Visual Retrieval of Compound Queries

D.Phil. Thesis

Robotics Research Group
Department of Engineering Science
University of Oxford

Supervisor:
Professor Andrew Zisserman
Dr. Relja Arandjelović

Yujie Zhong
Harris Manchester College

2018

Abstract

There are now vast number of images available on the Internet or in personal collections. As a result, searching for images based on their visual content has many useful applications. In this thesis, we focus on compound query image retrieval. Namely, the query can consist of objects of different natures, a set of objects of the same type or multiple examples of an object, and the goal is to retrieve (based on visual content) images that match the query from a large image corpus. However, compound query retrieval is very challenging, as it may require the system to handle queries of different object types. Furthermore, the retrieval should be real-time with high performance.

The first task we consider is to retrieve images containing both a target person and a target scene type from a large dataset of images. We propose a hybrid convolutional neural network architecture that produces place-descriptors that are aware of faces and their corresponding descriptors. We also propose an image synthesis system to render high quality fully-labelled face-and-place images which are used to train the network. To facilitate this research, we collect and annotate a dataset of real images containing celebrities in different places, which can be used to evaluate the retrieval system. We demonstrate significantly improved retrieval performance for compound queries using the new face-aware place-descriptors compared to baseline methods.

Set retrieval is another example of compound query retrieval. Namely, we wish to rank the images, given a set of query identities, such that those containing all the identities of the query are ranked first, followed by those which satisfy all but one of the query identities, and so on. To this end, we propose a network architecture to achieve the objective: it learns face descriptors and their aggregation over a set to produce a compact fixed length descriptor designed for set retrieval. We also explore the speed vs. retrieval quality trade-off for set retrieval using this compact descriptor. For evaluation, we collect and annotate a large dataset of images containing various numbers of celebrities, which we use and is publicly available.

Template-based face recognition, where a set of faces of the same subject is available, is now gaining attention as there are usually more than one examples for each subject in real-world situations. To tackle this problem, we propose a network architecture which aggregates and embeds the face descriptors produced by deep convolutional neural networks into a compact template representation. This compact representation requires minimal memory storage and enables efficient similarity computation. The proposed architecture contains a novel GhostVLAD layer which enables the network to deal with poor quality images, *i.e.* informative images contribute more than the low quality ones. We also show that such quality weighting on the input faces emerges automatically. The performance of the network far exceeds the state-of-the-art on one of the most challenging public benchmarks.

Declaration

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.

Yujie Zhong

Acknowledgements

I would like to thank my supervisor Professor Andrew Zisserman for his guidance, support and encouragement. He is the best supervisor I can ever imagine. I hope that I can continue to learn from him even after my graduation. I would also like to thank my secondary supervisor, Doctor Relja Arandjelović, for his devotion and patience. He is extremely responsible and provides me invaluable advice.

I am thankful to everyone in VGG for making it such a nice environment to work in. I also thank my friends in Oxford, with whom I have a precious and joyful life during my PhD. Most importantly, I am grateful to my parents Xiujun and Jiaqiang, and my girl friend Wei for all their support and understanding.

Contents

Chapter 1

Introduction

1.1 Objectives and motivation

We have entered an age of information, in which the Internet has become an increasingly important component of our social lives. One of the key drivers of this transition has been the tremendous number of images that are uploaded to the Internet. For example, billions of new photos are uploaded every day to Facebook by over two billion monthly active users and roughly 35 billion images were shared on Instagram prior to 2018. However, this vast supply of visual information content is mostly unannotated: many users who share their images do not wish to spend their time supplying corresponding descriptive captions or tags. As a consequence, searching through this large number of images based on the limited annotations available becomes unreliable. In this work, we focus on an alternative solution, that is, to search based on visual content.

In computer vision, content-based image retrieval has gained attention for many years. Most work has focused on single-query image retrieval, such as searching for an object category, a particular building or a target person. Sometimes, however, such simple form of queries is not sufficient and more complicated queries are required.

Suppose you want to find a photo of your friend in a cathedral in your personal

collection of images, or you are a news producer and want to find a photo of 'Barack Obama on the beach' to illustrate an article, where the required image is somewhere in a corpus of 100k images. In this case, we wish to search for a particular person in a target place based on the visual content from a large image corpus without any annotations.

Consider another situation: suppose we wish to retrieve all images in a very large collection of personal photos that contain a particular set of people, such as a group of friends or a family. Then we would like the retrieved images that contain all of the set to be ranked first, followed by images containing subsets: *e.g.* if there are three friends in the query, then first would be images containing all three friends, then images containing two of the three, followed by images containing only one of them.

The above examples of queries can not be solved efficiently and accurately using current image retrieval systems which only handle single-query searching problems. As a consequence, different from most existing single-query image retrieval research, the objective of this thesis is to investigate searching large scale image collections visually for *compound queries* of these types – the query may be a combination of objects of different nature (*e.g.* faces and places), or multiple objects of same kinds (*e.g.* a set of identities).

Specifically, the first problem we tackle in this thesis is retrieving images containing both a target person (via the face) and a target place, as shown in Figure 1.1. The key investigation here is how to rank the database images based on two very different and independent object types. The second situation we consider is an example of a *set retrieval problem*: each image contains a set of elements (faces in this case), and we wish to order the images according to a query (on multiple identities) such that those images satisfying the query completely are ranked first (*i.e.* those images that contain all the identities of the query), followed by images that satisfy all but one of

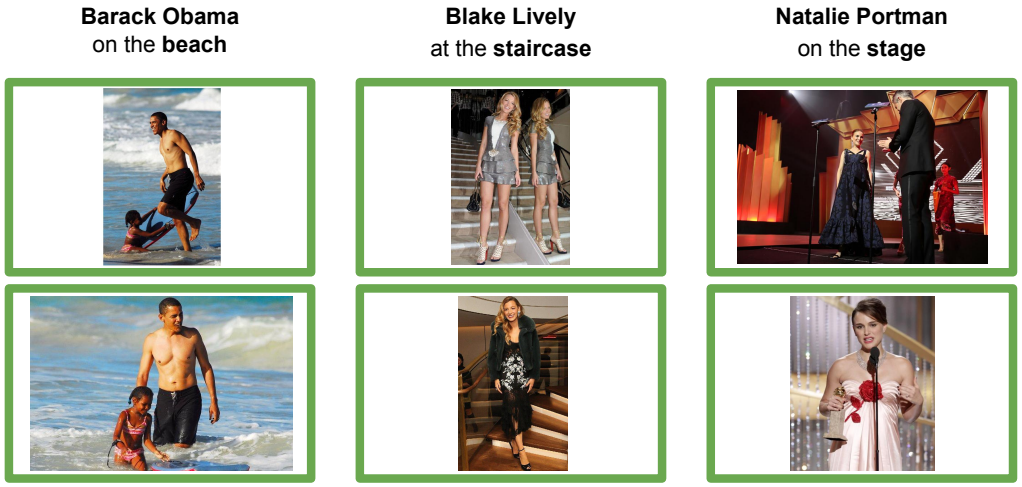


Figure 1.1: **Three examples of retrieving faces in places.** The top two images retrieved from the CIP dataset (introduced in Chapter 3) are shown for each compound query shown on top of each column. The results are obtained using the descriptors generated by the CNN introduced in Chapter 5.

the query identities, *etc.* An illustration of set retrieval is shown in Figure 1.2.

In this thesis, we also consider another related problem – set-based face recognition. For the identification task of this problem, the database consists of a large number of image sets and each image set corresponds to a subject. Given a set of images of a query subject, the aim is to rank the database image sets and identify the query subject. Therefore, the face identification task is in fact a searching problem, and it can be seen as a task similar to the compound query retrieval where the query comprises of multiple images of the same identity.

There are three specific objectives we aim to achieve in this thesis for compound query retrieval problems. First, we would like it to happen in real time with minimal memory storage required for the database. Second, we wish to achieve high retrieval accuracy. Third, we would like the retrieval system to be able to handle novel classes, *i.e.* not only the pre-defined classes that occur in the training dataset, but also any new classes defined by the users. The three aspects are discussed in detail in the following.

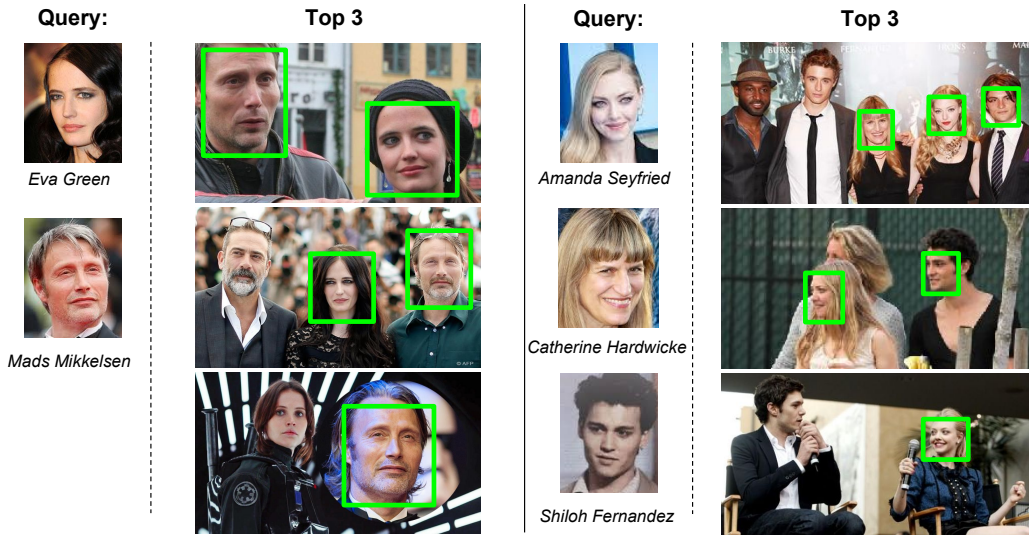


Figure 1.2: **Two examples of set retrieval.** The query faces are given on the left of each example column, together with their names (only for reference). The top ranked image in each case contains all the faces in the query. Lower ranked images partially satisfy the query, and contain progressively fewer faces of the query. The results are obtained using the compact set retrieval descriptor generated by the CNN architecture introduced in Chapter 6, by searching over 200k images of the *Celebrity Together* dataset introduced in Chapter 3.

Real-time and memory-efficient retrieval. If we consider the situation where the dataset is very large, containing millions or billions of images, two crucial aspects for real time retrieval are: first, an efficient algorithm is used when searching for images that satisfy the query. Second, all operations should take place in memory (not reading from disk). However, for problems like set retrieval, storing a fixed length vector for each individual face in memory is prohibitively expensive at this scale. But this cost can be significantly reduced if a fixed length vector is stored only for each set of faces in an image (since there are far fewer images than faces). As well as reducing the memory cost this also reduces the run time cost of the search since fewer vectors need to be scored. So, one of the questions we investigate in this thesis is the following: can we represent the multiple faces in an image as a *single vector* with little loss of set-retrieval performance? If so, then the cost of both memory and retrieval can be significantly reduced as only one vector per *image* (rather than one per *face*) has to be stored and scored.

Retrieval quality. For the problem of retrieving faces in places, simply combining the ranking lists of faces and places is far from ideal (*i.e.* the baselines in Chapter 5), since the face scores and place scores are not comparable, and the place descriptors and face descriptors are completely independent. To alleviate this problem, we wish to learn better place descriptors and face descriptors by coupling them, as well as the calibration between their scores. For the set retrieval problem, representing multiple faces using a single descriptor may lead to the drop in retrieval performance, and this is expected as information is lost. Therefore, in this thesis, we seek an embedding and aggregation method that can minimize the information loss or even improve the retrieval performance. Furthermore, this approach is applicable to different situations where the individual descriptors may come from different or the same classes. In the first case, descriptors of different classes should maintain their discriminability and the interference among descriptors should be as low as possible. While in the situation where descriptors are from the same classes, the method should focus more on the informative examples and down-weight the low-quality ones within each image set.

Open-world requirement. In order to be useful in practice, the retrieval system should not be restricted to a closed set of classes. Instead, it should be capable of handling novel classes, because in practice users may query for new object classes or identities that are not in the training dataset. Two methods are considered in this thesis. The first solution is an on-the-fly retrieval algorithm (Chatfield and Zisserman (2012), Chatfield et al. (2014b)) where, given any new query classes, the corresponding models can be learnt online using the images downloaded from the Internet. Another approach is to represent new queries as descriptors directly extracted from the images of the queries.

1.2 Key challenges

Image retrieval for compound queries is still a challenging and little researched problem. We list some key challenges in this research topic.

No suitable evaluation benchmarks. For the problem of retrieving faces in places (Chapter 5) and set retrieval (Chapter 6), there is no existing dataset suitable for evaluation. As a result, collecting and annotating new datasets is necessary.

Lack of quality training data. For the task of retrieving faces in places, it is not possible to gather enough training data due to the extremely low yield (Chapter 3). Furthermore, achieving a good coverage of faces in places is impossible as images of many face-place combinations simply do not exist – *e.g.* searching the Internet for 'Angela Merkel in an ice rink' gives no relevant images. These two problems can severely affect the performance of deep CNNs.

Coupling descriptors of different types. A naive approach to the problem of searching a face in a place would be to train a classifier for the face of interest, say 'Anne Hathaway', and the place, say a supermarket, obtain a ranked list of images for each based on the classifier score, and then combine the lists in some way using the rankings or scores (Shaw and Fox (1994)). However, such combinations often have poor performance as an image with a high classifier score for a place, may have a very low score (and so low rank) for a face, and vice versa (for example, images that contain very recognizable faces usually have a large face proportion and much less information about the background scene). Therefore, learning to couple the two different types of descriptors is crucial for the retrieval performance.

Real-time querying. For the set retrieval problems considered in Chapter 6 and 7, we wish to query in real-time. A straightforward method to tackle multiple images per set is to store per-image descriptors extracted from each face image (or frame in a video), and compare every pair of images between sets at query time (Schroff et al. (2015), Taigman et al. (2014)). However, this type of approach can be memory-consuming and prohibitively slow, especially for searching tasks in large-scale datasets. Therefore, an aggregation method that can produce a compact representation for the set or template is desired. Furthermore, this representation should support efficient computation of similarity and require minimal memory storage.

Maintaining individual discriminability in compact representation. As we mentioned previously, aggregation of descriptors for a speed gain inevitably leads to an information loss. To obtain the best of both worlds, we need to learn a discriminative set-level representation that preserves as much information as possible after aggregation.

Face recognition. Face recognition can be a challenging task as faces may have various poses, hair styles, expression, illumination and so on. Hence images of the same person can look very different. As a result, the learnt face representation should be discriminative for the person despite the variations.

Variation in image quality. For unconstrained template-based face recognition considered in Chapter 7, one of the key problems in real-world situations is that some faces in a template may be of low quality – for example, low resolution, or blurred, or partially occluded. These low-quality images are distractors and are likely to hurt the performance of the face recognition system if given equal weight as the other (good quality) faces. Therefore, the method that handles compound query retrieval should be able to reduce the impact of such distracting images and focus on the informative

ones.

1.3 Contributions

In this section, we list the main contributions made in this thesis.

1. Two large-scale face-related datasets. As described in Chapter 3, we build a *Celebrity in Places* (CIP) dataset and a *Celebrity Together* (CT) dataset, to facilitate research in compound query retrieval. The CIP dataset can be used as a benchmark for evaluating the performance of a system in retrieving a particular face in a particular place, whereas the CT dataset is an evaluation dataset for the set retrieval problem. The images of both datasets are downloaded from the Internet and manually annotated using Mechanical Turk.

2. An image synthesis engine for face replacement. In Chapter 4, to alleviate the lack of training data with both identity labels and place labels, we develop an image generation pipeline to automatically synthesize high quality images for any face-place combination. We show the synthetic data is good enough to train the CNN (proposed in Chapter 5) to perform the task well on real images.

3. A two-stream network for retrieving faces in places. In Chapter 5, we design a method to efficiently retrieve images containing a target person in a target place. More importantly, we learn a two-stream (one for face and place respectively) CNN to produce place descriptors that are face-aware and face-descriptor-aware. The architecture is simple yet powerful – two streams are only coupled by the loss. we demonstrate that our network significantly outperforms the strong baselines on the CIP dataset collected in Chapter 3.

4. An efficient and accurate ranking scheme for set retrieval. In Chapter 6, we operationalize the problem of retrieving a set of query identities by scoring each face in each photo of the collection as to whether they are one of the identities in the query. Each face is represented by a fixed length vector, and identities are scored by the similarity between the face descriptors and the query descriptors followed by logistic regression classifiers. Based on this setting, we propose to first have an initial ranking using the extremely efficient *descriptor-per-set* method (*i.e.* one fix-length descriptor per set), followed by a re-ranking step on the top images using the accurate *descriptor-per-element* method. The ranking scheme provides a trade-off between speed and accuracy, and is useful in practice.

5. A compact image representation for set retrieval. In Chapter 6, we design a CNN architecture to learn a compact set representation for the task of retrieving a set of identities. It can take any number of input faces and produce a compact fixed-length descriptor to represent the image set. Moreover, the learnt set-level representation contains the information of each element as much as possible, by minimizing the interference between elements during network training. We show that this powerful representation beats the baselines significantly on the CT dataset introduced in Chapter 3.

6. A CNN architecture for template-based face recognition. In Chapter 7, we adopt a network architecture similar to that in Chapter 6 for the task of unconstrained template-based face recognition. We show that this network embeds face descriptors such that the resultant template-descriptors are more discriminative than the original descriptors. The learnt representation is efficient in both memory and query speed, *i.e.* it only stores one compact descriptor per template, regardless of the number of face images in a template, and the similarity between two templates is simply measured as the scalar product (*i.e.* cosine similarity) of two template descriptors.

7. A novel CNN layer for noise removal. In Chapter 7, we extend the NetVLAD layer where it is designed such that it can decrease the contribution of noisy images to the final representation, and thus enable the CNN to focus on the more informative images. Note that this property of downweighting low-quality images emerges automatically, without explicitly providing the quality of images to the CNN. With this novel layer, our networks outperform state-of-the-art methods by a large margin on currently the most challenging public face recognition benchmarks.

1.4 Publications

The publications (including the ones under review) related to the research conducted in this thesis are listed in the following.

- Y. Zhong, R. Arandjelović, A. Zisserman

  Faces in Places: Compound query retrieval. BMVC, 2016.

- Y. Zhong, R. Arandjelović, A. Zisserman

  Compact aggregation for set retrieval. ECCV Workshop, 2018. *Best Paper Award*

- Y. Zhong, R. Arandjelović, A. Zisserman

  GhostVLAD for set-based face recognition. ACCV, 2018.

The publication shown below is excluded from this thesis, since the topic of that paper is not directly relevant to the thesis topic of compound retrieval.

- M. Malaspina, Y. Zhong

  Image-matching technology applied to fifteenth-century printed book illustration. Lettera Matematica, Springer, 2017.

Chapter 2

Literature review

In this chapter we review some recent work related to this thesis. In Section 2.1, we first review the development of deep learning and convolutional neural networks (CNN). In the past few years, deep learning has surpassed many traditional learning algorithms and hand-crafted features and achieved the state-of-the-art performance in many machine learning areas. We focus on CNNs as they are the most widely-used deep neural networks in computer vision, upon which the majority of our work is built.

Section 2.2 reviews methods for text retrieval and how compound queries are handled, which are similar to the baseline compound query retrieval methods in the visual domain considered in this thesis. Section 2.3 reviews the literature on the development of descriptor aggregation methods which are used to obtain compact image representations for image retrieval. In Chapter 6 and 7, we borrow these ideas of image retrieval methods using compact image representation. In the second half of Section 2.3, we review methods on category-level image retrieval, from methods using shallow features to those based on deep learning, which form the basis of compound query retrieval investigated in this thesis,

Lastly, in Section 2.4, we move on to the work on face recognition, especially on template-based unconstrained face recognition. We also list some recent face datasets.

2.1 Deep learning

In this section, we review recent developments in deep learning, especially on convolutional neural networks (CNN) which are widely used in computer vision. We then review some applications of deep learning, including face recognition and scene recognition.

Deep learning is a form of machine learning that makes use of multiple layers of non-linear transformations to extract feature representations of low to high level concepts. This stack of layers forms a deep neural network which can model complex relationships and learn powerful data representations. Neural networks are in fact inspired by interaction of neurons in a biological nervous system. For example, visual concepts grow gradually from low level to high level in human brains, which is what neural networks mimic. Another example is that a neuron in human brains only generates a spike of activity to an axon when it receives enough charge, and this is effectively what activation functions in neural networks aim to achieve.

There are various types of neural networks, such as feed-forward neural networks in which data passes through without looping, and recurrent neural networks which take inputs in a sequence and update their hidden units based on the past inputs. CNNs are a particular type of feed-forward neural networks. One of the main reasons that CNNs work so well is the advantage of end-to-end training - they are trained exactly for the task at hand. This is in contrast to the previous hand-crafted methods where people design the feature extraction rules manually. In the following, our review will focus on CNNs which play an important role in this thesis.

2.1.1 Convolutional neural networks

CNNs are a particular type of deep neural networks which contain multiple convolutional layers. Typically, a convolutional layer consists of a set of filters (*i.e.* filter

bank), and it performs a convolution on the input image or the feature maps with the filters. These replicated filters achieve equivariance of the neuron activities with respect to the input. More clearly, equivariance means that the learnt representation changes according to the change in the spatial location of the features. This property is desired as we wish the representation to be consistent for the same patterns or objects with regards to their spatial location. Specifically, convolutional layers achieve equivariance to translation, but not scale and rotation. Furthermore, this design greatly reduces the number of weights to be learnt compared to that of fully-connected (FC) layers. A FC layer is a layer in which neurons have connections to all activations in the previous layer. It therefore tends to have a large number of weights. A convolutional layer or a FC layer is usually followed by a nonlinear unit (*i.e.* activation function) which enables the CNN to learn nonlinear transformations. The most common activation functions include the sigmoid function, the tanh function and the rectifier linear unit (ReLU). In particular, the ReLU is used in most modern CNNs. The ReLU enables CNNs to learn more discriminative representations by injecting non-linearity. Namely, it sets all the negative activations to zero. More importantly, the ReLU does not have the vanishing gradient problem as with the sigmoid and tanh function, since the gradients of positive inputs are always equal to 1. Besides, CNNs usually include multiple pooling layers (max or average pooling) which down-sample the intermediate feature maps in order to capture high level semantics of images. Max-pooling layers can achieve spatial invariance of neuron activities with respect to the image, by capturing the maximum in each pooling region. Lastly, a dropout layer (Hinton et al. (2012)) is a regularization layer for reducing overfitting in CNNs by preventing complex co-adaptations on training data.

Training CNNs. CNNs have trainable parameters (*i.e.* convolutional filter weights and FC weights), and we wish to find the optimal values of the weights for the end task which is defined by the loss function. During training, we minimize the loss with

respect to weights by back-propagation algorithms (Rumelhart et al. (1986)). Optimization can be done via 'gradient-based' optimization schemes such as stochastic gradient descent (SGD), Adagrad (Duchi et al. (2011)), AdaDelta (Zeiler (2012)), RMSprop (Tieleman and Hinton (2012)) and Adam (Kingma and Ba (2014)). Based on the chain rule, back-propagation enables the computation of the gradients of weights of each layer in a CNN with respect to the learning objective. For efficient optimization, SGD is a commonly used method which updates the weights of the network after each mini-batch (*i.e.* a small subset of the entire training set). Momentum (Rumelhart et al. (1986)) is often used together with SGD in order to speed up the convergence, as it stabilizes the direction of the updates towards the optimum in the weight space.

Training with synthesized images. Since CNNs are data-hungry, and obtaining a large number of labeled images requires a large amount of human labour and time, training CNNs on purely synthetic data and testing on real data has been done in the past for other tasks, such as text recognition and localization (Gupta et al. (2016), Jaderberg et al. (2014, 2016), Wang et al. (2012)), learning a universal feature extractor (Bilen and Vedaldi (2017)), training generative models (Dosovitskiy et al. (2015b), Yildirim et al. (2015)), and so on. Based on the success of some synthetic character datasets (de Campos et al. (2009)), Jaderberg et al. (2016) created a synthetic word data generator to emulate the distribution of scene text images. Jaderberg et al. (2016) also presented an end-to-end system for text localization and recognising text in natural scene images, and the networks in the system are trained purely on images generated by a synthetic text generation engine. Gupta et al. (2016) proposed a fast and scalable pipeline to generate synthetic images of text in clutter. This pipeline places synthetic text over existing background images according to the local 3D scene geometry, and thus the resulting images look natural without artifacts. Their proposed Fully-Convolutional Regression Network for text detection and bounding-box regression is trained using these synthetic images. Apart from text localization and

recognition, synthetic images have been used in other tasks. Dosovitskiy et al. (2015a) use synthetic chair renderings to train dense optical flow regression networks. The synthetic chairs are generated by applying affine transformations to images downloaded from Flickr and using a public set of 3D chair models. Synthetic data has also been used to train generative models. For instance, Dosovitskiy et al. (2015b) train a network that can be used to generate objects given object type, colour and viewpoint. As another example, Yildirim et al. (2015) regress pose parameters from face images using deep CNN features trained on synthetic face renderings. Another related topic on image synthesis is face replacement. Face replacement or swap has been investigated a lot before deep learning. For instance, Bitouk et al. (2008) proposed a pipeline to realize face swapping, in which they search for faces that look similar to the original face based on a variety of features such as pose, image resolution, lighting, colour and so on. In Chapter 4, we build an image synthesis pipeline based on the work mentioned above.

2.1.2 Network architectures

In this section, we briefly review some well-known CNN architectures which are highly related to this thesis.

AlexNet. Deep neural networks generally require a large amount of labelled training data, therefore they were underestimated by people for years due to the lack of sufficient training data. However in 2012, thanks to ImageNet (Deng et al. (2009)), it was shown possible to train deep CNNs and achieve prominent performance. The first successful CNN was AlexNet (Krizhevsky et al. (2012)).

AlexNet is similar to LeNet introduced by LeCun et al. (1989) which was used for digit recognition, but has a deeper structure (*i.e.* contains more layers) and more filters per layer. Specifically, AlexNet consists of five convolutional layers and two

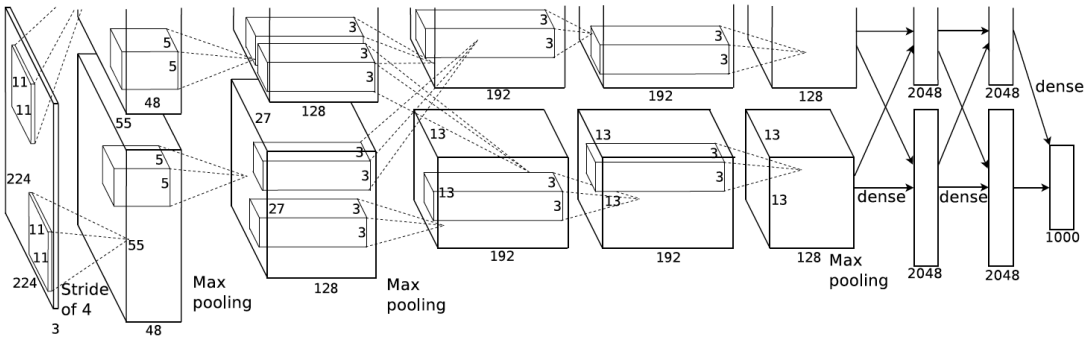


Figure 2.1: **Architecture of AlexNet.** Figure is taken from (Krizhevsky et al. (2012)).

fully-connected (FC) layers, with an additional FC layer at the end as classifiers. Each convolutional layer and FC layer is followed by a ReLU layer.

For the purpose of down-sampling, there are max-pooling layers before the second and third convolutional layer and brefore the first FC layer. The input of the network is an image of size 227 by 227, and the output feature before the classification layer is 4096-D. Dropout layers (Hinton et al. (2012)) are used after each FC layer in order to alleviate the problem of overfitting.

VGG networks. Another widely used type of deep CNN is VGGNet proposed by Simonyan and Zisserman (2015). As deeper networks are expected to have larger learning capacity and thus perform better, Simonyan and Zisserman (2015) proposed to replace large filters by very small filters (*i.e.* filters with 3×3 receptive fields) and, together with a careful initialization process, they managed to push the depth of CNN to 16 and even 19 layers. This is achieved by the fact that a 5×5 filter can be represented by a stack of two 3×3 filters, and a 7×7 filter can be represented by a stack of three 3×3 filters.

By replacing large filters with smaller ones, more nonlinear rectification layers can be used instead of a single one as in previous CNNs such as AlexNet and ZFNet (Zeiler and Fergus (2013)), and thus more discriminative functions can be learnt.

Furthermore, decomposing filters into a stack of 3×3 filters reduces the number of parameters of the network. For instance, a 7×7 convolutional layer with C channels has $49C^2$ parameters, while a stack of three 3×3 convolutional layers only requires $27C^2$ parameters. Apart from using smaller receptive fields and max-pooling layers with smaller stride (of 2), VGG networks follow a similar architecture as AlexNet, while achieving a much higher accuracy in tasks like classification and localization.

Inception Networks. At the same time, a more complex network architecture named 'inception network' was introduced by Szegedy et al. (2015). This architecture is inspired by multi-scale processing and designed based on the Hebbian principle. As Figure 2.2 shows, an Inception module consists of a 1×1, 3×3, 5×5 convolutional layer and a max-pooling layer. A 1×1 convolution happens before the 3×3 and 5×5 convolutions and after the max-pooling layer, which is used to reduce the number of channels of the feature maps resulting in less computations. The whole Inception network is a combination of all these Inception modules with their output filter banks concatenated into a single output vector forming the input of the next module. Further work by (Szegedy et al. (2016)) explored ways to increase the computational efficiency and the scalability of networks. To achieve such objectives, they proposed Inception-v2, which factorizes the convolutions in the original Inception network with some aggressive regularization. For example, a 3×3 filter is factorized into a 1×3 filter followed by a 3×1 filter. For Inception-v3, they further improved the network by adding a Batch-Normalization layer in the Auxillary Classifiers, and performed label smoothing during training. In the next year, Szegedy et al. (2017) proposed Inception-v4 and Inception-ResNet, in which residual learning is combined with inception blocks.

Residual networks. The depth of CNNs was further increased dramatically by He et al. (2015). They argued that deep networks should be able to achieve at least equal or even better results than shallow networks due to the fact that deeper networks

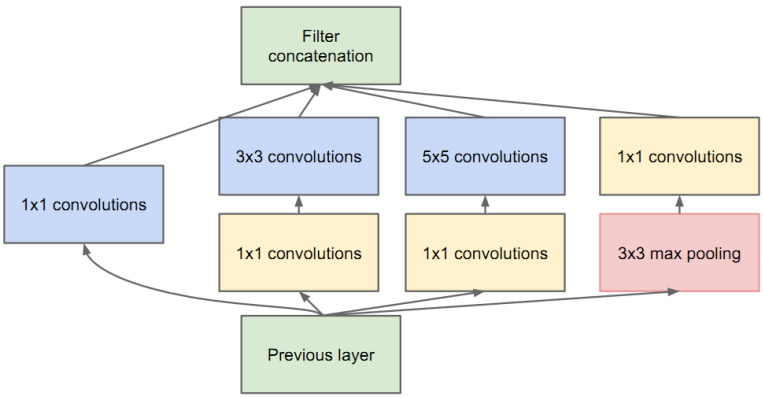


Figure 2.2: **Inception module with dimension reductions.**

should be able to compute the same functions as the shallower ones. However, this is not observed in practice due to difficulties in training. Therefore, to make the training easier, they proposed deep residual learning architecture (*i.e.* ResNet) to learn the residual mapping instead of the desired mapping directly. This can be achieved by introducing skip connections between convolutional blocks (*i.e.* stacks of convolutional layers). With this modification, it is easier to propagate information through the layers, as they argue that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping.

Similar to Inception networks, they also replaced the fully-connected layers on top of the network by a global average pooling layer to reduce the size of the network. He et al. (2015) were able to train such residual networks with up to a thousand convolutional layers. Typically, a residual network has either 34, 50, 101 or 152 layers. There have been some followup works such as ResNeXt (Xie et al. (2017)) and Inception-ResNet (Szegedy et al. (2017)).

Squeeze-and-Excitation networks. Hu et al. (2018) proposed to add a neural network unit called 'Squeeze-and-Excitation' (SE) block which adaptively recalibrates channel-wise feature responses in neural networks by explicitly modelling interdependencies between channels. Specifically, the feature maps first go through a squeeze

operation which aggregates across spatial dimensions to produce a channel descriptor. This descriptor embeds the global information of channel-wise activations, enabling information from the global receptive field of the network to be leveraged by lower layers. The squeeze operation is followed by an excitation operation, in which the global information is used to modulate the input feature maps. The feature maps are then recalibrated to produce a stack of new feature maps that are passed into subsequent layers. Networks with SE blocks are termed 'SENet'.

The abovementioned AlexNet, VGGNet, ResNet and SENet are used as feature extractors or backbone networks in most experiments conducted in this thesis.

2.2 Retrieval for Compound queries

There are not many works on retrieving images using compound queries. However, it is actually a common problem in some other modalities, such as text retrieval which we briefly review in this section.

2.2.1 Text retrieval

Text retrieval has been developed for many decades. In the following, we review two well-known text retrieval models.

Vector space model. A well-developed and widely-used approach for text retrieval is the 'Bag-of-Words' method (BoW (Manning et al. (2008))), which can be considered as a vector space model (Salton and McGill (1986)). In this approach, every document in the database is viewed as an orderless 'bag' of words, and represented as a high-dimensional histogram vector recording the word occurrences. Usually the dimension of this vector is equal to the number of words in a language. Since some words (such as 'a' and 'the') occur frequently in the documents even though they do not contain

much information, they should be down-weighted in the vector representation. The *tf-idf* weighting scheme (Manning et al. (2008)) is a well-known method to account for this frequency imbalance. Specifically, the words in the vector representation are weighted by their term frequency (tf) and the inverse document frequency (idf), and the inverse document frequency is computed as $log(N_D/N_i)$ where N_D is the number of database documents and N_i is the number of documents in which the ith word occurs. With such a *tf-idf* weighted BoW representation, the similarity between documents is simply measured by the cosine similarity.

Boolean model. Another most-adopted stream of text retrieval methods is the Boolean model (Lancaster and Gallup (1973), Lashkari et al. (2009)), which is based on Boolean logic and classical set theory. In this model, each document is represented by a subset of the whole set of index terms that characterize or describe the documents. Given a query, it first translates the query into a set of index terms, and then ranks the documents according to the size of the intersection between the index terms of the query and those of the documents. This model has clean and intuitive formalism, and is easy to implement. However, it suffers some drawbacks such as that all terms are equally weighted, unlike in a vector space model.

2.2.2 Compound query for text retrieval

The queries in text retrieval often contain more than one word, as people tend to use multiple key words (which may not be continuous or related) in order to cover more aspects of the target documents. This therefore can be seen as a compound query retrieval problem, as the retrieval system needs to handle multiple individual input query words or phrases. We now discuss how the two different models handle compound queries. For a vector space model (Salton and McGill (1986)) such as BoW, this is solved naturally, as the histogram vector in the BoW model is designed to

record the occurrences of all the words in the language. In this case, the query vector of multiple words is handled simply by a vector with multiple non-zero weighted elements. However, a shortcoming of the vector space model is that it can not guarantee that the documents containing all the query words are ranked higher than the rest, as the words are weighted. On the contrary, the Boolean model (Lancaster and Gallup (1973), Lashkari et al. (2009)) always ranks the documents that contain all the query texts highest (if there are any) due to the fact that all terms have equal weights. Unfortunately, these methods are not applicable to image retrieval as images are not composed of a set of fixed words or terms.

2.2.3 Learning on sets

The set retrieval problem considered in Chapter 6 is an example of compound query retrieval, in which we need to handle sets of vectors. There are also some works that explicitly deal with sets of vectors. We briefly review these works in this section. Kondor and Jebara (2003) developed a kernel between vector sets by characterising each set as a Gaussian in some Hilbert space. However, their set representation cannot currently be combined with CNNs and trained in an end-to-end fashion. Recently, based on deep learning, Zaheer et al. (2017) investigate a permutation-invariant objective functions for operating on sets. Their method boils down to average pooling of input vectors, which we compare to as a baseline in Chapter 6. Hamid Rezatofighi et al. (2017) consider the problem of predicting sets, *i.e.* having a network which outputs sets. This is different from the case we consider in Chapter 6 and Chapter 7, where a set of elements is an input to be processed and described with a single vector.

2.3 Image retrieval with a compact representation

In this section, we first review some work on descriptor aggregation for compact image representation used in image retrieval, and then proceed to methods on category-level

image retrieval. Although all the previous methods aim for image retrieval with a single query, they form a solid basis for the compound query retrieval which we tackle in this thesis. Lastly, we review some methods that specifically deal with sets, since the queries considered in Chapter 6 are in the form of image sets.

2.3.1 Descriptor aggregation

Aggregating local descriptors is a common strategy to produce image-level descriptors for image retrieval. The work we do in chapter 6 is inspired by this method and proposes a domain shift to our set retrieval work. Therefore in this section, we review methods that aggregate local descriptors assigned to the same clusters in the descriptor space to learn compact image representations.

Bag-of-Words image representation

The work of Sivic and Zisserman (2003) is the first one to apply text retrieval ideas to image search. For text retrieval framework, text documents are naturally broken down into words, whereas for image retrieval, no such natural segmentation exists for images. To overcome this obstacle, Sivic and Zisserman (2003) take advantage of the fact that a set of local descriptors can be extracted from an image, and introduce the concept of 'visual words' where local descriptors are vector quantized into a predefined vocabulary obtained using k-means clustering. Specifically, images are represented with sparse bag-of(-visual)-words histograms weighted by the tf-idf scheme. With this image representation, retrieval is performed efficiently through by using an inverted index. This bag-of-words model can be seen as the first image encoding method that produces a single vector for each image.

