Deep Face Recognition

Omkar M. Parkhi
omkar@robots.ox.ac.uk
Andrea Vedaldi
vedaldi@robots.ox.ac.uk
Andrew Zisserman
az@robots.ox.ac.uk

Visual Geometry Group
Department of Engineering Science
University of Oxford

Abstract

The goal of this paper is face recognition – from either a single photograph or from a set of faces tracked in a video. Recent progress in this area has been due to two factors: (i) end to end learning for the task using a convolutional neural network (CNN), and (ii) the availability of very large scale training datasets.

We make two contributions: first, we show how a very large scale dataset (2.6M images, over 2.6K people) can be assembled by a combination of automation and human in the loop, and discuss the trade off between data purity and time; second, we traverse through the complexities of deep network training and face recognition to present methods and procedures to achieve comparable state of the art results on the standard LFW and YTF face benchmarks.

1 Introduction

Convolutional Neural Networks (CNNs) have taken the computer vision community by storm, significantly improving the state of the art in many applications. One of the most important ingredients for the success of such methods is the availability of large quantities of training data. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [16] was instrumental in providing this data for the general image classification task. More recently, researchers have made datasets available for segmentation, scene classification and image segmentation [12, 33].

In the world of face recognition, however, large scale public datasets have been lacking and, largely due to this factor, most of the recent advances in the community remain restricted to Internet giants such as Facebook and Google etc. For example, the most recent face recognition method by Google [17] was trained using 200 million images and eight million unique identities. The size of this dataset is almost *three orders of magnitude* larger than any publicly available face dataset (see Table 1). Needless to say, building a dataset this large is beyond the capabilities of most international research groups, particularly in academia.

This paper has two goals. The first one is to propose a procedure to create a reasonably large face dataset whilst requiring only a limited amount of person-power for annotation. To this end we propose a method for collecting face data using knowledge sources available on the web (Section 3). We employ this procedure to build a dataset with over two million faces,

Dataset	Identities	Images
LFW	5,749	13,233
WDRef [4]	2,995	99,773
CelebFaces [25]	10,177	202,599

Dataset	Identities	Images
Ours	2,622	2.6M
FaceBook [29]	4,030	4.4M
Google [17]	8M	200M

Table 1: **Dataset comparisons:** Our dataset has the largest collection of face images outside industrial datasets by Goole, Facebook, or Baidu, which are not publicly available.

and will make this freely available to the research community. The second goal is to investigate various CNN architectures for face identification and verification, including exploring face alignment and metric learning, using the novel dataset for training (Section 4). Many recent works on face recognition have proposed numerous variants of CNN architectures for faces, and we assess some of these modelling choices in order to filter what is important from irrelevant details. The outcome is a much simpler and yet effective network architecture achieving near state-of-the-art results on all popular image and video face recognition benchmarks (Section 5 and 6). Our findings are summarised in Section 6.2.

2 Related Work

This paper focuses on face recognition in images and videos, a problem that has received significant attention in the recent past. Among the many methods proposed in the literature, we distinguish the ones that do not use deep learning, which we refer as "shallow", from ones that do, that we call "deep". Shallow methods start by extracting a representation of the face image using handcrafted local image descriptors such as SIFT, LBP, HOG [5, 13, 22, 23, 32]; then they aggregate such local descriptors into an overall face descriptor by using a pooling mechanism, for example the Fisher Vector [15, 20]. There are a large variety of such methods which cannot be described in detail here (see for example the references in [15] for an overview).

This work is concerned mainly with deep architectures for face recognition. The defining characteristic of such methods is the use of a CNN feature extractor, a learnable function obtained by composing several linear and non-linear operators. A representative system of this class of methods is *DeepFace* [29]. This method uses a deep CNN trained to classify faces using a dataset of 4 million examples spanning 4000 unique identities. It also uses a *siamese network* architecture, where the same CNN is applied to pairs of faces to obtain descriptors that are then compared using the Euclidean distance. The goal of training is to minimise the distance between congruous pairs of faces (*i.e.* portraying the same identity) and maximise the distance between incongruous pairs, a form of *metric learning*. In addition to using a very large amount of training data, DeepFace uses an ensemble of CNNs, as well as a pre-processing phase in which face images are aligned to a canonical pose using a 3D model. When introduced, DeepFace achieved the best performance on the Labelled Faces in the Wild (LFW; [8]) benchmark as well as the Youtube Faces in the Wild (YFW; [32]) benchmark. The authors later extended this work in [30], by increasing the size of the dataset by two orders of magnitude, including 10 million identities and 50 images per identity. They proposed a bootstrapping strategy to select identities to train the network and showed that the generalisation of the network can be improved by controlling the dimensionality of the fully connected layer.

