# Compact Deep Aggregation for Set Retrieval

Yujie Zhong[1], Relja Arandjelović[2], and Andrew Zisserman[1]

[1] Visual Geometry Group, Department of Engineering Science, University of Oxford, UK
{yujie,az}@robots.ox.ac.uk
[2] DeepMind
relja@google.com

**Abstract.** The objective of this work is to learn a compact embedding of a set of descriptors that is suitable for efficient retrieval and ranking, whilst maintaining discriminability of the individual descriptors. We focus on a specific example of this general 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.
To this end, we make the following contributions: first, we propose a CNN architecture – *SetNet* – 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, and the score of an image is a count of the number of identities that match the query; second, we show that this compact descriptor has minimal loss of discriminability up to two faces per image, and degrades slowly after that – far exceeding a number of baselines; third, we explore the speed vs. retrieval quality trade-off for set retrieval using this compact descriptor; and, finally, we collect and annotate a large dataset of images containing various number of celebrities, which we use for evaluation and will be publicly released.

## 1 Introduction

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. We would also like this retrieval to happen in real time.

This 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 the query identities, etc. An example of this ranking is shown in Fig. 1 for two queries.

We can operationalize this 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 logistic regression classifiers. But, consider


Fig. 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 this paper.

the situation where the dataset is very large, containing millions or billions of images each containing multiple faces. In this situation two aspects are crucial for real time retrieval: first, that all operations take place in memory (not reading from disk), and second that an efficient algorithm is used when searching for images that satisfy the query. The problem is that storing a fixed length vector for each *face* in memory is prohibitively expensive at this scale, but this cost can be significantly reduced if a fixed length vector is only stored 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, the question we investigate in this paper is the following: can we aggregate the *set of vectors* representing the multiple faces in an image into 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*) have to be stored and scored.

Although we have motivated this question by face retrieval it is quite general: there is a set of elements, each element is represented by a vector of dimension $D$, and we wish to represent this set by a single vector of dimension $D'$, where $D' = D$ in practice, without losing information essential for the task. Of course, if the 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: 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 [25] and VLAD [12]; 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 Sec. 2 where compact set-level descriptors are trained in an end-to-end manner using a ResNet-50 [8] as the base CNN.

Our second contribution is to introduce a dataset annotated with multiple faces per images. In Sec. 3 we describe a pipeline for automatically generating a labelled dataset of pairs (or more) of celebrities per image. This *Celebrity Together* dataset contains around 200k images with more than half a million faces in total. It will be publicly released.

The performance of the set-level descriptors is evaluated in Sec. 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, where images contain a variable number of faces, with many not corresponding to the queries, and explore efficient algorithms that can achieve immediate (real-time) retrieval on very large scale datasets.

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. We return to this point in the conclusion.

### 1.1 Related work

To the best of our knowledge, this paper is the first to consider the set retrieval problem. However, the general area of image retrieval has an extensive literature that we build on here.

One of the central problems that has been studied in large scale image instance retrieval is how to condense the information stored in multiple local descriptors such as SIFT [16], into a single compact vector to represent the image. This problem has been driven by the need to keep the memory footprint low for very large image datasets. An early approach is to cluster descriptors into visual words and represent the image as a histogram of word occurrences – bag-of-visual-words [25]. Performance can be improved by aggregating local descriptors within each cluster, in representations such as Fisher Vectors [13, 20] and VLAD [12]. In particular, VLAD – 'Vector of Locally Aggregated Descriptors' by Jégou et al. [12] and its improvements [2, 5, 14] was used

to obtain very compact descriptions via dimensionality reduction by PCA, considerably reducing the memory requirements compared to earlier bag-of-words based methods [18, 22, 25]. VLAD has superior ability in maintaining the information about individual local descriptors while performing aggressive dimensionality reduction.

VLAD has recently been adapted into a differentiable CNN layer, NetVLAD [1], making it end-to-end trainable. We incorporate a modified form of the NetVLAD layer in our SetNet architecture. An alternative, but related, very recent approach is the memory vector formulation of [10], but we have not employed it here as it has not been made differentiable yet.

Another strand of research we build on is category level retrieval, where in our case the category is a face. This is another classical area of interest with many related works [4, 21, 28, 30, 32]. For the case of faces, the feature vector is produced from the face region using a CNN trained to classify or embed faces [19, 24, 27].