However, storing per-feature level information of local descriptors such as the Bag-of-Words encoding (Sivic and Zisserman (2003)) can become unaffordable in terms of memory usage for very large image corpora, since it requires a large vocabulary

to ensure its good performance. Therefore, retrieval approaches that are scalable to tens of millions of images draw much attention. By representing each image as a very compact global descriptor and discarding local descriptor information, the database storage can be significantly reduced. Furthermore, retrieval can be performed simply by a multiplication between the matrix of the image database and the query image descriptor (*e.g.* Jégou and Chum (2012)) or by efficient nearest neighbour search (*e.g.* Jégou et al. (2011a), Muja and Lowe (2009)).

Some methods such as (Chum et al. (2007, 2008), Jégou et al. (2009a) and Jégou and Chum (2012)) obtain compact image representation by compressing BoW representation of an image, whereas another direction of research is to aggregate local descriptors followed by dimensionality reduction. In particular, the VLAD encoding (Jégou et al. (2010)) which falls into this category of compact dimensionality performs very well. In the following, we will introduce the Fisher Vector (FV) encoding (Perronnin and Dance (2007)) and the VLAD encoding in detail.

Fisher vectors

The Fisher Kernel was first introduced by Jaakkola and Haussler (1998) as a similarity measuring function operating on two sets of measurements. Perronnin and Dance (2007) applied the Fisher Vector, which is an improved form of Fisher Kernel, for image classification. Fisher vector was also applied to image retrieval by Jégou et al. (2010) and Perronnin et al. (2010a). Compared to sparse and high-dimensional BoW representations with a large vocabulary, the Fisher Vector is a dense and compact image representation. It assumes that local descriptors of an image are independent samples from a Gaussian Mixture Model (GMM). It is constructed by taking the derivative of local descriptors with respect to the mixture weights, means and the covariances of the GMM and by concatenating the derivatives.

Perronnin and Dance (2007) revealed that computing gradient with respect to the mixture weights corresponds to counting the number of descriptors assigned to each component (similar to the BoW representation). The derivative with respect to the mean of a GMM, on the other hand, corresponds to the weighted aggregation of all difference between local descriptors and the mean of that component. Further improvements to image retrieval using FVs include (Perronnin et al. (2010c) and Perronnin et al. (2010a)). Perronnin et al. (2010c) illustrated that irrelevant descriptors are discarded automatically under some assumptions, leaving only the image-related information. Perronnin et al. (2010a) incorporated tf-idf weighting when computing the similarity of two FVs, and applied power normalization (similar to Jégou et al. (2009b)) in order to reduce the effect of burstiness of descriptors.

Later, the Fisher Vector obtained a more compact form, as the derivatives with respect to both the Gaussian variance (Jégou et al. (2010, 2012)) and mixture weights (Perronnin et al. (2010c)) do not provide much useful information for image retrieval tasks. Therefore, the final dimensionality of the Fisher Vector became $2KD$, where K is the number of mixture components (similar to the 'visual words' in the BoW encoding) and D is the dimensionality of the local descriptors.

Vector of locally aggregated descriptors

The Vector of Locally Aggregated Descriptors (VLAD) is a global image descriptor aggregated from local descriptors, proposed by Jégou et al. (2010). It was motivated by the Fisher Vector, while the major difference is that it has visual vocabulary constructed by clustering local descriptors, instead of using a GMM for the Fisher Vector. Specifically, it hard-assigns local descriptors to the visual words of the pre-built vocabulary, and aggregates all the differences between local descriptors and its assigned visual word (cluster centre) – namely residuals. As it uses hard-assignment, it is faster than the FV encoding. The final dimension of the VLAD descriptor is

Figure 2.3: **Images and corresponding VLAD descriptors (with K=16 centroids).** The components of the descriptor are represented like SIFT, with negative components in red. Each box shows a residual of each cluster. Image taken from (Jégou et al. (2010)).

KD (similar to the Fisher Vector), where K and D are the number of visual words in the vocabulary and the dimension of the local descriptors respectively. Images with corresponding VLAD descriptors are displayed in Figure 2.3.

The major improvements for the VLAD happens in the normalization scheme. Signed Square Rooting (SSR) normalization, which is used to address bursty features, is first used by Perronnin et al. (2010a) for Fisher Vectors, and it is then also applied to VLAD vectors (Jégou and Chum (2012), Jégou et al. (2012)). Subsequent to SSR, Arandjelović and Zisserman (2013) introduced intra-normalization to address the problem of bursty features raised by Jégou et al. (2009b), *i.e.* avoiding domination of the similarity computation for VLAD vectors caused by a few large components in the VLAD vectors. In the same work, Arandjelović and Zisserman (2013) also introduced RootSIFT which significantly improves the retrieval performance, and a cluster adaptation algorithm to alleviate the problem of inconsistency between the cluster centres and the dataset. It is achieved without re-performing clustering by computing the adapted cluster centres as the mean of all local descriptors in the dataset.

In the original scheme, descriptors closer to the cluster centre have smaller residuals, whereas Delhumeau et al. (2013) proposed to L2-normalize each residual to contribute

equally to VLAD. Apart from sum aggregation as in the original VLAD, Chen et al. (2011) experimented on mean and median aggregation of residuals for each cluster, and revealed that mean aggregation works better than the others. They also proposed to remove descriptors that are close to the boundaries of clusters due to their instability of assignment.

Delhumeau et al. (2013) argued that power normalization is not rotationally invariant in the descriptor space, unlike other encoding steps. Therefore rotation has an impact on retrieval performance. They propose to apply a different rotation to each 'visual word' based on the Principal Component Analysis (PCA) basis of the 'visual word'. This PCA basis is expected to capture the main bursty patterns of the 'visual word', whereas one rotation per 'visual word' encourages the capture of more bursty patterns.

While PCA is a common method to reduce the dimensionality of VLAD vectors, Jégou and Chum (2012) proposed a different dimensionality reduction approach. They use the random initialization of k-means to capture multiple visual vocabularies and then compute multiple VLAD vectors using these vocabularies. The VLAD vectors are then concatenated followed by PCA and whitening to give the final representation. The resulting VLAD vectors suffer less quantization errors due to the use of multiple vocabularies.

Other methods which can improve retrieval performance include extracting dense local descriptors (Delhumeau et al. (2013), Zhao et al. (2013)) and storing geometric information in CVLADs (Zhao et al. (2013)). This is achieved by computing one VLAD vector per dominant orientation and concatenating them to obtain Covariant-VLAD (CVLAD) . The similarity between CVLADs is measured as the largest dot product among all possible rotations. Jégou and Zisserman (2014) proposed a different embedding method – T-embedding that aims at regularizing the individual contributions of the local descriptors in the final representation. Comparing to original local descriptors, the T-embedding increases the contrast between the similarity

scores of matching and mismatching local descriptors. Another very recent approach to improve image retrieval is the memory vector formulation of Iscen et al. (2017). They design an architecture for the database, which consists of several memory units, and each unit summarizes a fraction of the database by a single vector. The similarity of a query to one of the vectors stored in the memory unit is measured by a simple correlation with the memory unit's representative vector.

Aggregating CNN activations

With the success of deep CNNs in image classification (Krizhevsky et al. (2012)), CNNs can be considered as a powerful feature extractor used for many other tasks like image retrieval (Arandjelović et al. (2016), Babenko and Lempitsky (2015), Babenko et al. (2014), Gordo et al. (2016), Kalantidis et al. (2016), Mohedano et al. (2016, 2017), Radenović et al. (2016), Razavian et al. (2014, 2016), Tolias et al. (2015)). In this section, we discuss these image retrieval (mostly instance-level) approaches using deep CNN representations, most of which aggregate the representations of sub-patches on either the image or the feature maps from a CNN, to obtain the final image representation. In particular, our work for set retrieval (Chapter 6) is inspired by the aggregation of deep representations using NetVLAD (Arandjelović et al. (2016)).

As a straightforward method for using CNNs for image retrieval, Babenko et al. (2014) and Razavian et al. (2014) directly use the activations of the last fully-connected layer (FC) in the CNN as a global image representation. Gong et al. (2014) improve retrieval performance by pooling multiple CNN features from the last FC layer of different scales of the input image.

Instead of using the activations of the last fully-connected layer, Razavian et al. (2014) suggested that it makes more sense to extract the feature maps of the last convolutional layer, as they contain high-level semantic information of the whole image while preserving the representation of sub-patches of the image, which is useful for instance

retrieval. Razavian et al. (2014) applied global max-pooling on the feature maps to obtain the image representation. Similarly, Babenko and Lempitsky (2015) revealed that sum-pooling over the feature maps of the last convolutional layer achieves the best performance in image retrieval. They also suggested that using max-pooling, the scalar product between two image descriptors corresponds to the simple match kernel between two images. The obtained descriptor is then L2-normalized followed by PCA and whitening to form the final image representation.

Although the above-mentioned approaches match or even outperform the hand-crafted shallow image representations, they are not as good as models with spatial ranking, *i.e.* models that take into account the spatial configuration of the local descriptors. Tolias et al. (2015) proposed to extract multiple descriptors corresponding to different CNN response map regions at different scales, namely regional maximum activations of convolution (R-MAC), followed by PCA, whitening and sum-pooling. This R-MAC image representation can be seen as a descriptor aggregation approach operating on deep CNN feature maps. Furthermore, they proposed to use the integral image to perform approximate max-pooling over feature maps, in order to enable fast computation. The integral image also acts as a localizer of objects. This localization of objects can be used in the re-ranking stage, in a similar way as spatial re-ranking. Gordo et al. (2016) improve the R-MAC representation by training the deep network with a triplet loss in an end-to-end fashion. They also predict the location of informative regions by training a region proposal network. As another method that operates on CNN activations for object retrieval, Mohedano et al. (2016, 2017) proposed utilizing the BoW method to encode the convolutional features of CNNs. Furthermore, they used the assignment map for spatial re-ranking, and achieved promising results.

In contrast to those methods which consist of region selection and direct max-pooling of corresponding feature maps, Arandjelović et al. (2016) proposed NetVLAD – a CNN version of the VLAD encoding (section 2.3.1) – in order to benefit from the

great body of work in traditional image retrieval methods using shallow features. This VLAD-like layer is differentiable and fully trainable for end-tasks like place recognition (Arandjelović et al. (2016)), instance retrieval (Arandjelović et al. (2016)), action classification (Girdhar et al. (2017)) and video classification (Miech et al. (2017)). It is shown that NetVLAD outperforms sum and max pooling of CNN activations for the same dimensionality of image descriptors. To enable the NetVLAD layer to be differentiable, they replaced the hard-assignment of descriptors to clusters (as in the original VLAD) by a soft-assignment. Moreover, they separate the assignment weights and the cluster centres to provide greater flexibility of the layer.

2.3.2 Category-level object retrieval

Image search and annotation for *single queries* has gained attention for many years, including finding particular people by their face (Chen et al. (2013), Parkhi et al. (2015), Schroff et al. (2015), Wu et al. (2011)), or specific objects (Arandjelović and Zisserman (2012b), Jégou et al. (2011b)), object categories (Chatfield et al. (2015), Perronnin et al. (2012)), or scene categories (Zhou et al. (2014a)), or other types of objects such as handwriting (Chatbri et al. (2018)) and texts in e-business images (Zhou et al. (2018)). In this section, we review the development of category-level object retrieval.

Image encoding methods. Before going into category retrieval, we first take a look at the ways that people encode images in image classification (Perronnin et al. (2010a,b,c, 2012)), which play a very important role in category retrieval. Those methods in general train classifiers for each category first, then rank the database images according to their classifier scores. In (Perronnin et al. (2010a)) the Fisher Vector is used as image representation (see Section 2.3.1 for details), while in (Perronnin et al. (2010c)), the Fisher Kernel is improved for large-scale image classification. As kernel

machines are difficult to scale to large datasets, Perronnin et al. (2010b) proposed to perform an explicit mapping of the data and to directly learn linear SVM classifiers in the new space. Furthermore, a benchmark of several objective functions for large-scale image classification is proposed in (Perronnin et al. (2010a)). In object categories retrieval, different from object instances retrieval, features are extracted in a dense grid manner. Those local features are then encoded to produce an image-level descriptor on which classification or retrieval is performed. In (Chatfield et al. (2011)), a rigorous evaluation of several popular encodings for BoW (Sivic and Zisserman (2003)) models is carried out, including histogram encoding, kernel codebook encoding (Philbin et al. (2008), van Gemert et al. (2008)), Locality-constrained linear (LLC) encoding (Wang et al. (2010)), improved Fisher encoding (Perronnin et al. (2010c)) and the super vector encoding (Zhou et al. (2010)). For more details about the Fisher vector encoding, see Section 2.3.1.

On-the-fly category retrieval. In category-level object retrieval, the database can not be queried with the input query image directly as in instance retrieval. This is where on-the-fly methods come into play. On-the-fly object category retrieval is achieved by bridging instance-level object retrieval and image classification (Chatfield and Zisserman (2012), Chatfield et al. (2014b)). It trains classifiers for a query category as in image classification, using some images downloaded from image search engines at run-time, and then applies the classifiers on the database image descriptors for searching. This novel on-the-fly category-level object retrieval system (Chatfield and Zisserman (2012)) is shown in Figure 2.4. The proposed retrieval system can search for object categories in a large unannotated image database in real time. More importantly, it overcomes the closed world problem where object category recognition systems are restricted to only those certain classes that occur in the manually curated training datasets. This is achieved by enabling both fast learning of any novel object category model, and fast retrieval of images from the large datasets. The on-the-fly

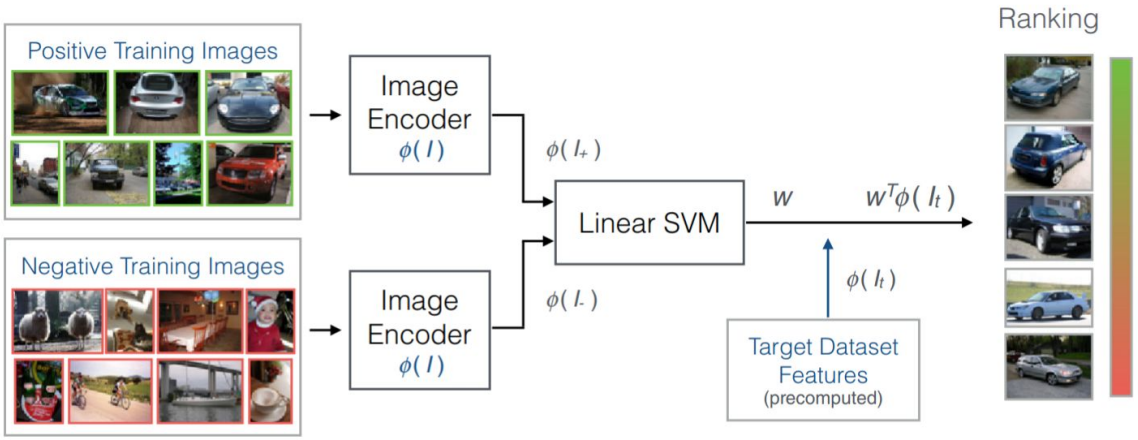


Figure 2.4: **Object category retrieval pipeline.** Positive and negative training images can be either obtained from the target dataset, or from some other source such as Google Image search. A linear SVM is then trained online using these images, and it is then applied on the dataset descriptors. Figure taken from (Chatfield et al. (2014b)).

retrieval procedure consists of four stages. First, positive training images are obtained using Google image search starting from a text query. Second, a descriptor is computed for each of the positive images. Third, a linear SVM classifier is trained using extracted positive image descriptors and a pool of negative images with pre-computed descriptors. Lastly, database images are ranked by the learnt classifier. This category retrieval system enables users to search for new categories in real-time since training and applying SVMs is fast.

From shallow to deep. With the success of deep CNNs in image classification tasks (Krizhevsky et al. (2012), Simonyan and Zisserman (2015)), there are many works in comparing deep CNN features with shallow hand-crafted features. Chatfield et al. (2014a) conducted a careful evaluation of CNNs and explore and compare different deep architectures. They show that deep architectures outperform the shallow methods by a large margin in image classification, and their performance can be further improved by fine-tuning. In order to benefit from the great power of deep learning, Chatfield et al. (2014b) proposed to use a CNN with GPU training to build an on-the-fly category retrieval system. The retrieval system shows superior perfor-

mance to previous systems in terms of precision, memory requirements and speed. The pipeline is divided into a CPU-based front-end and a GPU-based back-end. In the off-line pre-processing, the CNN features for the target dataset images are pre-computed using the CNN architecture with a 128-dimensional feature output (Chatfield et al. (2014a)). The negative image feature features are also pre-computed. For the online part, the back-end runs in parallel with the front-end, and it trains the ranking model (*i.e.* the SVM as before) and applies the model to the dataset. Importantly, the training starts as soon as the first positive image has been downloaded and is received from the front-end. Meanwhile, the front-end will continue downloading new images from the Internet, therefore the size of the positive image pool and the diversity of the extracted crops is constantly increasing.

The retrieval work revised in this section form a solid base upon which the retrieval algorithms developed in this thesis are built.

2.4 Face recognition

Face recognition plays an important role in this thesis, as we take it as our test bed in most of the work done here. In this chapter, we review the development of face recognition based on deep learning. In particular, we first briefly review single-image based face recognition, and then focus more on face recognition based on image sets.

2.4.1 Face recognition based on a single image

Face recognition has been developed for many years. As the performance of deep learning surpassed all the hand-crafted, shallow features in image classification, people also started to apply it to face recognition in 2014 (Sun et al. (2014b), Taigman et al. (2014)). Since then, the face recognition community follows the development of deep network architectures in image classification (see Section 2.1.2), and applies

more powerful CNNs to the field. Apart from different feature extractors, most face recognition researchers focus on the improvement of loss functions. Therefore, for image-based face recognition, we mainly review the development of loss functions in the following. To better differentiate these loss functions, we categorize them into two types: Euclidean-distance-based losses and angular-margin-based losses.

Euclidean-distance-based losses. Euclidean-distance-based losses aim to increase the inter-variance between different classes while decreasing the intra-variance within the same classes. A representative loss is triplet loss, which enforces a margin between the distance between an anchor and a positive sample and the distance between the anchor and a negative sample. The first work that applied deep learning and triplet loss to face recognition is (Schroff et al. (2015)), and other work including (Crosswhite et al. (2017), Parkhi et al. (2015), Sankaranarayanan et al. (2016)) also use this loss. Another popular loss that falls into this category is the contrastive loss. It is applied to face recognition in (Sun et al. (2014a, 2015a,b, 2016)). The main difference between the contrastive loss and the triplet loss is that it considers the absolute distance between positive pairs as well as negative pairs. However, these two losses suffer instability in training due to the difficulties in selecting samples and margins. Wen et al. (2016) propose the 'centre loss' that compresses intra-variance while enabling a more stable training procedure. Similar work to this include (Deng et al. (2017), Wu et al. (2017)).

Angular-margin-based losses. In 2017, researchers started to investigate the normalization of features and/or weights of CNNs in face recognition, and that is when angular-margin-based losses come into play. The motivation of this loss is to separate the learned face features by a large angular/cosine distance. Liu et al. (2016) propose a L-Softmax loss by reformulating the softmax loss into a large-margin version. To overcome the difficulty in the convergence of the L-Softmax loss, the A-Softmax loss

(Liu et al. (2017a)) is proposed which normalizes the weights such that the normalized CNN features lie on a hypersphere. Similarly, Liu et al. (2017b) propose SphereNet that has hyperspherical convolution in the network and effectively separates the features of different classes by a margin in the angles. Another very recent work (Zheng et al. (2018)) introduces a convex loss function – Ring loss – that enables the network weights to learn to produce features of a certain norm.

The face recognition techniques and losses based on single image are considered as strong baselines in Chapter 7.

2.4.2 Face recognition based on image sets

In the following, we review face recognition based on image sets or videos, instead of a single image. For the problem of face recognition based on sets of images or videos, early approaches which make use of sets of face examples (extracted from different images or video frames) aim to represent image sets as manifolds (Arandjelovic and Cipolla (2006), Huang et al. (2015), Kim et al. (2007), Lee et al. (2003), Turaga et al. (2011), Wang and Forsyth (2008), Yang et al. (2013)) or convex hulls (Cevikalp and Triggs (2010)), and measure the dissimilarity between image sets as the distance between these spaces.

Later methods represent face sets more efficiently using a single fixed-length descriptor. For example, Parkhi et al. (2014) first detect faces in each frame and then aggregate local descriptors such as RootSIFT (Arandjelović and Zisserman (2012a)) extracted from the face crops using the FV encoding (Perronnin et al. (2010a)) to obtain a single descriptor per face track. Some work, on the other hand, aims at capturing the underlying data distribution of the image sets. For example, in (Wang et al. (2015)), each image set is first modelled with a Gaussian Mixture Model which consists of a number of Gaussian components. To measure the distance between Gaussians, they use a kernel defined on the Riemannian manifold to perform discriminant

analysis on the corresponding Gaussian components.

Since the success of deep learning in image-based face recognition (Parkhi et al. (2015), Schroff et al. (2015), Sun et al. (2014b), Taigman et al. (2014)), simple strategies for face descriptor aggregation prevailed, such as average- and max-pooling (Chen et al. (2015), Parkhi et al. (2015)). However, none of these strategies are trained end-to-end for face recognition as typically only the face descriptors are learnt, while aggregation is performed post hoc. From a different perspective, Wang et al. (2017b) propose to model the image sets by the corresponding set covariance matrix computed based on the image features extracted from a CNN. The CNN is then trained to maximize the discriminability of the set covariance matrix. In terms of training schemes, they either minimize the weighted Log-Euclidean metric between two set covariance matrices, or regress the corresponding log-covariance vectors.

A few methods go beyond simple pooling by computing a weighted average of face descriptors based on some measure of per-face example importance. For example, Goswami et al. (2014) train a module to predict human judgement on how memorable a face is, and use this memorability score as the weight. In (Yang et al. (2017)), an attention mechanism is used to compute face example weights, so that the contribution of low quality images to the final set representation is down-weighted. However, these methods rely on pretrained face descriptors and do not learn them jointly with the weighting functions.

Two other recent papers are quite related in that they explicitly take account of image quality: Hassner et al. (2016) first bins face images of similar quality and pose before aggregation; whilst Liu et al. (2017c) introduces a fully end-to-end trainable method which automatically learns to down-weight low quality images.

Another recent work that is worth mentioning is (Rao et al. (2017)), which aggregate the raw images instead of the image descriptors extracted from the frames. Specifi-

cally, given a video, they propose to use a network to synthesize one or a few more discriminative aggregated images, which are then passed to a fixed CNN to produce video descriptors. The network is trained on a combination of multiple losses, including a discriminative and an adversarial loss.

2.4.3 Face datasets

The collection and annotation of face datasets can be time-consuming and requires a large amount of human effort. Hence, some early face datasets were relatively small. Since the rise of deep learning, face recognition gained a boost in its performance. Apart from the deep networks, this improvement also relies on large scale labeled face datasets (see Table 2.1 for an overview). A representative and widely-used face dataset is the Labeled Faces in the Wild (LFW) dataset (Huang et al. (2007)) published in 2007. This dataset has over five thousand identities and in total 13,000 images. Later, some large scale face datasets were collected. In 2014, two large face dataset were introduced: the CelebFaces+ dataset (Sun et al. (2014b)) and the CASIA-WebFace dataset (Yi et al. (2014)). The CelebFace+ has over 200k images containing about 10k identities, while the CASIA-WebFace dataset contains nearly 500k images of over 10k people. In 2015, the first public million-scale face dataset – the VGGFace dataset – is introduced by Parkhi et al. (2015). The dataset contains 2.6 million images covering 2622 identities, with around 1000 samples per identity on average. This dataset has a curated version which further removes the label noise. The curated version contains 800,000 images with roughly 305 images per identity. Both the CASIA-WebFace and the VGGFace dataset were designed for training only and not considered as benchmarks.

In 2016, the MegaFace dataset (Kemelmacher et al. (2016)), a large-scale face dataset, was released for face recognition evaluation. This dataset contains up to a million distractor images in the gallery image set for testing, and has 4.7 million images

Datasets	# of subjects	# of images	# of images per subject	manual labelling	year
LFW (Huang et al. (2007))	5,749	13,233	1/2.3/530	-	2007
YTF (Wolf et al. (2011))	1,595	3,425 videos	-	-	2011
CelebFaces+ (Sun et al. (2014b))	10,177	202,599	19.9	-	2014
CASIA-WebFace (Yi et al. (2014))	10,575	494,414	2/46.8/804	-	2014
IJB-A (Klare et al. (2015))	500	5,712 images 2,085 videos	11.4	-	2015
IJB-B (Whitelam et al. (2017))	1,845	11,754 images 7,011 videos	36.2	-	2017
IJB-C (Maze et al. (2018))	3,531	31,334 images 11,779 videos	36.3	-	2018
VGGFace (Parkhi et al. (2015))	2,622	2.6 M	1,000	-	2015
MegaFace (Kemelmacher et al. (2016))	690,572	4.7 M	3/7/2469	-	2016
MS-Celeb-1M (Guo et al. (2016))	100,000	10 M	100	-	2016
UMDFaces (Bansal et al. (2017b))	8,501	367,920	43.3	Yes	2016
UMDFaces-Videos (Bansal et al. (2017a))	3,107	22,075 videos	-	-	2017
VGGFace2 (Cao et al. (2017))	9,131	3.31 M	80/362.6/843	Yes	2018
Facebook (Taigman et al. (2015))	10 M	500 M	50	-	-
Google (Schroff et al. (2015))	8M	200 M	25	-	-

Table 2.1: **Statistics for recent public face datasets.** The three entries in the 'per subject' column are the minimum/average/maximum per subject.

covering roughly 672k identities in its training set. Although this dataset has a large number of identities, it has only 7 images per identity on average and thus lacks intra-class face variation. As a result, the MegaFace Challenge uses the subset of FaceScrub (Ng and Winkler (2014)) and FG-NET for evaluating the effect of pose and age variations. The test set consists of 4,000 images of 80 subjects from Face Scrub and 975 images of 82 identities from FG-NET.

As another face dataset serving both training and testing purposes, MS-Celeb-1M dataset (Guo et al. (2016)) was introduced by Microsoft in 2016. This dataset contains 10 million images from 100k celebrities. Similar to the MegaFace dataset (Kemelmacher et al. (2016)), the MS-Celeb-1M dataset also suffers a limitation in intra-class variation as it has on average 81 images per identity. Moreover, this dataset has relatively high label noise, due to the fact that images are directly downloaded using a search engine without human filtering.

The IARPA Janus Benchmark-A (IJB-A) which was introduced by Klare et al. (2015) and the Benchmark-B (IJB-B) datasets introduced by Whitelam et al. (2017) are two

benchmark datasets aiming to evaluate face detection, face search and verification systems in a template-based situation. These two datasets are used as test benchmarks in Chapter 7.

A latest well-rounded face dataset – VGGFace2 dataset – was released by Cao et al. (2017). The VGGface2 dataset contains 3.31 million images from over 9k identities with around 360 images per person on average. As another advantage, it has large pose, age and ethnicity variations. Furthermore, it has very little label noise as it went through a careful manual annotation process. This property can significantly improve the performance of networks trained on it, according to a recent work (Chen et al. (2018)). UMDFaces (Bansal et al. (2017b)) which was released in the same year contains 367k images, covering 8.5k people.

The YouTube Face (Wolf et al. (2011)) and UMDFaces-Videos datasets (Bansal et al. (2017a)), unlike the datasets that aim for image-based face recognition, are released for face recognition in unconstrained videos. The YouTube Face has about 1.6k people and 3,425 videos, whereas UMDFaces-Videos contains 3.1k identities and 22k videos.

The above datasets are publicly available, whereas Facebook and Google have large private face datasets. For example, Facebook (Taigman et al. (2015)) trained a face recognition system using 500 million images of more than 10 million subjects. In the same year, Google (Schroff et al. (2015)) also trained one with up to 200 millions images of 8 million identities.

Chapter 3

Dataset collection and annotation

Since the rise of deep learning in computer vision, large scale annotated datasets have been in high demand. This is because deep CNNs generally require enormous amount of labelled data for training. Therefore, more and more datasets have been released to fulfill this requirement, including general object datasets (Deng et al. (2009), Everingham et al. (2010), Lin et al. (2014)), face datasets (Guo et al. (2016), Huang et al. (2007), Kemelmacher et al. (2016), Parkhi et al. (2015)) and scene datasets (Xiao et al. (2010), Zhou et al. (2014a)) *etc.* In this chapter, we focus on face-related datasets. While some object datasets such as Microsoft COCO (Lin et al. (2014)) provide images containing multiple objects with corresponding class labels, all the face datasets with identity labels only provide face crops of individual identities, which restricts the training and testing to tasks based on single face such as face recognition, pose and age estimation. On the other hand, those face datasets which contain whole images and multiple faces do not have identity labels for each face, as they are used for tasks like face detection, and identity labels are not necessary. Therefore, for tasks like compound query retrieval (*e.g.* retrieving particular faces in particular places as described in Chapter 5), and retrieving sets of identities (in which the database images should be ranked such that those images containing all the query identities are ranked first, followed by images that containing a subset of of the query identities, as described in Chapter 6), previous face datasets are not suitable.

To this end, we collect and annotate two datasets that can be used as an evaluation benchmarks for compound query retrieval.

The first dataset is called the *Celebrity in Places* (CIP) dataset. It contains whole images with both identity labels (of celebrities) and scene labels. Therefore it can be used as evaluation benchmark for retrieving particular identities in particular scenes, and also for face recognition and place recognition. The second dataset is called *Celebrity Together* (CT) dataset. It contains multiple detected and labelled faces in whole images. It mainly differs from the existing face datasets that it contains multiple faces with labels, and the whole image is preserved, instead of only providing face crops. This dataset, therefore, can be used to evaluate retrieval systems for tasks like set retrieval (*i.e.* query consists of sets of identities) and face detection. Both datasets are publicly accessible and available on the VGG (Visual Geometry Group) website.

In this chapter, we first describe the CIP dataset in detail in Section 3.1, including statistics of the dataset (Section 3.1.1) and how the dataset is collected and annotated (Section 3.1.2). In Section 3.2, we proceed to the CT dataset, and explain the details of dataset collection process. Lastly, a conclusion is drawn in Section 3.3.

3.1 Celebrity in Places

In this section, we introduce the *Celebrity in Places* dataset. The *Celebrity in Places* dataset contains 35.8k images covering 4611 celebrities, and it forms our retrieval test set in Chapter 5. Figure 3.1 shows examples from the dataset. We first provide an statistical overview of this dataset, and then describe how this dataset was built by a combination of web search, CNN filtering, de-duplication and Mechanical Turk annotation.

Figure 3.1: **Example images from the CIP dataset.** The celebrities include politicians like David Cameron and Barack Obama, film stars like Emma Watson and Audrey Hepburn, singers like Ricky Martin, athletes like Tiger Woods and Pau Gasol, etc. The places shown above cover (in order) airport terminal, banquet, beach, boat, desert, football stadium, golf course, ice rink, coffee shop, conference room, staircase, hospital, kitchen, office room, stage and supermarket (selected from Places205 dataset (Zhou et al. (2015))).

3.1.1 CIP dataset distribution and statistics

Here we provide the statistics of the collected dataset in detail, including the class distribution of faces and places respectively. As Table 3.1 shows, the CIP dataset consists of two parts, split according to the celebrity name lists (*i.e.* one contains 2622 identities from VGGFace Dataset and one complementary to it with 1989 identities). Both parts contain the same 16 place classes. In total, there are 35.8k images covering 4611 celebrities.

Part	total # of images	# of people	# of places
1	15.2k	2622	16
2	20.6k	1989	16
whole	35.8k	4611	16

Table 3.1: **Statistics of the CIP dataset.** Part 1 has exactly the same people classes as VGGFace Dataset, while part 2 includes another set of famous people. Part 1 and part 2 share the same place classes that are selected from Places205 Dataset.

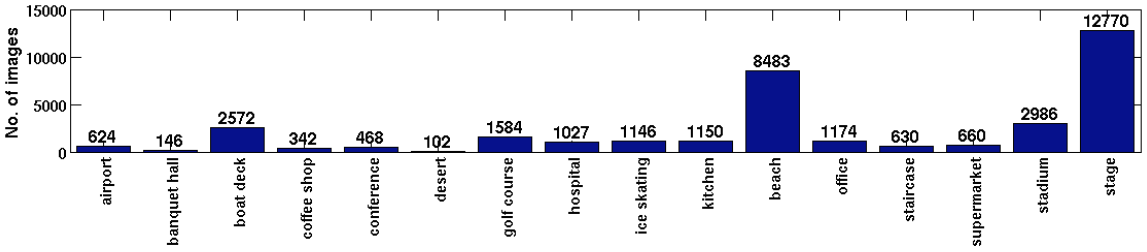


Figure 3.2: **Place class distribution for the CIP dataset** This dataset covers 16 place classes and they are highly unbalanced.

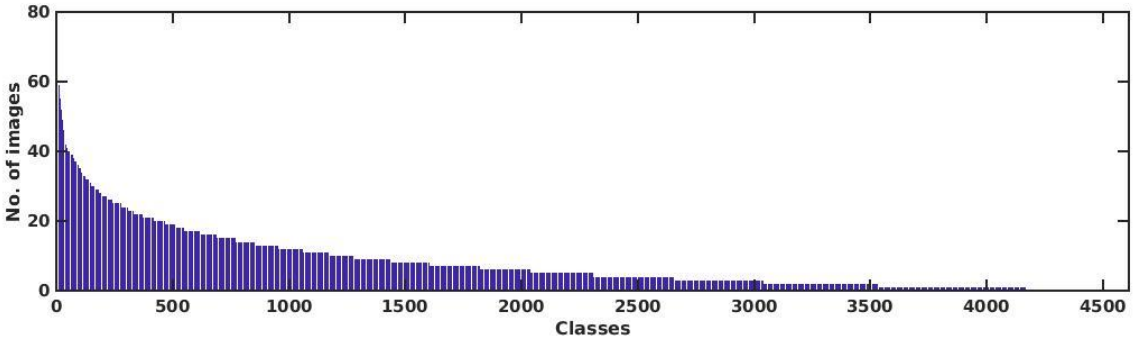


Figure 3.3: **Face class distribution for the CIP dataset.** There are 4611 celebrities in the CIP dataset. The celebrity classes in the chart are ordered based on the number of images in that class.

Class Distribution. Figure 3.2 and Figure 3.3 show the place classes distribution and the celebrity class distribution of our CIP dataset. Both the place and people classes are quite unbalanced – this is to be expected for celebrities due to variation in their popularity. However, this imbalance is not a problem since the dataset is not used for network training but as an evaluation benchmark.

3.1.2 Dataset collection and annotation

The dataset is obtained in four stages: (i) initial downloading using Google Image Search based on a text query; (ii) initial filtering using CNN; (iii) duplicate removal and (iv) manual checking using Mechanical Turk.

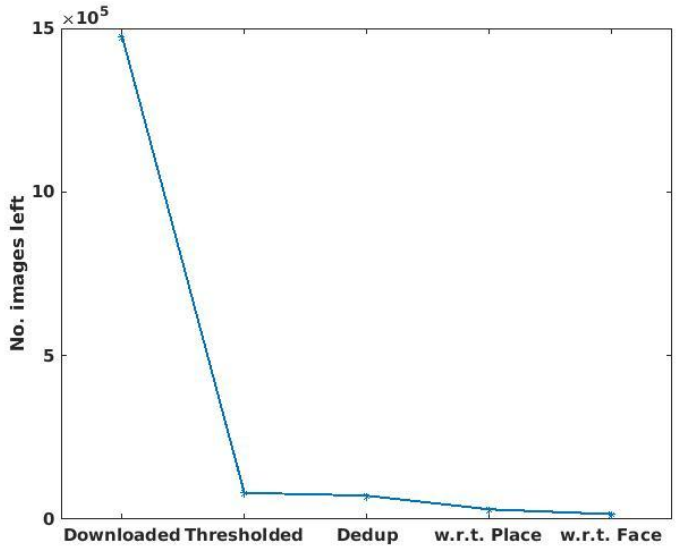
Query text selection. The query texts used for the Google Image Search are the pairwise combinations of a celebrity name list and a places list. The celebrity name list is compiled from two sources. The first list has 2622 celebrities from the VGGFace Dataset (Parkhi et al. (2015)). The second contains 1989 celebrities that are not included in the VGGFace Dataset, but are nonetheless very famous (these celebrities were excluded from the VGG Face Dataset to avoid overlap with prior face datasets such as LFW (Huang et al. (2007)) and YouTube Face (Wolf et al. (2011))). The scene list contains 16 place classes selected from the Places205 Dataset (Zhou et al. (2014b)) classes. These classes are the ones for which celebrities are more likely to have photos taken (celebrities are very unlikely to be photographed in some places, such as computer room or creek). They are airport terminal, banquet hall, boat deck, coffee shop, conference room, desert, golf course, hospital room, ice skating rink, kitchen, ocean, office, staircase, supermarket, stadium and stage (and synonyms of these).

CNN filtering. Images are downloaded using the selected string combinations as queries to the Google Image Search. However, the returned images for a text search composed of a person name and a place are very noisy – far more so than searches on the individual name or place. This is probably a reflection of the lack of joint face-place annotation in the caption of many images on the web. There are several reasons for that. One is that the place key word may not appear in the image caption, so the image can not be retrieved by Google. Other reasons could include that the

search engine only focuses on the person or the place, or a combination itself is actually unrealistic as there are no images on the Internet for that celebrity in that place at all. Hence each downloaded image has two weak labels (place and person) according to its corresponding query words. To avoid checking all images, we follow the annotation procedure suggested in Parkhi et al. (2015): first, CNN image classifiers are trained for each of the 16 place classes (using images from the Places205 dataset), and used to rank the downloaded images. Then only images of that place-class above a classifier threshold are preserved. This procedure substantially reduces the number of images that need to be annotated, and does not remove many relevant images.

