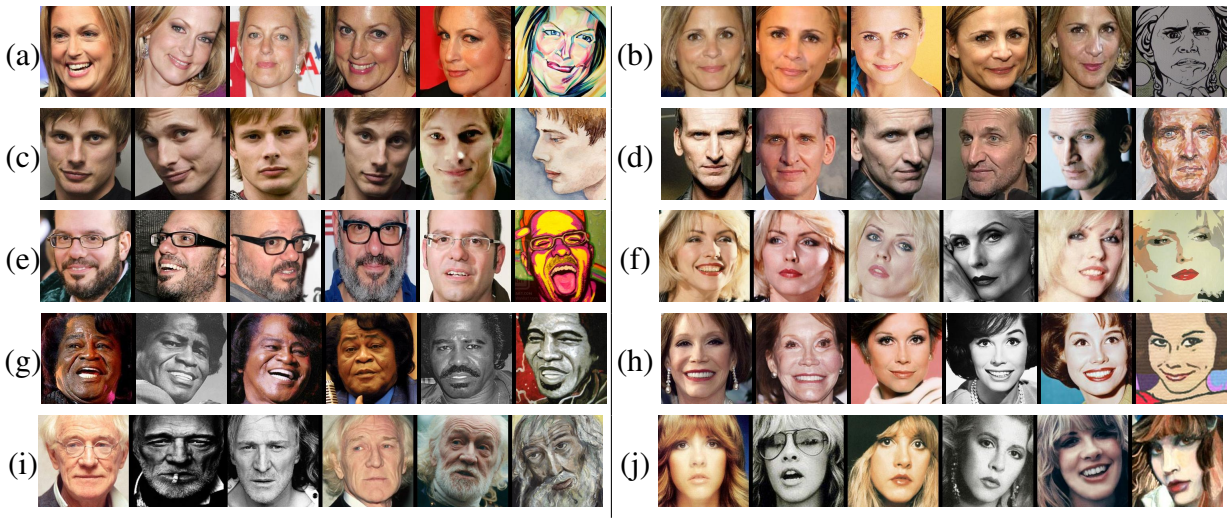


Figure 6.4: *Successful (Top 5) retrievals using a CNN representation. In each case, the five query photos are shown beside the top retrieved painting. (a) Alexandra Wentworth, (b) Amy Sedaris, (c) Bradley James, (d) Christopher Ecclestone, (e) David Cross, (f) Deborah Harry, (g) James Brown, (h) Mary Tyler Moore, (i) Richard Harris, (j) Stevie Nicks.*

from others) typically outperforms embedding learning for low k but is thereafter surpassed. Re@k on NPG is generally higher than that on DEV for all methods. This is probably because some DeviantArt-style paintings are highly abstract and difficult to retrieve. Interestingly, the embedding matrix performs very well for National Portrait Gallery retrieval, despite having been learnt on DeviantArt style paintings.

CNN Results. The first thing to note is that CNN results always exceed those of Fisher Vectors, even after learning. Interestingly, additional learning for CNN features has negligible effect on performance (this is why further CNN experiments are absent from Table 6.3). The network training has probably already captured the discriminative aspects of a person's face, remarkably to such an extent that the stylistic differences and low level statistics of paintings are of little consequence. The features also demonstrate invariance to pose: notice in Fig. 6.4(c) that the side-profile painting of Bradley James has been retrieved using front-profile images. In the case of the Deborah Harry (f) painting where much of the facial outline is missing, the discriminative eyes and lips have been picked out.

6.5 Retrieving Photos with Paintings

The focus of the majority of this chapter has been: starting with photos of a person, retrieve a painting of that person. Here, we instead try to retrieve photos starting with the paintings, to observe how reciprocal the adaptation problem is.

Evaluation. For each of the 1,088 people featured in DEVret paintings, 75 photos are crawled from Google Image Search. These are supplemented with distractor photos to form a retrieval set of 97,545 photos. Photos are retrieved from this set using each of the 1,088 DEVret paintings as a single query; photos are ranked using the L2 distance between CNN features of the photos and the painting. This proves to be highly successful and some example retrievals are given in Fig. 6.5 along with the Average Precisions (AP) of retrieval.

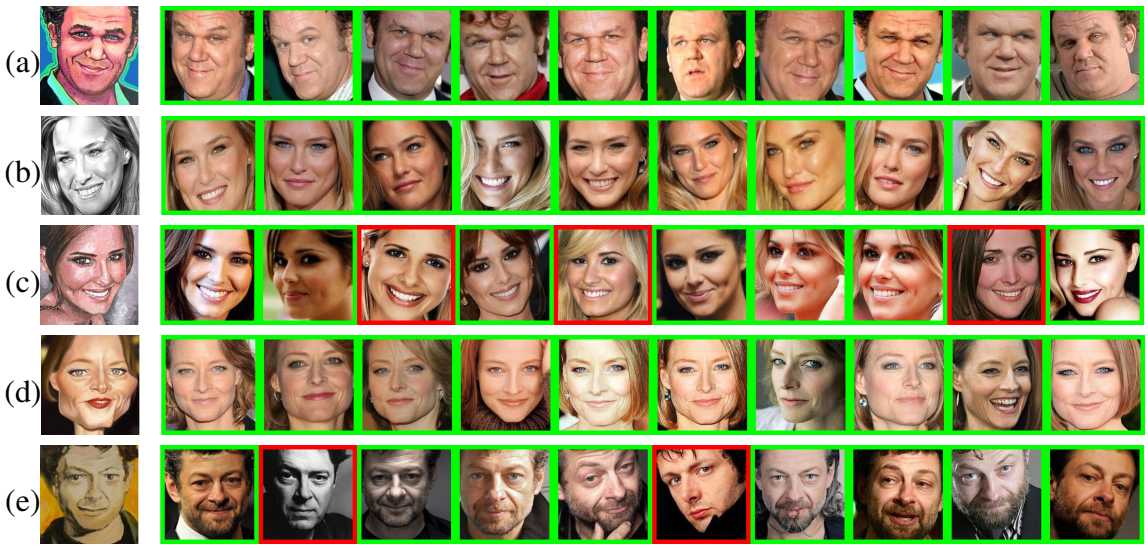


Figure 6.5: *Photo retrieval using a single painting. Each row from left to right shows the painting used for retrieval followed by the 10 highest ranked photos. Correct retrievals have a green border and incorrect retrievals a red one. (a) John C. Reilly AP: 0.90, (b) Bar Refaeli AP: 0.63, (c) Cheryl Fernandez-Versini AP: 0.42, (d) Jodie Foster AP: 0.56, (e) Andy Serkis AP: 0.47*

6.6 Finding Doppelgängers in Art

In this section, we return to our first goal: Given a photo of a person, we would like to retrieve a painting of a very similar looking person. To qualitatively evaluate this problem, we form a set of 40,000 paintings by applying a face detector to the entirety of the 'Your Paintings' [2]

dataset and filtering the paintings with the highest classifier scores. This set is queried with photos of famous people known **not** to be present among the paintings, using L2 distances between the CNN features. Some example retrievals are found in Fig. 6.6. Notice that the results are uncanny: the portraits and photos have very similar facial features.

Incorporating Attributes. Consider not just being able to find a similar looking painting, but one that also has a given attribute, such as 'frowning'. Motivated by this, we combine this retrieval with attribute classifiers, such that the painting retrieved y satisfies:

$$\underset{y}{\operatorname{argmin}} ||x - y||_2^2 - \lambda w_a y \tag{6.4}$$

where x is the query photo and w_a is an attribute classifier. λ adjusts the influence of the attribute classifier on retrieval. We demonstrate how retrieval changes with λ in Fig. 6.7: as λ increases, so does the extent of the attribute in the retrieved painting.

Implementation Details. Classifiers are learnt for 50 of the attributes listed in [84]: For a given attribute, photos are obtained by crawling Google Image Search with the attribute name as a query. The CNN features extracted from these photos are used as positive examples in a linear SVM with photos of the 49 other attributes as negatives to learn a classifier. These classifiers are then applied to the set of 40,000 paintings.

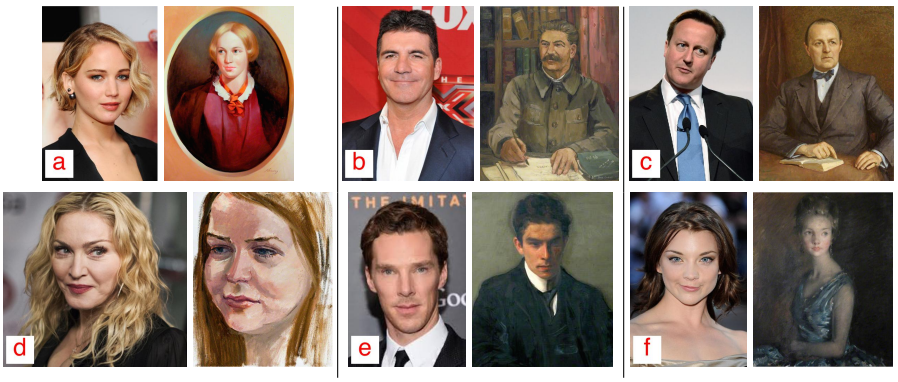


Figure 6.6: *Photos of famous people and their closest matching portrait from 'Your Paintings'. (a) Jennifer Lawrence, (b) Simon Cowell, (c) David Cameron, (d) Madonna, (e) Benedict Cumberbatch, (f) Natalie Dormer.*

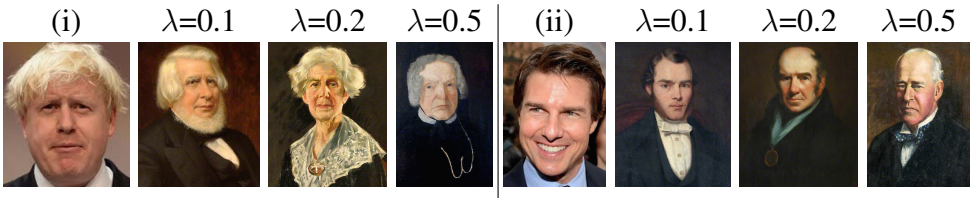


Figure 6.7: *Top retrieved paintings for (i) Boris Johnson and (ii) Tom Cruise as λ is increased for (i) an 'Old face' Classifier and (ii) a 'frowning' Classifier respectively.*

6.7 Summary

In this work, we have shown that it is possible to retrieve paintings of people starting with photos of the same person (and vice versa), and that for a Fisher Vector face representation, discriminative learning can significantly increase performance. We have further shown that CNN features produced from a network learnt entirely on photos are able generalize remarkably well to paintings of many different styles. Furthermore, the similarity between these features can be used to find photos and paintings of people that look eerily similar.