Also relevant are works that explicitly deal with sets of vectors. Kondor and Jebara [15] 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, Zaheer et al. [31] investigate permutation-invariant objective functions for operating on sets, although their method boils down to average pooling of input vectors, which we compare to as a baseline. Rezatofighi et al. [7] consider the problem of predicting sets, i.e. having a network which outputs sets, rather than our case where a set of elements is an input to be processed and described with a single vector.

## 2 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. There are two objectives:

**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.

### 2.1 SetNet architecture

In this section we introduce our CNN architecture, designed to aggregate multiple element (face) descriptors into a single fixed-size set representation. The *SetNet* architecture (Fig. 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.

**Feature extraction.** The first part is a modified ResNet-50 [8] 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 $2048 \times 128$ after the global average pooling layer in the original ResNet, in order to obtain a lower dimensional face descriptor (full details of the architecture are given in the supplementary material). In this part, individual element descriptors $(x_1, \ldots, x_F)$ are obtained from ResNet, where $F$ is the number of faces in an image.

**Feature aggregation.** Face features are aggregated into a single vector $V$ using a NetVLAD layer (illustrated in Fig. 3, and described below in Sec. 2.2). 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 [5] 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 [9], and L2-normalized to produce the final set representation $v_{set}$.

**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. Sec. 2.3 describes the appropriate loss and training procedure in more detail.

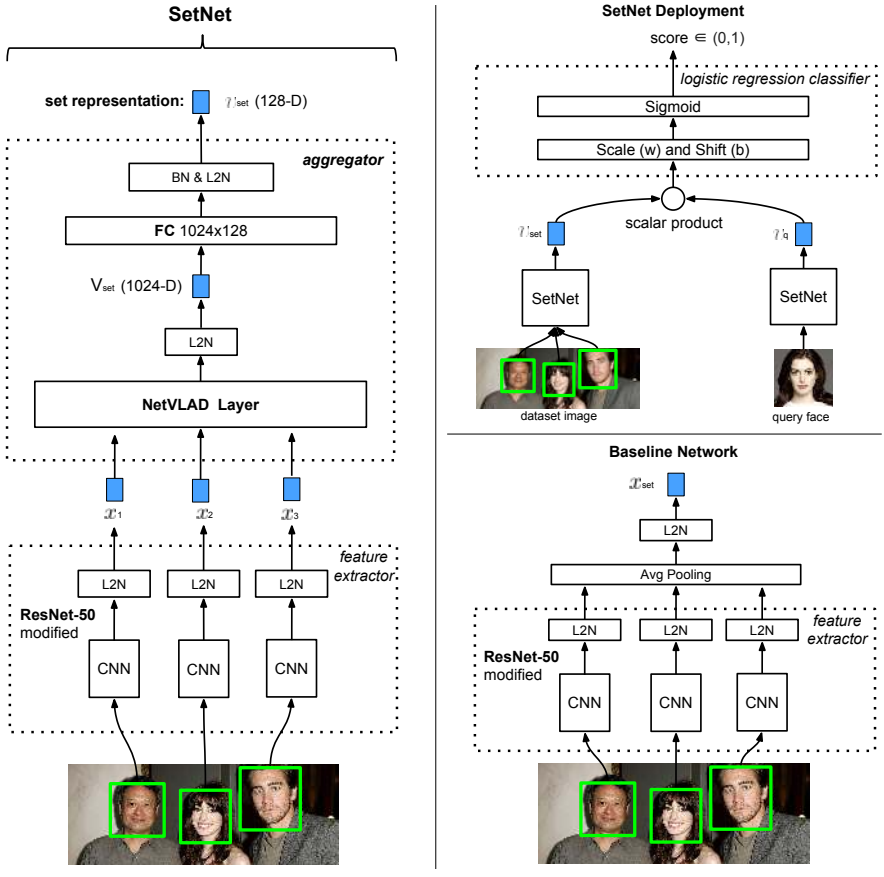



Fig. 2: **SetNet architecture and training. Left**: SetNet – features are extracted from each face in an image using a modified ResNet-50. They are aggregated using a modified NetVLAD layer into a single 1024-D vector which is then reduced to 128-D via a fully connected dimensionality reduction layer, 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.