De-duplication. After CNN screening, the remaining dataset contains many duplicates. Annotating duplicates can waste lots of time and human effort. To alleviate this duplicate problem, exact and near duplicate images are removed by clustering VLAD descriptors (Arandjelović and Zisserman (2013), Jégou et al. (2010)) of all images. The de-duplication stage proceeds as follows. First, 128-D dense SIFT features (Lowe (2004)) are extracted for each image, which are then used to build a visual vocabulary. Second, each image is encoded using this vocabulary to give a fixed-length VLAD descriptor. Third, a nearest neighbour search is performed for each of these global image descriptors to form a undirected graph, in which the vertices denote the images and the edges denote the similarity between two connected images. In particular, only the neighbours with a similarity score (*i.e.* cosine similarity between two VLAD descriptors) over a certain threshold are preserved, with a maximum of 20 neighbours per image. Fourth, all the connected components (which are subgraphs in which any two vertices are connected to each other by edges) are computed from this undirected graph, which are considered as clusters (*i.e.* a group of images whose similarity scores between each other are higher than the threshold we set). Lastly, only one image per cluster is retained, and thus all the duplicates are removed.

(a) Entire process



(b) Close-up

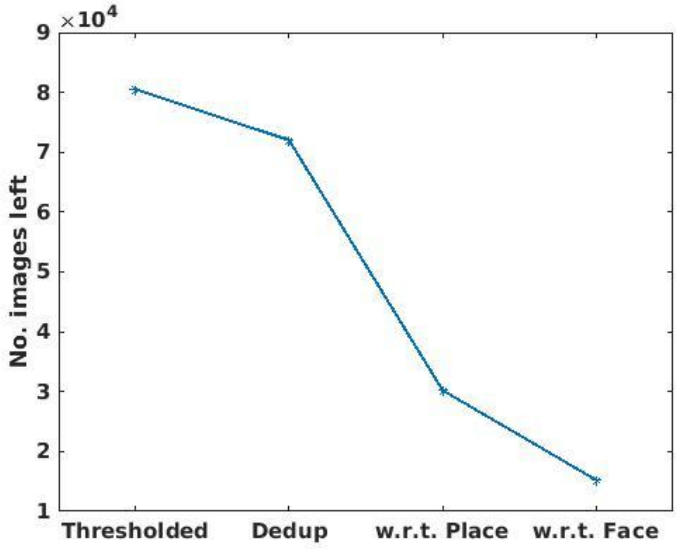


Figure 3.4: **Number of images remained after each step in the annotation pipeline of the CIP dataset (part 1).** (a) With the number of downloaded images; (b) Without the number of downloaded images. Note that 'thresholded' means CNN filtered.

Verification using Mechanical Turk. The remaining images after filtering by pre-trained CNN and de-duplication are sent to the annotators. We observe that only a small fraction of the these remaining images actually contain the named person in the specified place. Mechanical Turk (MT) is used to obtain the images that do. Three options are provided to MT for each remaining image: yes, no and not sure if the image contains the target scene or person.

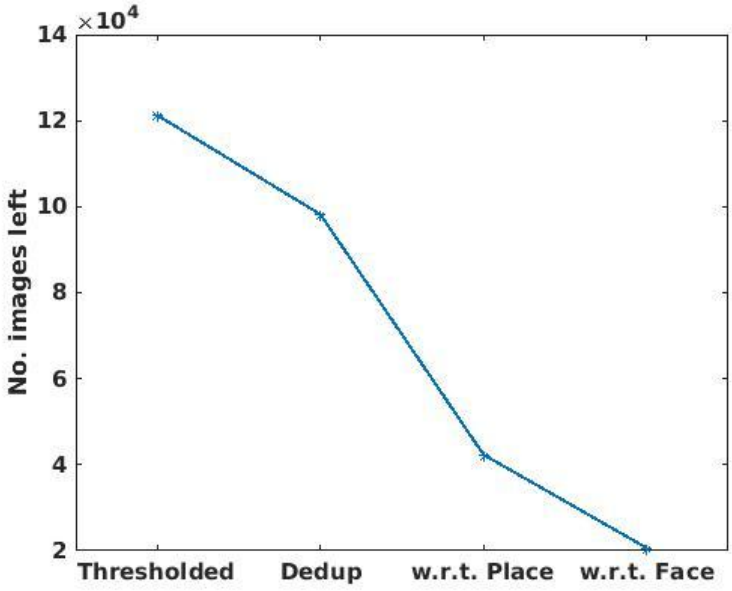


Figure 3.5: **Number of images remained after each step in the annotation pipeline of the CIP dataset (part 2).** Note that 'thresholded' means CNN filtered.

The annotation stage consists of two steps. First, the annotators are asked to annotate the images with respect to places, and images with incorrect place labels are removed. Second, the annotators then annotate the remaining images (which contain the correct place label) with respect to the celebrities (*i.e.* faces). The reason for annotating in this order is that it is much easier for the Mechanical Turk to annotate with respect to places than faces, as a Mechanical Turk is unlikely to know all the celebrities considered in this dataset. For the celebrities that a Mechanical Turk is not familiar with, the Mechanical Turk would have to look up them on the Internet, which would slow down the annotation.

At the end, we only preserve the images that are sure to be correctly labelled to form our final dataset. As Figure 3.4 illustrates, the number of remaining images decreases significantly at each step of the filtering. Initially, over 2.5 million images (including part1 and part2) are downloaded, whereas only 38k images in total survive (*i.e.* with true face and place labels). This is a very low yield, and exemplifies how noisy the original downloaded images are. We can also see that de-duplication is a necessary

step to save a lot of human effort. However, the most important and time-saving step is CNN filtering. For example, we include the number of initial downloaded images of the CIP dataset (part 1) in Figure 3.4. As we can see, by incorporating CNN in the initial filtering, we manage to reduce the number of useful images from 1.5 million to around 80k. The same thing happens for the second part of the dataset, as demonstrated in Figure 3.5. Finally, the CIP dataset has 38k images in total and includes 4611 different celebrities and 16 places.

To summarize, we should take care of several aspects to ensure the quality of the dataset while maintaining high efficiency during dataset collection and annotation. For instance, selecting text strings for an image search engine can be tricky for complex queries. We first need to filter out some useful combinations before starting the image downloading, to avoid wasting time in processing the downloaded images. As another important step, it is beneficial to make use of CNNs to help refining the downloaded images before proceeding to the human annotation stage. Sometimes the CNN-refining procedure can screen most false positives and thus save lots of human efforts.

Further example images of *Celebrity in Places* dataset are shown in Figure 3.6.

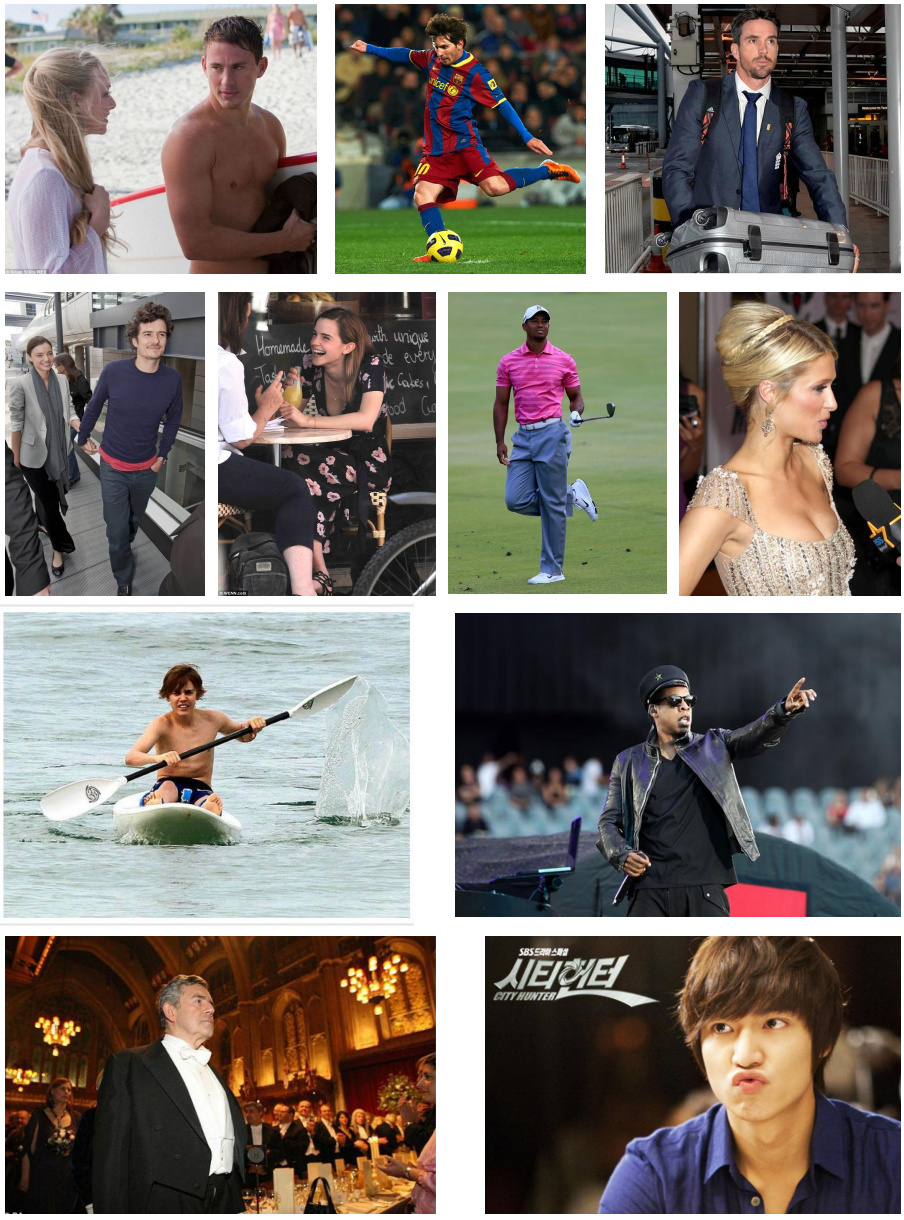


Figure 3.6: **Example images from the CIP dataset.** The celebrities shown above include Amanda Seyfried, Lionel Messi, Orlando Bloom, Justin Bieber, Jay Z and so on. The places cover beach, football stadium, airport terminal, coffee shop, golf course, boat, stage, banquet *etc.*

3.2 Celebrity Together

In this section, we introduce another new dataset, the *Celebrity Together* dataset. It contains images that portray multiple celebrities simultaneously (Fig. 3.7 shows a few samples), making it ideal for testing set retrieval methods which is investigated in Chapter 6. Unlike the other face datasets, which exclusively contain individual face crops, *Celebrity Together* is made of full images with multiple labelled faces. It contains 194k images and 546k faces in total, averaging 2.8 faces per image. The distribution of the number of faces per image is shown in Table 3.2 and the distribution of the number of labelled celebrities per image is recorded in Table 3.3.

Figure 3.7: **Example images of the *Celebrity Together* Dataset.** Note that only those celebrities who appear in the VGGFace Dataset are listed, as the rest are labelled as 'unknown'. (a) Amy Poehler, Anne Hathaway, Kristen Wiig, Maya Rudolph. (b) Bingbing Fan, Blake Lively. (c) Kat Dennings, Natalie Portman, Tom Hiddleston. (d) Kathrine Narducci, Naturi Naughton, Omari Hardwick, Sinqua Walls. (e) Helena Christensen, Karolina Kurkova, Miranda Kerr. (f) Adam Levine, Blake Shelton, CeeLo Green. (g) Aamir Khan, John Abraham, Ranbir Kapoor. (h) Amanda Seyfried, Anne Hathaway, Hugh Jackman.

The dataset is created with the aim of containing multiple people per image, which makes the image collection procedure much more complex than when building a single-face-per-image dataset, such as in (Parkhi et al. (2015)). The straightforward strategy of (Parkhi et al. (2015)), which involves simply searching for celebrities using an online image search engine, is inappropriate: consider an example of searching for 'Natalie Portman' on Google Image Search – *all* top ranked images contain only her and no other person. Here we explain how to overcome this barrier to collect images that

No. faces / image	2	3	4	5	>5
No. images	113k	43k	19k	9k	10k

Table 3.2: **Distribution of faces / image.**

No. celeb / image	1	2	3	4	5	>5
No. images	88k	89k	12k	3k	0.7k	0.3k

Table 3.3: **Distribution of annotations / image.**

depict multiple celebrities.

3.2.1 Dataset annotation Details

Different to the process described in Section 3.1, the collection pipeline of the dataset mainly comprises of five steps: (i) query string selection; (ii) increasing pair diversity; (iii) image downloading and filtering; (iv) de-duplication and (v) filtering by CNN and mechanical turk. However, the collection of this dataset is more difficult, especially in the query string selection stage and face annotation stage, as we need to search for images containing multiple faces in our celebrity list with a diversity in celebrity combinations. Each step plays an important role in minimizing the label noise while maintaining annotation efficiency. The whole pipeline is discussed in the following.

Search string selection. We use the list of 2622 celebrities from the VGGFace Dataset (Parkhi et al. (2015)) and aim to download images containing at least two celebrities each. Since it is inappropriate to query internet image search engines for single celebrities, here we explain how to obtain sets of celebrities. A straightforward approach would be to query for all pairs of celebrities; assuming top 100 images are downloaded for each query, this would result in 300 million images, which is clearly prohibitively time-consuming to download and exhaustively annotate. We use the fact that only a small portion of the name pairs is actually useful as not all pairs of celebrities have photos taken together. To obtain a list of plausible celebrity pairs, we consider each celebrity as a "seed" in turn. For each seed-celebrity, a search is

performed for '`seed-celebrity` and' to obtain a list of images of the seed-celebrity together with another person. The meta information associated with these images (image caption in Google Image Search) is then scanned for other celebrity names, producing a list of (seed-celebrity, celebrity-friend) pairs.

Increasing pair diversity. However, one important procedure is that some celebrities may have a strong connection to one or more particular celebrities, which could prevent us from obtaining a diverse list of celebrity friends. For instance, almost all the top 100 images returned by 'Beyoncé and' are with her husband Jay Z. Therefore, to prevent such celebrity friends dominating the top returned results, a secondary search is conducted by explicitly removing pairs found by the first-round search, *i.e.* the query term now is '`seed-celebrity` and `-friend1 -friend2` ...', where `friend1`, `friend2` .. are those found in the first search. We do not perform more searches after the second round, as we find that the number of new pairs obtained in the third-round search is minimal.

Image download and filtering. Once the name pairs are obtained, we download the images and meta information from the image search engine, followed by name scanning again to remove false images. Face detection is performed afterwards to further refine the dataset by removing images with fewer than two faces.

De-duplication. Removal of near duplicate images is performed using the same method described in Section 3.1.2, and images in common with the VGGFace Dataset (Parkhi et al. (2015)) are removed as well.

Annotation by CNN and human. Each face in each image is assigned to one of 2623 classes: 2622 celebrity names used to collect the dataset, or a special "unknown person" label if the person is not on the predefined celebrity list; the "unknown"

people are used as distractors (46%). As the list of celebrities is the same at that used in the VGGFace Dataset (Parkhi et al. (2015)), we use the pre-trained VGGFace CNN (Parkhi et al. (2015)) to aid with annotation by using it to predict the identities of the faces in the images. Combining the very confident CNN classifications with our prior belief of who is depicted in the image (*i.e.* the query terms), results in many good quality automatic annotations, which further decreases the manual annotation effort and costs. But choosing thresholds for deciding which images need annotation can be tricky, as we need to consider two cases: (i) if a face in an image belongs to one of the query names used for downloading that image, or (ii) if the face does not belong to the query names. The two cases require very different thresholds, as explained in the following.

Identities in the query text. We first consider the case where the predicted face (using a Face CNN) matches the corresponding query words for downloading that image. In this case, there is a very strong prior that the predicted face is correctly labelled as the predicted celebrity was explicitly searched for. Therefore, if the face is scored higher than 0.2 by the CNN, it is considered as being correctly labelled, otherwise it is sent for human annotation. As another way of including low scoring predictions, if one of the query names is ranked within top 5 out of the 2622 celebrities by the CNN, the face is also sent for annotation. A face that does not pass any of the automatic or manual annotation requirements, is considered to be an "unknown" person.

Identities not in the query text. The next question is: if a face is predicted to be one of the 2622 celebrities with a high confidence (CNN score) but it does not match any query celebrities, can we automatically label it as a celebrity without human annotation? In this situation, the CNN is much less likely to be correct as the predicted celebrity was not explicitly searched for. Therefore, it has to pass a much

stricter CNN score threshold of 0.8 in order to be considered to be correctly labelled without the need for manual annotation. On the other hand, empirically we find that a prediction scored below 0.4 is always wrong, and is therefore automatically assigned the "unknown" label. The remaining case, a face with score between 0.4 and 0.8, should be manually annotated, but to save time and human effort we simply remove the image from the dataset.

After CNN and human annotation, only 194k images are preserved out of over 3 million images downloaded. There are 2622 identities labelled, and the rest with a 'unknown' label. Further example images of *Celebrity Together* are shown in Figure 3.8.

3.3 Conclusion

In this chapter, we have introduced two face datasets, the *Celebrity in Places* dataset and the *Celebrity Together* dataset. The first dataset can be used as a test set for evaluating the performance of in retrieving a particular identity in a particular place, whereas the second dataset can be used as a set retrieval (*i.e.* retrieving sets of identities and ranking the images based on the amount of overlap between the query set and the identities in the image) benchmark. Note that the yield of annotating the *Celebrity in Places* dataset is low, making the dataset not suitable for training. However, this problem is solved in Chapter 4.

Figure 3.8: **Example images of the *Celebrity Together* Dataset.** Only those celebrities who appear in the VGGFace Dataset are listed, as the rest are labelled as 'unknown'. From left to right, the images in each row contain the following celebrities. **1st row**: Lisa Edelstein, Richard Schiff; Lea Michele, Taylor Lautner; Andrew Garfield, Emma Stone; Kate Micucci, Riki Lindhome; Abigail Breslin, Sofia Vassilieva. **2nd row**: Julie Walters, Celia Imrie, Helen Mirren; Josh Hutcherson, Jena Malone; Joanna Krupa, Joyce Giraud, Katie Cleary. **3rd row**: Anil Kapoor, Irrfa Khan; Amy Poehler, Kristen Wiig, Maya Rudolph; Luke Evans, Sarah Gadon. **4th row**: Jesse Metcalfe, Patrick Duffy, Julie Gonzalo, Josh Henderson, Linda Gray, Jordana Brewster; Jenna Ushkowitz, Lea Michele; Andrew Rannells, Josh Gad. **5th row** Kristen Stewart, Billy Burke; Marilu Henner, Judd Hirsch.

Chapter 4

Synthesizing datasets

For the problem of retrieving particular identities in target places described in Chapter 5, we need a large number of labeled images (in terms of both identities and scene types) to train the proposed hybrid CNN. However, collecting and annotating images with labels on both identities and places is prohibitively difficult, mainly because of two reasons: first, filtering and annotating the noisy downloaded images is very time-consuming and requires a huge amount of human effort, especially in our case where two labels are needed for each image (*i.e.* identities and places). Second, celebrities do not necessarily have photos or images in different places. Moreover, certain types of celebrities usually appear in a similar scene types. For example, singers frequently show up on the stage, and politicians are most likely to have photos in the office or conference room; while sports players are usually on the sports ground. Therefore, the yield of the image filtering and annotation is terribly low (see Chapter 3) and the resultant dataset is highly unbalanced in terms of the identities classes and place classes. These two problems, can severely negatively impact the CNN performance, and hence the performance of the network.

In order to address these two problems mentioned above, we propose to synthesize training images, instead of collecting real images from the web. Furthermore, we can, to a large extent, control the class balance if we synthesize our training dataset.

Since it is straightforward to obtain images labelled with places (*e.g.* from the Places dataset (Zhou et al. (2014a))), our goal is to replace the unknown faces in place-labelled images with labelled celebrity faces; since then both place and face identity are labelled for that image. The challenge is to replace the face automatically without introducing artifacts.

In this chapter, we first introduce the source and target datasets which are involved in the synthesis in Sec. 4.1; and then in Sec. 4.2, we describe the face replacement method step by step. In Sec. 4.3, we give statistics of the generated dataset, followed by a few examples of synthetic images in Sec.4.4.

4.1 Source and target datasets

A "target" image is the image whose place label is known but which contains unknown faces. One unknown face in the "target" image will be replaced with known faces coming from a "source" image, thus creating an image with face and place labels.

Target places dataset. There are several scene datasets available as target dataset in our task. The first dataset for scene recognition was the Scene15 database (Lazebnik et al. (2006)). This dataset contains only 15 scene categories and only several hundreds of images for each class. The MIT Indoor67 dataset (Quattoni and Torralba (2009)) has 67 categories on indoor places., while we do not want to restrict our synthetic dataset to be on indoor scenes only. The SUN database (Xiao (2010)) has a wide coverage of scene categories. It is provides 397 categories, however, containing only around 100 images per category. Therefore, the Places dataset (Zhou et al. (2014a)) becomes our best option as it fits our requirements for the target dataset perfectly: it contains a wide range of scene categories (476 in total) and more than 7 million scene-centric images.

Although the Places205 dataset (Zhou et al. (2014a)) provides us many place categories, we find that only a small subset of them is actually useful for our task, *i.e.* only the places where celebrities are likely to have images. Therefore, we keep the same 16 place classes as in the *Celebrity in Places* dataset (see Chapter 3). Furthermore, in order to be able to replace faces, images which do not contain a face are filtered out.

Source faces dataset. We choose to make use of the VGGFace dataset (Parkhi et al. (2015)) as our source face dataset. Details of VGGFace dataset are described in Chapter 2. Due to the limited time before paper submission deadline, we use the 500k manually annotated subset of this dataset as source face images which will be pasted into the target images.

4.2 Automatic Face Replacement

In this section, we describe our proposed face replacement pipeline. Our objective is to replace a face in a target image with a face sourced from another image. The pipeline mainly contains two parts as illustrated in Figure 4.1: (i) selecting candidate faces from the 500k potential source images, and (ii) replacing the target face with the selected source face.

4.2.1 Face replacement

In the first part of this pipeline, our aim is to select candidates that have the most similar appearance and pose to the target face. The intuition behind this part is that smaller changes reduce the risk of introducing artifacts. For example, replacing the face of a man with a black skin with a white lady's face would probably end up with a terrible synthetic image. Therefore, we need this part to propose and select reasonable source faces before the replacement. This selection proceeds in three stages, described

Figure 4.1: **Automatic synthetic rendering pipeline.** Replacing the face of the unknown person in the airport with the celebrity face (Gage Golightly). (a) Face detection and 36 landmark points annotation on the target image. (b) Top K most similar images (based on FC7 feature vectors) in the face source dataset. (c) Pose descriptors are computed based on face landmarks for these top K images and their flipped version. Top N images in the re-ranked list based on pose similarity are qualified as source faces for replacement. (d) During face replacement, a similarity transformation is computed between corresponding landmark points to map the source face onto the target one. (e) Poisson Editing is applied to produce natural-looking synthetic images. At this stage, the original face is replaced by the labelled source face in the image. (f) As a quality control, the synthesized image is passed to the Face CNN and retained only if the new face can still be recognized.

in the following.

Face detection and feature extraction. The first step is to detect faces in each target image using the face detector introduced by Mathias et al. (2014), and localize the 36 facial landmarks, by using the algorithm of Xiong and De la Torre (2013). After detecting the faces, we then extract the activations from the last FC layer (before the classification layer) from a Face CNN, as a feature representing the face.

Searching for source candidates. The second step is to find the top K appearance-similar source faces, where appearance-similarity is measured using the 4096-dimensional Face-CNN descriptor extracted from the last FC layer. In other words, we search for faces that could be confused with each other for recognition.

Re-ranking by pose similarity. Given these K candidates, we then re-rank them by their similarity in pose, where pose is measured by the Euclidean distance between every pair of facial keypoints of each face. In practice this delivers source faces that have similar shape and skin colour to the target. Bad quality face crops are removed from the candidate list to ensure the quality of the source faces. Basically, source face candidates must be correctly predicted by the Face CNN with its probability over a certain threshold. This threshold is to ensure that the source face belongs to its class with a good quality.

4.2.2 Face replacement

Given a candidate source face, the face replacement also consists of three steps, explained in the following.

Similarity transformation. First, as the source face and the target face are in different scale and angle, we need to find the transformation between the source and target face using the 36 facial keypoints detected in the previous step. We compute a similarity transformation as this is less distorting than an affine transformation (and the face poses are close). Any distortion of the face introduced in the transformation would easily make it unrecognizable. The similarity transformation is defined as

$$x' = Ax + t$$

where x' is the transformed coordinates and x is the original coordinates, A is an orthogonal matrix and t is a translation vector.

Face blending. The next step is to blend the source face into the target image. This is achieved using Poisson Editing described by Perez et al. (2003). Poisson Editing computes image interpolation using a guidance vector field based on solving Poisson equations. With Poisson Editing, one can seamlessly clone one part of an image onto another, which are difficult to achieve with conventional image-domain techniques. This step is very important as otherwise the pasted face would look unnatural due to different lighting condition, skin colour *etc.*

Quality control. The last step in this pipeline is to check that the replaced face can be recognized using a face classifier trained for that identity. This is a check that no artifacts have been introduced that can hurt the recognition of that face. The synthetic image is preserved only if the replaced face has a classification score higher than a threshold. In our experiments, we find 0.1 to be a reasonable threshold. This step is crucial as it acts as a quality control. The result of this pipeline is that the source face is blended in quite naturally to the target image. Some examples are shown in Figure 4.2.

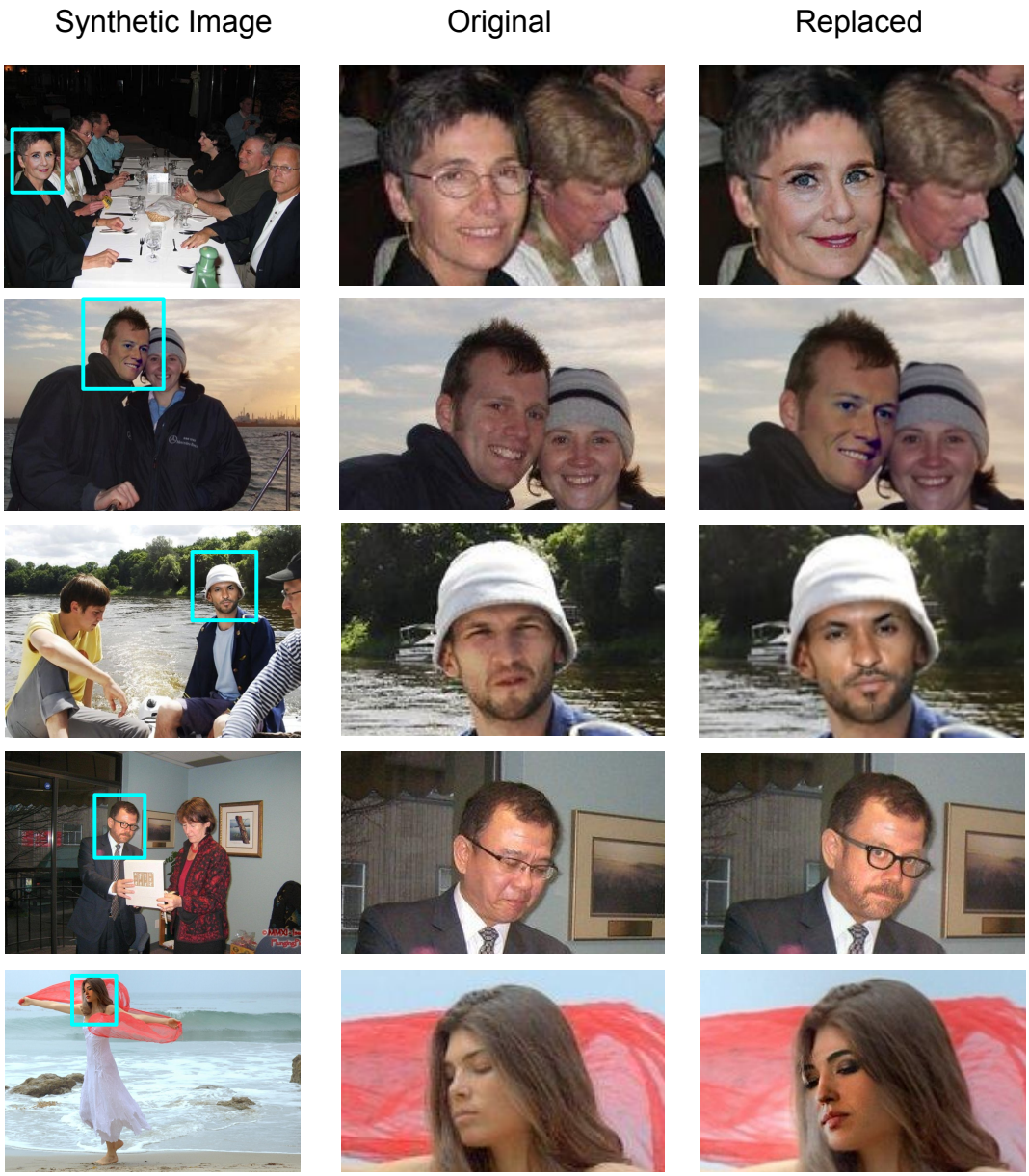


Figure 4.2: **Example images from the synthetic dataset.** The first column shows the synthetic images and the replaced faces inside the blue boxes. The second and the third columns show close-ups of the original and the replaced faces, respectively. The automatic face search and replacement system successfully deals with different lighting condition and head poses, as well as accessories like hats and glasses. Further examples are given at Web.

4.3 Synthetic dataset statistics

The dataset produced has three splits: 178k training images, 8.7k validation images and 15.9k test images. We have in total 250k images synthesized, with only images

above a reasonable CNN score threshold retained. The images are class balanced for face classes as this synthetic dataset is generated under certain rules to ensure the maximum number of images per face class and the diversity of face classes for each target image. The three sets (*i.e.* training set, validation set and test set) share the same 500 face classes and 16 place classes with a similar distribution, while images in our synthetic test set have 100 faces classes that the CNN has never seen during training. However, there are no instances in common at all between the training and validation sets (i.e. no target place-images or source faces). This synthetic dataset is significantly larger than the downloaded *Celebrity In Places* dataset described in the Chapter 3.

4.4 Examples of synthetic training data

More example images from our synthetic training dataset are shown in the following. Figure 4.3 and Figure 4.4 compare the original image with the synthetic one. Images on the left are all original images in which the source celebrity face crop is shown on the bottom right corner. Synthetic images are shown on the right column and the red boxes indicate the replaced face. As illustrated by the examples, the automatic face search and replacement pipeline generates high-quality synthetic images which look very natural. Surprisingly, the system can even replace the original coloured face with a grey-level face without introducing artifacts. Furthermore, the system manages to preserve the hair of the original person and blend it with the new face, which can be considered as a data-augmentation of face recognition, as people's hair can vary and should not affect the recognition performance.

4.5 Summary

In this chapter, we describe an automatic pipeline to generate a synthetic training dataset which is used to train deep CNNs for retrieving faces in places (Chapter 5). By using our proposed pipeline, we show the possibility of synthesizing high-quality images. With this synthetic dataset, we manage to train the hybrid CNNs described in Chapter 5, and achieve prominent results. Apart from research, this image synthesis system can also be applied for other purposes. For example, it can be used for creating interesting pictures in which our friends are next to their favourite celebrities.

Figure 4.3: **Further examples from the synthetic training dataset.** Left: original images in which the source celebrity face crop is shown on the bottom right corner. Right: synthetic images and the red boxes indicate the replaced face.

Figure 4.4: **Further examples from the synthetic training dataset.** Left: original images in which the source celebrity face crop is shown on the bottom right corner. Right: synthetic images and the red boxes indicate the replaced face.

Figure 4.5: **Further examples from the synthetic training dataset.** Left: original images in which the source celebrity face crop is shown on the bottom right corner. Right: synthetic images and the red boxes indicate the replaced face.

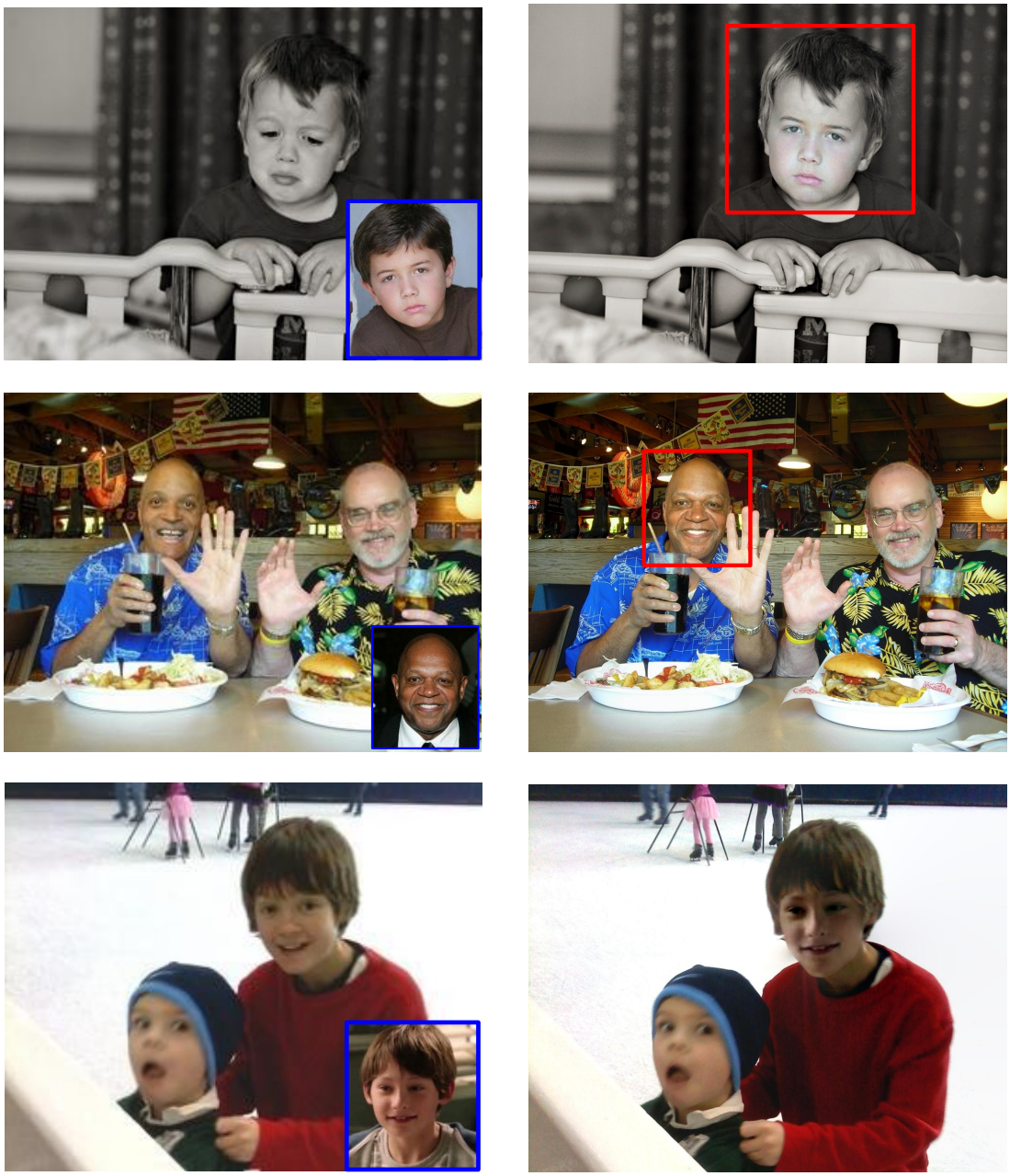


Figure 4.6: **Further examples from the synthetic training dataset.** Left: original images in which the source celebrity face crop is shown on the bottom right corner. Right: synthetic images and the red boxes indicate the replaced face.

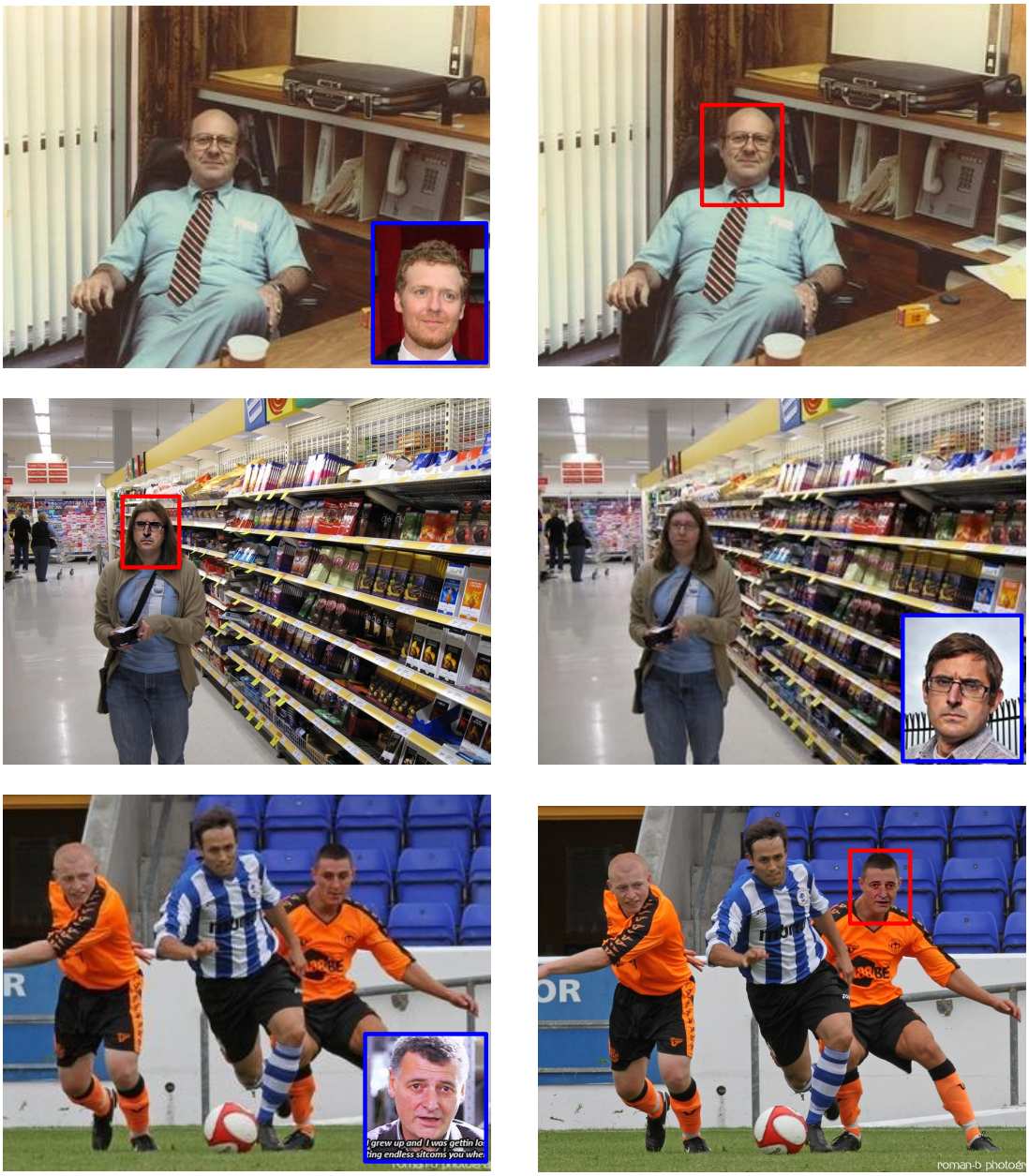


Figure 4.7: **Further examples from the synthetic training dataset.** Left: original images in which the source celebrity face crop is shown on the bottom right corner. Right: synthetic images and the red boxes indicate the replaced face.