Web demo. The online demo based on techniques explained in this chapter is available at `http://zeus.robots.ox.ac.uk/facepainting/index`.

Chapter 7: On-the-fly retrieval of faces

This chapter explores learning on-the-fly starting from a text query for immediate retrieval from large scale video datasets – although the methods are equally applicable to image datasets. Imagine that you have a large corpus of videos, such as a TV station archive or a large archive of internet videos like "The Moving Image Archive", and you need to find shots containing a particular person (and the corpus, of course, lacks complete meta-data describing who is in each video). We describe here a system to meet this need. Our goal is to be able to search for anyone in the corpus based on their face [76, 113, 136, 169]. A second aspect of the objective is to be able to achieve results immediately, and this requires that learning of a category occurs 'on-the-fly' at search time, as well as the scalable and immediate search of large scale datasets. Putting the two together allows a user to start with a text query, learn a visual model for the specified category, and then search an unannotated dataset on its visual content with results retrieved within seconds.

Our approach is to use discriminative classification for building models of different people with the architecture of Fig. 7.1, including positive and negative training data and a linear SVM classifier. Given a textual query, positive training data is obtained "on-the-fly" from the web sources such as Google Image Search and Bing. Face detection representation and classifier training using this data is done in real time. The trained classifier is then applied to the video archives to produce the relevant ranking. The fundamental unit of retrieval is a face-track as compared to that of images in traditional retrieval applications.

To ensure real time performance of the system, most of the video archive processing is done in advance (off-line) while only the query specific tasks are carried out in real time (on-line). In the following section, we describe the on-the-fly search system in detail, explaining in

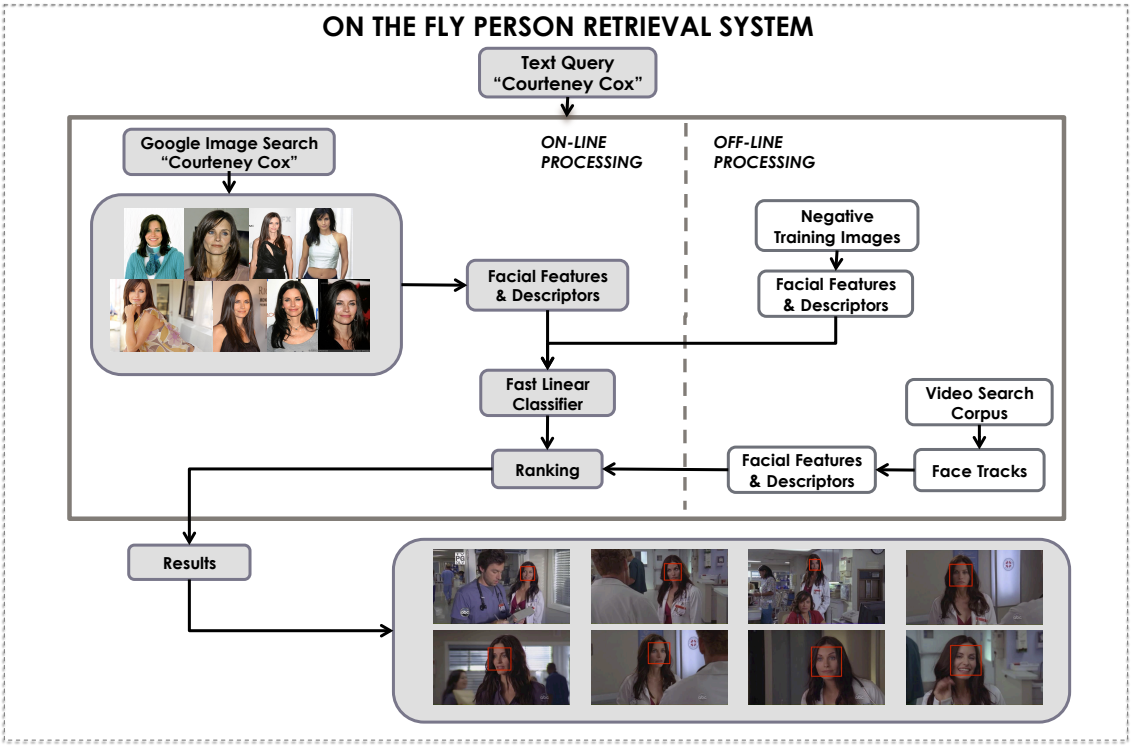


Figure 7.1: ***Architecture of the on-the-fly object category retrieval pipeline.*** *Positive training images are downloaded on-the-fly from Google Image search by the frontend component, and are then fed to the backend for processing. In the backend, the images are encoded and after a fixed timeout of τ seconds all encoded images are fed to a Linear SVM for batch training (along with a fixed pool of negative features pre-computed during the pre-processing phase). Finally, the linear SVM model $\mathbf{w}$ is applied to the pre-computed features of the target dataset, and the resulting classification scores sorted in descending order to produce the output ranking.*

particular how the design choices and parallel architecture allow the complete process from a text query to results to be carried out in matter of seconds for a corpus of millions of detected faces. Sect. 7.3 then gives a quantitative performance evaluation of the system.

7.1 Building an on-the-fly video face retrieval system

This section describes the architecture of an on-the-fly system. The key issues in building such a system are what to compute and store in advance vs. on-line, and the memory-speed trade-off. The on-the-fly system architecture is shown in Fig. 7.1. In order to achieve the on-the-fly person-specific training and retrieval it is necessary to carry out much of the processing in advance of a query. We describe the off-line steps next, followed by the processing that occurs at run time once a query is typed into the search window.

7.1.1 Off-line preprocessing

The off-line part of the system pre-computes non query-specific features in order for the run-time query specific operations to be fast. Non query-specific processing includes computing feature vectors for the entire video corpus as well as feature vectors for a set of negative training samples. The entire video collection is processed in three steps. In the first step, faces are detected in every frame of every video and linked together to form face tracks. In the second step, false tracks are removed and an exemplar face detection is selected to represent each face track. In the final step, feature vectors are computed for the exemplar face of each track to be used for ranking by the SVM classifier.

Video processing. We preprocess all videos in our target dataset with the objective of detecting all faces within a shot and associating all detections of the same person into contiguous face tracks.

Face detection and tracking A face-track is a temporal connection of detected faces of a single person. Faces are detected using the OpenCV frontal face detector and are linked together using KLT tracks as described in [48]. The facial landmark detector of [48] is used to

detect nine facial landmark points. These points are used in the face representation and also for selecting the best face to represent the track. Due to erroneous face detections there will be a number of false face-tracks generated. A significant proportion of these can be removed simply based on their length. Face tracks are also filtered out based on the score of the facial landmark detector. The remaining tracks are then each represented by combining temporally sampled detections from the track with selecting the 'best' detection based on the facial landmark detector score. This score is a measure of confidence of facial feature locations and proves to be useful for selecting suitable candidates. Table 7.1 gives statistics on the number of detections and face-tracks in the datasets used here. Face detection and tracking are discussed in more detail in Sect. 2.1.

Face representation. We check suitability of two feature representations for the large scale retrieval task. The first representation is a lightweight gradient magnitude based descriptor. This descriptor is a modification of the patch intensity based descriptor of Everingham *et al.* [48]. Similar to their method, the descriptor is computed by concatenation of image statistics around the facial landmarks. Specifically, the gradient magnitudes in a circular window around landmarks in 2D aligned face images are concatenated together to form a 4096 dimensional vector. This approach produces one feature vector per frame and therefore we investigate different strategies to represent face track which consider both memory consumption and time complexity of the inference. The other apporach is the Fisher Vector Track representation from Chapter 3. Following the procedure explained there, we compute GMM over the training data described below and use the track pooling strategy to obtain one descriptor per track.

Processing negative training data. Negative training face images are pre-downloaded from the Internet and are kept the same for all queries. We use a publicly available dataset on the web as the negative training set (Sect. 7.2). The assumption is that the negative data will not contain a significant number of faces from the particular person we wish to search for. The face detector, facial feature detector and appearance representation pipeline described above is applied to each of the negative images to produce a feature vector.

7.1.2 On-line processing.

This part performs all query specific tasks which include: querying and downloading images from the Internet; computing feature vectors from the downloaded images to provide positive training samples; training the classifier; and ranking and displaying the results.

Given a textual query, the corresponding top $K \sim 100$ images are retrieved from the Google Image Search. Since our search is restricted for identities, we enable the image filtering option on the Google Image Search and ask it to return images with "Face" filter applied to the results. The face detector, facial feature detector and appearance representation pipeline is then applied to each of the downloaded images to obtain a feature vector for every detected face. These features are then used as positive visual training data for our object category. Along with the fixed pool of pre-computed negative training data described before, these are used to train a linear SVM $\langle \mathbf{w}, \phi(I) \rangle$ by fitting $\mathbf{w}$ to the available training data by minimizing an objective function balancing a quadratic regularizer and the hinge-loss. The parameter C in the SVM is set to a constant value of $C = 10$ by cross validation on a hold out set.

Having learnt a classifier for our identity, we then apply it to our dataset features, computed in the pre-processing step. Following this, the output classification function is used as a ranking metric, and the images are presented to the user in descending order of these scores.

7.2 Datasets and evaluation protocol

We use three datasets for the qualitative evaluation of our approach, details of which are given below.