## 2.2 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 [1].

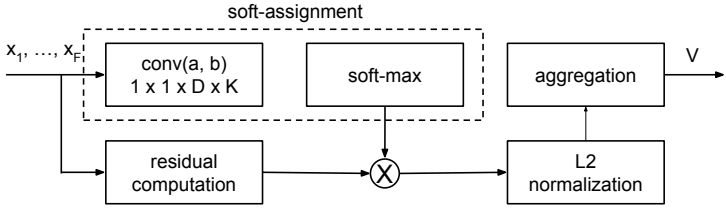



Fig. 3: **NetVLAD layer.** Illustration of the NetVLAD layer [1], corresponding to equation (1), and slightly modified to perform L2-normalization before aggregation; see Sec. 2.2 for details.

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

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

where $\{a_k\}$, $\{b_k\}$ and $\{c_k\}$ are trainable parameters for $k \in [1, 2, \ldots, K]$. 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. Finally, the vector is L2-normalized.

### 2.3 Loss function and training procedure

In order to achieve the two objectives outlined at the beginning of Sec. 2, a Multi-label logistic regression loss is used. Suppose a particular training image contains $F$ faces, and the mini-batch consists of faces for $P$ identities. Then in a forward pass at training time, the descriptors for the $F$ 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 $f$ 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 $F$ 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:

$$-\sum_{f=1}^{P} y_f \log(\sigma(w(v_f^T v_{set}) + b)) + (1 - y_f) \log(1 - \sigma(w(v_f^T v_{set}) + b)) \tag{2}$$

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 $f$ 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.

### 2.4 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 [3]. 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. (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 is 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 [3] using softmax loss. The NetVLAD layer, with $K = 8$ clusters, is initialized using k-means as in [1]. 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 (Sec. 2.3).

**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 [29]), 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 $224 \times 224$ 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


Fig. 4: **Example images of the *Celebrity Together* Dataset.** Note that only those celebrities who appear in the VGG Face 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. Additional examples of the dataset images are given in the supplementary material.

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.

## 3 'Celebrity Together' dataset

A new dataset, *Celebrity Together*, is collected and annotated. It contains images that portray multiple celebrities simultaneously (Fig. 4 shows a sample), making it ideal for testing set retrieval methods. 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 image collection and annotation procedures are explained next.

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 [19]. The straightforward strategy of [19], which involves simply searching for celebrities on 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 with multiple celebrities in them.

**Search string selection.** We use the list of 2622 celebrities from the VGG Face Dataset [19] 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 to search for. 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.

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

Table 1: **Distribution of faces / image.**

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

Table 2: **Distribution of annotations / image.**

**Image download and filtering.** The list of celebrity pairs is used to query Google Image Search and download the images, which are then filtered using the meta information to contain both queried celebrities, and a face detector [17] is used to filter out images which contain fewer than two faces. De-duplication is performed using the method of [19], and images in common with the VGG Face Dataset [19] are removed as well.

**Image annotation.** Since the same list of celebrities is used as in the VGG Face Dataset [19], we use the pre-trained Deep Face CNN [19] to aid with annotation. 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. Remaining images are annotated manually using Mechanical Turk. Full details of the annotation procedure are provided in the supplementary material. Note that not all faces in the dataset correspond to the 2622 queried celebrities, and these are labelled as "unknown" people, but are kept in the dataset as distractors; 41% of all faces are distractors. Table 1 shows the breakdown of the total number of faces appearing in each image, while Table 2 shows the same breakdown but for celebrities (i.e. people with known identities).

## 4 Experiments and results

In this section we investigate three aspects: first, in Sec. 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 Sec. 4.2. Third, the trade-off between time complexity and set retrieval quality is investigated in Sec. 4.3.

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 [19] (used for testing, e.g. for forming the *Celebrity Together* dataset) and the VGGFace2 Dataset [3] (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.

### 4.1 Stress test

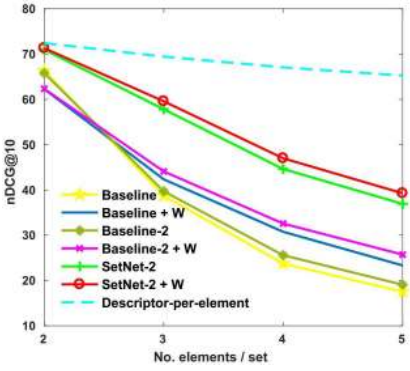
In this test, we aim to investigate how different models perform with increasing number of faces per set (image) in the test dataset. The effects of varying the number of faces per set used for training are also studied.