Chapter 5

Faces in places: compound query retrieval

The objective of this chapter is high performance (precision and recall) retrieval for compound queries consisting of a 'face' and a 'place' in large scale image datasets. Examples are shown in Figure 5.1. This ability has a range of daily applications, for instance, searching photos in personal collections and news accessing using key words *etc.*

Searching for a target face and a target place is a particular example of compound query retrieval. For such compound query retrieval problems, there must be a way to merge either the query features or ranking lists to give a final ranking. However, the place descriptors and face descriptors are independent of each other, and thus not comparable. Instead of trying to find the best combination method for such independent and incomparable scores, we *choose* a combination rule, and then design and train a deep CNN to optimize the classification score for this rule. This formulation of the problem as training a CNN to achieve a common classification is one of the key contributions of this chapter.

As will be seen in the sequel, training the CNN to generate place-descriptors (feature vectors) for a common classification with face-descriptors encourages the place-descriptors to be 'face aware', in the sense that they are able to ignore the face regions

Figure 5.1: **Examples of the top two retrieved images for various compound queries on the CIP test set.**

in an image and concentrate on the class of the place information. They are also 'face-descriptor aware' as they are calibrated to cooperate with the face-descriptors.

We show in section 5.4, that the face aware place-descriptors lead to a substantial improvement in compound query retrieval performance, compared to baselines with state-of-the-art descriptors used independently.

This chapter is organized as follows: Sec. 5.1 proposes a scalable and practical solution to the problem of compound query retrieval, and Sec. 5.2 describes the design of our two network architectures based on the objectives we wish to achieve. Sec. 5.3 explains the implementation details for network initialization and training, as well as testing. Experimental results are reported in Sec. 5.4 showing the prominent improvement of our network, followed by the exploration of what has been learnt during the joint

training of the face and place streams in Sec. 5.5. Sec. 5.6 lists some retrieval examples from the 'Celebrity in Places' dataset (Chapter 3). Finally, Sec. 5.7 discusses the problem we encountered when the network is used in practice, and how it is solved.

5.1 Compound query retrieval

Combination rule. Given a compound query, we choose to combine ranked lists by assigning each image the minimum of the individual scores obtained from each query. Taking the minimum is natural way to combine the scores, since the task is to find the face *and* the place, and taking the minimum penalizes the non-existence of either of the two objects of interest. Suppose we search for 'Barrack Obama on the beach', an image with Obama in a church may score extremely high for face but low for place. In this case if we take, for instance, the sum of the two query scores to represent this dataset image, it is very likely to rank higher than some target images with both face and place scoring not so high due to reasons such as low resolution of the face or occlusion on the background. However, by taking the minimum, the combined scores are never dominated by one of the queries. Images will only be ranked highly if both its face and place are correct.

For the retrieval system to be useful, we would like to be able to answer queries which have not been seen at the training stage, containing potentially novel faces and novel places. Therefore, standard approaches such as pre-tagging the database with a fixed 'vocabulary' of concepts (here faces/places), as commonly done for the TRECVID semantic indexing challenge (Paul et al. (2011)), is inappropriate.

To this end, we represent each database image using a set of feature vectors, one for the place and one for each face in the image. Given a new compound query, a classifier is learnt online which is then used to rank the images. Since it is very unlikely that we can obtain a sufficient amount of training images depicting the compound

query ('Barrack Obama on the beach'), but it is easy to obtain images of the face ('Barrack Obama') and the place ('beach'), the face and place classifiers are trained independently. They are then used to obtain two scores for each database image, one for a face and one for a place, and these scores are combined and sorted to obtain the final ranked list.

To search for the target images, each database image is scored by two classifiers, and then they are ranked by the minimum between the two scores. Clearly, the two scores have to be properly calibrated in order for this operation to make sense – the calibration is explicitly addressed in our network architecture (section 5.2).

In terms of efficiency, since linear (SVM) classifiers are used and the image descriptors are pre-computed, images can be classified and ranked for compound queries using standard methods (Chatfield et al. (2015), Jégou et al. (2011b)) which enable image datasets of millions of images to be searched in a fraction of a second. More details in SVM classifier training are discussed in section 5.3.

5.2 Network design

Our objective is to train a network which produces a better feature representation for places by making them interact with the face-descriptors. The new descriptors should serve two major purposes – firstly, place descriptors should be "aware" of the existence of faces, and the place descriptor should "suppress" the face information and focus mainly on the other areas of the image.

Secondly, the place descriptors should be "aware" of the face *descriptors*. That means the descriptors should be amenable to the combination rule we chose – the relevance of a database image to the compound query is computed as the minimum of the query face and query place classification scores (section 5.1). By explicitly incorporating this rule during training, the place-descriptors can be learnt such that the classification

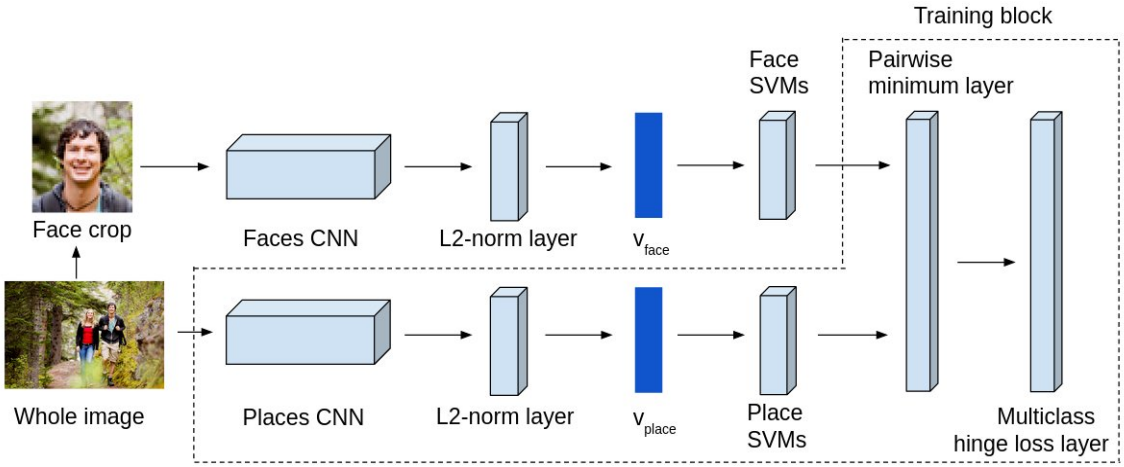


Figure 5.2: **Hybrid network architecture with coupled loss.** The network is used to generate face-descriptors and face-aware (FA) place-descriptors (the vectors v_{face} and v_{place}) for the face crop and the whole image respectively. These descriptors are used to represent the images in the target dataset. The operation of the network is to first compute FC7 feature vectors in each stream for the face crop and whole image respectively; then the vectors are L2-normalized to form v_{face} and v_{place}. The network is trained with additional layers (right of the feature vectors) that mimic the query-time operation of ranking the dataset images using linear classifiers (SVMs) and combining the face and place classifier scores by taking their minimum. During training, parameters of all the layers within the dotted block are updated.

scores are calibrated.

The coupling of the face and place descriptors, addressing both aspects of the new features, is achieved using our hybrid networks. Two variations of our proposed network are discussed in the following.

5.2.1 Hybrid CNN – coupled loss

Here we introduce the first of our hybrid network architecture which has a coupled loss – *Hybrid-CL* (shown in figure 5.2) – and explain how it addresses both two challenges discussed in the previous section. The training of the network is described in Section 5.3.

Two streams. The network has two streams, one for the place and one for face descriptor extraction. The place stream is an AlexNet pre-trained on the Places205 dataset (Zhou et al. (2014b)), and the face stream is a VGG-16 very-deep network trained on the VGG Face Dataset (Parkhi et al. (2015)). The two networks are cropped at layer FC7 and the outputs are L2-normalized to form the descriptors, v_{face} and v_{place}. These descriptors are used to train the face and place SVM classifiers.

SVM classification layer. The normalized descriptors are passed through the classification layers independently. The two classification FCs are constructed by stacking binary SVMs (which are trained using extracted features from corresponding place CNN and face CNN). Therefore, the size of the place SVM layer is $N_p * L_p$, where N_p is the number of place classes and L_p is the length of v_{place}, and similarly for the face SVM layer. During training, the place SVM layer is updated by the loss back-propagated from the layer described next.

Pairwise-minimum layer. On the top of the network, the scores for the face and place classes are combined into the scores for the face-place "product" classes, i.e. scores for all face-place combinations. For example, if there are 100 face classes and 10 places classes, then there are 1000 product classes. In order to mimic the query-time operation, the score of a face-place combination is the minimum of the scores for the two classes.

Loss layer. Finally, the face-place scores are passed through a loss layer, where a multiclass hinge loss proposed by Crammer and Singer (2001) is applied. As we know, softmax logistic loss is a most widely used loss for classification tasks, as it includes a normalization (*i.e.* softmax operation) which is beneficial for the optimization. However, in our network we use the multiclass hinge loss instead. This is because we aim to match the training of the classification layer with that of the classifiers during

query time. At run-time, a place SVM classifier and a face SVM classifier are trained for the given query place and face respectively. During network training, we thus aim to learn the place CNN such that the place classification layer acts as a stack of binary SVMs, by using a multiclass hinge loss which is also used for training SVMs.

Therefore, the two streams are actually coupled by the pairwise-minimum layer. During training, the error signal is back-propagated through both streams, and thus both streams can learn to produce corresponding face and place descriptors that are comparable to each other. This calibration refers to the fact that the place descriptors are 'aware' of face descriptors. On the other hand, the place stream is trained to recognize the scene even though a part of the image is occupied by the face(s), especially when the face takes a large portion of the image. More discussion regarding this learning is in Section 5.5.

Design choices. During training, we update the place stream only, whilst the face stream is fixed. The reason for learning the place stream only is because the face descriptors are only computed from a detection window around the face, and so are less influenced by the scene. In contrast, the place descriptors are extracted from the entire image and so can take into account that there should be a face in the image, and that a person will occlude a part of the scene. Moreover, the calibration can still be learnt between face and place descriptors even though only the place descriptors are updated during training.

5.2.2 Hybrid CNN - interaction FC

Here we introduce the second version of our hybrid network architecture – *Hybrid-FC*. This architecture aims to explicitly enable the face and place descriptors to be aware of each other (*i.e.* mutually aware) through learning. To achieve this, two additional fully-connected layers are added in the architecture which is detailed in the following.

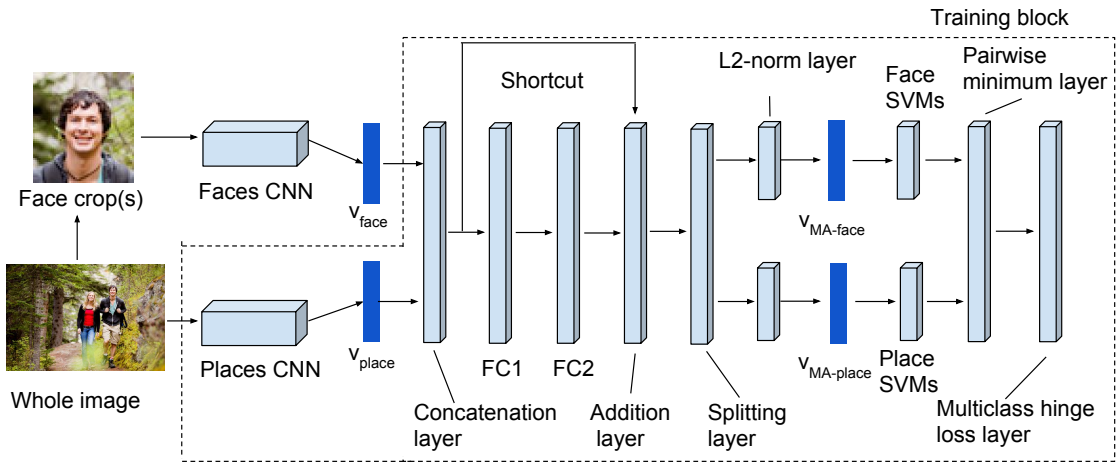


Figure 5.3: **Hybrid network architecture with interaction FC.** The first version of our hybrid network contains interaction FCs which explicitly couple the face and place descriptors. The network is used to generate two "mutually-aware" (MA) feature vectors ($v_{\text{MA-face}}$ and $v_{\text{MA-place}}$) for the face crop and whole image respectively. These feature vectors are used to represent the images in the target dataset. The operation of the network is to first compute face and place feature vectors, v_{face} and v_{place}, for the face crop and whole image respectively; then the vectors are concatenated and passed through the interaction fully-connected layers (along with a shortcut connection), and are finally split back into the face and place parts, and L2-normalized to form the MA face and place feature vectors. The network is trained with additional layers (right of the vertical dotted line) that mimic the query-time operation of ranking the dataset images using linear classifiers (SVMs) and combining the face and place classifier scores by taking their minimum.

Similar to *Hybrid-CL*, this *Hybrid-FC* also has two input streams, one for place and one for face feature extraction. These produced features (v_{face} and v_{place} in Figure 5.3) are used to train the face classifiers and place classifiers when a compound query is issued at run-time.

Interaction fully-connected layers. The two input streams are concatenated into a single feature. The feature then passes through two new fully-connected layers (FCs), where two streams interact and become aware of each other. Furthermore, the FCs can also learn to calibrate the features. In the spirit of (He et al. (2016)), we also add a shortcut such that the output of the final FC is summed with v_{IN} to form v_{MA}. The shortcut connection simply acts as an identity mapping and stabilizes

the network at the beginning of training. We found it to be effective at preventing overfitting because the CNN can now just focus on changing the original v_{IN} features to achieve the task, just like ResNet (He et al. (2016)).

The mutually-aware feature v_{MA} is split into the face ($v_{\text{MA-face}}$) and place ($v_{\text{MA-place}}$) components, which are used as features for all database images. The rest of the network has the same layers as *Hybrid-CL*, which are used during training.

5.2.3 Architecture comparison

There are two differences between two proposed hybrid CNNs. First, *Hybrid-CL* is simpler, as the face and place features produced by the corresponding networks directly pass to the classification layers; whereas for *Hybrid-FC*, the face and place features are concatenated and they interact across two FCs before splitting. Therefore, the face and place features in *Hybrid-CL* are only coupled through the loss without explicit interaction. Second, the design of *Hybrid-CL* is more consistent between training and testing. At run-time, the face and place classifiers are trained separately using v_{face} and v_{place} extracted from the examples of the query, which exactly matches the *Hybrid-CL* is trained. While for *Hybrid-FC* at training time, the classification layers are trained to cope with $v_{MA-face}$ and $v_{MA-place}$, whereas the classifiers used at query time are trained on v_{face} and v_{place} and then applied on the MA features. Although we hope the run-time classifiers will also work on MA features (because the face descriptors do not change much between the independent features and the mutually-aware features as we observed, and only the place descriptors change), this could still cause a calibration problem.

Regarding our design choices, we find that for *Hybrid-FC*, $v_{MA-place}$ differ significantly from v_{place}, whilst $v_{MA-face}$ and v_{face} are quite similar (though not identical). This observation proves that training only the place stream is sufficient.

5.3 Network training and Implementation Details

Network training. The hybrid CNNs are trained using the synthetic datasets generated in Chapter 4. Given a synthetic image with both face and place label, the target face crop is passed to the face stream in the hybrid network and the whole image is passed to the place stream. The network is thus trained to correctly predict both the face and place label of that image.

Implementation Details. Faces are detected using the method of (Mathias et al. (2014)). The Place-CNN and Face-CNN both produce FC7 feature vectors that are 4096 dimensional. The output of the pairwise minimum layer is a vector of length $N_f * N_p$, where N_f and N_p are the number of face and place classes respectively in the training set.

Training SVM classifiers. The linear SVM classifiers (for each face and place class) are trained in a one-vs-rest manner, where descriptors from all the other classes are the negatives. A weighted loss is used to balance the positive and negative sets. The SVM weight vectors are L2-normalized. These classifiers are used to rank the test images at query time, and are also used to initialize the SVM layers for training the network. This is important, as it leads to faster convergence.

Network training. Augmentation is used in both the face and place stream. For the place stream, the original full size image is resized so that its shorter side is 256 pixels, and a 227 pixel crop randomly selected (such that the crop contains the target face). The face stream is similarly resized, but the crop size is 224 pixels. Both the original place image and its face crop are flipped horizontally with 50% probability.

The place SVMs in the classification layer are L2-normalized at each gradient descent step to mimic the query time situation where the SVMs are L2 normalized. The network is trained for 5 epochs using dropout at a rate of 0.5 for the FC layers. Each epoch takes about 3 hours using a single GPU, with an average training speed of 18 images/sec. MatConvNet (Vedaldi and Lenc (2015)) is used for the implementation.

5.4 Experimental evaluation

In this section we describe the test datasets, the evaluation protocol, and the baselines, followed by quantitative and qualitative evaluation.

Test Datasets. We test the compound query retrieval on two test sets – our new "Celebrities in Places" (CIP) dataset (Chapter 3), and the test subset of the synthetic dataset (Chapter 4). To provide more detailed results, we divide the queries for each test set based on whether the query face class has been seen at training time or not. Namely, the performance on "seen" classes demonstrates the ability of the system to search for faces which are already known to the system (though the specific face instances have not been used at training), while good performance on the "unseen" queries provides a stronger proof of generalization of the system's ability to answer any unconstrained query. To make the tests more challenging, we also augment each dataset with real distractor images from the Places205 dataset; the distractors do not contain any celebrities and therefore are negatives for all test queries. The same 16 place classes are used throughout. The statistics of the two test sets are shown in table 5.1.

Test set name	Query face classes		Target images		Distractors	Total
	unseen	seen	unseen	seen		
Synthetic	100	500	7.6k	8.3k	58k	73.8k
Celebrity in Places	792	223	1.7k	0.6k	58k	73.1k

Table 5.1: **Test sets statistics.** "Seen" and "unseen" correspond to query face classes which have or have not been seen by the network at training time, respectively. Target images are the set of all images which are positive for some query, while distractors are images which are negatives for all queries. For each query, there are at least 1 and at most 21 target images.

Evaluation Protocol. To evaluate the quality of compound query retrieval, for a given query we consider as positives only images that contain both the query face and place, and all other images are considered as negatives, even if they contain the query face or query place. For a given query, we compute Average Precision (AP) as well as recall at rank 5 (i.e. recall is deemed to be one if a positive is retrieved within the top 5 ranked images, otherwise 0), and these are averaged across queries to obtain the mean Average Precision (mAP) and the mean recall@5. To disentangle the quality of face and place retrieval, we also report the individual mAPs for faces and places separately.

Baselines. We compare our results with those of three late-fusion approaches that use independent face and place descriptors, instead of our face-aware place descriptors. The baselines differ in the Place-CNN used, and in the method of calibration. The first, *Places-L2norm* uses the Place-CNN trained on the entire Places205 dataset, with calibration by L2-normalising the weight vector for both the place and face SVMs (before combining the scores using the minimum score rule). The other two baselines are stronger – the Place-CNN is retrained to recognize only the 16 place classes using only the images in the Places205 dataset that contain faces (this improves the performance of scene classifications for images containing people). These two baselines differ only in the method of calibration: one, *Places-F-Platt*, is calibrated using Platt (Platt (1999)); the other, *Places-F-L2norm*, is calibrated using L2-normalization of the weight vector.

test set	descriptors	**faces in places**		faces only		places only
		unseen	seen	unseen	seen	
Syn	Places-L2norm	0.480 / 0.874	0.363 / 0.569	0.899	0.692	0.521
	Places-F-L2norm	0.533 / 0.859	0.426 / 0.649			0.534
	Places-F-Platt	0.491 / 0.830	0.330 / 0.557			
	Hybrid-FC	0.584 / 0.925	0.552 / 0.763			0.539
	Hybrid-CL	0.655 / 0.941	0.617 / 0.817			**0.604**
	Hybrid-CL-Aug	**0.676 / 0.946**	**0.655 / 0.846**	**0.914**	**0.743**	

Table 5.2: **Retrieval mAP and Recall@5 on the synthetic test set.** The results are reported in the format of 'mAP / Rec@5'. 'Aug' means that an augmentation is applied on the face feature vectors by taking the mean of 10 crops (single values are mAP only).

5.4.1 Retrieval results

Table 5.2 and Table 5.3 report the performance of all the baseline descriptors and our proposed two hybrid networks, one with interaction FC (*Hybrid-FC*) and the other with coupled loss (*Hybrid-CL*). As the tables reveal, the features generated by both of our hybrid CNN (*Hybrid-FC* and *Hybrid-CL*) perform much better than the three baselines.

Hybrid-FC. Interestingly, for the 'places only' task, *Hybrid-FC* achieves a higher mAP (*i.e.* 0.464) than the Places205 CNN (Zhou et al. (2015)) specifically fine-tuned for the 16 place classes images with faces.

The reason that the baselines have a higher mAP on the synthetic test set than on CIP (a mAP of 0.480 vs 0.381) is due to an implementation detail: the synthetic pipeline uses a threshold on the face classifier trained on independent $v_{\mathbf{face}}$ to select images, and so the synthetic test faces are well suited to the baseline descriptors. The performance gap between the baselines and the MA features becomes much larger when testing on the real target images in the CIP test set. Most importantly, the MA features generalize well to the 'unseen' face classes, and achieve a very high Rec@5 for both test sets.

test set	descriptors	**faces in places**		faces only		places only
		unseen	seen	unseen	seen	
CIP	Places-L2norm	0.381 / 0.550	0.325 / 0.502	0.693	0.699	0.381
	Places-F-L2norm	0.435 / 0.605	0.373 / 0.574			0.441
	Places-F-Platt	0.420 / 0.609	0.239 / 0.388			
	Hybrid-FC	0.603 / 0.777	0.529 / 0.711			0.464
	Hybrid-CL	0.640 / 0.807	0.577 / 0.760			**0.514**
	Hybrid-CL-Aug	**0.675 / 0.867**	**0.593 / 0.777**	**0.741**	**0.737**	

Table 5.3: **Retrieval mAP and Recall@5 on the 'Celebrity In Places' test set.** The results are reported in the format of 'mAP / Rec@5'. 'Aug' means that an augmentation is applied on the face feature vectors by taking the mean of 10 crops (single values are mAP only).

Hybrid-CL. Surprisingly, with a simpler architecture, our proposed hybrid network with coupled loss achieves even better performance than that with interaction FC. As is evident from the results, our new place descriptors perform much better than the three baselines. The best baseline is *Places-F-L2norm* for both 'faces in places' and 'places only'. As expected, fine-tuned Place-CNN (*Places-F-L2norm* and *Places-F-Platt*) have a higher 'places only' mAP than the original Place-CNN (*Places-L2norm*). Furthermore, the fact that *Places-F-Platt* performs worse than *Places-F-L2norm* for 'faces in places' task indicates that L2-normalization of the classification weight vectors is a better calibration method than Platt-scaling. Notice, for the 'places only' task, our models achieve a higher mAP (0.514) than the Places205 CNN specifically fine-tuned for the 16 place classes images with faces (0.441). This supports our argument that it is not just calibration that is being learnt, it is also a better face-aware place descriptor.

For the 'faces in places' task, the performance gap between the baselines and our method becomes more significant when testing on the real target images in the CIP test set. Most importantly, our descriptors generalize well to work with the 'unseen' face classes, and achieve a very high Rec@5 (0.807 and 0.760) for both test sets. Finally, we can improve the performance further with augmentation (mAP of 0.675 and 0.593) by taking the mean descriptor from 10 crops for faces.

In general, images of supermarkets, beaches and stages are well retrieved by compound-queries, whereas others are sometimes confused (*e.g.* football stadium and golf course are confused due to the large green areas). Furthermore, the face-aware place descriptors have a much better performance than the baselines when faces do not take up too much of the image (when they are too large, the background provides too little information on the scene).

As the results show, the hybrid network with coupled loss outperforms the one with interaction FC, while being simpler in terms of the structure and containing less parameters. Therefore, all the results and retrieval examples in the remaining sections of this chapter are based on our loss-coupled network (*Hybrid-CL*).

5.5 What has been learnt?

To illustrate the difference between the face-aware and original (i.e. trained on Places205 Dataset) place descriptors, we search for nearest neighbours in the entire Places205 Dataset using only the place descriptors. As shown in Figure 5.4, when querying with an image that contains faces, the most similar images returned by the face-aware descriptors do not necessarily contain large faces but do match the place class. In contrast, the original place descriptors find scenes containing faces, but often with the wrong place class. This reveals that the new descriptors tend to ignore the face and focus more on the background information, and consequently are more accurate for the compound query.

5.6 Retrieval examples

Retrieval results from the mutually-aware features and best baseline method (*Places-F-L2norm*) are compared in this section. For each search query, the top 4 images in the ranking list are displayed using the (hybrid-CNN) mutually-aware features (top

Figure 5.4: **The top ranking images by nearest neighbour search on place descriptors.** For each query image, the upper row uses the original Place-CNN descriptors, whilst the lower row uses face-aware descriptors. The first query: airport terminal, the second query: boat. The green box indicates the same scene as the query image, whereas red indicates a different scene.

row) and baseline method (bottom row). Green boxes denote positive retrieval while red ones are negatives.

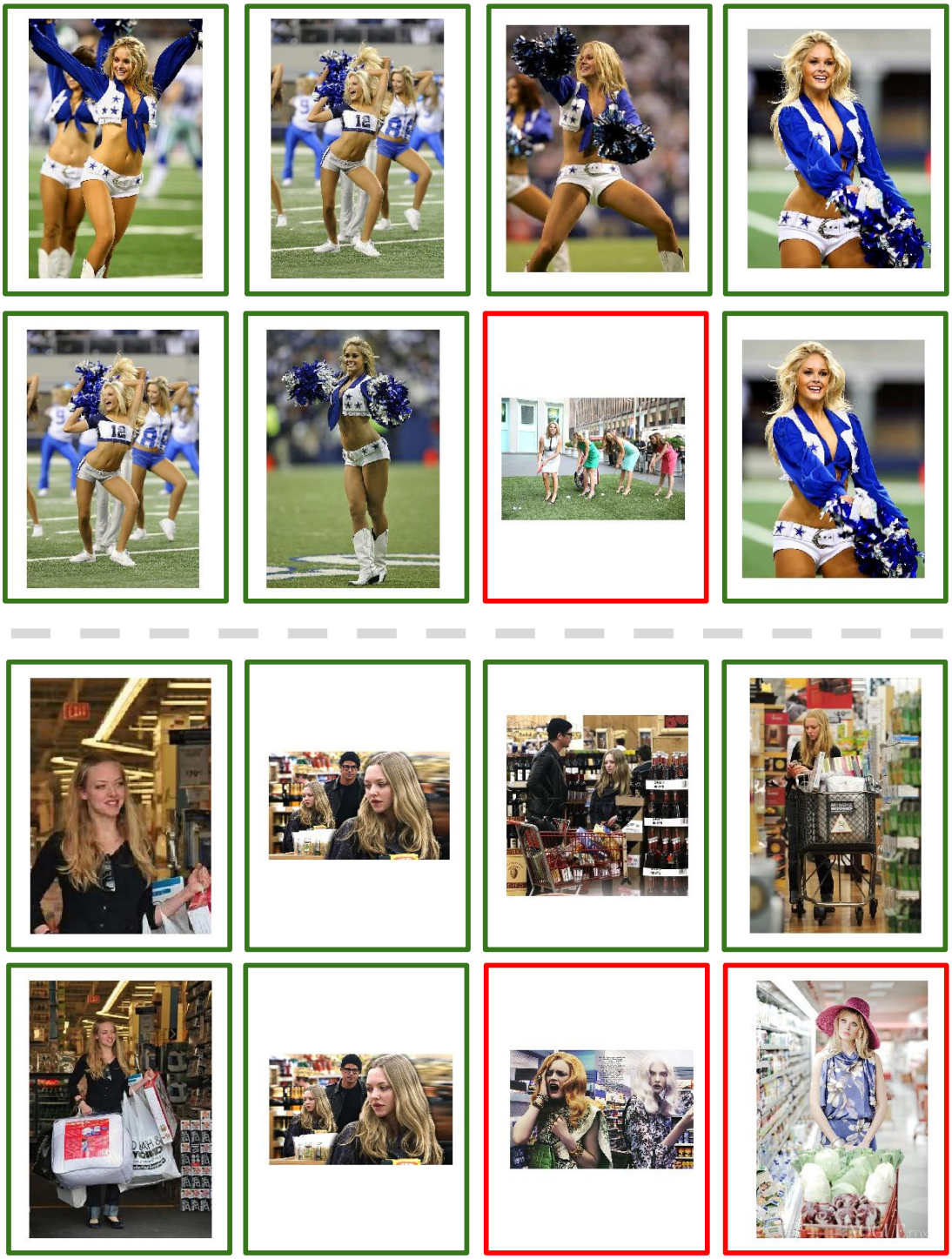


Figure 5.5: Top 4 retrieval results by the hybrid-CNN (top row) and baseline features (bottom row). Queries are (1) Abigail Klein in the stadium (2) Amanda Seyfried in the supermarket.

Figure 5.6: Top 4 retrieval results by the hybrid-CNN (top row) and baseline features (bottom row). Queries are (1) Anil Kapoor at the airport terminal (2) Annie Clark on the stage.

5.7 Network deployment and matching classification scores

In this section, we will describe a problem we encountered when the learnt hybrid network was deployed and used on a real-world dataset – the BBC News dataset – and provide a solution to it. Finally, a few retrieval examples are shown to illustrate the effectiveness of this solution.

BBC News Dataset. The BBC News dataset contains all the news programmes broadcast from 2007 to 2012 by the British Broadcasting Corporation (BBC), covering BBC1, BBC2, BBC3, BBC4, BBC Parliament and BBC News 24. Its programme genres include News, Factual, Politics, *etc.* The dataset contains about 20 million frames, including 5 million key frames which are encoded by our retrieval system.

5.7.1 Mismatching between training and testing

The network is used in four steps for any novel datasets. First, to encode the dataset, face descriptors and place descriptors are extracted from each image in the dataset in an offline manner. Second, given a query identity and a query place, a face classifier and a place classifier are trained separately at run-time. Third, the face classifier is applied to the face descriptors of the dataset, while the place classifier is applied to the place descriptors, which gives two scores per dataset image. Finally, a ranked list is obtained by taking the minimum between the face score and place score for each image.

However, when the network is applied on the BBC News dataset, we observe that the classification scores of the place classifiers are in general lower than those of the face classifiers, *i.e.* the place scores and face scores are not quite comparable. This problem can harm the retrieval performance. For example, for a particular query

identity and a query place, an image containing the query identity in the query place may have a place score of 0.45 and face score of 0.8, whereas an image containing only the query place but a different identity might score 0.5 and 0.5 for both place and face. By taking the minimum, the second image (with image score 0.5) will be ranked higher than the first image (with image score 0.45), even though the second image only contains the target place. This failing example is caused by the score overlap between the positive place scores and the negative face scores, which is due to the fact that face scores are in general higher than those of places.

We find that, even though in our network design the place scores and face scores are calibrated for the combination rule (*i.e.* minimum), at test time for any new query places, the on-the-fly trained classifier would produce scores that are lower than the scores produced by the classification layer in our hybrid network during training. One cause of this problem is that the parameters of the classification layer in the network are learnt without any weighting of the negative and positive training samples; whilst at run-time, for each place, the SVM classifier is trained such that the weights of the negative samples are the ratio between the number of positive and negative samples. As we only update the place stream of our hybrid network, this problem only happens for the place classifiers. Notably, this problem does not happen in the experiments on CIP dataset (Section 5.4.1), since we use the weights of the classification layer in the hybrid networks as our query classifiers for simplicity.

We visualize the score distribution of face and place for the training set (*i.e.* synthetic dataset generated in Chapter 4) to clearly illustrate the problem. As shown in Figure. 5.7 (top), the face scores produced by the classification layer (*i.e.* FC8) and the SVM classifiers trained at run-time share a similar distribution. On the other hand, there is an obvious difference between the place score distribution produced by the classification layer in the network and the SVM classifiers. A similar situation is also observed on our test set – 'Celebrity in Places', as Fig. 5.8 shows.

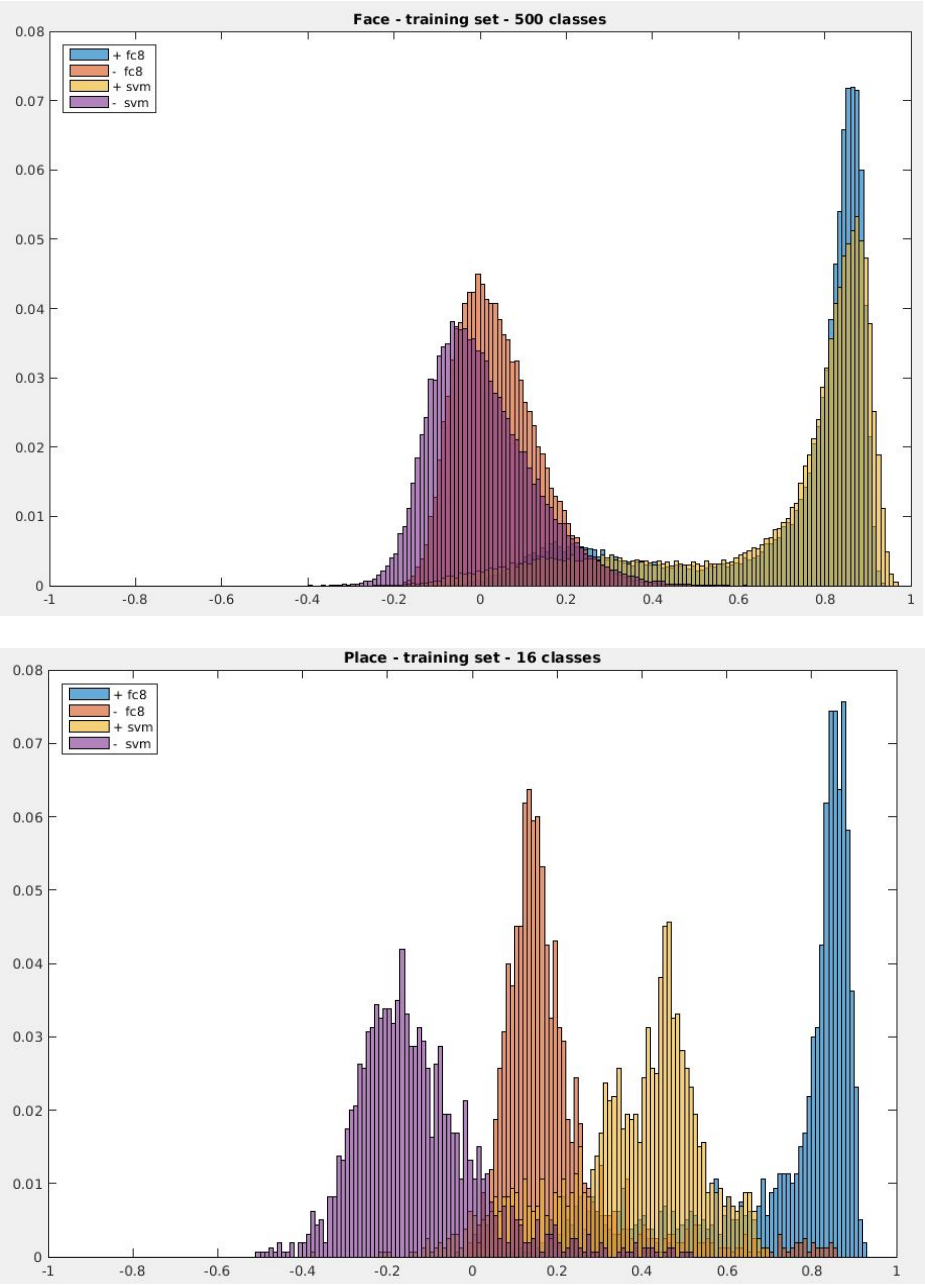


Figure 5.7: **Distribution of face (top) and place (bottom) scores for the synthetic training dataset.** '+ fc8' and '-fc8' denote the scores produced by the classification layer of our hybrid network for positive and negative samples respectively. Similarly, '+ svm' and '- svm' denote the scores obtained by SVM classifiers used at run-time.