Buffy. This is formed of 22 episodes of season 5 of the 'Buffy the Vampire Slayer', and forms the core of our primary evaluation dataset. Different versions of this data have appeared in literature before [48, 137, 152] (Sect. 5.6). We use the data publicly available at [16] that contains face-tracks and associated ground truth for all 22 episodes. Ground truth annotations are provided for the tracks of the six primary characters in the series, namely: Buffy, Willow, Xander, Giles, Tara and Anya. The first ten episodes are used for training purposes and the

Dataset	Hours	Faces	Tracks
Buffy	16.5	1,212,471	21,053
TRECVID 2012 INS	188	479,004	13,171
TRECVID 2013 INS	435	5,141,166	149,225
BBC News	10,132	6M	2.1M

Table 7.1: Face retrieval dataset details. *Number of faces detected (Faces) and face-tracks (Tracks) in different datasets used for evaluation.*

remaining episodes are used for testing.

TRECVID 2012 INS. This dataset was introduced for the TRECVID instance search competition in 2012. It consists of videos downloaded from Flickr under the creative commons license. The face-tracks of this dataset are used for negative training data in the experiments.

TRECVID 2013 INS. This dataset was introduced as a part of TRECVID instance search competition in 2013. The dataset consists of about 214 episodes from the BBC television series 'EastEnders'. The face-tracks from the 78 episodes of this dataset are used as distractors in the experiments to make the task of recognition harder. Since the data is collected from a disjoint TV series, it makes it a perfect choice as a distractor for the the Buffy dataset.

BBC News. This is a video dataset provided by the BBC comprising of their news broadcasts. This covers all news programs broadcast over all BBC channels from 6 pm until midnight from 2007 to 2012. It consists of 10,132 hours of footage from 17,401 different programs, and is represented by 5,297,206 key frames.

For each character, the quality of the trained model is assessed using *Precision @ k*, i.e. the fraction of the top-k ranked results that are classified correctly, and also the *Average Precision*. Finally, we further evaluate performance qualitatively over the large scale BBC News dataset.

7.3 Experiments

Our objective is to assess the quality of the retrieved face tracks for a given character. There are two specific goals: first, to find the best representation of a face-track; second, to assess the

Character	Train Tracks	Test Tracks	Grad. Feat			FV
			LM	Max	Avg	
Buffy	2000	2179	0.960 (0.520)	0.909 (0.526)	0.970 (0.518)	0.970 (0.784)
Giles	504	630	0.960 (0.350)	1.000 (0.450)	0.818 (0.279)	1.000 (0.798)
Xander	795	841	0.990 (0.463)	0.990 (0.519)	0.960 (0.399)	1.000 (0.813)
Willow	720	1146	0.899 (0.311)	0.869 (0.361)	0.838 (0.255)	1.000 (0.748)
Tara	318	619	0.879 (0.273)	0.939 (0.342)	0.869 (0.230)	1.000 (0.697)
Anya	449	762	0.869 (0.251)	0.869 (0.285)	0.778 (0.204)	1.000 (0.715)
Mean	-	-	0.926 (0.361)	0.929 (0.414)	0.872 (0.314)	0.995 (0.759)

Table 7.2: Face retrieval experiments. *Prec @ 100 and Average Precision (in brackets) for different characters and experiments. Train and Test track statistics are from the Buffy Dataset. With the addition of distractors from the TV13INS data, the total number of test tracks is 124,761. All the performance figures are reported on this combined test set. The track representation experiments using Gradient Descriptors (Grad. Feats) show that selecting the maximum score (Max) amongst all detections from a track is the best strategy. Selecting one frame based on facial landmark score (LM) performs comparably in terms of Prec @ 100 while reducing memory and computational requirements.*

suitability of Google Images for training character specific models.

The training procedure is common to all the experiments described below. Given a character, a linear SVM classifier is trained using all tracks belonging to that character from the training data as positive examples while tracks of all other characters and all tracks from TRECVID 2012 INS are used as negative training examples. The trained classifier is used to rank tracks from the test split of the Buffy dataset combined with the whole of the TRECVID 2013 INS dataset.

We compare four methods of representing and scoring face-tracks. The first three are variants of the intensity gradient descriptor discussed above whilst the fourth uses the Fisher Vector face-track representation (Chapter 3), more specifically, the methods are:

LM. We select one representative frame of the track using the maximum facial landmark detector confidence score, and represent the track by the feature vector of the face in the selected frame. The advantage of this method is that there is just one SVM score computation required per track.

Max. Every face in the track is scored using the SVM, and the maximum score obtained is assigned to the track. Unlike the *LM* method, feature vectors for *all* faces of the track must be stored as they are required for classification.

Avg. A single vector representation is computed for the entire track by averaging over all face descriptors. The track is then scored by the SVM of the average vector (due to linearity, this is equivalent to averaging the scores over all faces). As with the *LM* method, only a single feature vector needs to be stored per track.

FV. Using the track representation techniques described in Chapter 3, a single Fisher Vector is computed for the whole track.

Character	**Ground Truth**		**Google**	
	Grad	FV	Grad	FV
Buffy	0.960	0.970	0.626	0.677
Giles	0.960	1.000	0.162	0.182

Table 7.3: Comparison of training data sources for face retrieval. *Per character Prec @ 100 for classifiers trained on different data sources. The images available for a character from Google vary in quality. For the primary character of the series 'Buffy', Google returns sufficient images to train a classifier well. For a secondary character, such as 'Giles', the performance reduction observed is due to the lower quality of training images available from Google.*

Table 7.2 shows the performance of each of these methods over the combined Buffy + TRECVID 2013 INS dataset. For the intensity gradient descriptor, it can be seen that taking the maximum score for all frames in a track (*Max*) performs best. However, the Fisher Vector representation outperforms all other methods, whilst only producing one feature per track in contrast to the *Max* method.

Table 7.3 compares performance of a classifier trained on ground truth data to the one trained using images obtained from Google. For the characters where Google can return both sufficient and accurate images to train a classifier, results are comparable to the ground truth classifier, whilst performance drops in cases where Google can not provide sufficient and accurate images for the character. We suggest alternative methods to tackle noisy web training data below in Sect. 7.6.

7.4 System Architecture

So far we have discussed the system design, training and performance. We now discuss the details of implementing such a system as a real-time web application. The architecture of the system can be broadly split into a frontend and a backend component.

Frontend Component. The frontend component is in charge of presenting the web-interface, managing requests and converting textual queries to visual training data by downloading images from Google Image search. It is written in Python.

Obtaining positive training images. The image downloader component is implemented in Python using co-routines to maximize the rate at which training images can be downloaded. Our target is to download between 20 and 150 training images, depending on the search modality. We do this by first requesting the first 300 results from Google, and then impose a timeout of τ_{image} $\sim$100ms on the download time of each image, which ensures that no undue time is wasted retrieving images from slow servers. A global timeout is also set between τ_{global} $\sim$1 and 5 seconds depending on how many training images are required. Features are then computed in real-time and in parallel over multiple CPU cores and stored in memory for training. We use between 80-100 images for training the classifiers.

Backend Components. This component manages the process of producing a ranked list of results given a query. This involves detecting faces, landmarks and features for the downloaded images, training a SVM classifier and ranking the database tracks. This component is written using C++ for speed and efficiency, as they are the most computationally expensive part of the system. Additionally, ranking uses a parallel approach with multiple computation threads over the target dataset tracks. We use the gradient features discussed above due to their low memory footprint.

Compression. Optionally, these already very compact codes can be compressed further using binary compression methods. We explore the use of *Product Quantization (PQ)* which has been

widely used as a compression method for image features [72, 119], and works by splitting the original feature into Q-dimensional sub-blocks, each of which is encoded using a separate vocabulary of cluster centres pre-learned from a training set. We explore compression using $Q = 4$-dimensional sub-blocks. This results in a further $16\times$ compression compared to original memory requirement.

7.5 Web-based Demo System

The demonstration of our proposed on-the-fly system is done on a news video dataset provided by the BBC. The details of the BBC News dataset, along with the memory requirements of each search modality are summarized in Table 7.4. Figure 7.3 shows example results of face retrieval on this dataset. In this case the training images are sourced from Google instead of from a curated dataset in the experiements. As can be seen, we succeed in retrieving faces of a specific person with varying expressions and illumination.

Dataset Details	No. Programs	17,401
	Hours	10,132
	Keyframes	5.3M
	Face Tracks	2.1M
Memory Req.	Feats.	32 GB
	Feats. (with PQ)	2 GB

Table 7.4: ***BBC News dataset statistics and memory requirements.*** *The BBC News dataset is sourced from all BBC News footage broadcast from 6 pm until midnight from 2007 to 2012, and isused for qualitative evaluation and the on-the-fly system. In the case of product quantized (PQ) figures, sub-quantizers 4 dimensions are used.*

7.6 Applications and extensions

In this section, we overview a number of applications and extensions of the system for multi-dimensional exploration of large scale archieves.