**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 Sec. 2.4, 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 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 Fig. 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 Sec. 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.

**Results.** From the results in Fig. 5 it is clear that *SetNet-2+W* and *SetNet-3+W* outperform all baselines. A similar trend also happens for nDCG@30; results for this are included in the supplementary material. 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 when there are more than 2 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 community as it is usually found to be very helpful [2, 11], but has also been used




(a)

| Scoring model | 2/set | 3/set | 4/set | 5/set |
|---|---|---|---|---|
| Baseline | 66.3 | 38.8 | 23.7 | 17.5 |
| Baseline-2 | 65.8 | 39.7 | 25.6 | 19.0 |
| Baseline-3 | 65.9 | 39.4 | 24.8 | 18.4 |
| SetNet-2 | 71.0 | 57.7 | 44.6 | 36.9 |
| SetNet-3 | 71.9 | 57.9 | 44.7 | 37.0 |
| Baseline + W | 62.3 | 42.3 | 30.8 | 23.3 |
| Baseline-2 + W | 62.3 | 44.1 | 32.6 | 25.7 |
| Baseline-3 + W | 62.1 | 44.0 | 32.3 | 25.6 |
| SetNet-2 + W | 71.3 | 59.5 | 47.0 | 39.3 |
| SetNet-3 + W | 71.8 | 59.8 | 47.1 | 39.3 |
| Desc-per-element | 72.4 | 69.4 | 67.1 | 65.3 |

(b)

Fig. 5: **Stress test comparison of different models.** There are 100 query sets, each with two identities. (a) nDCG@10 for different number of elements (faces) per set (image) in the test dataset. (b) Table of nDCG@10 of stress test. Columns corresponds to the four different test datasets defined by the number of elements (faces) per set.

recently to improve CNN representations [23, 26]. 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 Fig. 5b, 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.

### 4.2 Evaluating on the Celebrity Together dataset

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

**Test dataset.** The dataset is described in Sec. 3. To increase the retrieval difficulty, random 355k distractor images are sampled from the MS-Celeb-1M Dataset [6], 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 resultant test dataset are shown in Table 3. 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

| | Celebrity Together | Dist. from MS1M | Total |
|---|---|---|---|
| Images | 194k | 355k | 549k |
| Faces | 546k | 1M | 1546k |

Table 3: **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 [6].

| 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 4: **Set retrieval performance on the test set.** $Q$ is the number of identities in the query. There are 500 queries with $Q = 2$, and 500 with $Q = 3$. $N_{ex}$ is the number of available face examples for each identity. $N_{qd}$ is the number of descriptors actually used for querying.

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 (Sec. 4.1) *Baseline-2+W*, is used as the main comparison method.

**Results.** Table 4 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. Fig. 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 4 (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.

### 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 $Q$, $F$ and $N$ be the number of query

| | | Without query averaging | | | | With query averaging | | | |
|---|---|---|---|---|---|---|---|---|---|
| Scoring method | $N_r$ | nDCG@10 | nDCG@30 | Timing | Speedup | nDCG@10 | nDCG@30 | Timing | Speedup |
| Desc-per-set | - | 59.1 | 59.4 | 0.11s | 57.5× | 58.7 | 59.4 | 0.01s | 635× |
| Desc-per-set + Re. | 100 | 76.5 | 70.1 | 0.13s | 48.8× | 76.4 | 70.1 | 0.03s | 212× |
| Desc-per-set + Re. | 1000 | 84.2 | 80.0 | 0.20s | 31.8× | 84.1 | 80.0 | 0.10s | 63.5× |
| Desc-per-set + Re. | 2000 | 85.3 | 81.4 | 0.28s | 22.7× | 85.3 | 81.4 | 0.18s | 35.3× |
| Desc-per-element | - | 85.4 | 81.7 | 6.35s | - | - | - | - | - |

Table 5: **Retrieval speed *vs* quality trade-off with varied number of re-ranking images.** 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 Sec. 4.2.

identities, average number of faces per dataset image, and the number of dataset images, respectively, and let the face descriptor be $D$-dimensional. Recall that our SetNet produces a compact set representation which is also $D$-dimensional, and $D = 128$ throughout.