Therefore, we propose a solution to match the place scores obtained by SVM classifiers used at run-time with those of the classification layer used during the training of the hybrid network, which will be discussed in the following.

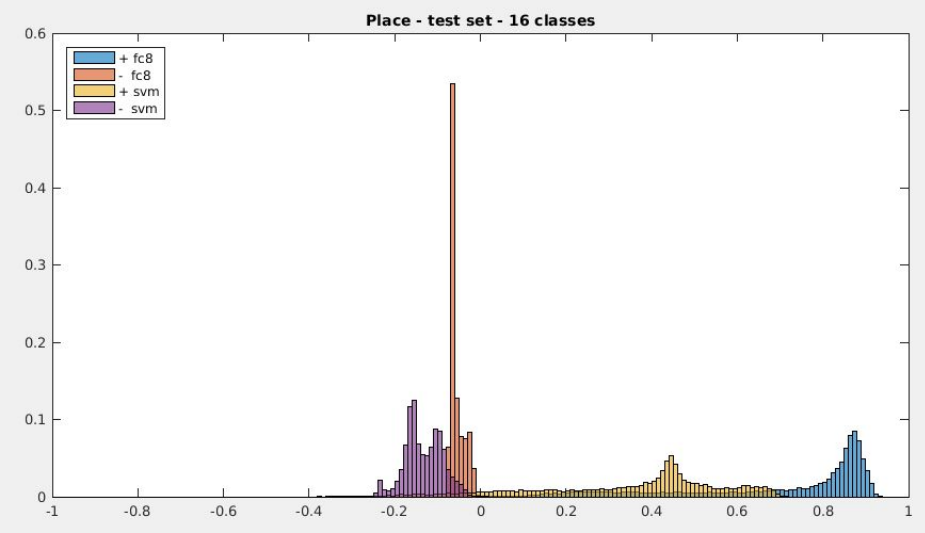


Figure 5.8: **Distribution of place scores for the 'Celebrity in Places' dataset.** '+ fc8' and '-fc8' denote the scores produced by the classification layer of our hybrid network for positive and negative samples respectively. Similarly, '+ svm' and '- svm' denote the scores obtained by SVM classifiers used at run-time.

5.7.2 Score matching by regression

We propose an approach to transform the scores of the SVM classifiers trained at run-time, such that the score distribution are closer to that produced by the classification layer in the hybrid network during training. The transformation is defined as Eq. 5.1, where s_{svm} is the score obtained by the SVM classifier, s_t is the transformed score, and a, b are the parameters of the linear transformation to be learnt.

$$s_t = s_{svm} \times a + b \tag{5.1}$$

Learning by regression. The parameters of the transformation can be learnt efficiently by linear regression. The training data we used are the classification scores (of both positive and negative matching) obtained from the synthetic image training set introduced in Chapter 4, which are also used to train the hybrid network. The objective of the learning is to minimize the average squared difference between the transformed SVM classification scores and the scores produced by the classification

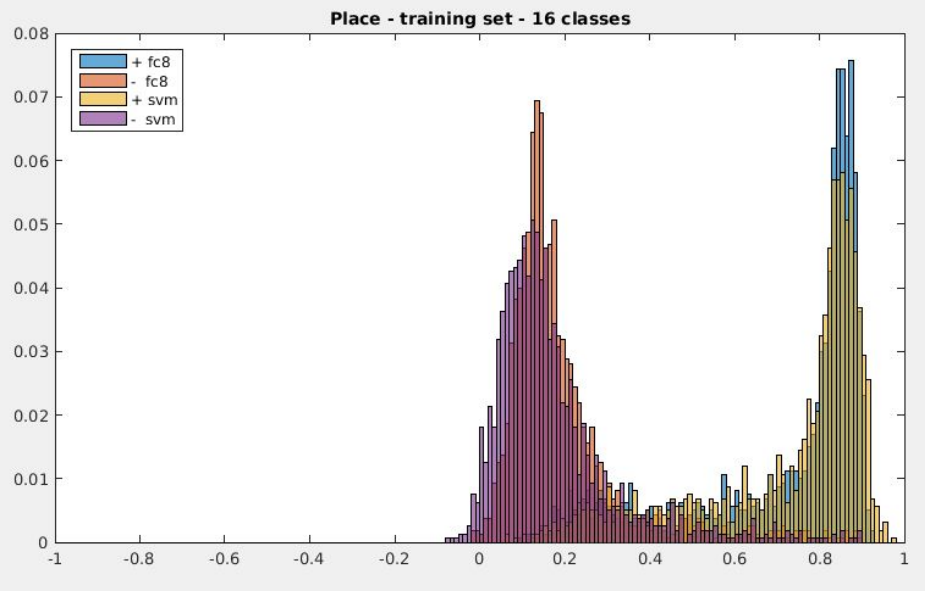


Figure 5.9: **Distribution of transformed place scores for the synthetic training dataset.** '+ fc8' and '-fc8' denote the scores produced by the classification layer of our hybrid network for positive and negative samples respectively. Similarly, '+ svm' and '- svm' denote the scores obtained by SVM classifiers used at runtime.

layer in the network.

We then apply the learnt linear transformation to the SVM classification scores on the training set (Fig. 5.9) and our test set – 'Celebrity In Places' (Fig. 5.10). As we can see, the score distribution of the transformed SVM classification scores is quite similar to that of classification layer in the network. On the test set, the positive scores are quite close to each other (*i.e.* overlap by a large amount),while the negative scores are still arpart from each other. However, in practice we find that matching the positive scores is more important, and the transformation seems to improve the retrieval results a lot on the BBC News dataset.

5.7.3 Retrieval examples on BBC News dataset

After matching the scores of the SVM classifiers to the score range during training by score regression, three retrieval examples are shown in Fig. 5.11, Fig. 5.12 and Fig. 5.13.

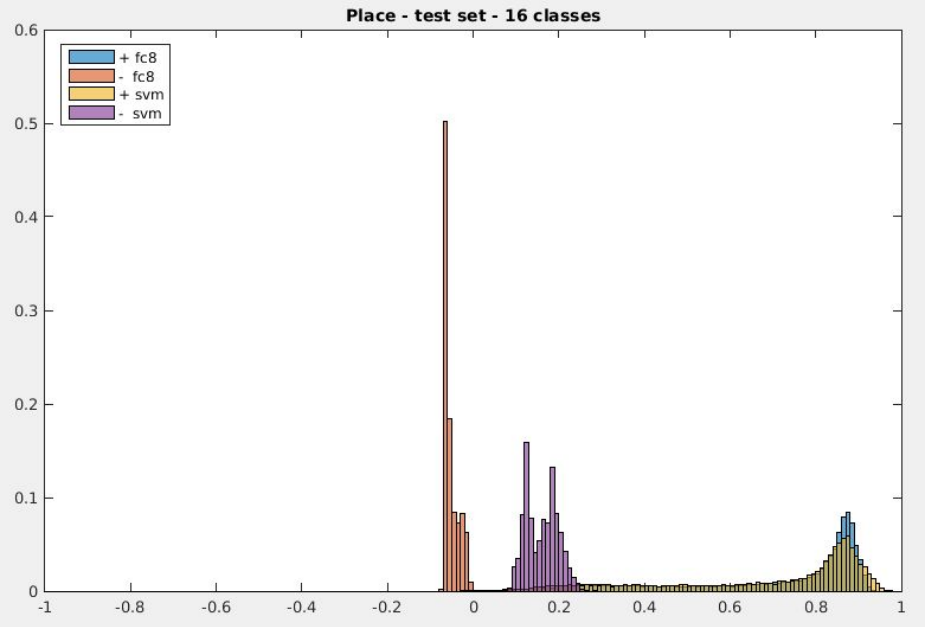


Figure 5.10: **Distribution of transformed place scores for the 'Celebrity in Places' dataset.** '+ fc8' and '-fc8' denote the scores produced by the classification layer of our hybrid network for positive and negative samples respectively. Similarly, '+ svm' and '- svm' denote the scores obtained by SVM classifiers used at run-time.

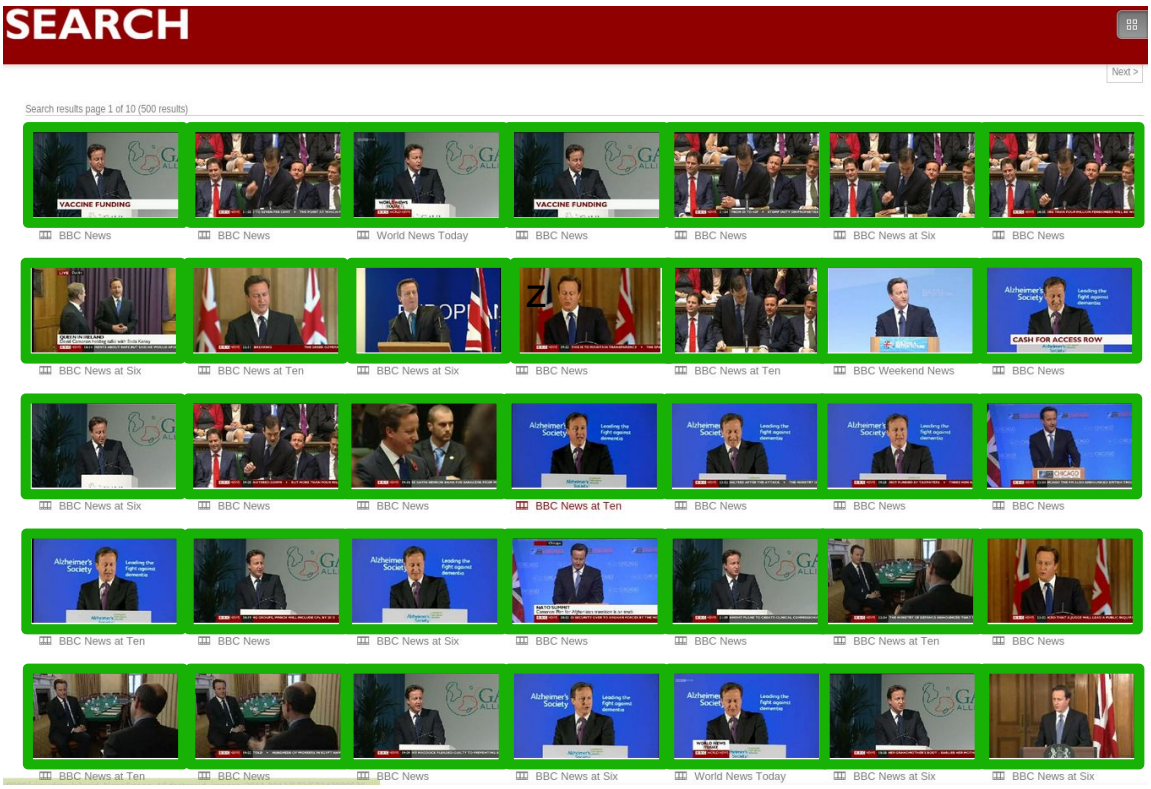


Figure 5.11: **Retrieval examples on the BBC News dataset.** Query is 'David Cameron in the conference room'. Green boxes indicate the target images.

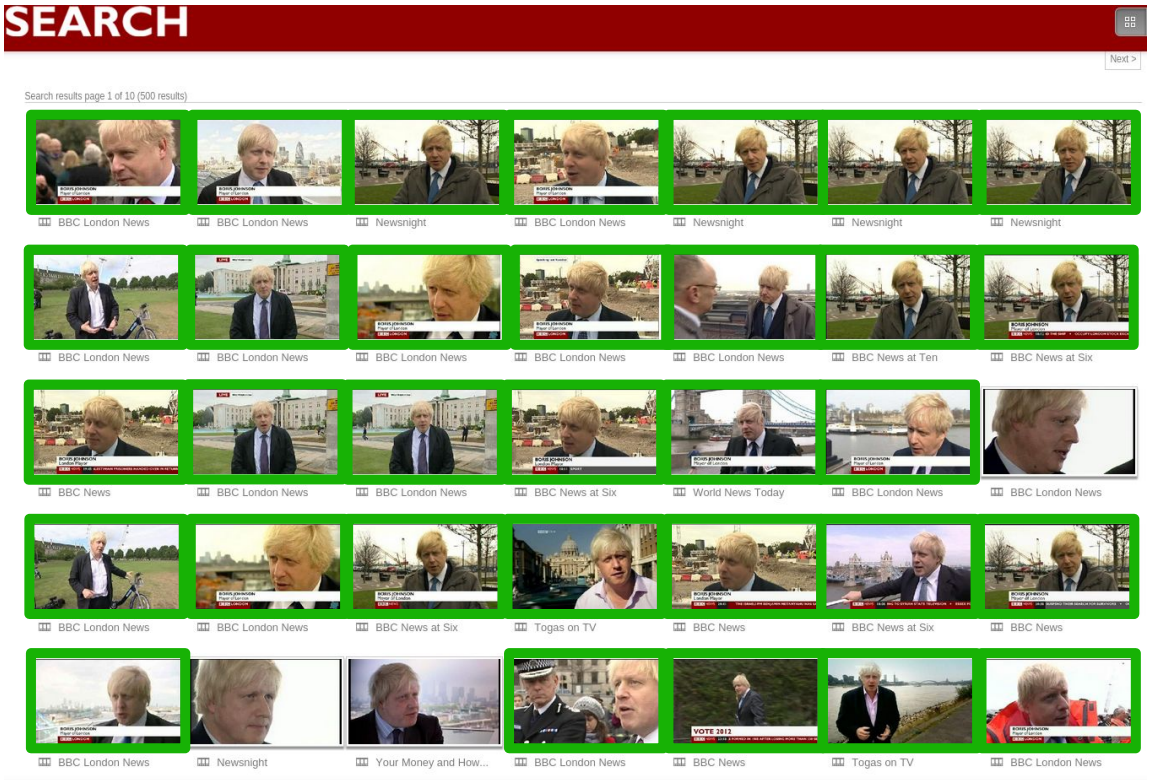


Figure 5.12: **Retrieval examples on the BBC News dataset.** Query is 'Boris Johnson outdoors'. Green boxes indicate the target images.

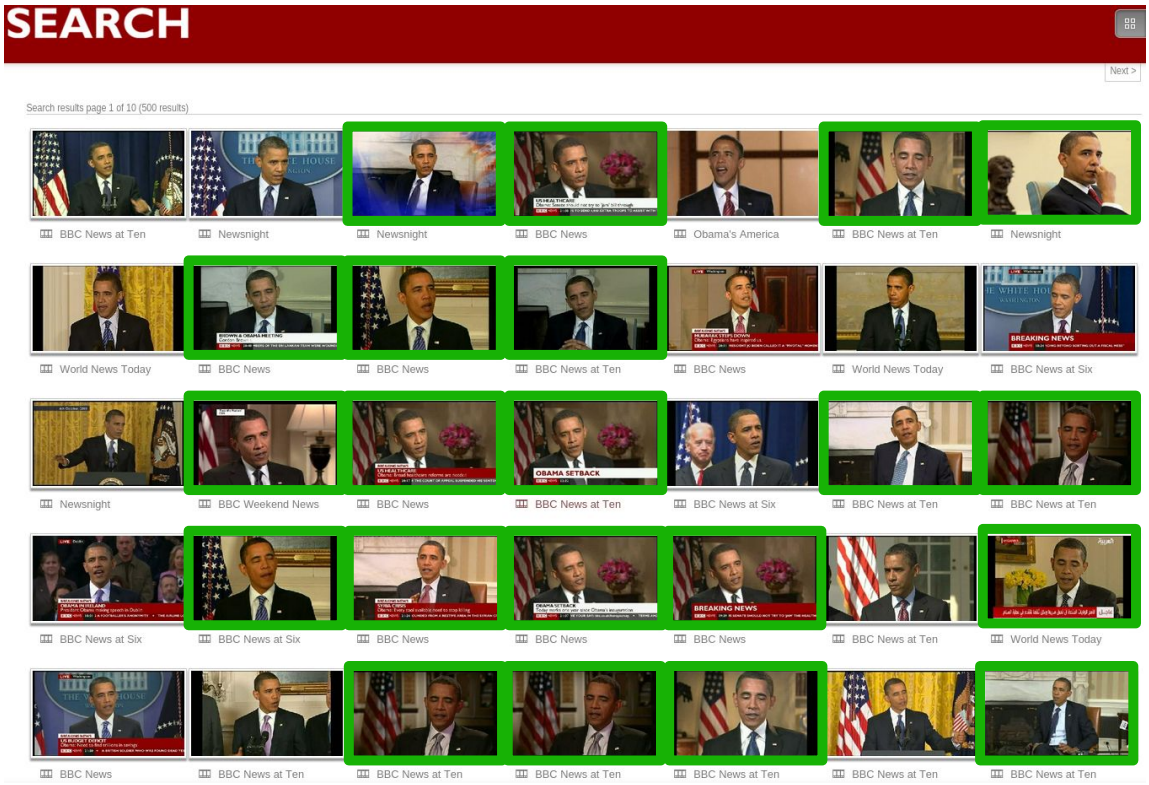


Figure 5.13: **Retrieval examples on the BBC News dataset.** Query is 'Barack Obama in the office'. Green boxes indicate the target images.

5.8 Summary

Given *any* face-in-a-place compound query, our goal in this chapter is to rank images containing that face-in-a-place combination highly. We have proposed a compound query retrieval method which searches for particular people in particular scenes in a large image dataset. We also present two hybrid networks which are trained for the task. In the first hybrid network, faces and places pre-trained networks generate place descriptors that are aware of faces and face descriptors, through a connection in the loss during training. In the second network, face and place features become mutually-aware during training through two fully-connected layers. Finally, we demonstrate that both networks outperform the baselines significantly, and show that the simple loss-coupled network can achieve leading results.

Chapter 6

Compact aggregation for set retrieval

The objective of this chapter is to learn a compact embedding of a set of descriptors that is suitable for efficient retrieval and ranking for the set retrieval problem, whilst maintaining discriminability of the individual descriptors. As discussed in Chapter 1, set retrieval refers to the task in which each set contains elements (*e.g.* a set can be an image and elements can be faces in the image), and we wish to order the sets according to a query on multiple elements, such that those sets that contain all the elements of the query are ranked first, followed by sets that satisfy all but one of the query elements, *etc.* We focus on a specific example of this general set retrieval problem – that of retrieving images containing multiple faces from a large scale dataset of images. Here the set consists of the face descriptors in each image, and given a query for multiple identities, the goal is then to retrieve, in order, images which contain all the identities, all but one, *etc.* An example of this ranking is shown in Figure 6.1 for two queries.

Although we have motivated this question by face retrieval, it is quite general: there is a set of elements where each element is represented by a vector of dimension D_F, and we wish to represent this set by a single vector of dimension D, where $D = D_F$ in practice, without losing information essential for the retrieval task. Of course, if the

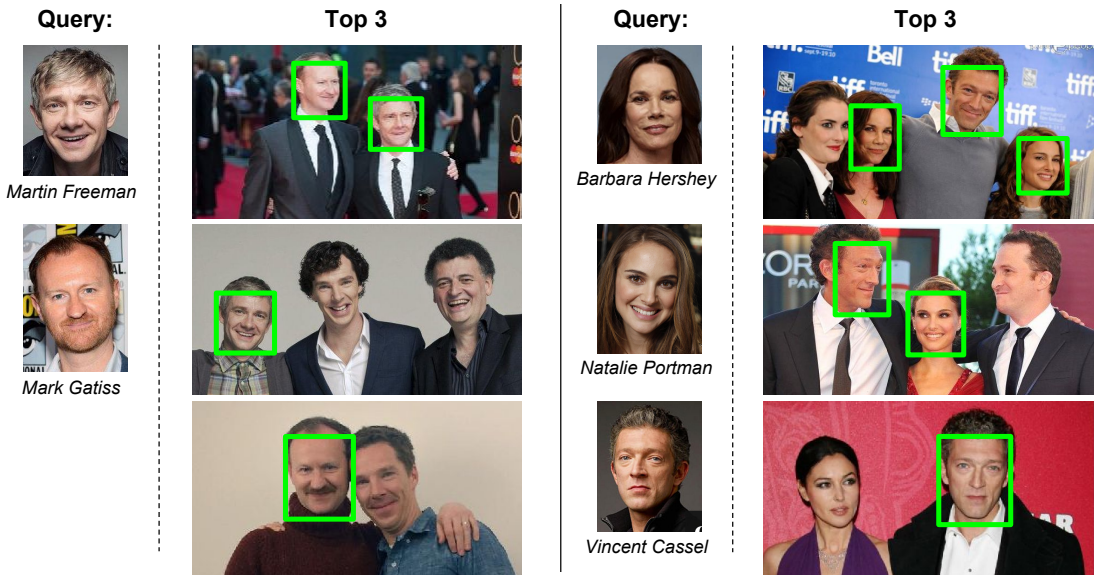


Figure 6.1: **Images ranked using set retrieval for two example queries.** The query faces are given on the left of each example column, together with their names (only for reference). Left: a query for two identities; right: a query for three identities. The first ranked image in each case contains all the faces in the query. Lower ranked images partially satisfy the query, and contain progressively fewer faces of the query. The results are obtained using the compact set retrieval descriptor generated by the SetNet architecture, by searching over 200k images of the *Celebrity Together* dataset introduced in Chapter 3.

total number of elements in all sets, N, is such that $N \leq D$ then this certainly can be achieved provided that the set of vectors are orthogonal. However, we will consider the situation commonly found in practice where $N \gg D$, *e.g.* D is small, typically 128 (to keep the memory footprint low), and N is in the thousands.

We make the following contributions in this chapter: first, we introduce a trainable CNN architecture for the set-retrieval task that is able to learn to aggregate face vectors into a fixed length descriptor in order to minimize interference, and also is able to rank the face sets according to how many identities are in common with the query using this descriptor. To do this, we propose a paradigm shift where we draw motivation from image retrieval based on local descriptors. In image retrieval, it is common practice to aggregate all local descriptors of an image into a fixed-size image-level vector representation, such as bag-of-words (Sivic and Zisserman (2003)) and VLAD (Jégou et al. (2010)); this brings both memory and speed improvements over storing all local

descriptors individually. We generalize this concept to set retrieval, where instead of aggregating local interest point descriptors, set element descriptors are pooled into a single fixed-size set-level representation. For the particular case of face set retrieval, this corresponds to aggregating face descriptors into a set representation. The novel aggregation procedure is described in Section 6.1 where compact set-level descriptors are trained in an end-to-end manner using ResNet-50 (He et al. (2016)) as the base CNN.

In Section 6.2, we describe the training procedure and the loss applied on top of the network architecture. Section 6.3 then explains the implementation details of the network at both training and test time. The performance of the set-level descriptors is evaluated in Section 6.4. We first 'stress test' the descriptors by progressively increasing the number of faces in each set, and monitoring their retrieval performance. We also evaluate retrieval on the *Celebrity Together* dataset (introduced in Chapter 3), where images contain a variable number of faces, with many not corresponding to the queries.

The second contribution of this chapter is exploring the speed *vs* retrieval quality trade-off for set retrieval using this compact descriptor, and proposing efficient algorithms that can achieve immediate (real-time) retrieval on very large scale datasets while maintaining a good retrieval quality.

6.1 SetNet – a CNN for set retrieval

As described in the previous section, using a single fixed-size vector to represent a set of vectors is a highly appealing approach due to its superior speed and memory footprint over storing a descriptor-per-element. In this section, we propose a CNN architecture, *SetNet*, for the end-task of set retrieval.

6.1.1 Network architecture

There are two objectives to be achieved in the design of the CNN:

1. To learn the element descriptors together with the aggregation in order to minimise the loss in face classification performance between using individual descriptors for each face, and an aggregated descriptor for the set of faces. This is achieved by training the network for this task, using an architecture combining ResNet for the individual descriptors together with NetVLAD for the aggregation.

2. To be able to rank the images using the aggregated descriptor in order of the number of faces in each image that correspond to the identities in the query. This is achieved by scoring each face using a logistic regression classifier. Since the score of each classifier lies between 0 and 1, the score for the image can simply be computed as the sum of the individual scores, and this summed score determines the ranking function.

As an example of the scoring function, if the search is for two identities and an image contains faces of both of them (and maybe other faces as well), then the ideal score for each relevant face would be one, and the sum of scores for the image would be two. If an image only contains one of the identities, then the sum of the scores would be one. The images with higher summed scores are then ranked higher and this naturally orders the images by the number of faces it contains that satisfy the query.

To deploy the set level descriptor for retrieval in a large scale dataset, there are two stages:

Offline: SetNet is used to compute face descriptors for each face in an image, and aggregate them to generate a set-vector representing the image. This procedure is carried out for every image in the dataset, so that each image is represented by a

single vector.

At **run-time**, to search for an identity, a face descriptor is computed for the query face using SetNet, and a logistic regression classifier used to score each image based on the scalar product between its set-vector and the query face descriptor. Searching with a set of identities amounts to summing up the image scores of each query identity.

The *SetNet* architecture (Figure 6.2) conceptually has two parts: (i) each face is passed through a feature extractor network separately, producing one descriptor per face; (ii) the multiple face descriptors are aggregated into a single compact vector using a modified NetVLAD layer, followed by a trained dimensionality reduction. At training time, we add a third part which emulates the run-time use of logistic regression classifiers. All three parts of the architecture are described in more detail next.

6.1.2 Feature extraction

The first part is a modified ResNet-50 (He et al. (2016)) chopped after the global average pooling layer. The ResNet-50 is modified to produce 128-D vectors in order to keep the dimensionality of our feature vectors relatively low (we have not observed a significant drop in face recognition performance from the original 2048-D descriptors). The modification is implemented by adding a fully-connected (FC) layer of size 2048×128 after the global average pooling layer in the original ResNet, in order to obtain a lower dimensional face descriptor.

ResNet Modification. In detail, the modification (as shown in Figure 6.3) is implemented by adding a fully-connected (FC) layer of size 2048×128 after the global average pooling layer in the original ResNet, such that the size of the output feature vector of the modified ResNet is 128-D instead of 2048-D. This additional fully-

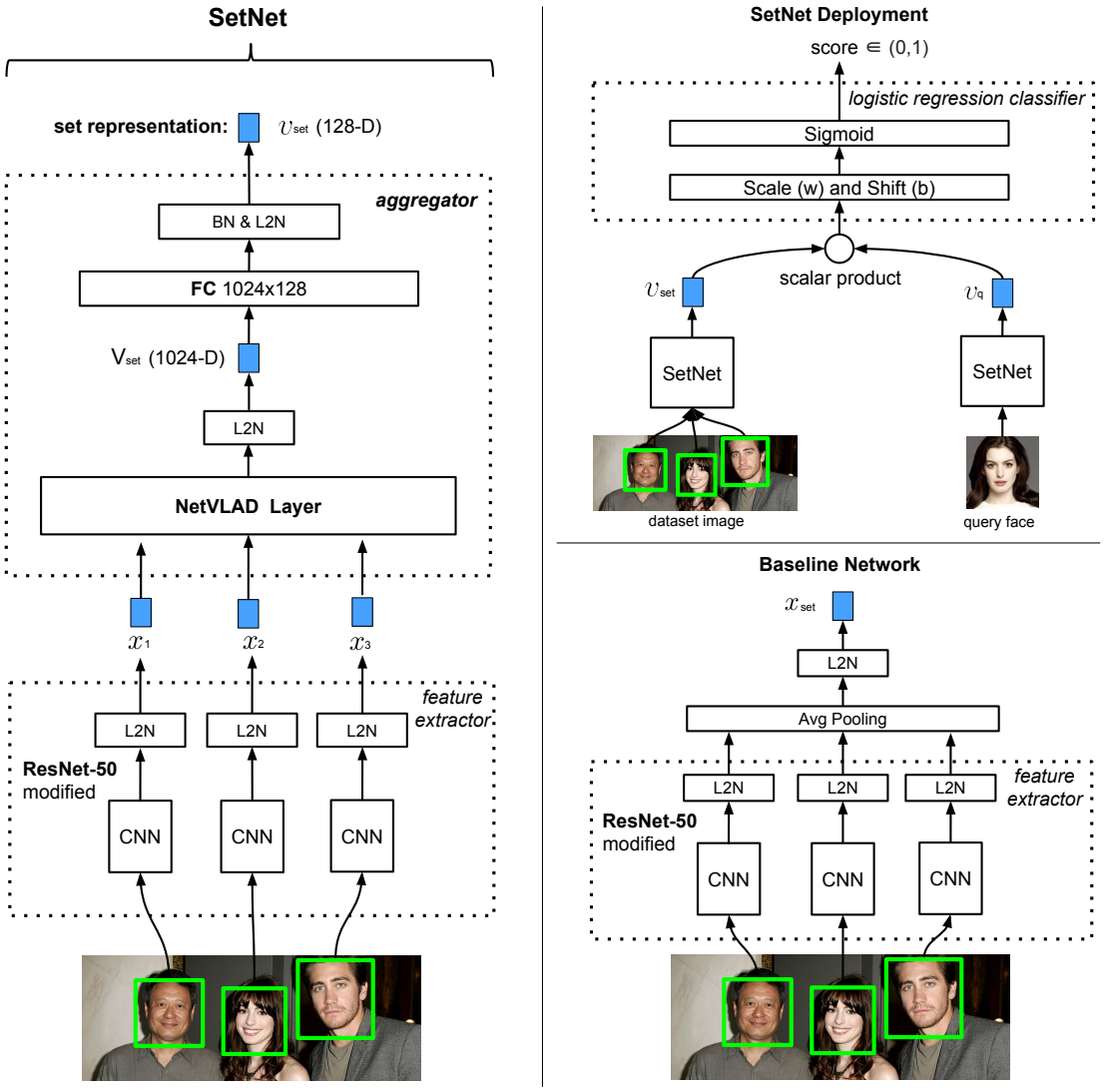


Figure 6.2: **SetNet architecture and training. Left**: SetNet – features are extracted from each face in an image using a modified ResNet-50. They are aggregated using a modified NetVLAD layer into a single 1024-D vector which is then reduced to 128-D via a fully connected dimensionality reduction layer, and L2-normalized to obtain the final image-level compact representation. **Right (top)**: at test time, a query descriptor, v_q, is obtained for each query face using SetNet (the face is considered as a single-element set), and the dataset image is scored by a logistic regression classifier on the scalar product between the query descriptor v_q and image set descriptor v_{set}. The final score of an image is then obtained by summing the scores of all the query identities. **Right (bottom)**: the baseline network has the same feature extractor as SetNet, but the feature aggregator uses an average-pooling layer, rather than NetVLAD.

connected layer essentially acts as a dimensionality reduction layer if we consider the input 2048-D feature vector as a stack of feature maps of spatial size 1×1. This modification introduces around 260k extra weights to the network, and we do not

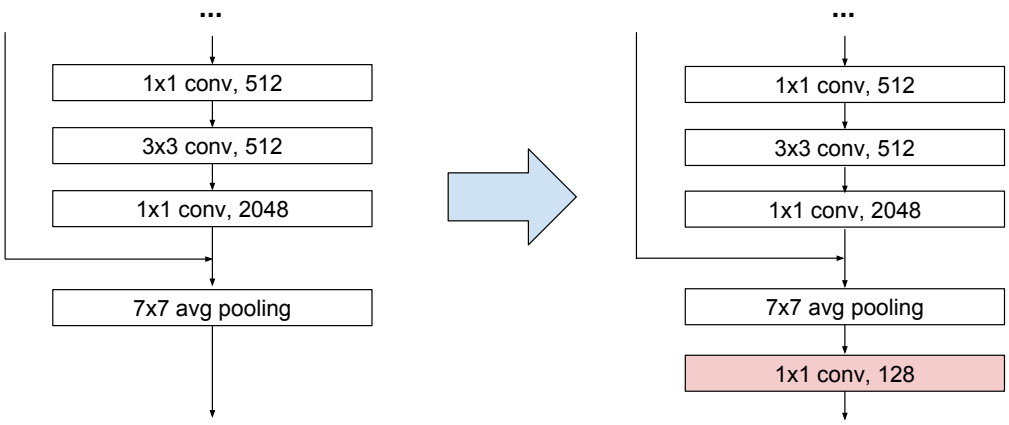


Figure 6.3: **Modification of ResNet-50.** The modification is highlighted in red, where an extra fully-connected layer is added after the global average pooling layer to output 128-D features.

observe a prominent drop in the face classification performance using an output of compact 128-D features compared to the original 2048-D feaures.

It is important to note that training the ResNet-50 128-D network on the VGGFace2 dataset (Cao et al. (2018)) provides a very strong baseline (referred to as *Baseline* in the stress test). Although the objective of this chapter is not face classification, to demonstrate the performance of this baseline network, we test it on the public IARPA Janus Benchmark A (IJB-A dataset) released by Klare et al. (2015). Specifically, we follow the same test procedure for the 1:N Identification task described in (Cao et al. (2018)). In terms of the true positive identification rate (TPIR), it achieves 0.975, 0.993 and 0.994 for the top 1, 5 and 10 ranking respectively, which is on par with the state-of-the-art networks in (Cao et al. (2018)).

In this part, individual element descriptors $(x_1, \ldots, x_{Nf})$ are obtained from ResNet, where N_F is the number of faces in an image.

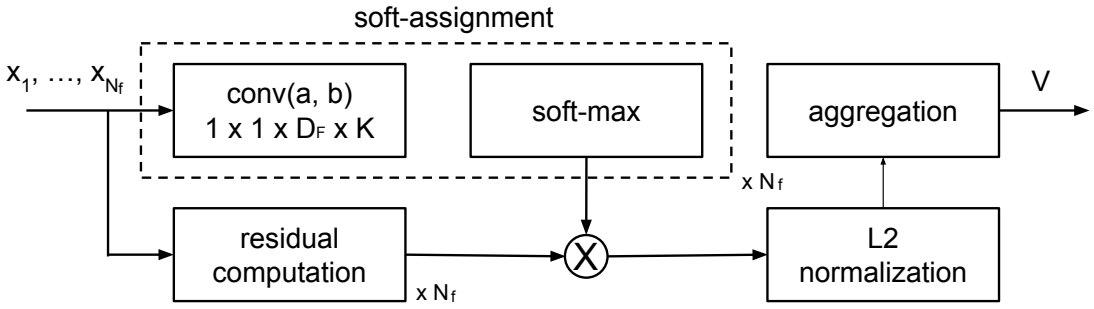


Figure 6.4: **NetVLAD layer.** Illustration of the NetVLAD layer (Arandjelović et al. (2016)), corresponding to equation (6.1), and slightly modified to perform L2-normalization before aggregation; see Section 6.1.4 for details.

6.1.3 Feature aggregation

Face features are aggregated into a single vector V using a NetVLAD layer (illustrated in Figure 6.4, and described below in Section 6.1.4). The NetVLAD layer is slightly modified by adding an additional L2-normalization step – the total contribution of each face descriptor to the aggregated sum (*i.e.* its weighted residuals) is L2-normalized in order for each face descriptor to contribute equally to the final vector; this procedure is an adaptation of residual normalization (Delhumeau et al. (2013)) of the vanilla VLAD to NetVLAD. The NetVLAD-pooled features are reduced back to 128-D by means of a fully-connected layer followed by batch-normalization (Ioffe and Szegedy (2015)), and L2-normalized to produce the final set representation v_{set}. This final L2-normalization of feature vectors also matches training (Section 6.2) with test time situation, similar to (Wang et al. (2017a), Zhong et al. (2016)).

Training block. At training time, an additional logistic regression loss layer is added to mimic the run-time scenario where a logistic regression classifier is used to score each image based on the scalar product between its set-vector and the query face descriptor. Note, SetNet is used to generate both the set-vector and the face descriptor. Section 6.2 describes the appropriate loss and training procedure in more detail.

6.1.4 NetVLAD trainable pooling

NetVLAD has been shown to outperform sum and max pooling for the same vector dimensionality, which makes it perfectly suited for our task. Here we provide a brief overview of NetVLAD. For full details please refer to (Arandjelović et al. (2016)).

For N_f D_F-dimensional input descriptors $\{x_i\}$ and a chosen number of clusters K, NetVLAD pooling produces a single $D_F \times K$ vector V (for convenience written as a $D_F \times K$ matrix) according to the following equation:

$$V(j,k) = \sum_{i=1}^{N_f} \frac{e^{a_k^T x_i + b_k}}{\sum_{k'} e^{a_{k'}^T x_i + b_{k'}}} (x_i(j) - c_k(j)) \tag{6.1}$$

where $\{a_k\}$, $\{b_k\}$ and $\{c_k\}$ are trainable parameters for $k \in [1, 2, \ldots, K]$. Note that j denotes the jth cluster. The first term corresponds to the soft-assignment weight of the input vector x_i for cluster k, while the second term computes the residual between the vector and the cluster centre. Finally, the vector is L2-normalized.

6.2 Network training and Loss function

In order to achieve the two objectives outlined at the beginning of this Chapter, a Multi-label logistic regression loss is used. Suppose a particular training image contains N_f faces, and the mini-batch consists of faces for P identities. Then in a forward pass at training time, the descriptors for the N_f faces are aggregated into a single feature vector, v_{set}, using the SetNet architecture, and a face descriptor, v_f, is computed using SetNet for each of the faces of the P identities. The training image is then scored for each face N_f by applying a logistic regressor classifier to the scalar product $v_{set}^T v_f$, and the score should ideally be one for each of the N_f identities in the image, and zero for the other $P - F$ faces. The loss measures the deviation from this ideal score, and the network learns to achieve this by maintaining the discriminability for individual face descriptors after aggregation.

In detail, incorporating the loss is achieved by adding an additional layer at training time which contains a logistic regression loss for each of the P training identities, and is trained together with the rest of the network.

Multi-label logistic regression loss. For each training image (set of faces), the value of the loss is:

$$-\sum_{f=1}^{P} y_f \log(\sigma(w(v_f^T v_{set}) + b)) + (1 - y_f) \log(1 - \sigma(w(v_f^T v_{set}) + b)) \tag{6.2}$$

where $\sigma(s) = 1/(1+\exp(-s))$ is a logistic function, P is the number of face descriptors (the size of the mini-batches at training), and w and b are the scaling factor and shifting bias respectively of the logistic regression classifier, and y_f is a binary indicator whether face N_f is in the image or not. Note that multiple y_f's are equal to 1 if there are multiple faces which correspond to the identities in the image.