Facial attribute search. On-the-fly face classification can also be used for retrieving face-tracks with specific attributes such as a mustache, beard, glasses, gender etc, by simply using

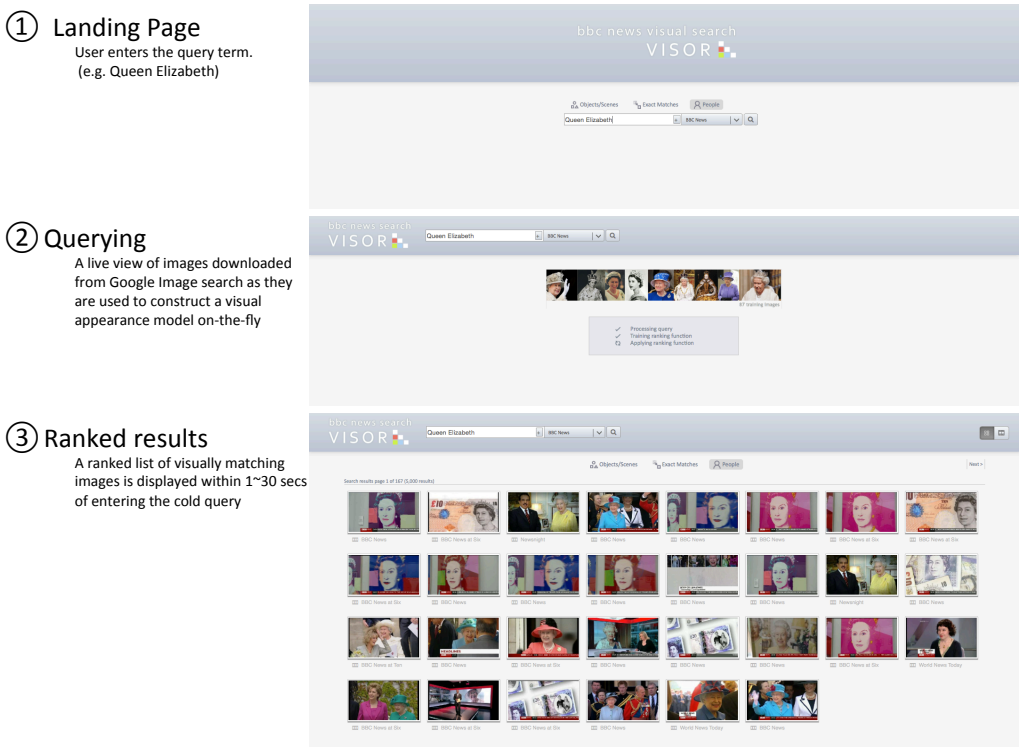


Figure 7.2: ***Web-based on-the-fly demo system.*** *(1) the user enters a text query term and selects a search modality, (2) images are downloaded from Google Image search and used to train an appearance model on-the-fly, (3) ranked results over the target dataset are returned. A live demo is available online – see the text for details.*

these for the text query, rather than specifying a person (by name) as in the case of identity retrieval. This simple technique enables users to explore the content along other dimensions. Figure 7.4 shows several facial attribute examples on the BBC News dataset.

Uber classifiers. The on-the-fly paradigm can be complemented by persistent classifiers learnt from curated datasets. These are termed 'uber' classifiers. The need for such classifiers is two fold: first, classifiers trained on-the-fly are subject to changes in the results of the image search engines – content on the internet can change rapidly, e.g. for trending topics, and also slowly, e.g. with returns dominated by more recent (and thus older) faces of a particular actor or politician. Second, training classifiers offline avoids the compromise between speed and accuracy, so far larger training sets can be employed. For example, if the goal is to retrieve all occurrences of a politician in a static (archive) dataset, then using recent photos from a web search may perform poorly, but a classifier trained on a corpus of images spanning several decades would

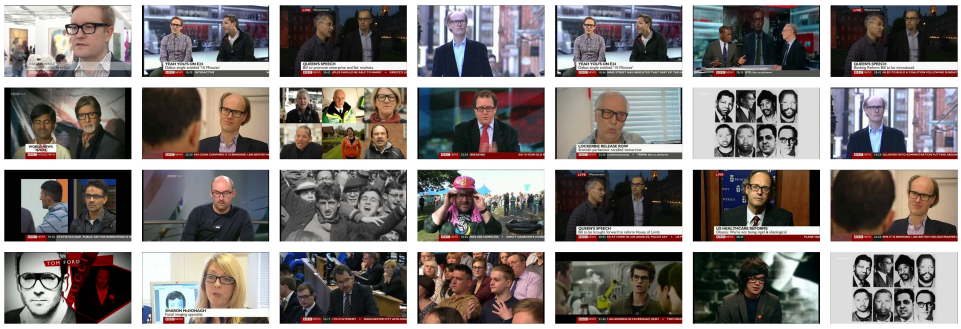
(a) Barack Obama

(b) Margaret Thatcher

(c) Vladimir Putin

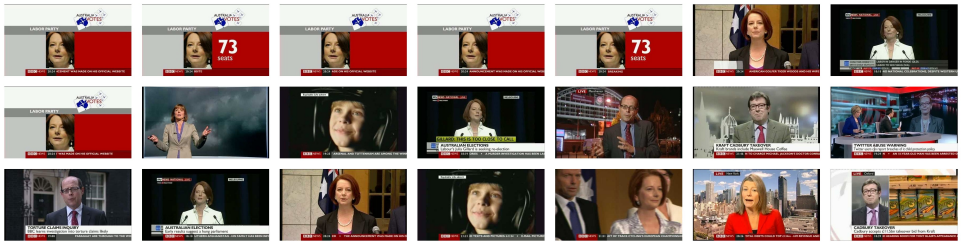
Figure 7.3: Face retrieval search examples. *Top results for queries Barack Obama, Margaret Thatcher and Vladimir Putin on the BBC News dataset. Results shown are the frames from the top ranked tracks ranked using the classifier trained on images downloaded from the Google. As can be seen, the proposed method returns very accurate results.*

(a) Attribute search; query: Beard



(b) Attribute search; query: Thick Black Spectacles

Figure 7.4: Face retrieval attribute search examples. *Top search results for facial attribute queries 'beard' and 'black spectacles' respectively on the BBC News dataset.*



(a) On-the-fly face search; query: Julia Gillard

(b) Uber classifier face search; query: Julia Gillard

Figure 7.5: Face uber classifier example. *The top search results when queried for the former Australian prime minister Julia Gillard. On-the-fly search results (top row) are quite poor while the results with an uber classifier (bottom row) trained on curated data are much better. Results are on the BBC News dataset.*

likely perform better.

Figure 7.5 shows one such example. Searching for the former Australian prime minister a few years after her tenure does not result in good results because of the poor training data available from the web. However, after manually curating images from relevant websites, the uber classifier yields superior results.

Curation of training images. As explained in the Sect. 7.3, search results can get affected by the noisy images coming from the web. The system provides an interface to curate this training data and retrain the classifier removing any noise. Fig. 7.6 shows an example.

Relevance Feedback. At times, one may feel the need to find similar results to a particular result or a group of results. We provide this functionality where the user can select example images from the returned results to further explore the database. We train a classifier based on user feedback and present the results back to the user. The user is allowed to give positive as well as negative feedback. Fig. 7.7 shows an example.

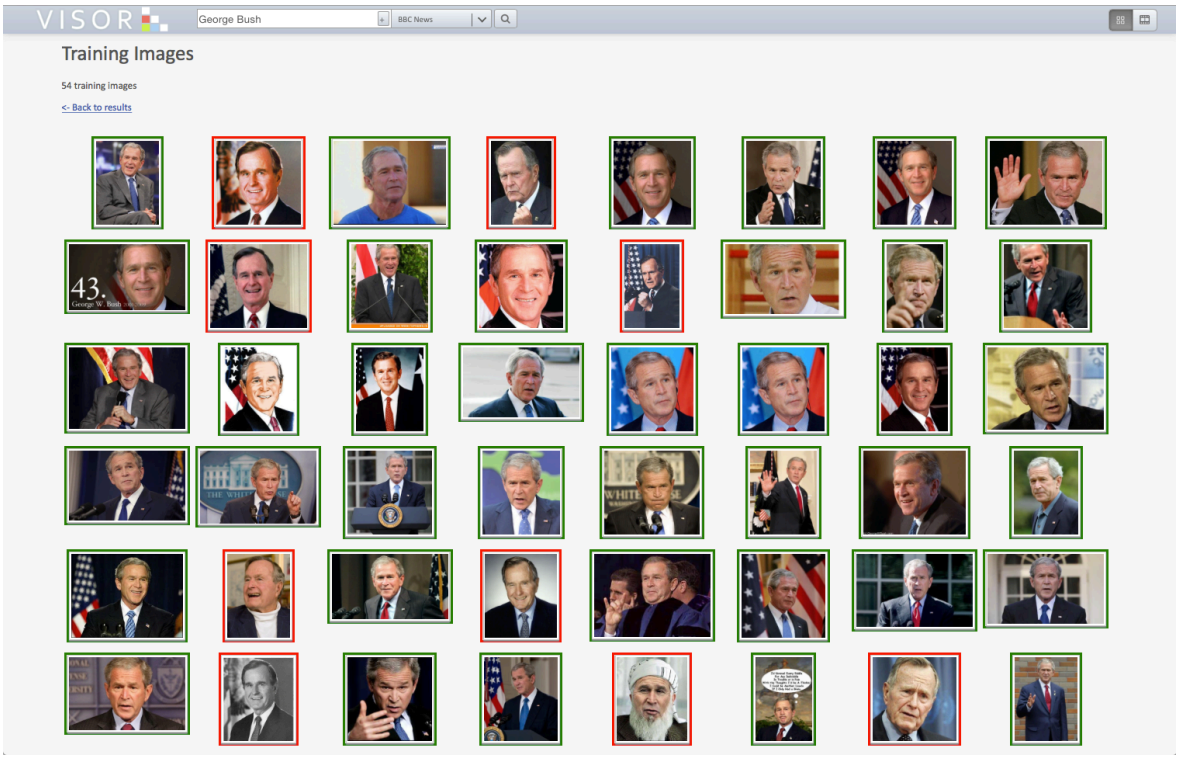
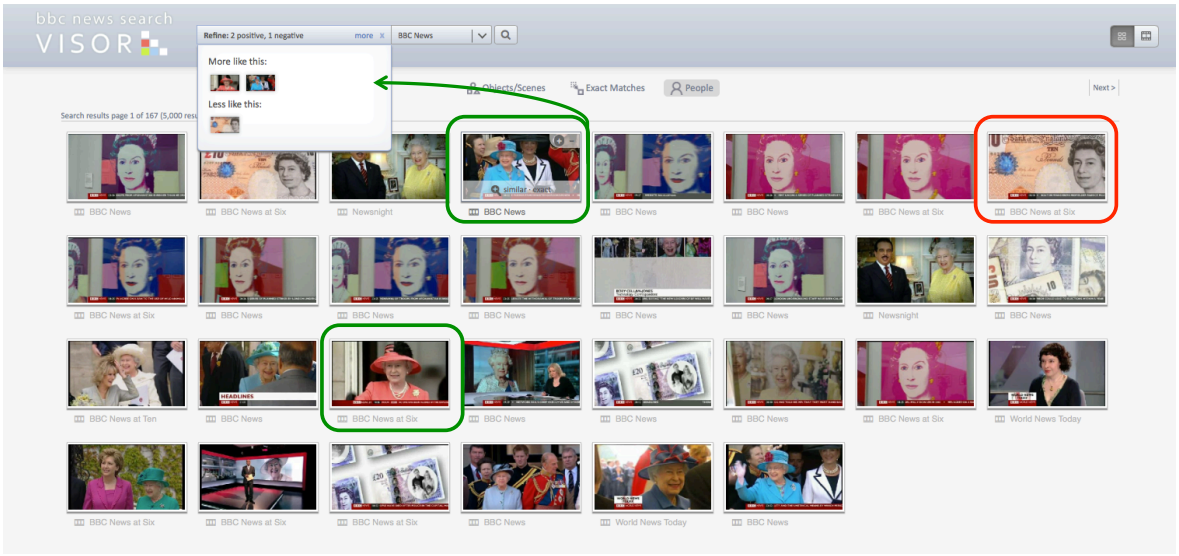
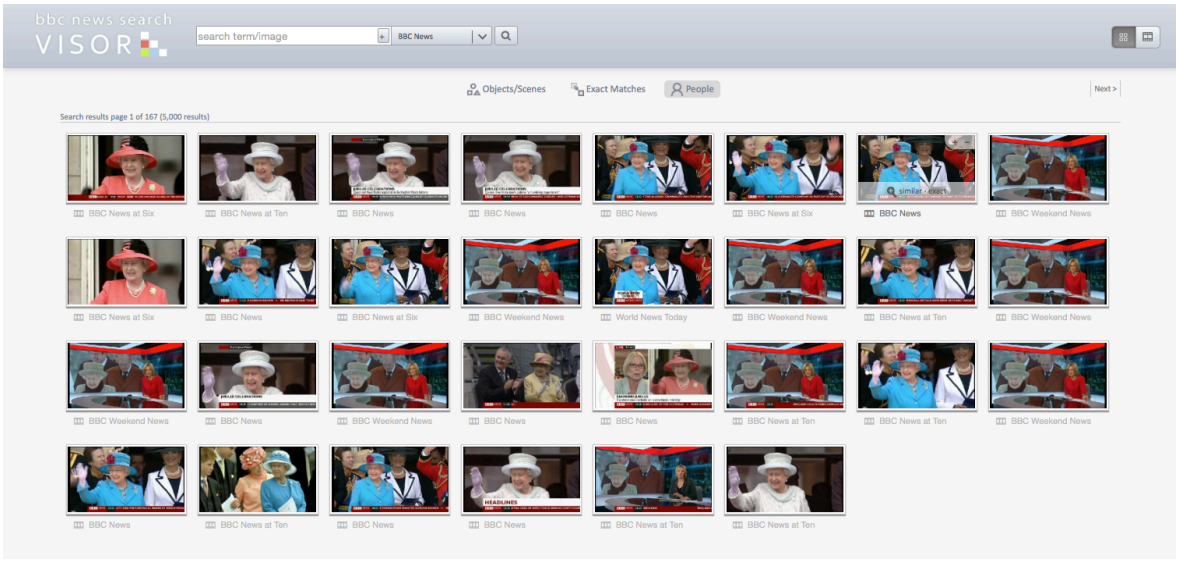