The DeepFace work was extended by the DeepId series of papers by Sun *et al.* [24, 25, 26, 27], each of which incrementally but steadily increased the performance on LFW and

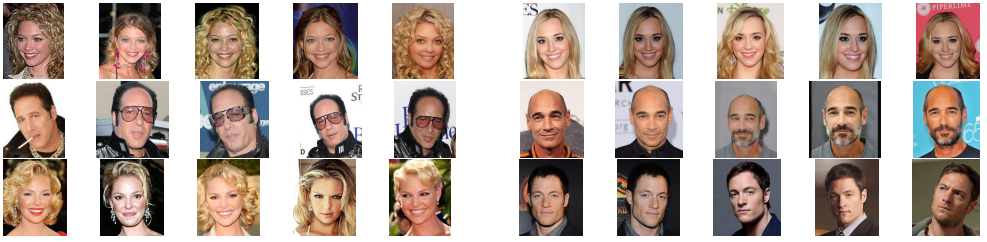

Figure 1: Example images from our dataset for six identities.

YFW. A number of new ideas were incorporated over this series of papers, including: using multiple CNNs [25], a Bayesian learning framework [4] to train a metric, multi-task learning over classification and verification [24], different CNN architectures which branch a fully connected layer after each convolution layer [26], and very deep networks inspired by [19, 28] in [27]. Compared to DeepFace, DeepID does not use 3D face alignment, but a simpler 2D affine alignment (as we do in this paper) and trains on combination of CelebFaces [25] and WDRef [4]. However, the final model in [27] is quite complicated involving around 200 CNNs.

Very recently, researchers from Google [17] used a massive dataset of 200 million face identities and 800 million image face pairs to train a CNN similar to [28] and [18]. A point of difference is in their use of a "triplet-based" loss, where a pair of two congruous (a, b) and a third incongruous face c are compared. The goal is to make a closer to b than c; in other words, differently from other metric learning approaches, comparisons are always relative to a "pivot" face. This matches more closely how the metric is used in applications, where a query face is compared to a database of other faces to find the matching ones. In training this loss is applied at multiple layers, not just the final one. This method currently achieves the best performance on LFW and YTF.

3 Dataset Collection

In this section we propose a multi-stage strategy to effectively collect a large face dataset containing hundreds of example images for thousands of unique identities (Table 1). The different stages of this process and corresponding statistics are summarised in Table 2. Individual stages are discussed in detail in the following paragraphs.

Stage 1. Bootstrapping and filtering a list of candidate identity names. The first stage in building the dataset is to obtain a list of names of candidate identities for obtaining faces. The idea is to focus on celebrities and public figures, such as actors or politicians, so that a sufficient number of distinct images are likely to be found on the web, and also to avoid any privacy issue in downloading their images. An initial list of public figures is obtained by extracting males and females, ranked by popularity, from the *Internet Movie Data Base* (IMDB) celebrity list. This list, which contains mostly actors, is intersected with all the people in the *Freebase knowledge graph* [1], which has information on about 500K different identities, resulting in a ranked lists of 2.5K males and 2.5K females. This forms a candidate list of 5K names which are known to be popular (from IMDB), and for which we have attribute information such as ethnicity, age, kinship etc. (from the knowledge graph). The total of 5K images was chosen to make the subsequent annotation process manageable for a

small annotator team.