6.3 Implementation details

This section gives full details of the training procedure, including how the network is used at run-time to rank the dataset images given query examples.

Training data. The network is trained using faces from the training partition of the VGGFace2 dataset (Cao et al. (2018)). This consists of 8631 identities, with on average 360 face samples for each identity.

Balancing positives and negatives. For each training image (face set) there are many more negatives (the $P - F$ identities outside of the image set) than positives (identities in the set), *i.e.* most y_f's in eq. (6.2) are equal to 0 with only a few 1's. To restore balance, the contributions of the positives and negatives to the loss function are down-weighted by their respective counts.

Initialization and pre-training. A good (and necessary) initialization for the network is obtained as follows. The face feature extraction block is pretrained for single face classification on the VGGFace2 Dataset (Cao et al. (2018)) using softmax loss. The NetVLAD layer, with $K = 8$ clusters, is initialized using k-means as in Arandjelović et al. (2016). The fully-connected layer, used to reduce the NetVLAD dimensionality to 128-D, is initialized by PCA, *i.e.* by arranging the first 128 principal components into the weight matrix. Finally, the entire SetNet is trained for face aggregation using the Multi-label logistic regression loss (Section 6.2).

Training details. Training requires face set descriptors computed for each image, and query faces (which may or may not occur in the image). The network is trained on synthetic face sets which are built by randomly sampling identities (e.g. two identities per image). For each identity in a synthetic set, two faces are randomly sampled (from the average of 360 for each identity): one contributes to the set descriptor (by combining it with samples of the other identities), the other is used as a query face, and its scalar product is computed with all the set descriptors in the same mini-batch. In our experiments, each mini-batch contains 84 faces. Stochastic gradient descent is used to train the network (implemented in MatConvNet (Vedaldi and Lenc (2015))), with weight decay 0.001, momentum 0.9, and an initial learning rate of 0.001 for pre-training and 0.0001 for fine-tuning; the learning rates are divided by 10 in later epochs.

Training faces are resized such that the smallest dimension is 256 and random 224×224 crops are used as inputs to the network. To further augment the training faces, random horizontal flipping and up to 10 degree rotation is performed. At test time, faces are resized so that the smallest dimension is 224 and the central crop is taken.

Dataset retrieval. Suppose we wish to retrieve images containing multiple query faces (or a subset of these). First, a face descriptor is produced by *SetNet* for each query face. The face descriptors are then used to score a dataset image for each query identity, followed by summing the individual logistic regression scores to produce the final image score. A ranked list is obtained by sorting the dataset images in non-increasing score order. In the case where multiple face examples are available for a query identity, the multiple descriptors produced by *SetNet* are simply averaged and L2-normalized to form a richer descriptor for that query identity.

6.4 Experiments and results

In this section we investigate three aspects: first, in Section 6.4.1 we study the performance of different models (SetNet and baselines) as the number of faces per image in the dataset is increased. Second, we compare the performance of SetNet and the best baseline model on the real-world *Celebrity Together* dataset in Section 6.4.2. Third, the trade-off between time complexity and set retrieval quality is investigated in Section 6.4.3. Lastly, we compare the descriptor orthogonality in Section 6.4.4.

Note, in all the experiments, there is no overlap between the query identities used for testing and the identities used for training the network, as the VGG Face Dataset (Parkhi et al. (2015)) which is used for testing, e.g. for forming the *Celebrity Together* dataset) and the VGGFace2 Dataset (Cao et al. (2018)) which is used for training share no common identities.

Evaluation protocol. We use Normalized Discounted Cumulative Gain (nDCG) to evaluate set retrieval performance, as it can measure how well images containing *all* the query identities and also *subsets* of the queries are retrieved. For this measurement, images have different relevance, not just a binary positive/negative label; the relevance of an image is equal to the number of query identities it contains. We report nDCG@10

and nDCG@30, where nDCG@N is the nDCG for the ranked list cropped at the top N retrievals. nDCGs are written as percentages, so the scores range between 0 and 100.

6.4.1 Stress test

In this test, we aim to investigate how different models perform with increasing number of faces per set (image) in the test dataset. The effects of varying the number of faces per set used for training are also studied.

Test dataset synthesis. To this end, a base dataset with 64k face sets of 2 faces each is synthesized, using only the face images of labelled identities in the *Celebrity Together* dataset. A random sample of 100 sets of 2 identities are used as queries, taking care that the two queried celebrities do appear together in some dataset face sets. To obtain four datasets of varying difficulty, 0, 1, 2 and 3 distractor faces per set are sampled from the unlabelled face images in *Celebrity Together* Dataset, taking care to include only true distractor people, *i.e.* people who are not in the list of labelled identities in the *Celebrity Together* dataset. Therefore, all four datasets contain the same number of face sets (64k) but have a different number of faces per set, ranging from 2 to 5. Importantly, by construction, the relevance of each set to each query is the same across all four datasets, which makes the performance numbers comparable across them.

Methods. The *SetNet* models are trained as described in Section 6.2, where the suffix *'-2'* or *'-3'* denotes whether 2- or 3-element sets are used during training. For baselines, the optional suffix *'+W'* indicates whether the face descriptors have been whitened and L2-normalized before aggregation; whereas for SetNet, *'+W'* means that the set descriptors are whitened. For example, *SetNet-2+W* is a model trained

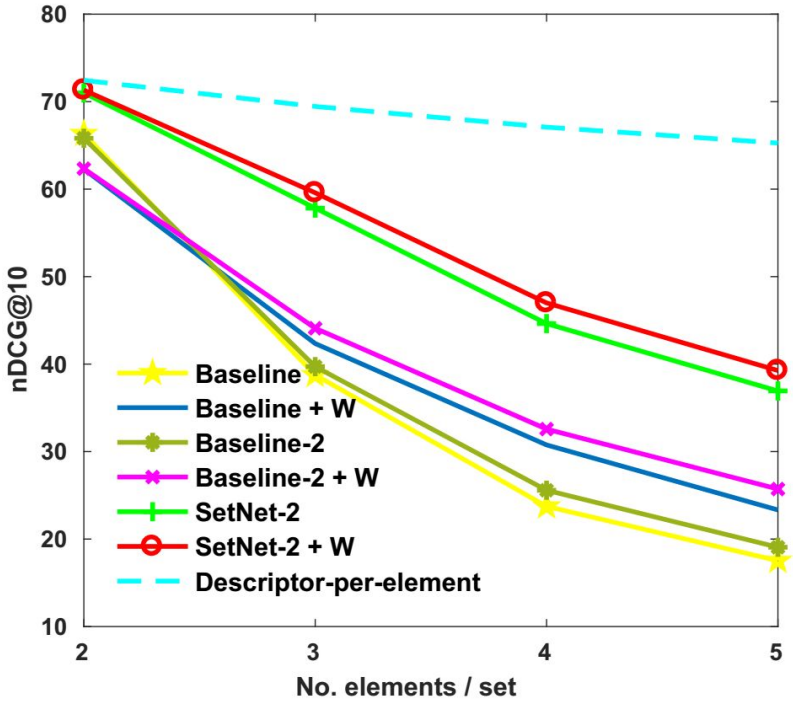


Figure 6.5: **nDCG@10 of stress test comparison of different models.** There are 100 query sets, each with two identities. Horizontal axis denotes the number of elements (faces) per set (image) in the test dataset.

with 2-element sets and whitening. Baselines follow the same naming convention, and use ResNet-50 with average-pooling (*i.e.* the same feature extractor network, data augmentation, optional whitening, *etc.*), where the architectural difference from the SetNet is that the aggregator block is replaced with average-pooling followed by L2-normalization (as shown in Figure 6.2). The baselines are trained in the same manner as SetNet. The exceptions are *Baseline* and *Baseline+W*, which simply use ResNet-50 with average-pooling, but no training. For reference, an upper bound performance (*Descriptor-per-element*, see Section 6.4.3 for details) is also reported, where no aggregation is performed and all descriptors for all elements are stored. In this test, one randomly sampled face example per query identity is used to query the dataset. The experiment is repeated 10 times using different face examples, and the nDCG scores are averaged.

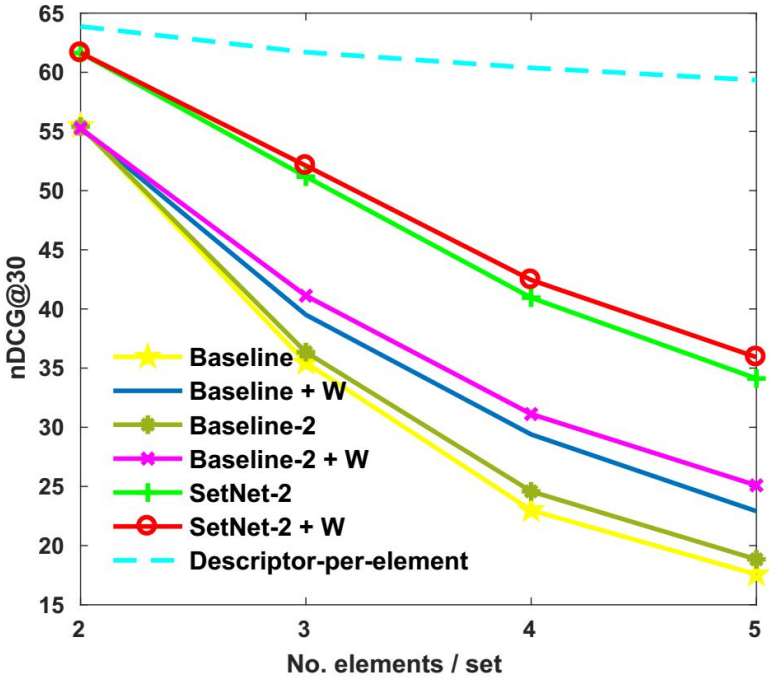


Figure 6.6: **nDCG@30 of stress test comparison of different models.** There are 100 query sets, each with two identities. Horizontal axis denotes the number of elements (faces) per set (image) in the test dataset.

Results. From the results in Figure 6.5 and Figure 6.6 it is clear that *SetNet-2+W* and *SetNet-3+W* outperform all baselines for both nDCG@10 and nDCG@30. As expected, the performance of all models decreases as the number of elements per set increases due to larger cross-element interference. However, *SetNet+W* deteriorates more gracefully (the margin between it and the baselines increases), demonstrating that our training makes SetNet learn representations which minimise the interference between elements. The set training is beneficial for all architectures, as *Baseline-2+W* and *Baseline-3+W* achieve better results than *Baseline+W* which is not trained for set retrieval.

Whitening improves the performance for all architectures for nDCG@10 when there are more than 2 elements per set, and for nDCG@30 for all number of elements per set, which is a somewhat surprising result since adding whitening only happens after the network is trained. However, using whitening is common in the retrieval com-

	nDCG@10				nDCG@30			
Scoring model	2/set	3/set	4/set	5/set	2/set	3/set	4/set	5/set
Baseline	66.3	38.8	23.7	17.5	55.5	35.5	23.0	17.6
Baseline-2	65.8	39.7	25.6	19.0	55.4	36.3	24.6	18.8
Baseline-3	65.9	39.4	24.8	18.4	55.3	35.9	23.9	18.3
SetNet-2	71.0	57.7	44.6	36.9	61.6	51.1	40.9	34.1
SetNet-3	71.9	57.9	44.7	37.0	61.5	51.2	41.0	34.2
Baseline + W	62.3	42.3	30.8	23.3	55.2	39.5	29.4	22.9
Baseline-2 + W	62.3	44.1	32.6	25.7	55.3	41.1	31.1	25.1
Baseline-3 + W	62.1	44.0	32.3	25.6	55.2	41.0	30.9	25.0
SetNet-2 + W	71.3	59.5	47.0	39.3	61.6	52.1	42.4	35.9
SetNet-3 + W	71.8	59.8	47.1	39.3	61.2	52.2	42.5	36.0
Desc-per-element	72.4	69.4	67.1	65.3	63.9	61.7	60.4	59.3

Table 6.1: **Table of nDCG@10 and nDCG@30 of stress test comparison of different models.** There are 100 query sets, each with two identities. Columns corresponds to the four different test datasets defined by the number of elements (faces) per set.

munity as it is usually found to be very helpful (Arandjelović and Zisserman (2013), Jégou and Chum (2012)), but has also been used recently to improve CNN representations (Radenović et al. (2016), Sun et al. (2017)). It is likely that whitening before aggregation is beneficial also because it makes descriptors more orthogonal to each other, which helps to reduce the amount of information lost by aggregation. However, *SetNet* gains much less from whitening, which may indicate that it learns to produce more orthogonal face descriptors. In the supplementary material we investigate the orthogonality of descriptors further by analysing the Gram matrix computed on the identities in the VGG Face Dataset. We observe that SetNet produces even more orthogonal descriptors than the whitened baselines.

It is also important to note that, as illustrated by Table 6.1, the cardinality of the sets used for training does not affect the performance much, regardless of the architecture. Therefore, training with a set size of 2 or 3 is sufficient to learn good set representations which generalize to larger sets.

	Celebrity Together	Distractors from MS1M	Total
No. images	194k	355k	549k
No. faces	546k	1M	1546k

Table 6.2: **Number of images and faces in the test dataset.** The test dataset consists of the *Celebrity Together* dataset and distractor images from the MS-Celeb-1M dataset (Guo et al. (2016)).

6.4.2 Evaluating on the Celebrity Together dataset

Here we evaluate the SetNet performance on the full *Celebrity Together* dataset.

Test dataset. The collection of the test dataset is described in Chapter 3. To increase the retrieval difficulty, random 355k distractor images are sampled from the MS-Celeb-1M Dataset (Guo et al. (2016)), as before taking care to include only true distractor people. The sampled distractor sets are constructed such that the number of faces per set follows the same distribution as in the *Celebrity Together* dataset. The statistics of the resulting test dataset are shown in Table 6.2. There are 1000 test queries, formed by randomly sampling 500 queries containing two celebrities and 500 queries containing three celebrities, under the restriction that the queried celebrities do appear together in some dataset images.

Experimental setup and baseline. In this test we consider two scenarios: first, where only one face example is available for each query identity, and second, where three face examples per query identity are available. In the second scenario, for each query identity, three extracted face descriptors are averaged and L2-normalized to form a single enhanced descriptor which is then used to query the dataset. In both scenarios the experiment is repeated 10 times using different face examples for each query identity, and the nDCG score is averaged. The best baseline from the stress test (Section 6.4.1) *Baseline-2+W*, is used as the main comparison method.

Scoring model	N_{ex}	N_{qd}	nDCG@10	nDCG@30
Baseline-2 + W	1	Q	50.0	49.4
SetNet-3 + W	1	Q	**59.1**	**59.4**
SetNet-3 + W w/ query avg.	1	1	58.7	**59.4**
Baseline-2 + W	3	Q	56.6	56.0
SetNet-3 + W	3	Q	**63.8**	**64.1**
SetNet-3 + W w/ query avg.	3	1	62.9	**64.1**

Table 6.3: **Set retrieval performance on the full *Celebrity Together* dataset with distractor images.** N_{ex} is the number of available face examples for each identity, while N_{qd} is the number of query descriptors used for querying. Q is the number of query identities.

Results. Table 6.3 shows that *SetNet-3+W* outperforms the best baseline for all performance measures by a large margin. Particularly impressive is the boost when only one face example is available for each query identity, where *Baseline-2+W* is beaten by 9.1% and 10.0% at nDCG@10 and nDCG@30 respectively. The results demonstrate that our trained aggregation method is indeed beneficial since it is designed and trained end-to-end exactly for the task in hand. The improvement is also significant for the second scenario where three face examples are available for each query identity, namely an improvement of 7.2% and 8.1% over the baseline. Figure 6.1 shows the top 3 retrieved images out of 549k images for two examples queries using SetNet (images are cropped for better viewing). The supplementary material contains many more examples.

Query averaging. We also investigate a more efficient method to query the database for multiple identities. Namely, we average the descriptors of all the query identities to produce a single descriptor which represents all query identities, and query with this single descriptor. In the second scenario, when three face examples are available for each query identity, all of the descriptors are simply averaged to a single descriptor. With this query representation we obtain a slightly lower nDCG@10 compared to the original method shown in Table 6.3 (62.9 vs 63.8), and the same nDCG@30 (64.1). However, as will be seen in the next section, this drop can be nullified by

re-ranking, making *query averaging* an attractive method due to its efficiency.

6.4.3 Efficient set retrieval

Our SetNet approach stores a single descriptor-per-set making it very fast though with potentially sacrificed accuracy. This section introduces alternatives and evaluates trade-offs between set retrieval quality and retrieval speed. To evaluate computational efficiency formally with the big-O notation, let Q, N_f and N be the number of query identities, average number of faces per dataset image, and the number of dataset images, respectively, and let the face descriptor be D_F-dimensional. Recall that our SetNet produces a compact set representation which is D-dimensional. In our case, $D = 128$ and $D_F = 128$ throughout. For convenience, we use D to represent both the dimensions of the element descriptors and the set-level descriptors in the following (since they are equal in our case).

Descriptor-per-set (SetNet). Storing a single descriptor per set is very computationally efficient as ranking only requires computing a scalar product between Q query D-dimensional descriptors and each of the N dataset descriptors, passing them through a logistic function, followed by scoring the images by the sum of similarity scores, making this step $O(NQD)$. For the more efficient *query averaging* where only one query descriptor is used to represent all the query identities, this step is even faster with $O(ND)$. Sorting the scores is $O(N \log N)$. Total memory requirements are $O(ND)$.

Descriptor-per-element. Set retrieval can also be performed by storing all element descriptors, requiring $O(NFD)$ memory. An image can be scored by obtaining all $Q \times F$ pairs of (query-identity, image-face) scores and finding the optimal assignment by considering it as a maximal weighted matching problem in a bipartite graph.

Instead of solving the problem using the Hungarian algorithm which has computational complexity that is cubic in the number of faces and is thus prohibitively slow, we use a greedy matching approach which is $O(QF \log(QF))$ per image. Therefore, the total computational complexity is $O(NQFD + NQF \log(QF) + N \log N)$. For our problem, we do not find any loss in retrieval performance compared to optimal matching, while being 7× faster.

Fast descriptor-per-element. Even the greedy method is too slow in practice, so we propose a faster more approximate version which relaxes the one-to-at-most-one matching of query identities with dataset face images. This can be achieved simply in $O(QF)$ time per image by computing a score for each query identity as the maximum of the query-face scores over all image faces, followed by summing of all query identity scores. The computational cost is $O(NQFD + NQF + N \log N)$.

Combinations by re-ranking. Borrowing ideas again from image retrieval Philbin et al. (2007), Sivic and Zisserman (2003), it is possible to combine the speed benefits of the faster methods with the accuracy of the slow descriptor-per-element method by using the former for initial ranking, and the latter to re-rank the top N_r results. The computational complexity is then equal to that of the fast method of choice, plus $O(N_r QFD + N_r QF \log(QF) + N_r \log N_r)$.

Experimental setup. The performance is evaluated on the 1000 test queries and on the same full dataset with distractors as in Section 6.4.2. N_r is varied in this experiment to demonstrate the trade-off between accuracy and retrieval speed. For the descriptor-per-element method, we use the Baseline + W features. We also discuss a naive approach of pre-tagging the dataset with a list of identities. The retrieval speed is measured as the mean over all 1000 test query sets. The test is implemented in Matlab, and the measurements are carried out on a Xeon E5-2667 v2/3.30GHz, with

Scoring method	N_r	Without query averaging nDCG@10	nDCG@30	Timing	Speedup
Desc-per-set	-	59.1	59.4	0.11s	57.5×
Desc-per-set + Re.	100	76.5	70.1	0.13s	48.8×
Desc-per-set + Re.	1000	84.2	80.0	0.20s	31.8×
Desc-per-set + Re.	2000	85.3	81.4	0.28s	22.7×
Fast Desc-per-element + Re.	2000	85.3	80.2	0.78s	8.1×
Desc-per-element	-	85.4	81.7	6.35s	1×

Table 6.4: **Retrieval speed *vs* quality trade-off with varied number of re-ranking images (without query averaging).** Retrieval performance, average time required to execute a set query and speedup over *Desc-per-element* are shown for each method. 'Re.' denotes re-ranking. The evaluation is on the 1000 test queries and on the same full dataset with distractors as in Section 6.4.2.

Scoring method	N_r	With query averaging nDCG@10	nDCG@30	Timing	Speedup
Desc-per-set	-	58.7	59.4	0.01s	635×
Desc-per-set + Re.	100	76.4	70.1	0.03s	212×
Desc-per-set + Re.	1000	84.1	80.0	0.10s	63.5×
Desc-per-set + Re.	2000	85.3	81.4	0.18s	35.3×
Desc-per-element	-	85.4	81.7	6.35s	1×

Table 6.5: **Retrieval speed *vs* quality trade-off with varied number of re-ranking images (with query averaging).** Retrieval performance, average time required to execute a set query and speedup over *Desc-per-element* are shown for each method. 'Re.' denotes re-ranking. The evaluation is on the 1000 test queries and on the same full dataset with distractors as in Section 6.4.2.

only a single thread.

Results. Table 6.4 and Table 6.5 shows set retrieval results for the various methods together with the time it takes to execute a set query. The full descriptor-per-element approach is the most accurate one, but also prohibitively slow for most uses, taking more than 6 seconds to execute a query. The descriptor-per-set (i.e. SetNet) approach with query averaging is blazingly fast with only 0.01s per query using one descriptor to represent all query identities, but sacrifices retrieval quality to achieve this speed. However, using SetNet for initial ranking followed by re-ranking achieves good results without a significant speed hit – the accuracy almost reaches that of the full slow descriptor-per-element, while being more than 35× faster. Moreover, its retrieval

performance is similar to the fast descriptor-per-element with re-ranking, and even better in nDCG@30, while achieving a large 4.3× speedup over it. Furthermore, by combining *desc-per-set* and *desc-per-element* it is possible to choose the trade-off between speed and retrieval quality, as appropriate for specific use cases. For a task where speed is crucial, *desc-per-set* can be used with few re-ranking images (*e.g.* 100) to obtain a 212× speedup over the most accurate method (*desc-per-element*). For an accuracy-critical task, it is possible to re-rank more images while maintaining a reasonable speed. Therefore, our method offers a trade-off between speed and retrieval quality, which is useful in practice.

Furthermore, its retrieval performance is similar to the fast descriptor-per-element with re-ranking, being worse in mAP and better in nDCG@30, while achieving a large 2.7× speedup over it.

Pre-tagging Baseline. Apart from descriptor-per-set and descriptor-per-element methods, to perform a well-rounded investigation of retrieval methods, we also consider another naive method – pre-tagging. This approach pre-tags all the dataset images offline with a closed-dictionary of known identities by deeming a person to appear in the image if the classification score is larger than a threshold. Set retrieval is then performed by ranking images based on the intersection between the query and image tags. Pre-tagging seems straightforward, however, it suffers from two large limitations. First, it is very constrained as it only supports querying for a fixed set of identities which has to be predetermined at the offline tagging stage. Second, it relies on making hard decisions when assigning an identity to a face, which is expected to perform badly, especially in terms of recall. In our test, by using descriptors of *Baseline + W*, pre-tagging achieves 65.7% and 68.6% for nDCG@10 and nDCG@30. It is significantly lower than the results of SetNet even with only 100 re-ranking images (*i.e.* 76.5% and 70.1% for nDCGs) in Table 6.4. Therefore, since our descriptor-per-set + re-ranking method achieves vastly superior performance as well as allowing

querying for novel identities, it is strongly preferred over pre-tagging.

6.4.4 Descriptor orthogonality

A whitening transformation decorrelates each dimension of the descriptors, and consequently descriptors will interfere less when added to each other. We expect SetNet to automatically learn such behaviour in order to minimize the interference between descriptors after aggregation. To verify this, we compute the gram matrix of 2622 identities in the VGG Face Dataset for both the baseline models and SetNet, and measure the deviation G_{dif} between the computed gram matrix and an identity matrix. G_{dif} is defined as equation (6.3).

$$G_{dif} = \|G - I\|_{Frob} \tag{6.3}$$

where G is the gram matrix and I is the identity matrix.

An identity matrix means all the considered identities are orthogonal to each other; therefore a smaller G_{dif} corresponds to more orthogonal descriptors. Note that whitening does not enforce orthogonality in this case, as the dimension of descriptors D_F (*i.e.* 128) is much smaller than the number of classes (*i.e.* 2622) considered. It only transforms the descriptors such that the covariance matrix of the input data becomes identity matrix. Whereas we compute gram matrix here instead of covariance matrix, and gram matrix measures how orthogonal the classes are to each other.

G_{dif} is computed in four steps: first, 100 faces are randomly sampled for each idientity in VGG Face Dataset, and a feature vector for each face sample is extracted from the network; second, a mean descriptor for each identity is obtained by averaging the 100 descriptors followed by L2-normalization; third, the gram matrix is computed by taking the scalar product between the mean descriptors of each possible pair of identities, *i.e.* gram matrix is of size 2622 $\times$ 2622. Finally, G_{dif} is computed as the

Model	Not whitened	Whitened
Baseline-2	416	297
SetNet-3	282	276

Table 6.6: **Difference between gram matrix and identity matrix G_{dif}.** The gram matrix is computed over 2622 identities in VGG Face Dataset.

Frobenius norm of the difference of the gram matrix and an identity matrix of the same size. We compute G_{dif} using non-whitened and whitened features of the best baseline network, *Baseline-2*, and SetNet. Notably, both networks are trained on VGGFace2 dataset (Cao et al. (2018)), and their corresponding whitening transformation are also computed on VGGFace2 dataset; whereas in this test, all the identities are from the VGGFace dataset (Parkhi et al. (2015)).

As Table 6.6 shows, whitening reduces G_{dif} for both *Baseline-2* and *SetNet-3*; especially for *Baseline-2*. More importantly, SetNet without whitening has even smaller G_{dif} than the whitened baseline. Therefore, from Table 6.6 we can see that SetNet learns to produce more discriminative descriptors, which is benificial for set retrieval with descriptor-per-set approach.

6.5 Retrieval Examples

To visualize the performance of the set retrieval system, the top ranked 5 images (ordered column-wise) for several queries are shown in Figure 6.7, 6.8, 6.9 (images are cropped for displaying purposes in order to see the faces better). Notably, the nDCG@5 (the quality of the top 5 retrievals) is equal to 1 for all examples, meaning that images are correctly ranked according to the size of their overlap with the query set.

6.6 Conclusion

We have considered a new problem of set retrieval, discussed multiple different approaches to tackle it, and evaluated them on a specific case of searching for sets of faces. Our learnt compact set representation, produced using the SetNet architecture and trained in a novel manner directly for the set retrieval task, beats all baselines convincingly. Furthermore, due to its high speed it can be used for fast set retrieval in large image datasets. The set retrieval problem has applications beyond multiple faces in an image. For example, a similar situation would also apply for the task of video retrieval when the elements themselves are images (frames), and the set is a video clip or shot.

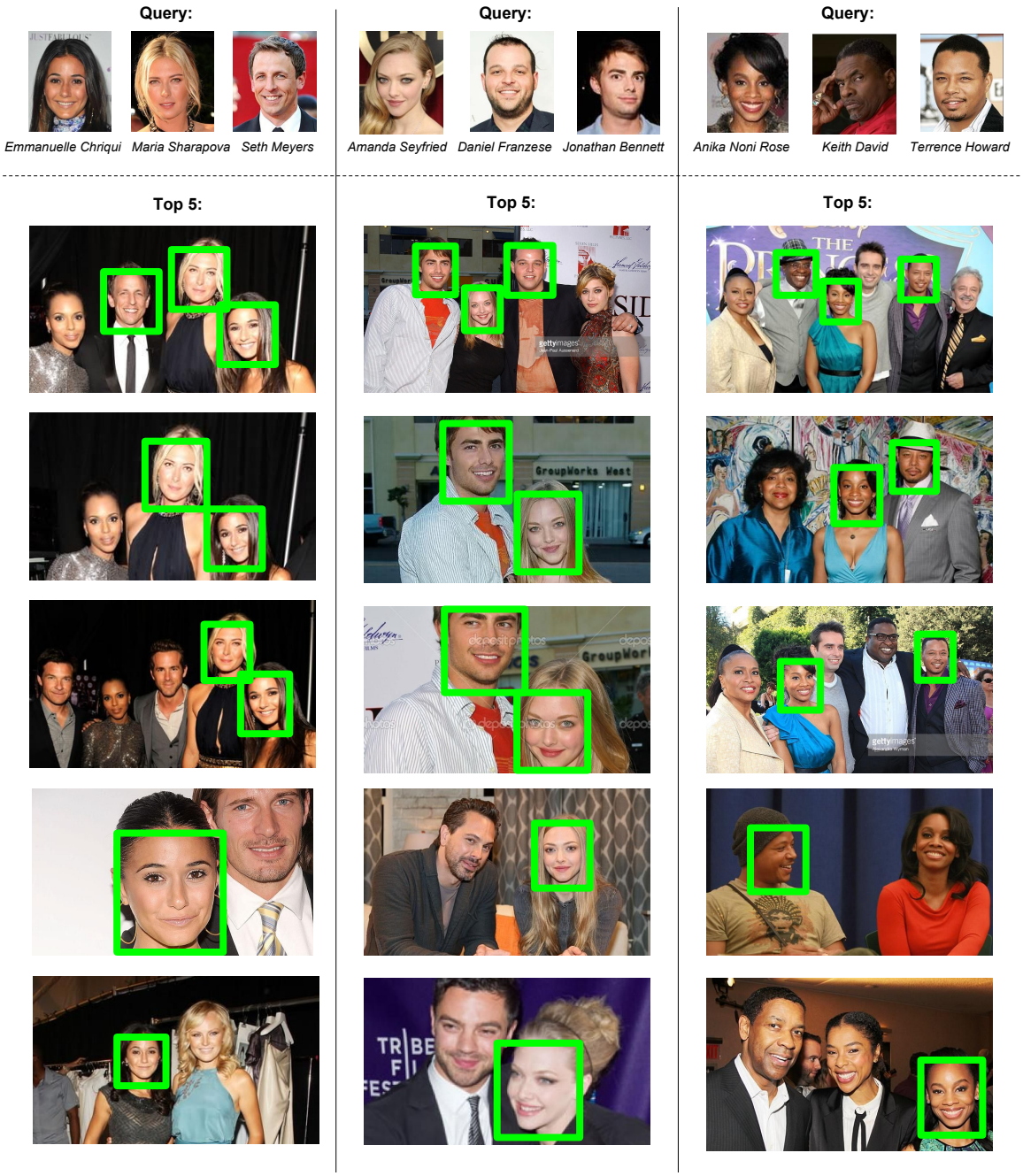


Figure 6.7: **Top 5 ranked images returned by the SetNet based descriptor-per-set retrieval.** The query identities are highlighted by green bounding boxes.

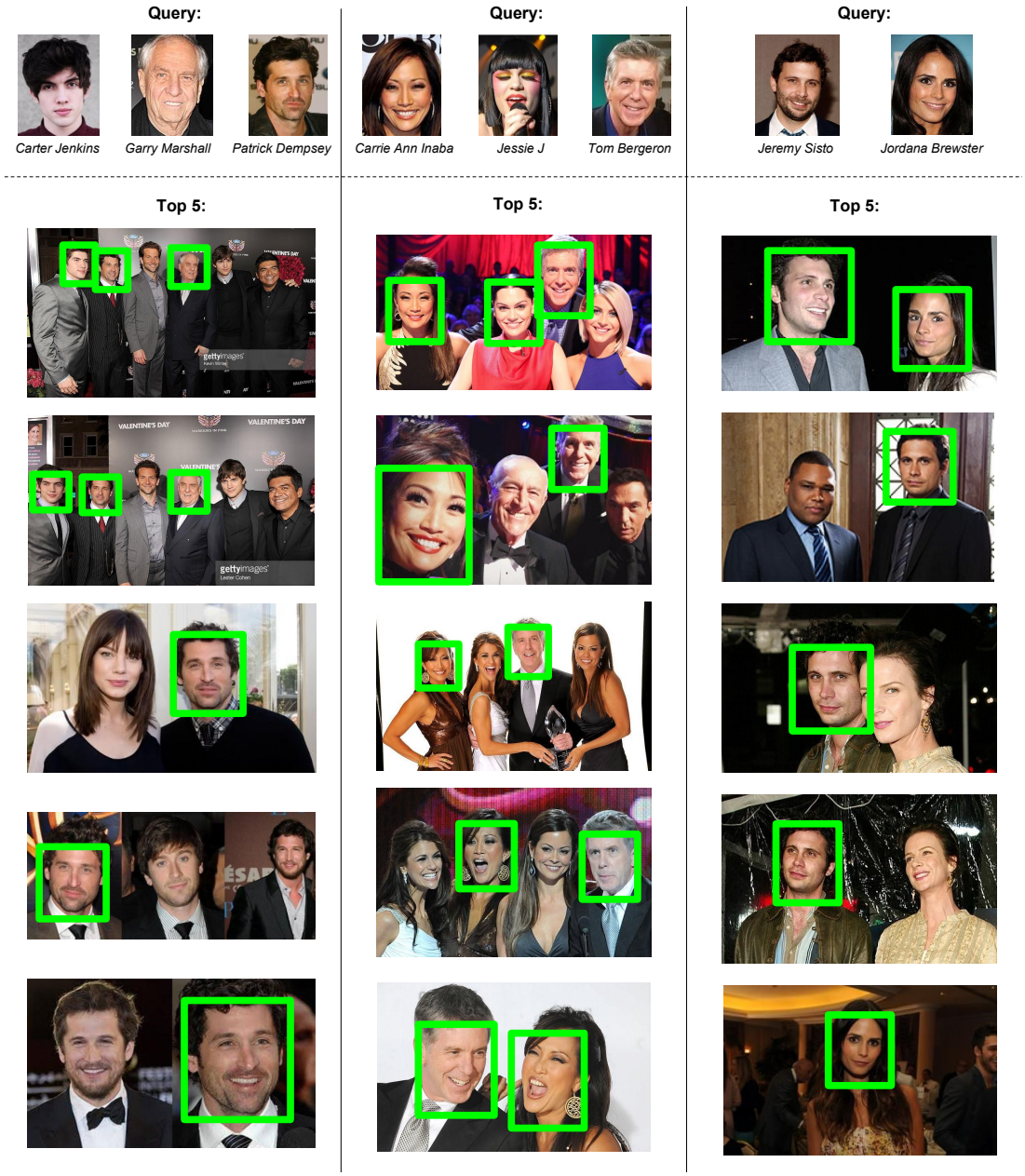


Figure 6.8: **Top 5 ranked images returned by the SetNet based descriptor-per-set retrieval.** The query identities are highlighted by green bounding boxes.

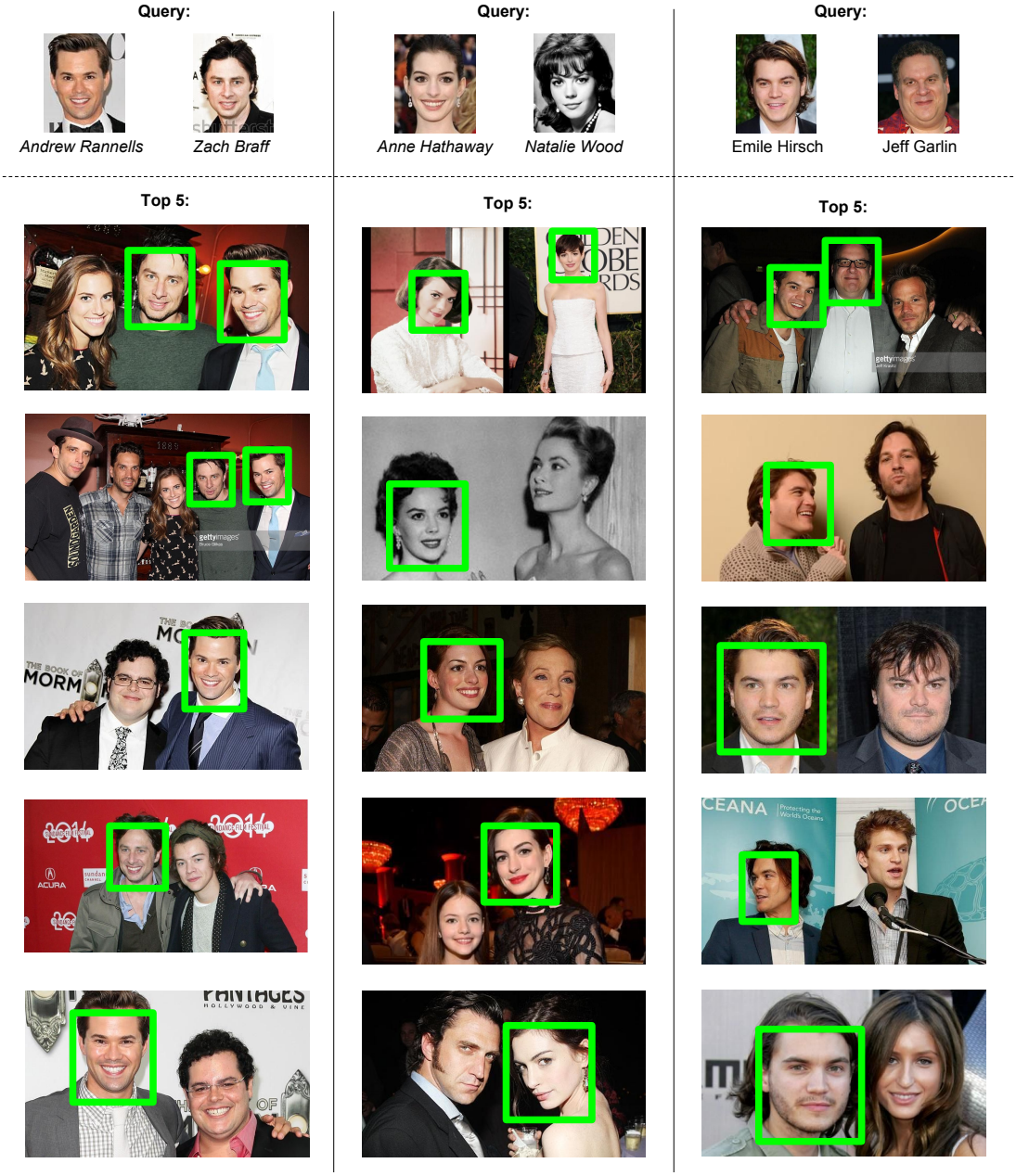


Figure 6.9: **Top 5 ranked images returned by the SetNet based descriptor-per-set retrieval.** The query identities are highlighted by green bounding boxes.