Figure 7.6: ***Training images curation.*** *Images coming from the web can be noisy. The user needs to be able to curate these images in order to train better models. Our system provides an interface for this curation. As it can be seen, for the query "George Bush", Google return images of both the father and the son. The user can then select appropriate images (Green) to retrain the classifier from.*



(a) Relevance feedback: Selection



(b) Relevance feedback: Results

Figure 7.7: Relevance feedback example *From the search results for the query "Queen Elizabeth" the user selects few results for further exploration (Green). The user can also provide negative feedback asking system to return results less similar to those selected (Red). The classifier is then retrained based on this feedback and results are return back to the user.*

Program view. As explained at the beginning, the entity for retrieval is a face track. A track is always confined within boundaries of a shot which forms default display. The default results page shows shots (tracks) ranked according to the likelihood of having the queried person. The view can be switched to "program mode" which shows the programs in which the person appears. Each program can further be explored giving exact locations at which the person appears. Fig. 7.8 shows an example of this approach. We use timing related metadata of the programs to furnish these results.

Interaction between search modalities. An image or video database can be browsed intuitively by switching between different search modalities. We have integrated our system with two other systems. One allows users to search for a specific instance such as Big Ben or the White House and logos such as National Lottery, Manchester United etc. The other allows users to search based on generic object categories. As a result, users can explore the results further potentially using a different modality. An example is shown in Figure 7.9 where a user transitions from face to instance search.

7.7 Summary

In this chapter, we have demonstrated approaches to build a large scale retrieval system for faces. We evaluated suitability of different approaches of representation discussed in previous chapters for this task. We also demonstrated integration of such a system with other approaches for searching through the large archives.

Web demo. An on-line demonstration of the method presented here is available at `http://www.robots.ox.ac.uk/~vgg/research/on-the-fly/`. The backend part of the system was developed as a part of this thesis, while the frontend was developed in collaboration with Relja Arandjelovic and Ken Chatfield.

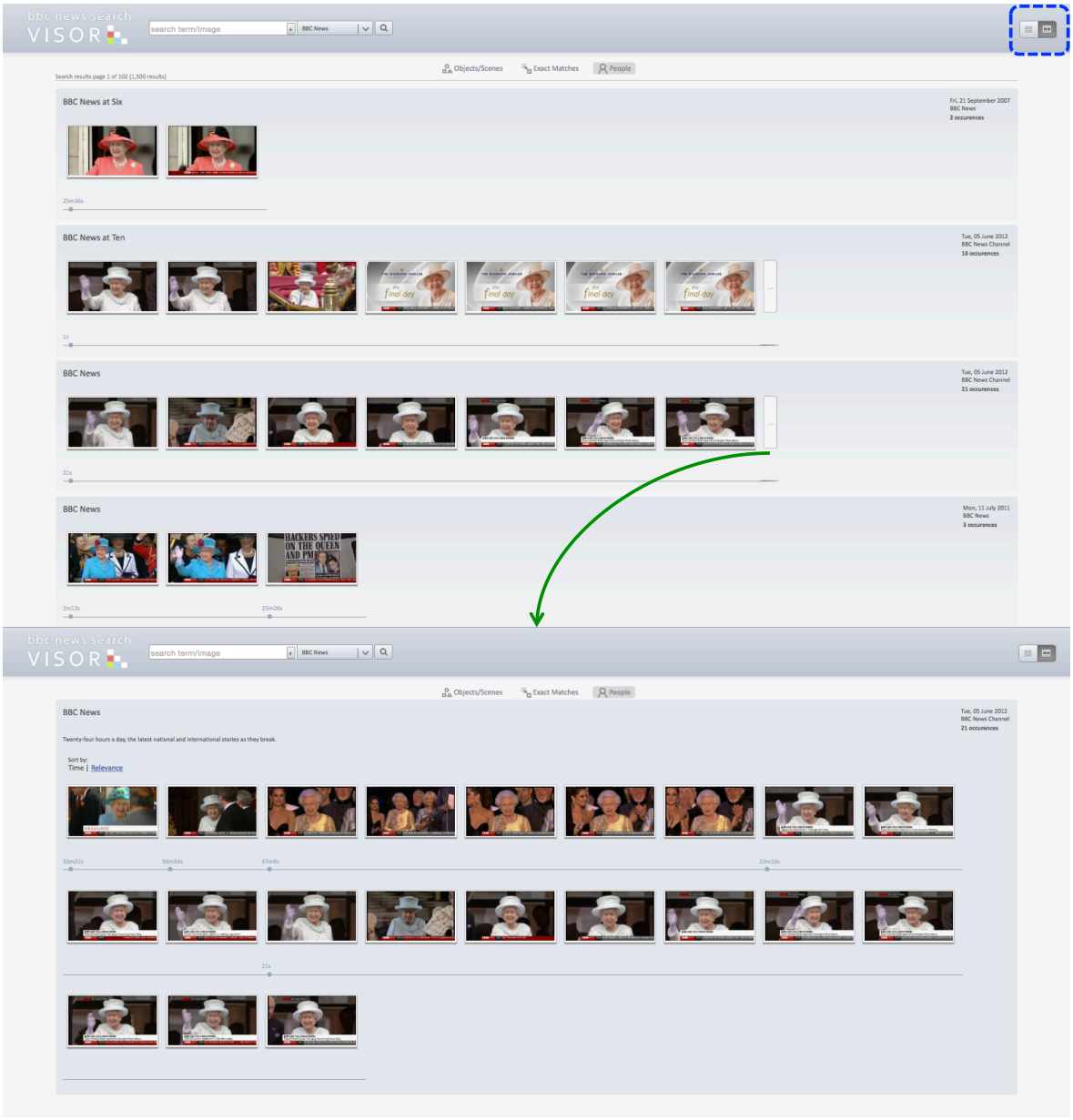


Figure 7.8: ***Exploring archives using the system.*** *The default view option for the results is shot based. The user can click on program view option (blue rectangle) to see the programs specific to their query. They can further explore any program to see all occurances of queried person in that program.*