The candidate list is then filtered to remove identities for which there are not enough distinct images, and to eliminate any overlap with standard benchmark datasets. To this end 200 images for each of the 5K names are downloaded using Google Image Search. The 200 images are then presented to human annotators (sequentially in four groups of 50) to determine which identities result in sufficient image purity. Specifically, annotators are asked to retain an identity only if the corresponding set of 200 images is roughly 90% pure. The lack of purity could be due to homonymy or image scarcity. This filtering step reduces the candidate list to 3,250 identities. Next, any names appearing in the LFW and YTF datasets are removed in order to make it possible to train on the new dataset and still evaluate fairly on those benchmarks. In this manner, a final list of 2,622 celebrity names is obtained.

Stage 2. Collecting more images for each identity. Each of the 2,622 celebrity names is queried in both Google and Bing Image Search, and then again after appending the keyword "actor" to the names. This results in four queries per name and 500 results for each, obtaining 2,000 images for each identity.

Stage 3. Improving purity with an automatic filter. The aim of this stage is to remove any erroneous faces in each set automatically using a classifier. To achieve this the top 50 images (based on Google search rank in the downloaded set) for each identity are used as positive training samples, and the top 50 images of all other identities are used as negative training samples. A one-vs-rest linear SVM is trained for each identity using the Fisher Vector Faces descriptor [15, 20]. The linear SVM for each identity is then used to rank the 2,000 downloaded images for that identity, and the top 1,000 are retained (the threshold number of 1,000 was chosen to favour high precision in the positive predictions).

Stage 4. Near duplicate removal. Exact duplicate images arising from the same image being found by two different search engines, or by copies of the same image being found at two different Internet locations, are removed. Near duplicates (e.g. images differing only in colour balance, or with text superimposed) are also removed. This is done by computing the VLAD descriptor [2, 9] for each image, clustering such descriptors within the 1,000 images for each identity using a very tight threshold, and retaining a single element per cluster.

Stage 5. Final manual filtering. At this point there are 2,622 identities and up to 1,000 images per identity. The aim of this final stage is to increase the purity (precision) of the data using human annotations. However, in order to make the annotation task less burdensome, and hence avoid high annotation costs, annotators are aided by using automatic ranking once more. This time, however, a multi-way CNN is trained to discriminate between the 2,622 face identities using the AlexNet architecture of [10]; then the softmax scores are used to rank images within each identity set by decreasing likelihood of being an inlier. In order to accelerate the work of the annotators, the ranked images of each identity are displayed in blocks of 200 and annotators are asked to validate blocks as a whole. In particular, a block is declared *good* if approximate purity is greater than 95%. The final number of good images is 982,803, of which approximately 95% are frontal and 5% profile.

Discussion. Overall, this combination of using Internet search engines, filtering data using existing face recognition methods, and limited manual curation is able to produce an accurate

Stage	Aim	Type	# of persons	# of images per person	total # images	Annotation effort	100%-EER
1	Candidate list generation	A	5,000	200	1,000,000	–	–
2	Image set expansion	M	2,622	2,000	5,244,000	4 days	–
3	Rank image sets	A	2,622	1,000	2,622,000	–	96.90
4	Near dup. removal	A	2,622	623	1,635,159	–	–
5	Final manual filtering	M	2,622	375	982,803	10 days	92.83

Table 2: **Dataset statistics after each stage of processing.** A considerable part of the acquisition process is automatically carried out. Type A and M specify whether the processing stage was carried out automatically or manually. The EER values are the performance on LFW for CNN configuration A trained on that stage of the dataset and compared using $l2$ distance.

large-scale dataset of faces labelled with their identities. The human annotation cost is quite small – the total amount of manual effort involved is only around 14 days, and only four days up to stage 4. Table 1 compares our dataset to several existing ones.

A number of design choices have been made in the process above. Here we suggest some alternatives and extensions. The Freebase source can be replaced by other similar sources such as DBPedia (Structured WikiPedia) and Google Knowledge Graph. In fact, Freebase will be shut down and replaced by Google Knowledge Graph very soon. On the image collection front, additional images can be collected from sources like Wikimedia Commons, IMDB and also from search engines like Baidu and Yandex. The removal of identities overlapping with LFW and YTF in stage 1 could be removed in order to increase the number of people available for the subsequent stages. The order of the stages could be changed to remove near duplicates before stage 2. In terms of extensions, the first stage of collection can be automated by looking at the distribution of pairwise distances between the downloaded images. An image class with high purity should exhibit a fairly unimodal distribution.