Chapter 7

GhostVLAD for set-based face recognition

The objective of this chapter is to learn a compact representation of image sets for template-based unconstrained face recognition. In the problem of template-based face recognition, a set of images ranging from one to several hundreds are available for each subject. A straightforward method to tackle multiple images per subject is to store per-image descriptors extracted from each face image (or frame in a video), and compare every pair of images between sets at query time (Schroff et al. (2015), Taigman et al. (2014)). However, this type of approach can be memory-consuming and prohibitively slow, especially for searching tasks in large-scale datasets. Therefore, an aggregation method that can produce a compact template representation is desired. Furthermore, this representation should support efficient computation of similarity and require minimal memory storage.

In this chapter, we again employ the NetVLAD layer as in Chapter 6 to learn a compact template representation, rather than simple average-pooling or max-pooling (Cao et al. (2018), Chen et al. (2015), Parkhi et al. (2015)). This is inspired by Jégou and Zisserman (2014), who reveal that traditional image retrieval encoding methods are able to increase the contrast between mated and unmated local descriptors, compared to the original descriptors. Therefore, we adopt a similar encoding approach in

the design of our network.

Another problem we confront for unconstrained face recognition is the large variation in the image quality. Images taken in a real-world situation can have low resolution, being in different lighting conditions or blurred. These low-quality images usually do not provide much information and can even distract the template representation. As a result, we introduce a novel layer to effectively down-weight the low-quality images and enable the template representation to focus on the informative images.

This chapter is organized as follow. In Section 7.1, we describe a network architecture which aggregates and embeds the face descriptors produced by deep convolutional neural networks into a compact fixed-length representation. This compact representation requires minimal memory storage and enables efficient similarity computation. Furthermore, we propose a novel GhostVLAD layer that includes *ghost clusters*, that do not contribute to the aggregation. We show that a quality weighting on the input faces emerges automatically such that informative images contribute more than those with low quality, and that the ghost clusters enhance the network's ability to deal with poor quality images. Section 7.2 describes how the networks are trained and provides implementation details. In Section 7.3, we explore how input feature dimension, number of clusters and different training techniques affect the recognition performance. Given this analysis, we train a network that far exceeds the state-of-the-art on the IJB-B face recognition dataset. This is currently one of the most challenging public benchmarks, and we surpass the state-of-the-art on both the identification and verification protocols. Lastly, a conclusion is drawn in Section 7.4.

7.1 Set-based face recognition

We aim to learn a compact representation of a person's face. Namely, we train a network which digests a set of example face images of a person, and produces a fixed-

length template representation useful for face recognition. The network should satisfy the following properties:

(1) Take any number of images as input, and output a fixed-length template descriptor to represent the input image set. (2) The output template descriptor should be compact (*i.e.* low-dimensional) in order to require little memory and facilitate fast template comparisons. (3) The output template descriptor should be discriminative, such that the similarity of templates of the same subject is much larger than that of different subjects.

We propose a convolutional neural network that fulfils all three objectives. (1) is achieved by aggregating face descriptors using a modified NetVLAD (Arandjelović et al. (2016)) layer, GhostVLAD. Compact template descriptors (2) are produced by a trained layer which performs dimensionality reduction. Discriminative representations (3) emerge because the entire network is trained end-to-end for face recognition, and since our GhostVLAD layer is able to down-weight the contribution of low-quality images, which is important for good performance (Goswami et al. (2014), Hassner et al. (2016), Yang et al. (2017)).

The network architecture and the new GhostVLAD layer are described in Section 7.1.1 and Section 7.1.2, respectively, followed by the network training procedure (Section 7.2.1) and implementation details (Section 7.2.2).

7.1.1 Network architecture

As shown in Figure 7.1, the network consists of two parts (which is similar to that introduced in Chapter 6): feature extraction, which computes a face descriptor for each input face image, and aggregation, which aggregates all face descriptors into a single compact template representation of the input image set. The two parts are explained in detail in the following.

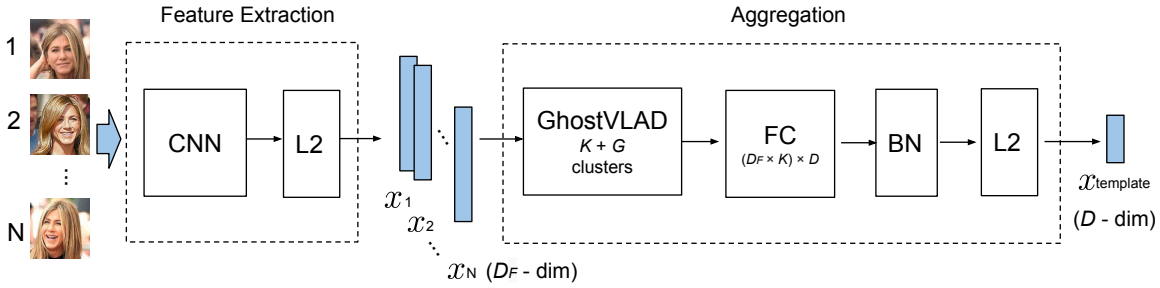


Figure 7.1: **Network architecture.** Input images in each template are first passed through a convolutional neural network (*e.g.* ResNet-50 or SENet-50 with an additional FC layer and L2-normalization) to produce a face descriptor per image. The descriptors are aggregated into a single fixed-length vector using the GhostVLAD layer. The final D-dimensional template descriptor is obtained by reducing dimensionality using a fully-connected layer, followed by batch normalization (BN) and L2-normalization.

Feature extraction. A neural network is used to extract a face descriptor for each input face image. Any network can be used in our learning framework, but in this chapter we opt for ResNet-50 (He et al. (2016)) or SENet-50 (Hu et al. (2018)). Both networks are cropped after the global average pooling layer, and an extra FC layer is added to reduce the output dimension to D_F. More details on the modification of the networks are referred to Section 6.1.1 in Chapter 6. We typically pick D_F to be low-dimensional (*e.g.* 128 or 256), and do not see a significant drop in face recognition performance compared to using the original 2048-D descriptors. Finally, the individual face descriptors are L2 normalized.

Aggregation. The second part uses GhostVLAD (Section 7.1.2) to aggregate multiple face descriptors into a single $D_F \times K$ vector (where K is a parameter of the method). To keep computational and memory requirements low, dimensionality reduction is performed via an FC layer, where we pick the output dimensionality D to be 128. The compact D-dimensional descriptor is then passed to a batch-normalization layer (Ioffe and Szegedy (2015)) and L2-normalized to form the final template representation $x_{template}$.

7.1.2 GhostVLAD: NetVLAD with ghost clusters

The key component of the aggregation block is our *GhostVLAD* trainable aggregation layer, which given N D_F-dimensional face descriptors computes a single $D_F \times K$ dimensional output. It is based on the NetVLAD (Arandjelović et al. (2016)) layer which implements an encoding similar to VLAD encoding (Jégou et al. (2010)), while being differentiable and thus fully-trainable. NetVLAD has been shown to outperform average and max pooling for the same vector dimensionality, which makes it perfectly suited for our task. Here we provide a brief overview of NetVLAD (for full details please refer to (Arandjelović et al. (2016))), followed by our improvement, GhostVLAD.

NetVLAD. Although explained in Chapter 6, we have a brief review here on the NetVLAD layer. For N D_F-dimensional input descriptors $\{x_i\}$ and a chosen number of clusters K, NetVLAD pooling produces a single $D_F \times K$ vector V (for convenience written as a $D_F \times K$ matrix) according to the following equation:

$$V(j,k) = \sum_{i=1}^{N} \frac{e^{a_k^T x_i + b_k}}{\sum_{k'=1}^{K} e^{a_{k'}^T x_i + b_{k'}}} (x_i(j) - c_k(j)) \tag{7.1}$$

where $\{a_k\}$, $\{b_k\}$ and $\{c_k\}$ are trainable parameters, with $k \in [1, 2, \ldots, K]$. The first term corresponds to the soft-assignment weight of the input vector x_i for cluster k, while the second term computes the residual between the vector and the cluster centre. The final output is obtained by performing L2 normalization.

GhostVLAD. We extend NetVLAD with "ghost" clusters to form *GhostVLAD*, as shown in Figure 7.2. Namely, we add further G "ghost" clusters which contribute to the soft assignments in the same manner as the original K clusters, but residuals between input vectors and the ghost cluster centres are ignored and do not contribute to the final output. In other words, the summation in the denominator of eq. (7.1) in-

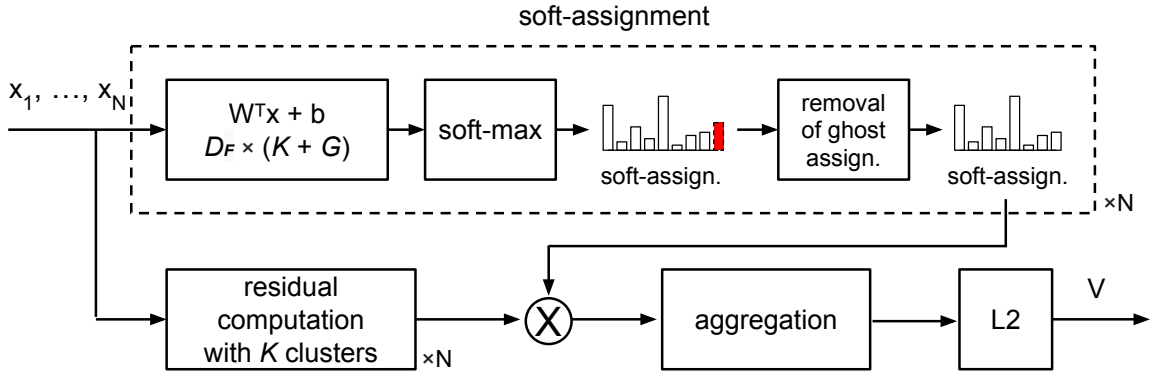


Figure 7.2: **GhostVLAD.** For each input descriptor, NetVLAD performs soft-assignment into K cluster centres, computed as a linear transformation followed by a soft-max. It then, for each cluster centre, aggregates all residuals between input descriptors and the cluster centre, weighted with the soft-assignment values. The final vector is produced as a concatenation of the per-cluster aggregated residuals; for more details see eq. (7.1) and (Arandjelović et al. (2016)). We introduce G "ghost" clusters in the soft-assignment stage, where the "ghost" assignment weight is illustrated with a dotted red bar (here we show only $G = 1$ ghost cluster). The ghost assignments are then eliminated and residual aggregation proceeds as with NetVLAD. This mechanism enables the network to assign uninformative descriptors to ghost clusters thus decreasing their soft-assignment weights for non-ghost clusters, and therefore reducing their contribution to the final template representation.

stead of to K goes to $K+G$, while the output is still $D_F \times K$ dimensional; this means $\{a_k\}$ and $\{b_k\}$ have $K+G$ elements each, while $\{c_k\}$ still has K. Another view is that we are computing NetVLAD with $K+G$ clusters, followed by removing the elements that correspond to the G ghost clusters. Note that GhostVLAD is a generalization of NetVLAD as with $G = 0$ the two are equivalent. As with NetVLAD, GhostVLAD can be implemented efficiently using standard convolutional neural network building blocks, *e.g.* the soft-assignment can be done by stacking input descriptors and applying a convolution operation, followed by a convolutional soft-max; for details see (Arandjelović et al. (2016)).

The intuition behind the incorporation of ghost clusters is to make it easier for the network to adjust the contribution of each face example to the template representation by assigning examples to be ignored to the ghost clusters. For example, in an ideal case, a highly blurry face image would be strongly assigned to a ghost clus-

ter, making the assignment weights to non-ghost clusters close to zero, thus causing its contribution to the template representation to be negligible; in Section 7.3.6 we qualitatively validate this intuition. However, note that we do not explicitly force low-quality images to get assigned to ghost clusters, but instead let the network discover the optimal behaviour through end-to-end training for face recognition.

7.2 Network training and implementation details

In this section we describe how to train the network for face recognition (but note that GhostVLAD is a general layer which can also be used for other tasks), followed by implementation details.

7.2.1 Network training

Training loss. Just for training purposes, we append the network with a fully-connected "classification" layer of size $D \times T$, where D is the size of the template representation and T is the number of identities available in the training set. We use the one-versus-all logistic regression loss as empirically we found that it converges faster and outperforms cross-entropy loss. The classification layer is discarded after training and the trained network is used to extract a single fixed-length template representation for the input face images.

Training with degraded images. For unconstrained face recognition, it is important to be able to handle images of varying quality that typically occur in the wild. The motivation behind our network architecture, namely the GhostVLAD layer, is to enable it to down-weight the influence of these images on the template representation. However, since our training dataset only contains good quality images, it is necessary to perform data augmentation in the form of image degradation, such as blurring or

compression (see Section 7.2.2 for details), in order to more closely match the varying image quality encountered at test time.

7.2.2 Implementation details

This section discusses full details of the training process, including training data, data augmentation, network initialization, *etc.*

Training data. We use face images from the training set of the VGGFace2 dataset (Cao et al. (2018)) to train the network. It consists of around 3 million images, covering 8631 identities. For each identity, there are on average 360 face images across different ages and poses. To perform set-based training, we form image sets on-the-fly by repeatedly sampling a fixed number of images belonging to the same identity.

Data augmentation. Training images are resized such that the smallest dimension is 256 and random crops of size 224×224 are used as inputs to the network. Further augmentations include random horizontal flipping and a random rotation of no greater than 10 degrees.

We adopt four methods to degrade images for training: isotropic blur, motion blur, decreased resolution and JPEG compression. Each degradation method has a probability of 0.1 to be applied to a training image, where, to prevent over-degradation, a maximum of two transformations per image is allowed. Isotropic blur is implemented using a Gaussian filter where the standard deviation is uniformly sampled between 6 and 16. For motion blur, the angle of motion is uniformly sampled between 0 and 359 degrees, and the motion length is fixed to 11. Resolution decrease is simulated by downscaling the image by a factor of 10 and scaling it back up to the original size. Finally, we add JPEG compression artefacts by randomly compressing the images to

one of three compression ratios: 0.01, 0.05 and 0.09.

Training procedure. The network can be trained end-to-end in one go, but, to make the training faster and more stable, we divide it into three stages; all stages only use the VGGFace2 (Cao et al. (2018)) dataset. In the first two stages, parts of the network are trained for single-image face classification (*i.e.* the input image set consists of a single image), and image degradation is not performed. Firstly, the feature extractor network is pre-trained for single-image face classification by temporarily (just for this stage) appending a classification FC layer on top of it, and training the network with the cross-entropy loss. Secondly, we train the whole network end-to-end for single-image classification with one-versus-all logistic regression loss, but exclude the ghost clusters from GhostVLAD because training images are not degraded in this stage. Finally, we add ghost clusters and enable image degradation, and train the whole network using image sets with one-versus-all logistic regression loss.

Parameter initialization. The non-ghost clusters of GhostVLAD are initialized as in NetVLAD (Arandjelović et al. (2016)) by clustering its input features with k-means into K clusters, where only non-degraded images are used. The G ghost clusters are initialized similarly, but using degraded images for the clustering; note that for $G = 1$ (a setting we often use) k-means simplifies to computing the mean over the features. The FC following GhostVLAD which performs dimensionality reduction is then initialized using the PCA transformation matrix computed on the GhostVLAD output features.

Training details. The network is trained using stochastic gradient descend with momentum, implemented in MatConvNet (Vedaldi and Lenc (2015)). The mini-batch consists of 84 face images, *i.e.* if we train with image sets of size two, a batch contains

42 image sets, one per identity. The initial learning rate of 0.0001 is used for all parameters apart from GhostVLAD's assignment parameters and the classification FC weights, for which we use 0.1 and 1, respectively. The learning rates are divided by 10 when validation error stagnates, while weight decay and momentum are fixed to 0.0005 and 0.9, respectively.

7.3 Experiments

In this section, we describe the experimental setup, investigate the impact of our design choices, and compare results with the state-of-the-art.

7.3.1 Benchmark datasets and evaluation protocol

Standard and challenging public face recognition datasets IJB-A (Klare et al. (2015)) and IJB-B (Whitelam et al. (2017)) are used for evaluation. In contrast to single-image based face datasets such as Cao et al. (2018), Guo et al. (2016), Kemelmacher et al. (2016), Parkhi et al. (2015), IJB-A and IJB-B are intended for template-based face recognition, which is exactly the task we consider in this work. The IJB-A dataset contains 5,712 images and 2,085 videos, covering 500 subjects; thus the average number of images and videos per subject are 11.4 and 4.2 videos, respectively. The IJB-B dataset is an extension of IJB-A with a total of 11,754 images and 7,011 videos from 1,845 subjects, as well as 10,044 non-face images. There is no overlap between subjects in VGGFace2, which we use for training, and the test datasets. Faces are detected from images and all video frames using MTCNN (Zhang et al. (2016)), the face crops are then resized such that the smallest dimension is 224 and the central 224×224 crop is used as the face example.

Evaluation protocol. We follow the standard benchmark procedure for IJB-A and IJB-B, and evaluate on "1:1 face verification" and "1:N face identification".

The goal of *1:1 verification* is to make a decision whether two templates belong to the same person, done by thresholding the similarity between the templates. Verification performance is assessed via the receiver operating characteristic (ROC) curve, *i.e.* by measuring the trade-off between the true accept rates (TAR) *vs* false accept rates (FAR).

For *1:N identification*, templates from the probe set are used to rank all templates in a given gallery. The performance is measured using the true positive identification rate (TPIR) *vs* false positive identification rate (FPIR) (*i.e.* the decision error trade-off (DET) curve) and *vs* Rank-N (*i.e.* the cumulative match characteristic (CMC) curve).

Evaluation protocols are the same for both benchmark datasets, apart from the fact that IJB-A defines 10 test splits, while IJB-B only has one split for verification and two galleries for identification. For IJB-A and for IJB-B identification, we report, as per standard, the mean and standard deviation of the performance measures.

7.3.2 Networks, deployment and baselines

Our networks. As explained earlier in Section 7.1.1, we use two different architectures as backbone feature extractors: ResNet-50 (He et al. (2016)) and SENet-50 (Hu et al. (2018)). They are cropped after global average-pooling which produces a $D_F = 2048$ dimensional face descriptor, while we also experiment with reducing the dimensionality via an additional FC, down to $D_F = 256$ or $D_F = 128$.

To disambiguate various network configurations, we name the networks as *Ext*-GV-S(-gG), where *Ext* is the feature extractor network (*Res* for ResNet-50 or *SE* for SENet-50), S is the size of image sets used during training, and G is the number of

ghost clusters (if zero, the suffix is dropped). For example, *SE-GV-3-g2* denotes a network which uses the SENet-50 as the feature extractor, training image sets of size 3, and 2 ghost clusters.

Network deployment. In the IJB-A and IJB-B datasets, there are images and videos available for each subject. Here we follow the established approach of (Cao et al. (2018), Crosswhite et al. (2017)) to balance the contributions of face examples from different sources, as otherwise a single very long video could completely dominate the representation. In more detail, face examples are extracted from all video frames, and their additive contributions to the GhostVLAD representation are down-weighed by the number of frames in the video.

The similarity between two templates is measured as the scalar product between the template representations; recall that they have unit norm (Figure 7.1).

Baselines. Our network is compared with several average-pooling baselines. The baseline architecture consists of a feature extractor network which produces a face descriptor for each input example, and the template representation is performed by average-pooling the face descriptors (with source balancing), followed by L2 normalization. The same feature extractor networks are used as for our method, ResNet-50 or SENet-50, abbreviated as *Res* and *SE*, respectively, with an optionally added FC layer to perform dimensionality reduction down to 128-D or 256-D. These networks are trained for single-image face classification, which is equivalent to stage 1 of our training procedure from Section 7.2.2, and also corresponds to the current state-of-the-art approach (Cao et al. (2018)) (albeit with more training data – see Section 7.3.4 for details and comparisons). No image degradation is performed as it decreases performance when combined with single-image classification training.

In addition, we train the baseline architecture SENet-50 with average-pooling using

our training procedure (Section 7.2.1), *i.e.* with image sets of size 2 and degraded images, and refer to it as *SE-2*.

7.3.3 Ablation studies on IJB-B

Here we evaluate various design choices of our architecture and compare it to baselines on the IJB-B dataset, as it is larger and more challenging than IJB-A; results on verification and identification are shown in Tables 7.1 and Table 7.2, respectively.

Feature extractor and dimensionality reduction. Comparing rows 1 *vs* 2 of the two tables shows that reducing the dimensionality of the face features from 2048-D to 128-D does not affect the performance much, and in fact sometimes improves it due to added parameters in the form of the dimensionality reduction FC. As the feature extractor backbone, SENet-50 consistently beats ResNet-50, as summarized in rows 2 *vs* 3.

Training for set-based face recognition. The currently adopted set-based face recognition approach of training with single-image examples and performing aggregation post hoc (*SE*, row 4) is clearly inferior to our training procedure which is aware of image sets (*SE-2*, row 5).

Learnt GhostVLAD aggregation. Using the GhostVLAD aggregation layer (with $G = 0$ *i.e.* equivalent to NetVLAD) together with our set-based training framework strongly outperforms the standard average-pooling approach, regardless of whether training is done with non-degraded images (*SE-GV-2*, row 8 *vs SE*, rows 3 and 4), degraded images (*SE-GV-2*, row 9 *vs SE-2*, row 5), or if a different feature extractor architecture (ResNet-50) is used (*Res-GV-2*, row 6 *vs Res*, row 2). Using 256-D *vs* 128-D face descriptors as inputs to GhostVLAD, while keeping the same dimension-

Row id	Network	D_F	D	K	G	No. faces	Deg.	1:1 Verification TAR (FAR=)			
								$1E-5$	$1E-4$	$1E-3$	$1E-2$
1	Res (Cao et al. (2018))	2048	2048	-	-	1	✗	0.647	0.784	0.878	0.938
2	Res	128	128	-	-	1	✗	0.646	0.785	0.890	0.954
3	SE	128	128	-	-	1	✗	0.670	0.803	0.896	0.954
4	SE	256	256	-	-	1	✗	0.677	0.807	0.892	0.955
5	SE-2	256	256	-	-	2	✓	0.679	0.810	0.902	0.958
6	Res-GV-2	128	128	8	0	2	✓	0.715	0.835	0.916	0.963
7	SE-GV-2	128	128	8	0	2	✓	0.721	0.835	0.916	0.963
8	SE-GV-2	256	128	8	0	2	✗	0.685	0.823	0.925	0.963
9	SE-GV-2	256	128	8	0	2	✓	0.738	0.850	0.923	0.964
10	SE-GV-2	256	128	4	0	2	✓	0.729	0.841	0.914	0.957
11	SE-GV-2	256	128	16	0	2	✓	0.722	0.848	0.921	0.964
12	SE-GV-3	256	128	8	0	3	✓	0.741	0.853	0.925	0.963
13	SE-GV-4	256	128	8	0	4	✓	0.747	0.852	0.922	0.961
14	SE-GV-3-g1	256	128	8	1	3	✓	0.753	0.861	**0.926**	0.963
15	SE-GV-4-g1	256	128	8	1	4	✓	**0.762**	**0.863**	**0.926**	0.963
16	SE-GV-3-g2	256	128	8	2	3	✓	0.754	0.861	0.926	**0.964**

Table 7.1: **Verification performance on the IJB-B dataset.** A higher value of TAR is better. D_F is the face descriptor dimension before aggregation. D is the dimensionality of the final template representation. K and G are the number of non-ghost and ghost clusters in GhostVLAD, respectively. 'No. faces' is the number of faces per set used during training. 'Deg.' indicates whether the training images are degraded. All training is done using the VGGFace2 dataset.

Row id	Network	D_F	D	K	G	No. faces	Deg.	1:N Identification TPIR				
								FPIR= 0.01	FPIR= 0.1	Rank-1	Rank-5	Rank-10
1	Res (Cao et al. (2018))	2048	2048	-	-	1	✗	0.701	0.824	0.886	0.936	0.953
2	Res	128	128	-	-	1	✗	0.688	0.833	0.901	0.950	0.963
3	SE	128	128	-	-	1	✗	0.712	0.849	0.908	0.949	0.963
4	SE	256	256	-	-	1	✗	0.718	0.854	0.908	0.948	0.962
5	SE-2	256	256	-	-	2	✓	0.717	0.857	0.909	0.949	0.962
6	Res-GV-2	128	128	8	0	2	✓	0.762	0.872	0.917	0.953	0.964
7	SE-GV-2	128	128	8	0	2	✓	0.753	0.880	0.917	0.953	0.964
8	SE-GV-2	256	128	8	0	2	✗	0.751	0.884	0.912	0.952	0.962
9	SE-GV-2	256	128	8	0	2	✓	0.760	0.879	0.918	0.955	0.964
10	SE-GV-2	256	128	4	0	2	✓	0.749	0.868	0.914	0.953	0.963
11	SE-GV-2	256	128	16	0	2	✓	0.759	0.879	0.918	0.954	**0.965**
12	SE-GV-3	256	128	8	0	3	✓	0.764	0.885	0.921	0.955	0.962
13	SE-GV-4	256	128	8	0	4	✓	0.752	0.878	0.914	0.952	0.960
14	SE-GV-3-g1	256	128	8	1	3	✓	0.770	**0.888**	**0.923**	**0.956**	**0.965**
15	SE-GV-4-g1	256	128	8	1	4	✓	**0.776**	**0.888**	0.921	**0.956**	0.964
16	SE-GV-3-g2	256	128	8	2	3	✓	0.772	0.886	0.922	0.957	0.964

Table 7.2: **Identification performance on the IJB-B dataset.** A higher value of TPIR is better. See caption of Table 7.4 for the explanations of column titles. Note, for readability standard deviations are not included here, but are included in Table 7.5 and 7.6 .

ality of the final template representation (128-D), achieves better results (rows 9 *vs* 7), so we use 256-D in all latter experiments.

Training with degraded images. When using our set-based training procedure, training with degraded images brings a consistent boost, as shown in rows 9 *vs* 8, since it better matches the test-time scenario which contains images of varying quality.

Number of clusters K. GhostVLAD (and NetVLAD) have a hyperparameter K – the number of non-ghost clusters – which we vary between 4 and 16 (rows 9 to 11) to study its effect on face recognition performance. It is expected that K shouldn't be too small so that underfitting is avoided (*e.g.* $K = 1$ is similar to average-pooling) nor too large in order to prevent over-quantization and overfitting. As in traditional image retrieval (Jégou et al. (2010)), we find that a wide range of K achieves good performance, with $K = 8$ being the best.

Ghost clusters. Introducing a single ghost cluster ($G = 1$) brings significant improvement over the vanilla NetVLAD, as shown by comparing *SE-GV-3-g1 vs SE-GV-3* (rows 14 *vs* 12) and *SE-GV-4-g1 vs SE-GV-4* (rows 15 *vs* 13). Using one ghost cluster is sufficient as increasing the number of ghost clusters to two does not result in significant differences (row 16 *vs* row 14). Ghost clusters enable the system to automatically down-weight the contribution of low quality images, as will be shown in Section 7.3.6, which improves the template representations and benefits face recognition.

Set size used during training. To perform set-based training, as described in Section 7.2.2, image sets are created by sampling a fixed number of faces for a subject; the number of sampled faces is another parameter of the method. Increasing the set size from 2 to 3 consistently improves results (rows 9 *vs* 12), while there is no clear

Network	Training dataset	D	1:1 Verification TAR		
			FAR=$1E-3$	FAR=$1E-2$	FAR=$1E-1$
Bin (Hassner et al. (2016))	ImNet+CAS	4096	0.631	0.819	-
NAN (Yang et al. (2017))	priv1	128	0.881 ± 0.011	0.941 ± 0.008	0.978 ± 0.003
QAN (Liu et al. (2017c))	VF+priv2	-	0.893 ± 0.039	0.942 ± 0.015	0.980 ± 0.006
SE (Cao et al. (2018))	VF2	2048	0.904 ± 0.020	0.958 ± 0.004	0.985 ± 0.002
SE (Cao et al. (2018))	MS+VF2	2048	0.921 ± 0.014	0.968 ± 0.006	0.990 ± 0.002
SE-GV-3	VF2	128	0.935 ± 0.016	$\mathbf{0.972 \pm 0.005}$	0.988 ± 0.002
SE-GV-4-g1	VF2	128	$\mathbf{0.936 \pm 0.015}$	$\mathbf{0.972 \pm 0.003}$	$\mathbf{0.990 \pm 0.002}$

Table 7.3: **Comparison with state-of-the-art for *verification* on the IJB-A dataset.** A higher value of TAR is better. D is the dimension of the template representation. 'VF', 'VF2', 'MS', 'ImNet+CAS', 'priv1' and 'priv2' refer to the VGGFace (Parkhi et al. (2015)), VGGFace2 (Cao et al. (2018)), MS-Celeb-1M (Guo et al. (2016)), ImageNet (Russakovsky et al. (2015)) and CASIA WebFace (Yi et al. (2014)), and private datasets used by (Yang et al. (2017)) and (Liu et al. (2017c)), respectively. Our best network, SE-GV-4-g1, sets the state-of-the-art by a significant margin on both datasets.

Network	Training dataset	D	1:1 Verification TAR			
			FAR=$1E-5$	FAR=$1E-4$	FAR=$1E-3$	FAR=$1E-2$
SE (Cao et al. (2018))	VF2	2048	0.671	0.800	0.888	0.949
SE (Cao et al. (2018))	MS+VF2	2048	0.705	0.831	0.908	0.956
SE-GV-3	VF2	128	0.741	0.853	0.925	0.963
SE-GV-4-g1	VF2	128	**0.762**	**0.863**	**0.926**	**0.963**

Table 7.4: **Comparison with state-of-the-art for *verification* on the IJB-B dataset.** A higher value of TAR is better. D is the dimension of the template representation. 'VF', 'VF2' and 'MS' refer to the VGGFace (Parkhi et al. (2015)), VGGFace2 (Cao et al. (2018)) and MS-Celeb-1M (Guo et al. (2016)) respectively. Our best network, SE-GV-4-g1, sets the state-of-the-art by a significant margin on both datasets.

winner between using 3 or 4 face examples (worse for $G = 0$, rows 12 *vs* 13, better for $G = 1$, rows 15 *vs* 14).

7.3.4 Comparison with state-of-the-art

In this section, our best networks, *SE-GV-3* and *SE-GV-4-g1*, are compared against the state-of-the-art on the IJB-A and IJB-B datasets. The currently best performing method (Cao et al. (2018)) is the same as our *SE* baseline (*i.e.* average-pooling of SENet-50 features trained for single-image classification) but trained on a much larger training set, MS-Celeb-1M dataset (Guo et al. (2016)), and then fine-tuned on

Network	Training dataset	D	1:N Identification TPIR	
			FPIR=0.01	FPIR=0.1
IJB-A				
Bin (Hassner et al. (2016))	ImNet+CAS	4096	0.875	-
NAN (Yang et al. (2017))	priv1	128	0.817 ± 0.041	0.917 ± 0.009
SE (Cao et al. (2018))	VF2	2048	0.847 ± 0.051	0.930 ± 0.007
SE (Cao et al. (2018))	MS+VF2	2048	0.883 ± 0.038	0.946 ± 0.004
SE-GV-3	VF2	128	0.872 ± 0.066	0.951 ± 0.007
SE-GV-4-g1	VF2	128	$\mathbf{0.884 \pm 0.059}$	$\mathbf{0.951 \pm 0.005}$
IJB-B				
SE (Cao et al. (2018))	VF2	2048	0.706 ± 0.047	0.839 ± 0.035
SE (Cao et al. (2018))	MS+VF2	2048	0.743 ± 0.037	0.863 ± 0.032
SE-GV-3	VF2	128	0.764 ± 0.041	0.885 ± 0.032
SE-GV-4-g1	VF2	128	$\mathbf{0.776 \pm 0.030}$	$\mathbf{0.888 \pm 0.029}$

Table 7.5: **Comparison with state-of-the-art for *identification* (TPIR *vs* FPIR) on the IJB-A and IJB-B datasets.** A higher value of TPIR is better. See caption of Table 7.3 for explanations of abbreviations. Our best network, SE-GV-4-g1, sets the state-of-the-art by a significant margin on both datasets.

VGGFace2.

From Tables 7.3 and 7.4, and 7.5, it is clear our GhostVLAD network (*SE-GV-4-g1*) convincingly outperforms previous methods and sets the new state-of-the-art for both identification and verification on both IJB-A and IJB-B datasets. The only state-of-the-art that our netowkr doesn't beat (but is on par with) is in TPIR at Rank-1 to Rank-10 for identification on IJB-A (shown in Table 7.6, but this is because IJB-A is not challenging enough and the TPIR values have saturated to a 99% mark. For the same measures on the more challenging IJB-B benchmark, our network is consistently the best. Furthermore, our networks produce much smaller template descriptors than the previous state-of-the-art networks (128-D *vs* 2048-D), making them more useful in real-world applications due to smaller memory requirements and faster template comparisons.

The results are especially impressive as we only train using VGGFace2 (Cao et al. (2018)) and beat methods which train with much more data, such as (Cao et al. (2018)) which combine VGGFace2 and MS-Celeb-1M (Guo et al. (2016)), *e.g.* TAR at FAR=$1E-5$ of 0.762 *vs* 0.705 for verification on IJB-B, and TPIR at FPIR=0.01

Network	Training dataset	D	1:N Identification TPIR		
			Rank-1	Rank-5	Rank-10
IJB-A					
Bin (Hassner et al. (2016))	ImNet+CAS	4096	0.846	0.933	0.951
NAN (Yang et al. (2017))	priv1	128	0.958 ± 0.005	0.980 ± 0.005	0.986 ± 0.003
SE (Cao et al. (2018))	VF2	2048	0.981 ± 0.003	$\mathbf{0.994 \pm 0.002}$	$\mathbf{0.996 \pm 0.001}$
SE (Cao et al. (2018))	MS+VF2	2048	$\mathbf{0.982 \pm 0.004}$	0.993 ± 0.002	0.994 ± 0.001
SE-GV-3	VF2	128	0.979 ± 0.005	0.990 ± 0.003	0.992 ± 0.003
SE-GV-4-g1	VF2	128	0.977 ± 0.004	0.991 ± 0.003	0.994 ± 0.002
IJB-B					
SE (Cao et al. (2018))	VF2	2048	0.901 ± 0.030	0.945 ± 0.016	0.958 ± 0.010
SE (Cao et al. (2018))	MS+VF2	2048	0.902 ± 0.036	0.946 ± 0.022	0.959 ± 0.015
SE-GV-3	VF2	128	0.921 ± 0.023	0.955 ± 0.013	0.962 ± 0.010
SE-GV-4-g1	VF2	128	$\mathbf{0.921 \pm 0.020}$	$\mathbf{0.956 \pm 0.013}$	$\mathbf{0.964 \pm 0.010}$

Table 7.6: **Comparison with state-of-the-art for *identification* (TPIR *vs* Rank) on the IJB-A and IJB-B datasets.** A higher value of TPIR is better. See caption of Table 7.3 for explanations of abbreviations. Our best network, SE-GV-4-g1, sets the state-of-the-art by a significant margin on both datasets.

of 0.776 *vs* 0.743 for identification on IJB-B. When considering only methods trained on the same data (VGGFace2), our improvement over the state-of-the-art is even larger: TAR at FAR=$1E-5$ of 0.762 *vs* 0.671 for verification on IJB-B, and TPIR at FPIR=0.01 of 0.776 *vs* 0.706 for verification on IJB-B.

The superiority of our GhostVLAD network (*SE-GV-4-g1*) over the state-of-the-art network (*SE-2048D (MS+VF2)*) is even clearer when we look at Figure 7.3. To visualize the improvement of our network in a more fair setting, we include the *SE-256D (VF2)* which is also trained on VGGFace2 dataset (Cao et al. (2018)) only, with compact output descriptor size (256-D) which is the same as the input descriptor size to the aggregation block in our network. As shown in Figure 7.3, the improvement of our network is even larger compared with *SE-256D (VF2)*.