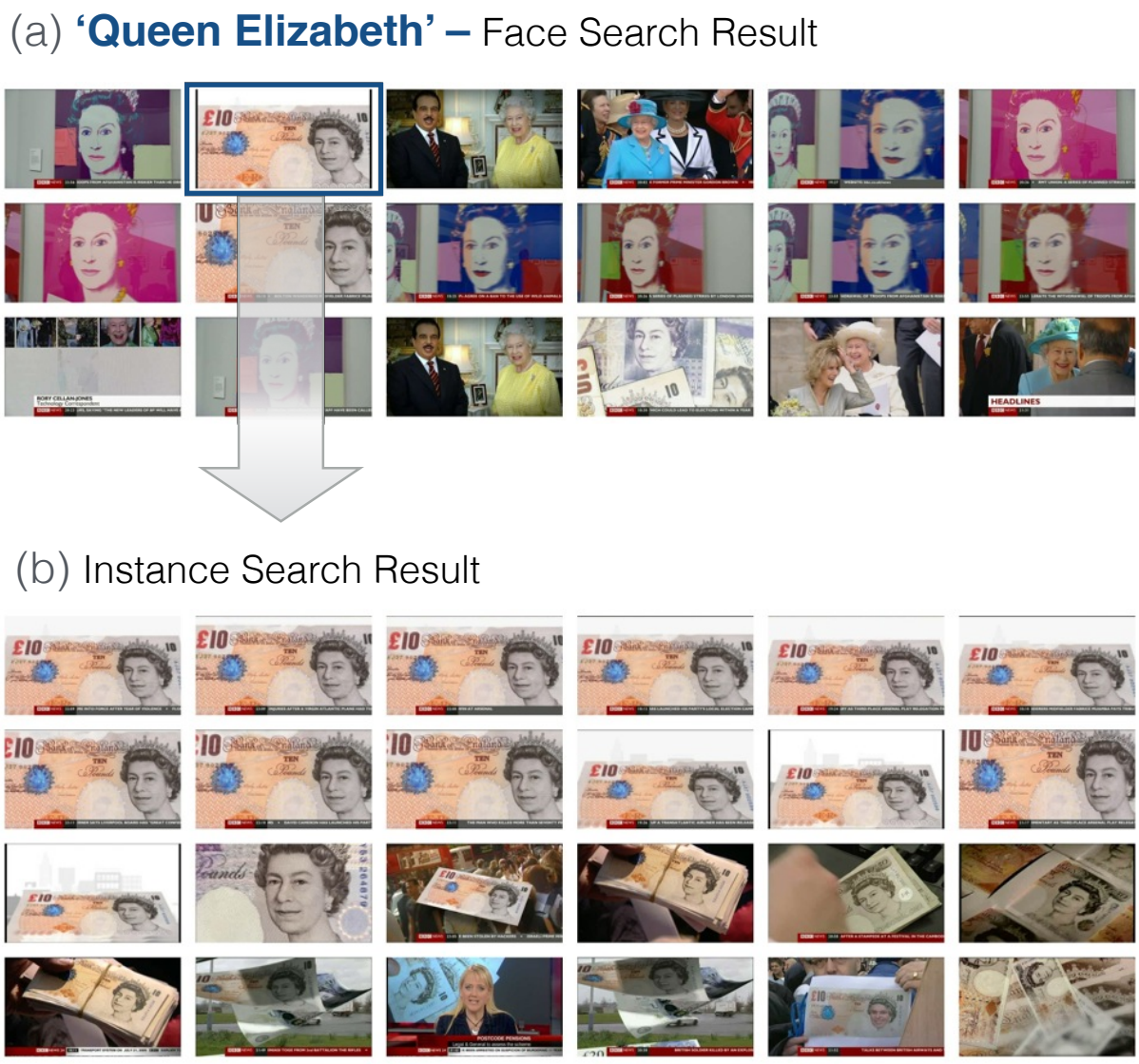


Figure 7.9: Interaction between face and instance search. *Users can effortlessly switch between search modalities. In this example, querying for Queen Elizabeth is done via our proposed face search (a), and the second result depicting the 10 pound note is used to issue an instance search query (b).*

Chapter 8: Summary and future work

In conclusion, we summarize the contributions of this thesis, analyze their impact in the community and comment on the possible areas for future work.

8.1 Shallow features (Chapter 3)

In Chapter 3, we investigated the use of dense features and their encodings for representing faces in images and videos. In both cases, we showed that using off-the-shelf components from standard image classification pipeline. (SIFT and Fisher Vector Encoding), can be successfully used for representing faces. These features implicitly learn about facial landmarks making careful handcrafting around some of these landmarks redundant. We presented detailed analysis of this feature pipeline bringing out key contributors in achieving greater performance. We also show that same findings are applicable to representing faces in videos as well. We show that features extracted on individual frames can be aggregated into a single feature vector. This brings tremendous reduction in memory footprint as compared to previous methods. We presented a fast encoding method for encoding Fisher Vectors with up to $6\times$ speedup with a negligible performance loss. The Gaussian Mixtures in Fisher Vectors were shown to implicitly track facial landmarks.

Another important contribution has been that of the discriminative dimensionality reduction. We presented a method to reduce the dimensionality of Fisher Vectors. The proposed method achieved orders of magnitude reductions in the dimensionality of these vectors without any loss in performance. We also presented a technique to further compress these vectors making them extremely compact yet efficient.

8.1.1 Impact and Future Work

Our presented methods achieved the-state-of-the-art performance on LFW and YTF datasets at the time of publication of the respective publications. Together, they have been cited by more than 150 publications. In terms of future work, Cimpoi *et al.* [33] recently showed that SIFT in Fisher Vector encoding can be replaced by local features obtained from CNN activations to achieve state-of-the art performance on various datasets. Whether this finding is applicable to face representation will be interesting to investigate. Simonyan *et al.* [133] presented a method introducing the notion of end to end learning to Fisher Vector encodings. Paulin *et al.* [111] presented a method to improve performance of Fisher Vectors by dataset augmentation. It will be an interesting research problem to see the impacts of these methods for the purpose of face representation.

8.2 Deep Features (Chapter 4)

Chapter 4 had two clear contributions. In the first we showed that massive datasets can be collected by harvesting knowledge available on the web. This can be done with minimal manual intervention while achieving low label noise. This procedure has been developed for faces, but is evidently suitable for other object classes as well as fine grained tasks. Using this dataset we investigate unreasonable effectiveness of CNNs for face representation. While doing so we achieve state-of-the-art performance on a very challenging public benchmarks. We show that sophisticated post-processing tasks like 3D frontalisation are not essential to achieving the best performance and networks trained for a specific task work better than those trained with a generic objective.

8.2.1 Impact and Future Work

The CNN models developed have been one of the top performing models of IARPA's JANUS program's benchmark evaluation tests beating several commercial off the shelf (COTS) systems. Our work opens lots of avenues for further research on this topic. Firstly, the effect of

different architectures on performance can be investigated. Also based on our findings, we believe that CNNs trained for specific tasks perform better than using FC layer activations from a network trained for another task. A network can be learnt end to end with a triplet loss or any other embedding loss to see the impact on the performance. The dataset collection procedure can be further simplified and dataset size can be further increased.

8.3 Weakly Supervised Learning (Chapter 5)

We have shown that stronger supervisory information can be obtained from an aligned transcript than that of previous methods. Moreover, from our implementation and comparison of bag properties over previous methods, we can conclude that a strategy of having tighter positive bags (i.e. fewer labels associated with the the bag) together with a greater quantity of negative data leads to a significant improvement in classification performance. Taken together with the ConvNet based face-track descriptor, the stronger supervision leads to near saturation of performance on the Buffy benchmark, and also to learning classifiers that generalize well to unseen episodes.

8.3.1 Future Work

The contributions we have introduced are applicable to other tasks that obtain supervision from aligned scripts, for example, to action recognition [46]. We would expect similar performance improvements in these cases as well. Additionally, with automatic labeling accuracies nearing saturation, it opens the possibility of investigating more sophisticated methods of inferring from videos such as answering questions based on the storyline.

8.4 Transfer Learning (Chapter 6)

In this work, we have shown the transfer learning capabilities of both representations introduced in the previous chapters. More specifically we show that transfer capabilities of the Fisher Vector face representation, can be improved significantly by discriminative learning. While the CNN features generalized remarkably well without any additional learning being required.

We also looked at use of these features to find paintings of similar looking people given a photograph and investigated the use of facial attributes in for this application.

8.4.1 Future Work

Similar to the work discussed above, there is a lot of scope for future work here. Carrying forward the artistic theme of the chapter and combining it with the recent work on inverting networks, one can investigate generating network impressions of famous identities. The retrieval work can be further improved to combine objects and identities together for retrieval.

8.5 Large scale retrieval (Chapter 7)

In Chapter 7 we looked at the building blocks of a large scale face retrieval system. We showed that appearance models can be learnt for celebrities in real time fostering the knowledge base on the web. We investigated the use of Fisher Vectors for this retrieval purposes and also looked at the suitability of web sources for obtaining training data for learning models. Additionally, we have also shown that these models can be pre-trained to mitigate the effects of the changing nature of the web data. Finally, we showed that such a system can also be used for searching based on facial attributes and can also be combined with other search modalities in order to improve capabilities of exploring large scale video archives.

8.5.1 Impact and Future Work

The demo system built is being used by the BBC to search their archives. Our work was featured in BBC R&D's blog post "Face recognition and new ways to search for archives". The overall system developed under this project consists of the on-the-fly approaches for three classes of queries: object instances, object categories and faces. The question, then, is what next? Can human pose be learnt on-the-fly from still images downloaded by search engines? or human actions [69]? There are also a number of standard competencies that are provided by modern search engines: such as composite queries (e.g. 'Obama standing outside the White House'), diversity of the returned results, and clustering of the returned results. These each

have a different twist when vision, rather than text, is the primary search and representation method. Developing these competencies for visual search opens up new research directions. The work done here also contributed to European Union's FP7 project AXES designed for searching through audiovisual archives.

Bibliography

[1] ABC News. http://abcnews.go.com/blogs/headlines/2012/11/man-finds-his-doppelganger-in-16th-century-italian-painting/.

[2] BBC – Your Paintings. http://www.bbc.co.uk/arts/yourpaintings/.

[3] DeviantArt. http://www.deviantart.com/.

[4] Freebase. http://www.freebase.com/.

[5] Internet movie database. http://www.imdb.com.

[6] R. Aly, R. Arandjelović, K. Chatfield, M. Douze, B. Fernando, Z. Harchaoui, K. McGuiness, N. O'Connor, D. Oneata, O. Parkhi, D. Potapov, J. Revaud, C. Schmid, J.-L. Schwenninger, D. Scott, T. Tuytelaars, J. Verbeek, H. Wang, and A. Zisserman. The AXES submissions at TrecVid 2013. In *TRECVID Workshop*, 2013.

[7] R. Aly, K. McGuinness, S. Chen, N. E. O'Connor, K. Chatfield, O. M. Parkhi, R. Arandjelović, A. Zisserman, B. Fernando, T. Tuytelaars, J. Schwenninger, D. Oneata, M. Douze, J. Revaud, D. Potapov, H. Wang, Z. Harchaoui, J. Verbeek, and C. Schmid. AXES at TRECVID 2012: KIS, INS, and MED. In *TREC 2012 Video Retrieval Evaluation*, 2012.