4 Network architecture and training

This section describes the CNNs used in our experiments and their training. Inspired by [19], the networks are "very deep", in the sense that they comprise a long sequence of convolutional layers. Such CNNs have recently achieved state-of-the-art performance in some of the tasks of the ImageNet ILSVRC 2014 challenge [16], as well as in many other tasks [7, 19, 28].

4.1 Learning a face classifier

Initially, the deep architectures ϕ are bootstrapped by considering the problem of recognising $N = 2,622$ unique individuals, setup as a N-ways classification problem. The CNN associates to each training image $\ell_t, t = 1,\ldots,T$ a score vector $\mathbf{x}_t = W\phi(\ell_t) + b \in \mathbb{R}^N$ by means of a final fully-connected layer containing N linear predictors $W \in \mathbb{R}^{N\times D}$, $b \in \mathbb{R}^N$, one per identity. These scores are compared to the ground-truth class identity $c_t \in \{1,\ldots,N\}$ by computing the empirical *softmax log-loss* $E(\phi) = -\sum_t \log\left(e^{\langle \mathbf{e}_{c_t}, \mathbf{x}_t\rangle} / \sum_{q=1,\ldots,N} e^{\langle \mathbf{e}_q, \mathbf{x}_t\rangle}\right)$, where $\mathbf{x}_t = \phi(\ell_t) \in \mathbb{R}^D$, and $\mathbf{e}_c$ denotes the one-hot vector of class c.

After learning, the classifier layer (W,b) can be removed and the score vectors $\phi(\ell_t)$ can be used for face identity verification using the Euclidean distance to compare them. However, the scores can be significantly improved by tuning them for verification in Euclidean space using a "triplet loss" training scheme, illustrated in the next section. While the latter is

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

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

Table 3: **Network configuration.** Details of the face CNN configuration A. The FC layers are listed as "convolution" as they are a special case of convolution (see Section 4.3). For each convolution layer, the filter size, number of filters, stride and padding are indicated.

essential to obtain a good overall performance, bootstrapping the network as a classifier, as explained in this section, was found to make training significantly easier and faster.

This is a kind of metric learning here. Minimise the 'similarity' for different people, and maximise if photos of same person.

vv

4.2 Learning a face embedding using a triplet loss

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

Our triplet-loss training scheme is similar in spirit to that of [17]. The output $\phi(\ell_t) \in \mathbb{R}^D$ of the CNN, pre-trained as explained in Section 4.1, is l^2-normalised and projected to a $L \ll D$ dimensional space using an affine projection $\mathbf{x}_t = W'\phi(\ell_t)/\|\phi(\ell_t)\|_2$, $W' \in \mathbb{R}^{L\times D}$. While this formula is similar to the linear predictor learned above, there are two key differences. The first one is that $L \neq D$ is not equal to the number of class identities, but it is the (arbitrary) size of the descriptor embedding (we set $L = 1,024$). The second one is that the projection W' is trained to minimise the empirical *triplet loss*

$$E(W') = \sum_{(a,p,n)\in T} \max\{0, \alpha - \|\mathbf{x}_a - \mathbf{x}_n\|_2^2 + \|\mathbf{x}_a - \mathbf{x}_p\|_2^2\}, \quad \mathbf{x}_i = W' \frac{\phi(\ell_i)}{\|\phi(\ell_i)\|_2}. \tag{1}$$

So this triplet loss is minimised by changing W'.

Note that, differently from the previous section, there is no bias being learned here as the differences in (1) would cancel it. Here $\alpha \geq 0$ is a fixed scalar representing a *learning margin* and T is a collection of *training triplets*. A triplet (a, p, n) contains an *anchor* face image a as well as a positive $p \neq a$ and negative n examples of the anchor's identity. The projection W' is learned on target datasets such as LFW and YTF honouring their guidelines. The construction of the triplet training T set is discussed in Section 4.4.