**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 $7\times$ faster.

**Combinations by re-ranking.** Borrowing ideas again from image retrieval [22, 25], 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 Sec. 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. In the supplementary material we also discuss a naive approach of pre-tagging the dataset with a list of identities.

**Results.** Table 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 $35\times$ faster. 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 $212\times$ 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.

## 5 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.

We have also introduced a new dataset, *Celebrity Together*, which can be used to evaluate set retrieval performance and to facilitate research on this new topic. The dataset will be released publicly.

**Acknowledgements** This work was funded by an EPSRC studentship and EPSRC Programme Grant Seebibyte EP/M013774/1.

# Bibliography


[1] Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: NetVLAD: CNN architecture for weakly supervised place recognition. In: Proc. CVPR (2016)

[2] Arandjelović, R., Zisserman, A.: All about VLAD. In: Proc. CVPR (2013)

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

[4] Chatfield, K., Lempitsky, V., Vedaldi, A., Zisserman, A.: The devil is in the details: an evaluation of recent feature encoding methods. In: Proc. BMVC. (2011)

[5] Delhumeau, J., Gosselin, P.H., Jégou, H., Pérez, P.: Revisiting the VLAD image representation. In: Proc. ACMM (2013)

[6] Guo, Y., Zhang, L., Hu, Y., He, X., Gao, J.: MS-Celeb-1M: A dataset and benchmark for large-scale face recognition. In: Proc. ECCV (2016)

[7] Hamid Rezatofighi, S., Kumar, B., Milan, A., Abbasnejad, E., Dick, A., Reid, I.: DeepSetNet: Predicting sets with deep neural networks. In: Proc. CVPR (2017)

[8] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. CVPR (2016)

[9] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proc. ICML (2015)

[10] Iscen, A., Furon, T., Gripon, V., Rabbat, M., Jégou, H.: Memory vectors for similarity search in high-dimensional spaces. IEEE Transactions on Big Data (2017)

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

[12] Jégou, H., Douze, M., Schmid, C., Pérez, P.: Aggregating local descriptors into a compact image representation. In: Proc. CVPR (2010)

[13] Jégou, H., Perronnin, F., Douze, M., Sánchez, J., Pérez, P., Schmid, C.: Aggregating local image descriptors into compact codes. IEEE PAMI (2011)

[14] Jégou, H., Zisserman, A.: Triangulation embedding and democratic aggregation for image search. In: Proc. CVPR (2014)

[15] Kondor, R., Jebara, T.: A kernel between sets of vectors. In: Proc. ICML. AAAI Press (2003)

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

[17] Mathias, M., Benenson, R., Pedersoli, M., Van Gool, L.: Face detection without bells and whistle. In: ECCV (2014)

[18] Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: Proc. CVPR. pp. 2161–2168 (2006)

[19] Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition. In: Proc. BMVC. (2015)

[20] Perronnin, F., Liu, Y., Sánchez, J., Poirier, H.: Large-scale image retrieval with compressed fisher vectors. In: Proc. CVPR (2010)

[21] Perronnin, F., Sánchez, J., Mensink, T.: Improving the Fisher kernel for large-scale image classification. In: Proc. ECCV (2010)

[22] Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: Proc. CVPR (2007)
[23] Radenović, F., Tolias, G., Chum, O.: CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples. In: Proc. ECCV (2016)
[24] Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: A unified embedding for face recognition and clustering. In: Proc. CVPR (2015)
[25] Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: Proc. ICCV. vol. 2, pp. 1470–1477 (2003)
[26] Sun, Y., Zheng, L., Deng, W., Wang, S.: SVDNet for pedestrian retrieval. In: Proc. ICCV (2017)
[27] Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deep-Face: Closing the gap to human-level performance in face verification. In: IEEE CVPR (2014)
[28] Torresani, L., Szummer, M., Fitzgibbon, A.: Efficient object category recognition using classemes. In: Proc. ECCV. pp. 776–789 (sep 2010)
[29] Vedaldi, A., Lenc, K.: MatConvNet: Convolutional neural networks for MATLAB. In: Proc. ACMM (2015)
[30] Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., Gong, Y.: Locality-constrained linear coding for image classification. In: Proc. CVPR (2010)
[31] Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., Smola, A.: Deep sets. In: NIPS. pp. 3391–3401 (2017)
[32] Zhou, X., Yu, K., Zhang, T., Huang, T.S.: Image classification using super-vector coding of local image descriptors. In: Proc. ECCV (2010)