[8] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems*, 2003.

[9] N. E. Apostoloff and A. Zisserman. Who are you? – real-time person identification. In *Proceedings of the 18th British Machine Vision Conference, Warwick*, 2007.

[10] O. Arandjelovic and A. Zisserman. Automatic face recognition for film character retrieval in feature-length films. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*, pages 860–867, 2005.

[11] R. Arandjelović and A. Zisserman. Multiple queries for large scale specific object retrieval. In *Proceedings of the British Machine Vision Conference*, 2012.

[12] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[13] R. Arandjelović and A. Zisserman. All about VLAD. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[14] M. Aubry, B. Russell, and J. Sivic. Painting-to-3D model alignment via discriminative visual elements. In *ACM Transactions of Graphics*, 2013.

[15] J. R. Barr, K. W. Bowyer, P. J. Flynn, and S. Biswas. Face recognition from video:a review. *International Journal of Pattern Recognition and Artificial Intelligence*, 26(05):1266002, 2012.

[16] M. Bauml, M. Tapaswi, and R. Stiefelhagen. A time pooled track kernel for person identification. In *Proceedings of the 11th International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*. IEEE, August 26-28 2014.

[17] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[18] R. Benenson, M. Matthias, R. Timofe, and L. Van Gool. Pedestrian detection at 100 frames per second. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[19] T. Berg and P. N. Belhumeur. Tom-vs-Pete classifiers and identity-preserving alignment for face verification. In *Proceedings of the British Machine Vision Conference*, 2012.

[20] T. Berg, A. Berg, J. Edwards, M. Mair, R. White, Y. Teh, E. Learned-Miller, and D. Forsyth. Names and Faces in the News. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC*, 2004.

[21] T. L. Berg and D. A. Forsyth. Animals on the web. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York*, 2006.

[22] A. Bergamo, L. Torresani, and A. Fitzgibbon. PiCoDes: Learning a compact code for novel-category recognition. In *Advances in Neural Information Processing Systems*, pages 2088–2096, 2011.

[23] B. Bhattarai, G. Sharma, F. Jurie, and P. Pérez. Latent max-margin metric learning for comparing video face tubes. In *Workshop on Biometrics Computer Vision and Pattern Recognition*, 2015.

[24] P. Bojanowski, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Finding actors and actions in movies. In *Proceedings of the International Conference on Computer Vision*, 2013.

[25] G. Bradski. Opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

[26] H. Cevikalp and B. Triggs. Face recognition based on image sets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[27] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *Proceedings of the British Machine Vision Conference*, 2011.

[28] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proceedings of the British Machine Vision Conference*, 2014.

[29] K. Chatfield, K. Simonyan, and A. Zisserman. Efficient on-the-fly category retrieval using convnets and gpus. In *Proceedings of the Asian Conference on Computer Vision*, 2014.

[30] K. Chatfield and A. Zisserman. Visor: Towards on-the-fly large-scale object category retrieval. In *Proceedings of the Asian Conference on Computer Vision*, Lecture Notes in Computer Science. Springer, 2012.

[31] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun. Bayesian face revisited: A joint formulation. In *Proceedings of the European Conference on Computer Vision*, pages 566–579, 2012.

[32] D. Chen, X. Cao, F. Wen, and J. Sun. Blessing of dimensionality: High dimensional feature and its efficient compression for face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[33] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[34] R. G. Cinbis, J. J. Verbeek, and C. Schmid. Unsupervised metric learning for face identification in TV video. In *Proceedings of the International Conference on Computer Vision*, pages 1559–1566, 2011.

[35] T. Cootes, G. Edwards, and C. Taylor. Active Appearance Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

[36] T. Cour, B. Sapp, and B. Taskar. Learning from ambiguously labeled images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[37] T. Cour, B. Sapp, and B. Taskar. Learning from partial labels. *Journal of Machine Learning Research*, 2011.

[38] E. J. Crowley, O. M. Parkhi, and A. Zisserman. Face painting: querying art with photos. In *Proceedings of the British Machine Vision Conference*, 2015.

[39] E. J. Crowley and A. Zisserman. In search of art. In *Workshop on Computer Vision for Art Analysis, ECCV*, 2014.

[40] E. J. Crowley and A. Zisserman. The state of the art: Object retrieval in paintings using discriminative regions. In *Proceedings of the British Machine Vision Conference*, 2014.

[41] G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[42] Z. Cui, W. Li, D. Xu, S. Shan, and X. Chen. Fusing robust face region descriptors via multiple metric learning for face recognition in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[43] M. Dantone, J. Gall, G. Fanelli, and L. Van Gool. Real-time facial feature detection using conditional regression forests. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[44] H. Daumé III and D. Marcu. Domain adaptation for statistical classifiers. *J. Artif. Intell. Res.(JAIR)*, 26:101–126, 2006.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[46] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce. Automatic annotation of human actions in video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[47] M. Everingham, J. Sivic, and A. Zisserman. "Hello! My name is... Buffy" – automatic naming of characters in TV video. In *Proceedings of the 17th British Machine Vision Conference, Edinburgh*, 2006.

[48] M. Everingham, J. Sivic, and A. Zisserman. Taking the bite out of automatic naming of characters in TV video. *Image and Vision Computing*, 27(5), 2009.

[49] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[50] M. Everingham and A. Zisserman. Identifying individuals in video by combining generative and discriminative head models. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, China*, 2005.

[51] M. Everingham and A. Zisserman. Regression and classification approaches to eye localization in face images. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, 2006.

[52] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2241–2248, 2010.

[53] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1), 2005.

[54] P. Felzenszwalb, D. Mcallester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[55] P. F. Felzenszwalb, R. B. Grishick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[56] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from Google's image search. In *Proceedings of the 10th International Conference on Computer Vision, Beijing, China*, 2005.

[57] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from internet image searches. *Proceedings of the IEEE*, 98(8):1453–1466, 2010.

[58] B. Fernando and T. Tuytelaars. Mining multiple queries for image retrieval: On-the-fly learning of an object-specific mid-level representation. In *Proceedings of the International Conference on Computer Vision*, 2013.

[59] B. Froba and A. Ernst. Face detection with the modified census transform. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, 2004.

[60] R. Girshick. Fast r-cnn. *arXiv:1504.08083*, 2015.

[61] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. 2014.

[62] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

[63] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? Metric learning approaches for face identification. In *Proceedings of the International Conference on Computer Vision*, 2009.

[64] M. Guillaumin, J. Verbeek, and C. Schmid. Multiple instance metric learning from automatically labeled bags of faces. In *Proceedings of the European Conference on Computer Vision*, pages 634–647, 2010.

[65] A. Hadid and M. Pietikäinen. Manifold learning for video-to-video face recognition. In *Biometric ID Management and Multimodal Communication*, 2009.

[66] M. Hoai and A. Zisserman. Talking heads: Detecting humans and recognizing their interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[67] C. Huang, S. Zhu, and K. Yu. Large scale strongly supervised ensemble metric learning, with applications to face verification and retrieval. *CoRR*, abs/1212.6094, 2012.

[68] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[69] N. Ikizler-Cinbis and S. Sclaroff. Web-based classifiers for human action recognition. *Multimedia, IEEE Transactions on*, 14(4):1031–1045, 2012.

[70] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems*, pages 487–493, 1998.

[71] P. Jain, B. Kulis, and I. S. Dhillon. Inductive regularized learning of kernel functions. In *Advances in Neural Information Processing Systems*, pages 946–954, 2010.

[72] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[73] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[74] H. Jégou, T. Furon, and J.-J. Fuchs. Anti-sparse coding for approximate nearest neighbor search. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2029–2032, 2012.

[75] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. P'erez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[76] L. Jin, S. Satoh, F. Yamagishi, D. Le, and M. Sakauchii. Person X detector. In *TRECVID Workshop*, 2004.

[77] A. Klaser, M. Marszalek, C. Schmid, and A. Zisserman. Human focused action localization in video. In *International Workshop on Sign, Gesture, Activity*, 2010.

[78] M. Köstinger, P. Wohlhart, P. Roth, and H. Bischof. Learning to recognize faces from videos and weakly relatedinformation cues. In *avss*, 2011.

[79] J. Krapac, J. Verbeek, and F. Jurie. Modeling spatial layout with fisher vectors for image categorization. In *Proceedings of the International Conference on Computer Vision*, pages 1487–1494, 2011.

[80] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1106–1114, 2012.

[81] V. Krüger, R. Gross, and S. Baker. Appearance-based 3-D face recognition from video. In *Proc. German Pattern Recognition Symp.*, 2002.

[82] B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[83] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Learning layered motion segmentations of video. *International Journal of Computer Vision*, 76:301–319, 2008.

[84] N. Kumar, A. C. Berg, P. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In *Proceedings of the International Conference on Computer Vision*, 2009.

[85] N. Kumar and S. Seitz. Photo recall: Using the internet to label your photos. In *The 23rd International Conference on World Wide Web companion*, April 2014.

[86] N. Kumar and S. Seitz. Photo recall: Using the internet to label your photos. In *2nd Workshop on Web-scale Vision and Social Media (VSM) at CVPR 2014*, June 2014.

[87] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York*, 2006.

[88] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[89] H. Li, G. Hua, J. Brandt, and J. Yang. Probabilistic elastic matching for pose variant face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[90] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[91] J. Li, G. Wang, and L. Fei-Fei. OPTIMOL: automatic Object Picture collecTion via Incremental MOdel Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[92] P. Li, Y. Fu, U. Mohammed, J. H. Elder, and S. J. D. Prince. Probabilistic models for inference about identity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):144–157, November 2012.

[93] W.-H. Lin, R. Jin, and A. Hauptmann. Web Image Retrieval Re-Ranking with Relevance Model. In *Proceedings of the IADIS International Conference WWW/Internet*, 2003.

[94] Y. Liu, D. Xu, I. W. Tsang, and J. Luo. Using large-scale web data to facilitate textual query based retrieval of consumer photos. In *Proceedings of the 17th ACM International Conference on Multimedia*, MM '09, pages 55–64, 2009.