4.3 Architecture

We consider three architectures based on the A, B, and D architectures of [19]. The CNN architecture A is given in full detail in Table 3. It comprises 11 blocks, each containing a linear operator followed by one or more non-linearities such as ReLU and max pooling. The first eight such blocks are said to be convolutional as the linear operator is a bank of linear filters (linear convolution). The last three blocks are instead called Fully Connected (FC); they are the same as a convolutional layer, but the size of the filters matches the size of the

input data, such that each filter "senses" data from the entire image. All the convolution layers are followed by a rectification layer (ReLU) as in [10]; however, differently from [10] and similarly to [19], they do not include the Local Response Normalisation operator. The first two FC layers output are 4,096 dimensional and the last FC layer has either $N = 2{,}622$ or $L = 1{,}024$ dimensions, depending upon the loss functions used for optimisation, either N-way class prediction (Section 4.1 or L-dimensional metric embedding (Section 4.2). In the first case, the resulting vector is passed to a softmax layer to compute the class posterior probabilities. Networks B and D are similar to A but include 2 and 5 additional convolution layers respectively.

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

4.4 Training

Learning the N-way face classifier (Section 4.1) follows the steps of [10] with the modifications suggested by [19]. The goal is to find the parameters of the network that minimise the average prediction log-loss after the softmax layer.

We describe the procedure for the CNN A configuration first, and then the variants for the B and D configurations. Optimisation is by stochastic gradient descent using mini-batches of 64 samples and momentum coefficient of 0.9 [11]. The model is regularised using dropout and weight decay; the coefficient of the latter was set to 5×10^{-4}, whereas dropout was applied after the two FC layers with a rate of 0.5. The learning rate was initially set to 10^{-2} and then decreased by factor of 10 when the validation set accuracy stopped increasing. Overall, the model was trained using three decreasing learning rates.

The weights of the filters in the CNN were initialised by random sampling from a Gaussian distribution with zero mean and 10^{-2} standard deviation. Biases were initialised to zero. The training images were rescaled such that the smaller of width and height was equal to 256. During training, the network is fed with random 224×224 pixel patches cropped from these images (where crops change every time an image is sampled). The data was further augmented by flipping the image left to right with 50% probability; however, we did not perform any colour channel augmentation as described in [19] and [10].

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

i.e. W'? For learning the embeddings using triplet loss (Section 4.2), the nework is frozen except the last fully-connecte layer implementing the discriminative projection. This layer is then learnt for 10 epochs using SGD with a fixed learning rate of 0.25. An epoch here contains all the possible positive pairs (a, p), where image a is considered the anchor and p its paired positive example. Choosing good triplets is crucial and should strike a balance between selecting informative (*i.e.* challenging) examples and swamping training with examples that are too hard. This is achieve by extending each pair (a, p) to a triplet (a, p, n) by sampling the image n at random, but only between the ones that violate the triplet loss margin. The latter is a form of hard-negative mining, but it is not as aggressive (and much cheaper) than choosing the *maximally violating* example, as often done in structured output learning.

At test time, the embedded descriptors $W'\phi(\ell_t)$ are compared in Euclidean distance for the purpose of face verification. In verification the goal is to tell wether two face images

ℓ_1 and ℓ_2 have the same identity or not; this is obtained by testing whether the distance $\|W'\phi(\ell_1) - W'\phi(\ell_2)\|_2$ between embedded descriptors is smaller than a threshold τ. This threshold is not provided by the training procedure outlined above and is instead learned separately to maximise the *verification accuracy* (Acc.– or rate of correctly classified pairs) on suitable validation data.

How to verify some person's identity. The acceptance threshold is

5 Datasets and evaluation protocols

In order to allow for a direct comparison to previous work, while our CNNs are trained on the new dataset of Section 3, evaluation is performed on existing benchmark datasets (which, by construction, contain different identities from our dataset).

The first dataset is **Labeled Faces in the Wild dataset (LFW) [18]**. It contains 13,233 images with 5,749 identities, and is the standard benchmark for automatic face verification. We follow the standard evaluation protocol defined for the "unrestricted setting" using data external to LFW for training, which in our case is the new face dataset. In addition to the verification accuracy Acc., we use the *Equal Error Rate* (EER) as an evaluation metric, defined as the error rate at the ROC operating point where the false positive and false negative rates are equal. The advantage on Acc. is that this measure is independent on the distance threshold τ.