7.3.5 Descriptor separation

To visualize the effect of image retrieval methods used in our task (*i.e.* NetVLAD), we plot the similarity scores (computed as the scalar product between two template descriptors) of all the template pairs in the verification task on the IJB-B dataset, as

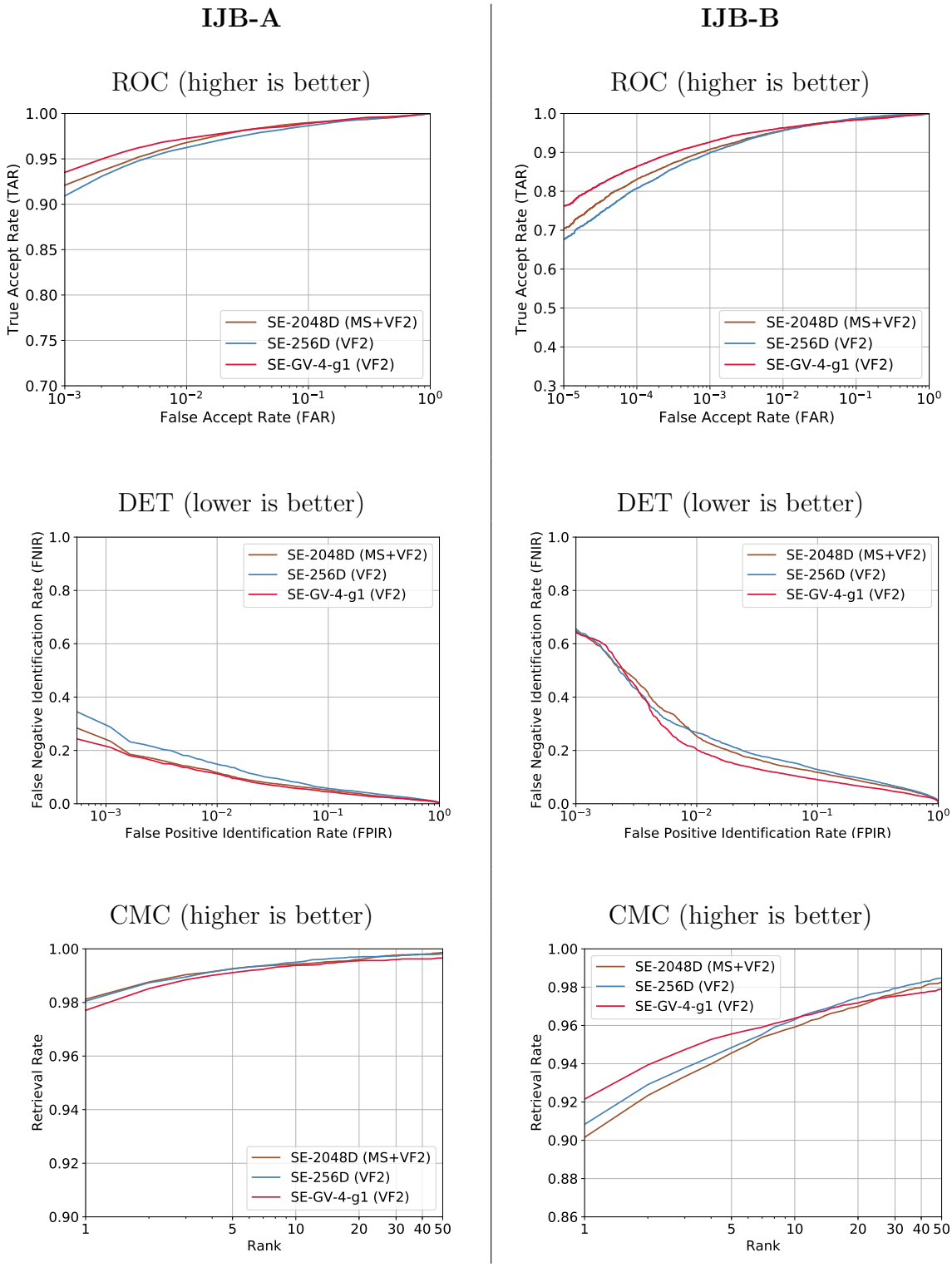


Figure 7.3: **Results on the IJB-A and IJB-B datasets.** Our *SE-GV-4-g1* network which produces 128-D templates, beats the best baseline (*SE* with 256-D templates) and the state-of-the-art trained on a much larger dataset (*SE* with 2048-D templates) trained on VGGFace2 and MS-Celeb-1M).

shown in Figure 7.4. As the number of positive pairs (*i.e.* templates of same identity) as much smaller than that of negative pairs (*i.e.* templates of different identities), we separate the two situations for better viewing.

As Figure 7.4 shows, the distribution of the scores of positive pairs are similar for both networks (*i.e. SE-256D* and *SE-GV-4-g1*), while for negative pairs, *SE-GV-4-g1* produces scores that are clearly smaller than *SE-256D*, with a smaller mean and smaller standard deviation (*i.e.* scores spreading with a smaller range). This results in a larger separation between the similarity between positive and negative template pairs, and therefore better performance. Consequently, we can see that NetVLAD encoding benefits our face recognition task by increasing the separation between template descriptors of same identities and different identities.

7.3.6 Analysis of ghost clusters

Addition of ghost clusters was motivated by the intuition that it enables our network to learn to ignore uninformative low-quality images by assigning them to the discarded ghost clusters. Here we evaluate this hypothesis qualitatively.

Recall that GhostVLAD computes a template representation by aggregating residual vectors of input descriptors, where a residual vector is a concatenation of per non-ghost cluster residuals weighted by their non-ghost assignment weights (Section 7.1.2). Therefore, the contribution of a specific example image towards the template representation can be measured as the norm of the residual.

Figure 7.5 show that our intuition is correct – the network automatically learns to dramatically down-weight blurry and low-resolution images, thus improving the signal-to-noise ratio. Note that this behaviour emerges completely automatically without ever explicitly teaching the network to down-weight low-quality images.

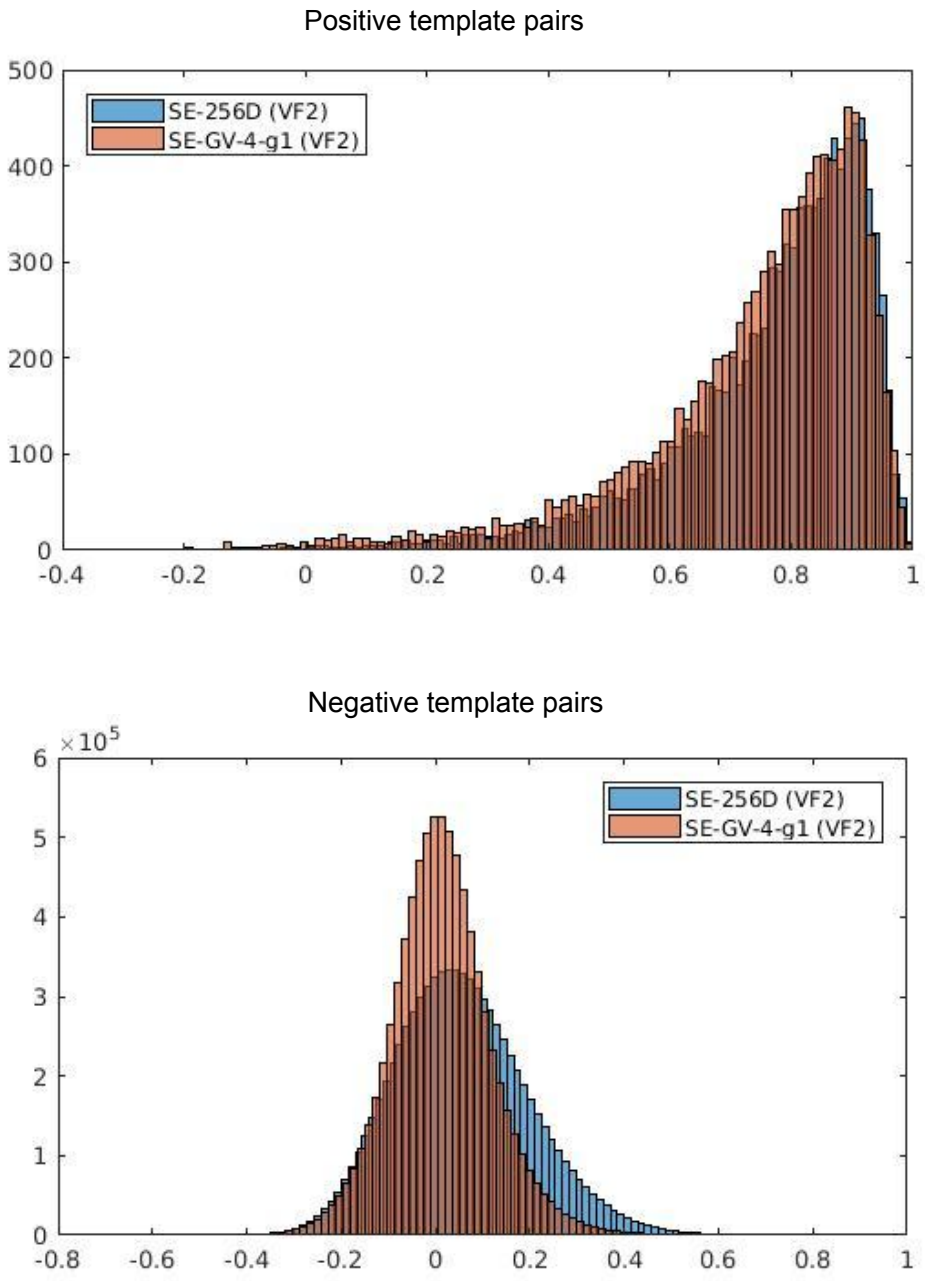


Figure 7.4: **Effect of NetVLAD encoding.** Top: similarity score distribution of positive template pairs (*i.e.* of same identities). Bottom: similarity score distribution of negative template pairs (*i.e.* of different identities). The similarity scores are computed for the verification task on the IJB-B dataset.

7.4 Conclusions

We introduced a neural network architecture and training procedure for learning compact representations of image sets for template-based face recognition. Due to the novel GhostVLAD layer, the network is able to automatically learn to weight face descriptors depending on their information content. Our template representations outperform the state-of-the-art on the challenging IJB-A and IJB-B benchmarks by a large margin.

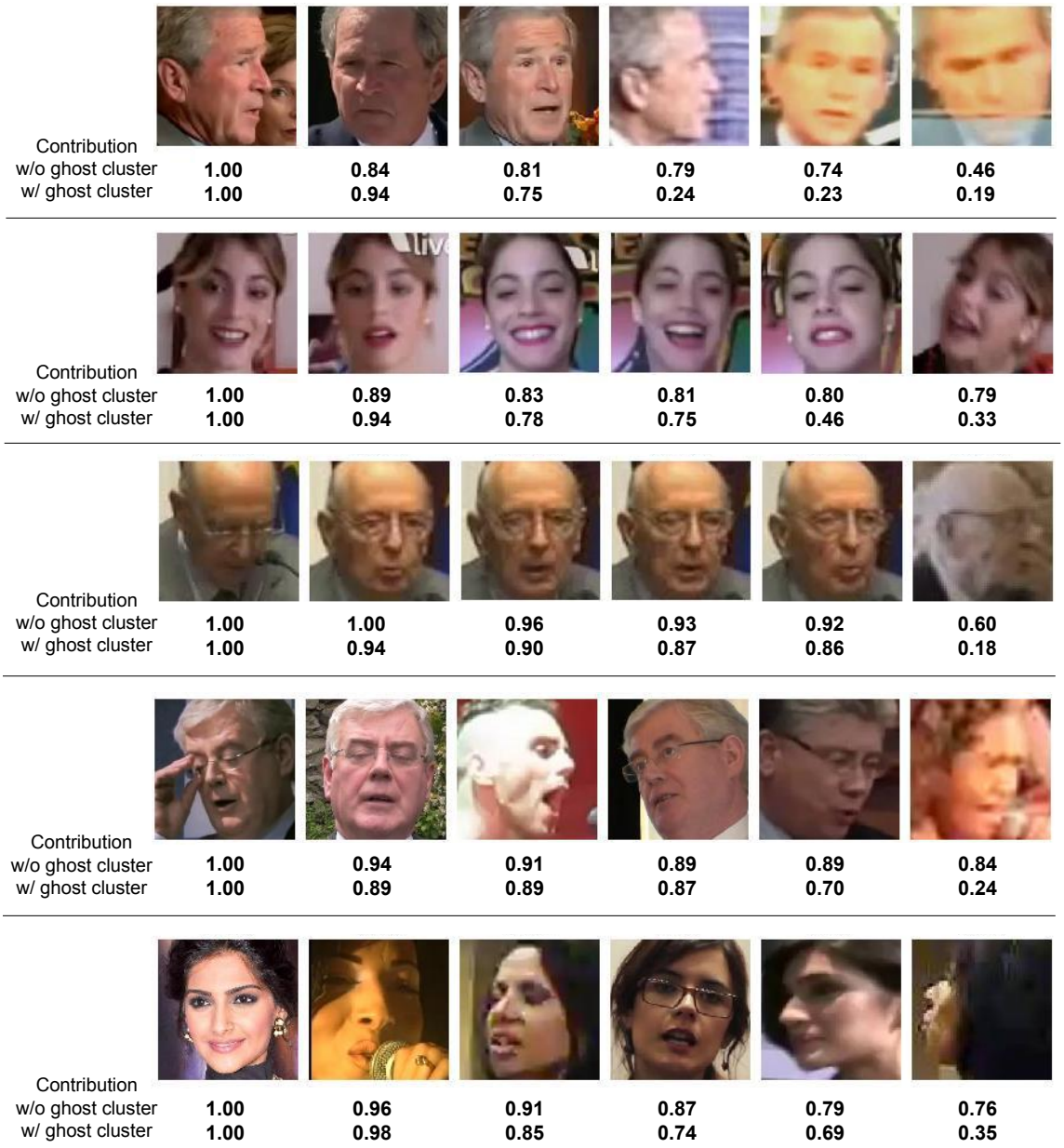


Figure 7.5: **Effect of ghost clusters.** Each row shows shows 6 images from a template in the IJB-B dataset. The contribution (relative to the max) of each image to the final template representation is shown (see Section 7.3.6 for details), for the cases of no ghost clusters ($G = 0$, network *SE-GV-3*) and one ghost cluster ($G = 1$, network *SE-GV-4-g1*) in GhostVLAD. Introduction of a single ghost cluster dramatically reduces the contribution of low-quality images to the template, improving the signal-to-noise ratio.

Chapter 8

Conclusion

In this chapter we summarize the achievements of the work illustrated in this thesis and discuss some future research directions.

8.1 Achievements

We have made five contributions to the research field in this thesis. First, we collect and annotate two large-scale datasets which can be used as benchmarks for face-related retrieval tasks. Second, an image synthesis engine is developed for replacing faces and generating high-quality images. Third, we propose a ranking method and train a CNN for retrieving target faces in target places. Fourth, a compact set representation is learnt for set retrieval problem, in which we focus on retrieving sets of identities. Lastly, a similar architecture used for set retrieval is applied to set-based face recognition and it achieves the state-of-the-art performance on currently one of the most challenging dataset. In the following, we will discuss each of the achievements in detail.

Chapter 3 introduced two new datasets – the *Celebrity in Places* dataset and the *Celebrity Together* dataset. The *Celebrity in Places* dataset can be used to evaluate the ability of searching target identities in target places. On the other hand, the

Celebrity Together dataset can be considered as an evaluation benchmark for retrieving a set of identities, rather than a single one. To our knowledge, they are the first datasets on their corresponding tasks. Both datasets are publicly accessible on the Internet.

In Chapter 4, we developed an image synthesis pipeline which can generate images with faces of labeled identities in different background scenes. The synthetic images look natural with no artifacts. Furthermore, the synthesizing process is fast and scalable to million of images. Although this pipeline is designed to generate images containing labeled faces in labeled places, we expect this pipeline to be useful for other tasks such as face recognition in which this synthesis can be considered as an data-augmentation method.

The problem of retrieving particular identities in target places is tackled in Chapter 5. We learn a new type of place descriptors that are aware of faces and face descriptors, and design a rule to combine the ranking scores of faces and places. Furthermore, the two-stream CNN that produces such place descriptors is trained to accommodate this combination rule. The new place descriptors have shown superior performance over several baseline methods on the *Celebrity Together* dataset described in Chapter 3. Of course, the methods developed are agnostic about the particular people and places, and can be applied to personal image and video collections, not just to celebrities in places. Also, the hybrid CNN architecture is independent of the underlying face and place CNNs used, and further performance boosts can be expected by replacing those CNNs with more powerful CNNs (such as ResNet (He et al. (2015))) in the future.

In Chapter 6 we proposed a new research topic named set retrieval. In particular, we focus on a specific task of set retrieval – retrieving a set of identities. In order to enable real-time retrieval, we learn a deep CNN that encodes and aggregates the faces in each set to produce a single fixed-length set-level descriptor. This set representation exceeds the strong baselines by a large margin. Moreover, we also explore the

relationship between retrieval quality and retrieval speed using this compact representation, which would be useful in practice. Note, although we have grounded the set retrieval problem as faces, the treatment is quite general: it only assumes that dataset elements are represented by vectors and the scoring function is a scalar product.

Chapter 7 applied a similar CNN architecture on a different computer vision task, *i.e.* set-based unconstrained face recognition. Based on this architecture, we further extend the NetVLAD layer to contain *ghost clusters*, such that the CNN is able to deal with low-quality images which exist a lot in an unconstrained scenario. Extensive experiments in this chapter demonstrated that the learnt template representation outperforms the state-of-the-art methods on a very challenging public benchmark. More importantly, the network architecture proposed could also be applied to other image-set tasks such as person re-identification. More generally, the idea of having a 'null' vector available for assignments in the proposed GhostVLAD layer could have applicability in many situations where it is advantageous to have a mechanism to remove noisy or corrupted data.

8.2 Potential directions of future work

In this section, we discuss some possibilities of future work.

Dataset collection. A straightforward extension to dataset collection and annotation work (Chapter 3) is to enlarge the current datasets with more identities. As both the *Celebrity in Places* dataset and the *Celebrity Together* dataset introduced in Chapter 3 are built based on the VGGFace dataset (Parkhi et al. (2015)), they could be easily extended by using the identities from other larger face datasets such as the VGGFace2 dataset (Cao et al. (2018)). Apart from identities, more types of places (rather than the existing 16 places) could also be looked into for the *Celebrity in Places* dataset.

Image synthesis system. The current image synthesis system presented in Chapter 4 has two major limitations. First, it is only capable to deal with frontal and near-frontal faces, but not profiles. This is mainly due to the limitation of facial landmark detector as it can not produce the landmarks for profile faces. In the future, we can easily solve this problem by using more advanced facial landmark detectors such as dlib (Kazemi and Sullivan (2014)). Another shortcoming of the current image synthesis system is the quality control stage for the synthesized images. So far we only filter the synthesized images based on the classification score of the face, *i.e.* the new face should be correctly recognized by the face CNN. However, there is no explicit assessment on how natural the new faces look. This might be done by using some evaluation metrics for generative models (*e.g.* (Olsson et al. (2018))) on the quality of synthesized images.

Faces in Places. There are several interesting extensions to consider for compound query retrieval described in Chapter 5. For example, the compound query can be a combination of identities and actions, which might be tackled by replacing the place stream in the hybrid CNN proposed in Chapter 5 by a network for action recognition (Feichtenhofer et al. (2016), Girdhar et al. (2017), Simonyan and Zisserman (2014)). Another possible combination would be category-level objects and instance-level objects, such as a monkey and a Coca-Cola can. For this particular task, an elegant and efficient solution would be encode both objects using the same convolutional layers, which can greatly reduce the computational complexity.

Set retrieval. In Chapter 6, we focus on a particular example of set retrieval: retrieving sets of identities. Further investigation can be conducted in closing the gap between the performance of *descriptor-per-set* and *descriptor-per-element*. So far the performance of the two retrieval methods are similar when combined with re-ranking stage. However, without re-ranking, *descriptor-per-set* is still much worse

than *descriptor-per-element*. It would be interesting to see new aggregation algorithms that can further reduce the performance gap. On the other hand, future work can also consider other set retrieval tasks, such as retrieving sets of general objects or video retrieval. Notice that for retrieving sets of identities, each set is an image and elements are the faces in each image; whereas for video retrieval, sets would become videos and elements are frames in videos.

Set-based face recognition. For the set-based face recognition problem considered in Chapter 7, a potential improvement would be to enable the network to remove images that do not belong the subject of the set. In the IJB-B dataset, as the images are taken in an unconstrained setting, there are noise on two aspects: images with low quality and images that should not be in the set (*i.e.* images of other subjects). The CNN architecture presented in Chapter 7 can only deal with low-quality images, but not false images. There has been some work that aim to solve this problem, *e.g.* (Xie and Zisserman (2018)). Therefore, it would be beneficial to include a similar module that can explore the relation between images in each set and thus remove the false images.

Bibliography

Research page. http://www.robots.ox.ac.uk/~vgg/research/faces-in-places/.

O. Arandjelovic and R. Cipolla. An information-theoretic approach to face recognition from face motion manifolds. *Image and Vision Computing*, 2006.

R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *Proc. CVPR*, 2012a.

R. Arandjelović and A. Zisserman. Multiple queries for large scale specific object retrieval. In *Proc. BMVC.*, 2012b.

R. Arandjelović and A. Zisserman. All about VLAD. In *Proc. CVPR*, 2013.

R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *Proc. CVPR*, 2016.

A. Babenko and V. Lempitsky. Aggregating local deep features for image retrieval. In *Proc. ICCV*, pages 1269–1277, 2015.

A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky. Neural codes for image retrieval. In *Proc. ECCV*, pages 584–599. Springer, 2014.

A. Bansal, C. Castillo, R. Ranjan, and R. Chellappa. The dos and donts for cnn-based face verification. *arXiv preprint arXiv:1705.07426*, 5, 2017a.

A. Bansal, A. Nanduri, C. Castillo, R. Ranjan, and R. Chellappa. Umdfaces: An annotated face dataset for training deep networks. In *Biometrics (IJCB), 2017 IEEE International Joint Conference on*, pages 464–473. IEEE, 2017b.

H. Bilen and A. Vedaldi. Universal representations: The missing link between faces, text, planktons, and cat breeds. *arXiv preprint arXiv:1701.07275*, 2017.

D. Bitouk, N. Kumar, S. Dhillon, P. Belhumeur, and S. Nayar. Face swapping: automatically replacing faces in photographs. In *ACM Transactions on Graphics (TOG)*, volume 27, page 39. ACM, 2008.

Q. Cao, L. Shen, W. Xie, O. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. *arXiv preprint arXiv:1710.08092*, 2017.

Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. VGGFace2: A dataset for recognising faces across pose and age. In *Proc. Int. Conf. Autom. Face and Gesture Recog.*, 2018.

H. Cevikalp and B. Triggs. Face recognition based on image sets. In *Proc. CVPR*, 2010.

H. Chatbri, K. Kameyama, P. Kwan, S. Little, and N. OConnor. A novel shape descriptor based on salient keypoints detection for binary image matching and retrieval. *Multimedia Tools and Applications*, pages 1–24, 2018.

K. Chatfield and A. Zisserman. Visor: Towards on-the-fly large-scale object category retrieval. In *Proc. ACCV*, Lecture Notes in Computer Science. Springer, 2012.

K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *Proc. BMVC.*, 2011.

K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proc. BMVC.*, 2014a.

K. Chatfield, K. Simonyan, and A. Zisserman. Efficient on-the-fly category retrieval using convnets and gpus. In *Proc. ACCV*, 2014b.

K. Chatfield, R. Arandjelović, O. M. Parkhi, and A. Zisserman. On-the-fly learning for visual search of large-scale image and video datasets. *International Journal of Multimedia Information Retrieval*, 2015.

D. Chen, S. Tsai, V. Chandrasekhar, G. Takacs, H. Chen, R. Vedantham, R. Grzeszczuk, and B. Girod. Residual enhanced visual vectors for on-device image matching. In *Asilomar*, 2011.

D. Chen, X. Cao, F. Wen, and J. Sun. Blessing of dimensionality: High dimensional feature and its efficient compression for face verification. In *Proc. CVPR*, 2013.

J. Chen, R. Ranjan, A. Kumar, C. Chen, V. Patel, and R. Chellappa. An end-to-end system for unconstrained face verification with deep convolutional neural networks. In *ICCV Workshops*, pages 118–126, 2015.

L. Chen, F. Wang, C. Li, S. Huang, Y. Chen, C. Qian, and C. Change. The devil of face recognition is in the noise. In *Proc. ECCV*, pages 765–780, 2018.

O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. In *Proc. CIVR*, 2007.

O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *Proc. BMVC.*, 2008.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR*, 2:265–292, 2001.

N. Crosswhite, J. Byrne, C. Stauffer, O. M. Parkhi, Q. Cao, and A. Zisserman. Template adaptation for face verification and identification. In *Proc. Int. Conf. Autom. Face and Gesture Recog.*, 2017.

T. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. *VISAPP*, 2009.

J. Delhumeau, P.-H. Gosselin, H. Jégou, and P. Pérez. Revisiting the VLAD image representation. In *Proc. ACMM*, 2013.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F. Li. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009.

J. Deng, Y. Zhou, and S. Zafeiriou. Marginal loss for deep face recognition. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPRW), Faces in-the-wild Workshop/Challenge*, volume 4, 2017.

A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proc. ICCV*, 2015a.

A. Dosovitskiy, J. Tobias Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *Proc. CVPR*, pages 1538–1546, 2015b.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010.

C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proc. CVPR*, 2016.

R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *Proc. CVPR*, volume 2, page 3, 2017.

Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *Proc. ECCV*, pages 392–407. Springer, 2014.

A. Gordo, J. Almazán, J. Revaud, and D. Larlus. Deep image retrieval: Learning global representations for image search. In *Proc. ECCV*, pages 241–257. Springer, 2016.

G. Goswami, R. Bhardwaj, R. Singh, and M. Vatsa. Mdlface: Memorability augmented deep learning for video face recognition. In *Biometrics (IJCB), 2014 IEEE International Joint Conference on*, pages 1–7. IEEE, 2014.

Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *Proc. ECCV*, pages 87–102. Springer, 2016.

A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *Proc. CVPR*, 2016.

S. Hamid Rezatofighi, B. Kumar, A. Milan, E. Abbasnejad, A. Dick, and I. Reid. DeepSetNet: Predicting sets with deep neural networks. In *Proc. CVPR*, 2017.

T. Hassner, I. Masi, J. Kim, J. Choi, S. Harel, P. Natarajan, and G. Medioni. Pooling faces: template based face recognition with pooled face images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 59–67, 2016.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proc. CVPR*, 2018.

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.

Z. Huang, R. Wang, S. Shan, and X. Chen. Projection metric learning on grassmann manifold with application to video based face recognition. In *Proc. CVPR*, pages 140–149, 2015.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015.

A. Iscen, T. Furon, V. Gripon, M. Rabbat, and H. Jégou. Memory vectors for similarity search in high-dimensional spaces. 2017.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, pages 487–493, 1998.

M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. In *Workshop on Deep Learning, NIPS*, 2014.

M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *IJCV*, 116(1):1–20, Jan 2016.

H. Jégou and O. Chum. Negative evidences and co-occurrences in image retrieval: the benefit of PCA and whitening. In *Proc. ECCV*, 2012.

H. Jégou and A. Zisserman. Triangulation embedding and democratic aggregation for image search. In *Proc. CVPR*, 2014.

H. Jégou, M. Douze, and C. Schmid. Packing bag-of-features. In *Proc. ICCV*, Sep 2009a.

H. Jégou, M. Douze, and C. Schmid. On the burstiness of visual elements. In *Proc. CVPR*, Jun 2009b.

H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Proc. CVPR*, 2010.

H. Jégou, M. Douze, and C. Schmid. Exploiting descriptor distances for precise image search. Technical report, INRIA, 2011a.

H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. P'erez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE PAMI*, 2011b.

H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local images descriptors into compact codes. *IEEE PAMI*, 2012.

Y. Kalantidis, C. Mellina, and S. Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *Proc. ECCV*, pages 685–701. Springer, 2016.

Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874, 2014.

I. Kemelmacher, S. Seitz, D. Miller, and E. Brossard. The megaface benchmark: 1 million faces for recognition at scale. In *Proc. CVPR*, pages 4873–4882, 2016.

T. Kim, O. Arandjelović, and R. Cipolla. Boosted manifold principal angles for image set-based recognition. *Pattern Recognition*, 40(9):2475–2484, 2007.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

B. Klare, B. Klein, E. Taborsky, A. Blanton, J. Cheney, K. Allen, P. Grother, A. Mah, and A. Jain. Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a. In *Proc. CVPR*, pages 1931–1939, 2015.

R. Kondor and T. Jebara. A kernel between sets of vectors. In *Proc. ICML*. AAAI Press, 2003.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.

F. W. Lancaster and E. Gallup. Information retrieval on-line. Technical report, 1973.

A. H. Lashkari, F. Mahdavi, and V. Ghomi. A boolean model in information retrieval for search engines. In *Information Management and Engineering, 2009. ICIME'09. International Conference on*, pages 385–389. IEEE, 2009.

S. Lazebnik, C. Schmid, and J Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proc. CVPR*, 2006.

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.

K. Lee, J. Ho, M. Yang, and D. Kriegman. Video-based face recognition using probabilistic appearance manifolds. In *Proc. CVPR*, volume 1, pages I–I. IEEE, 2003.

T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and L Zitnick. Microsoft coco: Common objects in context. In *Proc. ECCV*, pages 740–755. Springer, 2014.

W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-margin softmax loss for convolutional neural networks. In *Proc. ICML*, pages 507–516, 2016.

W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proc. CVPR*, volume 1, page 1, 2017a.

W. Liu, Y. Zhang, X. Li, Z. Yu, B. Dai, T. Zhao, and L. Song. Deep hyperspherical learning. In *NIPS*, pages 3950–3960, 2017b.

Y. Liu, J. Yan, and W. Ouyang. Quality aware network for set to set recognition. In *Proc. CVPR*, pages 5790–5799, 2017c.

D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2): 91–110, 2004.

C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

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

B. Maze, J. Adams, J. Duncan, N. Kalka, T. Miller, C. Otto, A. Jain, T. Niggel, J. Anderson, J. Cheney, et al. IARPA Janus Benchmark–C: Face dataset and protocol. 2018.

A. Miech, I. Laptev, and J. Sivic. Learnable pooling with context gating for video classification. *arXiv preprint arXiv:1706.06905*, 2017.

E. Mohedano, K. McGuinness, N. O'Connor, A. Salvador, F. Marqués, and X. Giro-i Nieto. Bags of local convolutional features for scalable instance search. In *ACM ICMR*, pages 327–331. ACM, 2016.

E. Mohedano, A. Salvador, K. McGuinness, X. Giró-i Nieto, N. OConnor, and F. Marqués. Object retrieval with deep convolutional. *Deep Learning for Image Processing Applications*, 31:137, 2017.

M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithmic configuration. In *Proc. VISAPP*, 2009.

H. Ng and S. Winkler. A data-driven approach to cleaning large face datasets. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 343–347. IEEE, 2014.

C. Olsson, S. Bhupatiraju, T. Brown, A. Odena, and I. Goodfellow. Skill rating for generative models. *arXiv preprint arXiv:1808.04888*, 2018.

O. M. Parkhi, K. Simonyan, A. Vedaldi, and A. Zisserman. A compact and discriminative face track descriptor. In *Proc. CVPR*. IEEE, IEEE, 2014.

O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *Proc. BMVC.*, 2015.

O. Paul, G. Awad, M. Michel, J. Fiscus, W. Kraaij, A. F. Smeaton, and G. Quéenot. TRECVID 2011 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *TRECVID*, 2011.

P. Perez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Transactions on Graphics*, 22(3):313–318, 2003.

F. Perronnin and D. Dance. Fisher kernels on visual vocabularies for image categorization. In *Proc. CVPR*, 2007.

F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *Proc. CVPR*, 2010a.

F. Perronnin, J. Sánchez, and Y. Liu. Large-scale image categorization with explicit data embedding. In *Proc. CVPR*, 2010b.

F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *Proc. ECCV*, 2010c.

F. Perronnin, Z. Akata, Z. Harchaoui, and C. Schmid. Towards good practice in large-scale learning for image classification. In *Proc. CVPR*, pages 3482–3489, 2012.

J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc. CVPR*, 2007.

J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proc. CVPR*, 2008.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning*, pages 185–208, 1999.

A. Quattoni and A. Torralba. Recognizing indoor scenes. In *Proc. CVPR*, 2009.

F. Radenović, G. Tolias, and O. Chum. Cnn image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In *Proc. ECCV*, pages 3–20. Springer, 2016.

Y Rao, J Lin, J Lu, and J Zhou. Learning discriminative aggregation network for video-based face recognition. In *Proc. CVPR*, pages 3781–3790, 2017.

A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN Features off-the-shelf: an Astounding Baseline for Recognition. *CoRR*, abs/1403.6382, 2014.

A. Razavian, J. Sullivan, S. Carlsson, and A. Maki. Visual instance retrieval with deep convolutional networks. *ITE Transactions on Media Technology and Applications*, 4(3):251–258, 2016.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, S. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, and F.F. Li. ImageNet large scale visual recognition challenge. *IJCV*, 2015.

G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.

S. Sankaranarayanan, A. Alavi, C. Castillo, and R. Chellappa. Triplet probabilistic embedding for face verification and clustering. *arXiv preprint arXiv:1604.05417*, 2016.

F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A unified embedding for face recognition and clustering. In *Proc. CVPR*, 2015.

J. A. Shaw and E. A. Fox. Combination of multiple searches. In *The Second Text REtrieval Conference (TREC-2)*, pages 243–252, 1994.

K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.

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

J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, volume 2, pages 1470–1477, 2003.

Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *NIPS*, pages 1988–1996, 2014a.

Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10,000 classes. In *Proc. CVPR*, pages 1891–1898, 2014b.

Y. Sun, D. Liang, X. Wang, and X. Tang. Deepid3: Face recognition with very deep neural networks. *arXiv preprint arXiv:1502.00873*, 2015a.

Y. Sun, X. Wang, and X. Tang. Deeply learned face representations are sparse, selective, and robust. In *Proc. CVPR*, pages 2892–2900, 2015b.

Y. Sun, X. Wang, and X. Tang. Sparsifying neural network connections for face recognition. In *Proc. CVPR*, pages 4856–4864, 2016.

Y. Sun, L. Zheng, W. Deng, and S. Wang. SVDNet for pedestrian retrieval. In *Proc. ICCV*, 2017.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D Anguelov, D. Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. CVPR*, 2015.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proc. CVPR*, pages 2818–2826, 2016.

C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

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.

Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Web-scale training for face identification. In *Proc. CVPR*, pages 2746–2754, 2015.

T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

G. Tolias, R. Sicre, and H. Jégou. Particular object retrieval with integral max-pooling of cnn activations. *arXiv preprint arXiv:1511.05879*, 2015.

P. Turaga, A. Veeraraghavan, A. Srivastava, and R. Chellappa. Statistical computations on grassmann and stiefel manifolds for image and video-based recognition. *IEEE PAMI*, 33(11):2273–2286, 2011.

J. C. van Gemert, J. M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders. Kernel codebooks for scene categorization. In *Proc. ECCV*, 2008.

A. Vedaldi and K. Lenc. MatConvNet: Convolutional neural networks for MATLAB. In *Proc. ACMM*, 2015.

F. Wang, X. Xiang, J. Cheng, and A. L. Yuille. Normface: L_2 hypersphere embedding for face verification. *Proc. ACMM*, 2017a.

G. Wang and D. Forsyth. Object image retrieval by exploiting online knowledge resources. *Proc. CVPR*, pages 1–8, 2008.

J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *Proc. CVPR*, 2010.

T. Wang, D. Wu, A. Coates, and A. Ng. End-to-end text recognition with convolutional neural networks. In *Proc. ICPR*, pages 3304–3308. IEEE, 2012.

W. Wang, R. Wang, Z. Huang, S. Shan, and X. Chen. Discriminant analysis on riemannian manifold of gaussian distributions for face recognition with image sets. In *Proc. CVPR*, pages 2048–2057, 2015.

W. Wang, R. Wang, S. Shan, and X. Chen. Discriminative covariance oriented representation learning for face recognition with image sets. In *Proc. CVPR*, pages 5599–5608, 2017b.

Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *Proc. ECCV*, pages 499–515. Springer, 2016.

C. Whitelam, E. Taborsky, A. Blanton, B. Maze, J. Adams, T. Miller, N. Kalka, A. Jain, J. Duncan, K. Allen, et al. Iarpa janus benchmark-b face dataset. In *CVPR Workshop on Biometrics*, 2017.

L. Wolf, Tal. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched backgroundsimilarity. In *Proc. CVPR*, 2011.

Y. Wu, H. Liu, J. Li, and Y Fu. Deep face recognition with center invariant loss. In *Proc. ACMM*, pages 408–414. ACM, 2017.

Z. Wu, Q. Ke, J. Sun, and H. Shum. Scalable face image retrieval with identity-based quantization and multireference reranking. *IEEE PAMI*, 33(10):1991–2001, 2011.

J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proc. CVPR*, pages 3485–3492. IEEE, 2010.

L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *JMLR*, 11:2543–2596, 2010.

S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proc. CVPR*, pages 5987–5995. IEEE, 2017.

W. Xie and A. Zisserman. Multicolumn networks for face recognition. *arXiv preprint arXiv:1807.09192*, 2018.

X. Xiong and F. De la Torre. Supervised descent method and its applications to face alignment. In *Proc. CVPR*, 2013.

J. Yang, P. Ren, D. Chen, F. Wen, H. Li, and G. Hua. Neural aggregation network for video face recognition. *arXiv preprint*, 2017.

M. Yang, P. Zhu, L. Van Gool, and L. Zhang. Face recognition based on regularized nearest points between image sets. In *Proc. Int. Conf. Autom. Face and Gesture Recog.*, 2013.

D. Yi, Z. Lei, S. Liao, and S. Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.

I. Yildirim, T. D. Kulkarni, W. A. Freiwald, and J. B. Tenenbaum. Efficient and robust analysis-by-synthesis in vision: A computational framework, behavioral tests, and modeling neuronal representations. In *Annual Conference of the Cognitive Science Society*, 2015.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017.

M. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10): 1499–1503, 2016.

W.-L. Zhao, H. Jégou, and G. Gravier. Oriented pooling for dense and non-dense rotation-invariant features. In *Proc. BMVC.*, 2013.

Y. Zheng, D. Pal, and M. Savvides. Ring loss: Convex feature normalization for face recognition. In *Proc. CVPR*, pages 5089–5097, 2018.

Y. Zhong, R. Arandjelović, and A. Zisserman. Faces in places: Compound query retrieval. In *Proc. BMVC.*, 2016.

B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, pages 487–495, 2014a.

B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene CNNs. In *ICLR*, 2015.

J. Zhou, K. McGuinness, and N. OConnor. A text recognition and retrieval system for e-business image management. In *International Conference on Multimedia Modeling*, pages 23–35. Springer, 2018.

X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *Proc. ECCV*, 2010.

Ziheng Zhou, Guoying Zhao, Xiaopeng Hong, and Matti Pietikäinen. A review of recent advances in visual speech decoding. *Image and vision computing*, 32(9): 590–605, 2014b.