[95] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[96] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-SVMs for object detection and beyond. In *Proceedings of the International Conference on Computer Vision*, 2011.

[97] M. Marin-Jimenez, A. Zisserman, and V. Ferrari. Heres looking at you, kid. detecting people looking at each other in videos. In *Proceedings of the British Machine Vision Conference*, 2011.

[98] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face detection without bells and whistle. In *ECCV*, 2014.

[99] K. McGuinness, S. Chen, M. Frappier, H. Lee, N. E. O'Connor, A. Smeaton, R. Aly, R. Ordelman, M. Kleppe, H. Beunders, R. Arandjelović, A. Vedaldi, A. Zisserman, M. Juneja, C. V. Jawahar, J. Schwenninger, S. Tschopel, and D. Schneider. AXES at TRECVID 2011. In *TREC 2011 Video Retrieval Evaluation*, 2011.

[100] H. Mendez-Vazquez, Y. Martinez-Diaz, and Z. Chai. Volume structured ordinal features with background similarity measure for video face recognition. In *International Conference on Biometrics (ICB)*, 2013.

[101] H. V. Nguyen and L. Bai. Cosine similarity metric learning for face verification. In *Proceedings of the Asian Conference on Computer Vision*, 2010.

[102] M.-E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York*, volume 2, pages 1447–1454, 2006.

[103] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *Proceedings of the European Conference on Computer Vision*, 2006.

[104] M. Osadchy, M. Miller, and Y. Lecun. Synergistic face detection and pose estimation with energy-based model. *Advances in Neural Information Processing Systems*, 2005.

[105] A. Ozerov, J. R. Vigouroux, L. Chevallier, and P. Pérez. On evaluating face tracks in movies. In *Proceedings of the IEEE International Conference on Image Processing*, 2013.

[106] U. Park, H. Chen, and A. Jain. 3d model-assisted face recognition in video. In *Proc. Canadian Conf. Computer and Robot Vision*, 2005.

[107] O. M. Parkhi, K. Simonyan, A. Vedaldi, and A. Zisserman. A compact and discriminative face track descriptor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, IEEE, 2014.

[108] O. M. Parkhi, A. Vedaldi, and A. Zisserman. On-the-fly specific person retrieval. In *International Workshop on Image Analysis for Multimedia Interactive Services*. IEEE, 2012.

[109] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *Proceedings of the British Machine Vision Conference*, 2015.

[110] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[111] M. Paulin, J. Revaud, Z. Harchaoui, F. Perronnin, and C. Schmid. Transformation Pursuit for Image Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[112] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *Proceedings of the European Conference on Computer Vision*, 2010.

[113] J. Philbin, A. Bosch, O. Chum, J. Geusebroek, J. Sivic, and A. Zisserman. Oxford TRECVID 2006 – notebook paper. In *TRECVID Workshop*, 2006.

[114] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss. The feret evaluation methodology for face-recognition algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[115] N. Pinto and D. Cox. Beyond simple features: A large-scale feature search approach to unconstrained face recognition. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, 2011.

[116] N. Pinto, J. J. DiCarlo, and D. D. Cox. How far can you get with a modern face recognition test set using only simple features? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[117] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, S. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and F. Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015.

[118] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *Proceedings of the European Conference on Computer Vision*, 2010.

[119] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[120] J. Sánchez, F. Perronnin, and T. Emídio de Campos. Modeling the spatial layout of images beyond spatial pyramids. *Pattern Recognition Letters*, 33(16):2216–2223, 2012.

[121] H. Schneiderman. Feature-centric evaluation for efficient cascaded object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[122] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting Image Databases from the Web. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):754–766, April 2011.

[123] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[124] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[125] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[126] G. Sharma, S. Hussain, and F. Jurie. Local higher-order statistics (LHS) for texture categorization and facial analysis. In *Proceedings of the European Conference on Computer Vision*, 2012.

[127] G. Sharma and P. Pérez. Epml: Expanded parts based metric learning for occlusion robust face verification. In *Proceedings of the Asian Conference on Computer Vision*, 2014.

[128] G. Sharma and P. Pérez. Latent max-margin metric learning for comparing video face tubes. In *Workshop on Biometrics Computer Vision and Pattern Recognition*, 2015.

[129] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[130] A. Shrivastava, T. Malisiewicz, A. Gupta, and A. Efros. Data-driven visual similarity for cross-domain image matching. *ACM Transaction of Graphics*, 2011.

[131] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman. Fisher Vector Faces in the Wild. In *Proceedings of the British Machine Vision Conference*, 2013.

[132] K. Simonyan, A. Vedaldi, and A. Zisserman. Descriptor learning using convex optimisation. In *Proceedings of the European Conference on Computer Vision*, 2012.

[133] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Fisher networks for large-scale image classification. In *Advances in Neural Information Processing Systems*, 2013.

[134] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[135] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[136] J. Sivic, M. Everingham, and A. Zisserman. Person spotting: Video shot retrieval for face sets. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, 2005.

[137] J. Sivic, M. Everingham, and A. Zisserman. "Who are you?" – learning person specific classifiers from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[138] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the 9th International Conference on Computer Vision, Nice, France*, volume 2, pages 1470–1477, 2003.

[139] J. Sivic and A. Zisserman. Efficient visual search for objects in videos. *Proceedings of the IEEE*, 96(4):548–566, 2008.

[140] J. Sivic and A. Zisserman. Efficient visual search of videos cast as text retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):591–606, 2009.

[141] J. Stallkamp, H. K. Ekenel, and R. Stiefelhagen. Video-based face recognition on real-world data. In *Proceedings of the International Conference on Computer Vision*, 2007.

[142] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. *Advances in Neural Information Processing Systems*, 2014.

[143] Y. Sun, L. Ding, X. Wang, and X. Tang. Deepid3: Face recognition with very deep neural networks. *CoRR*, abs/1502.00873, 2015.

[144] Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[145] Y. Sun, X. Wang, and X. Tang. Deeply learned face representations are sparse, selective, and robust. *CoRR*, abs/1412.1265, 2014.

[146] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[147] Y. Taigman and L. Wolf. Leveraging billions of faces to overcome performance barriers in unconstrained face recognition. *CoRR*, abs/1108.1122, 2011.

[148] Y. Taigman, L. Wolf, and T. Hassner. Multiple one-shots for utilizing class label information. In *Proceedings of the British Machine Vision Conference*, 2009.

[149] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deep-Face: Closing the gap to human-level performance in face verification. In *IEEE CVPR*, 2014.

[150] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Web-scale training for face identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[151] M. Tapaswi, M. Baeuml, and R. Stiefelhagen. "knock! knock! who is it?" probabilistic person identification in tv series. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[152] M. Tapaswi, C. Çağrı Çörez, M. Bäuml, H. Kemal Ekenel, and R. Stiefelhagen. Cleaning up after a Face Tracker: False Positive Removal. In *Proceedings of the IEEE International Conference on Image Processing*, 2014.

[153] L. Torresani and K. Lee. Large margin component analysis. In *Advances in Neural Information Processing Systems*, pages 1385–1392. MIT Press, 2007.

[154] L. Torresani, M. Szummer, and A. Fitzgibbon. Efficient object category recognition using classemes. In *Proceedings of the European Conference on Computer Vision*, pages 776–789, sep 2010.

[155] M. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991.

[156] A. Vedaldi and B. Fulkerson. VLFeat - an open and portable library of computer vision algorithms. In *Proceedings of the ACM Multimedia Conference*, 2010.

[157] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. *CoRR*, abs/1412.4564, 2014.

[158] P. Viola and M. Jones. Robust real-time object detection. In *International Journal of Computer Vision*, volume 1, 2001.

[159] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical report, California Institute of Technology, 2011.

[160] K. Q. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems*, 2006.

[161] P. Wohlhart, M. Köstinger, P. M. Roth, and H. Bischof. Multiple instance boosting for face recognition in videos. In *DAGM-Symposium*, 2011.

[162] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[163] L. Wolf, T. Hassner, and Y. Taigman. Descriptor based methods in the wild. In *Faces in Real-Life Images Workshop in European Conference on Computer Vision*, 2008.

[164] L. Wolf, T. Hassner, and Y. Taigman. Similarity scores based on background samples. In *Proceedings of the Asian Conference on Computer Vision*, 2009.

[165] L. Wolf and N. Levy. The svm-minus similarity score for video face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[166] Q. Wu, H. Cai, and P. Hall. Learning graphs to model visual objects across different depictive styles. In *Proceedings of the European Conference on Computer Vision*, 2014.

[167] Q. Wu and P. Hall. Modelling visual objects invariant to depictive style. In *Proceedings of the British Machine Vision Conference*, 2013.

[168] X. Xiong and F. De la Torre. Supervised descent method and its applications to face alignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[169] J. Yang, M. Chen, and A. Hauptmann. Finding person x: Correlating names with visual appearances. In *Proceedings of the ACM International Conference on Image and Video Retrieval*.

[170] J. Yang, R. Yan, and A. G. Hauptmann. Multiple instance learning for labeling faces in broadcasting news video. In *ACM Multimedia*, 2005.

[171] M. Yang, P. Zhu, L. Van Gool, and L. Zhang. Face recognition based on regularized nearest points betweenimage sets. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, 2013.

[172] Q. Yin, X. Tang, and S. J. Face recognition with learning-based descriptor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[173] Y. Ying and P. Li. Distance metric learning with eigenvalue optimization. *Journal of Machine Learning Research*, 2012.

[174] S. Zhang, Q. Huang, G. Hua, S. Jiang, and Q. Gao, W. Tian. Building contextual visual vocabulary for large-scale image applications. In *Proceedings of the ACM Multimedia Conference*, 2010.

[175] W. Zhao, R. Challappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys*, 35:399–458, 2003.

[176] S. Zhou, V. Krueger, and R. Chellappa. Face recognition from video: A condensation approach. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, 2002.

[177] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *Proceedings of the European Conference on Computer Vision*, 2010.

[178] X. Zhu and D. Ramanan. Face detection, pose estimation and landmark localization in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.