The second dataset is **YouTube Faces (YTF) [32]**. It contains 3,425 videos of 1,595 people collected from YouTube, with an average of 2 videos per identity, and is a standard benchmark for face verification in video. Again, we follow the standard evaluation protocol defined for the "unrestricted setting", and report EER.

6 Experiments and results

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

Implementation details. Our implementation is based on the MATLAB toolbox MatConvNet [31] linked against the NVIDIA CuDNN libraries to accelerate training. All our experiments were carried on NVIDIA Titan Black GPUs with 6GB of onboard memory, using four GPUs together. This is important due to the very significant memory footprint (and high complexity) of the very deep networks.

The CNN $\phi(\ell_t)$ contains all but the linear class predictor and softmax layers, outputting $D = 4{,}096$ descriptor vectors. Given a face image ℓ, four 224×224 pixel patches are cropped from the four corners and the centre with horizontal flip (i.e. 10 in total, as in [3]), and the feature vector from these are averaged. To enable multi-scale testing, the face is first scaled to three different sizes of 256, 384 and 512 pixels, and the cropping procedure repeated for each of these [19]. The resulting descriptor for the face is an average over all of these feature vectors.

Faces are detected using the method described in [14]. If face alignment is used, then facial landmarks are computed using the method of [6] and a 2D similarity transformation is applied to map the face to a canonical position.

For the YTF videos, K face descriptors are obtained for each video by ordering the faces by their facial landmark confidence score, and selecting the top K. Frontal faces are

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

Table 4: **Performance evaluation on LFW, unrestricted setting**. Training on the full dataset (F, stage 3), leads to a better performance than training on the curated dataset (C, stage 5). 2D alignment at the test time slightly improves the performance. Learning embedding for verification significantly boosts the performance. All results are obtained using $l2$ distance measure between the test samples.

2D aligned, but no alignment is used for profiles. Finally, the video is represented by the average of the K face descriptors.

6.1 Component analysis

This section evaluates the effect of different components of the system when training a network for face verification as in Section 4.4 and evaluating it on the LFW data. Table 4 summarises the results.

Dataset curation: First we analyse the curation effort for its effect on the performance of the network. We use dataset snapshots at stages 3 and 5 (Section 3), i.e. before and after curation, and test them using configuration A. The reason for selecting this configurations is that the networks can be trained from scratch. As can be seen, performance before curation is better (Table 4 rows 1 and 2). There are probably two reasons for this: first, it is better to have more data, even with label noise; and second, a more subtle point, some of the hard positives present in the stage 3 data get removed as a side effect of the curation process, and so the stage 5 data training does not benefit from these. **Alignment:** As can be seen from Table 4 rows 2 and 3, using 2D alignment on test images does improve the performance, but performing 2D alignment on the training data does not provide an additional boost – see Table 4 rows 4 and 5. **Architecture:** Next we vary the architecture of the network. We observe a slight boost in performance from configuration A to B (Table 4 rows 3,4) while configuration D fails to improve the results over configuration B (Table 4 rows 4,6). There are several possible reasons for this: the number of parameters in config. D is much more than B, due to the greater number of convolution layers. Also since network D is trained using fine-tuning of the new layers, setting parameters like learning rate and momentum becomes critical. Training the network from scratch is also an option which needs to be investigated in the future. **Triplet-loss embedding:** Learning a discriminative metric by minimising the triplet loss of Section 4.2 further improves performance by 1.8% (Table 4 row 7 vs. row 4). Note that this amounts to reducing the error rate by 68%.

6.2 Comparison with the state-of-the-art

LFW: Table 5 compares our results with the best results on LFW dataset, and also shows these as ROC curves. It can be observed that we achieve comparable results to the state of the art using much less data and a much simpler network architecture. **YTF:** Table 6 shows

No.	Method	Images	Networks	Acc.
1	Fisher Vector Faces [21]	-	-	93.10
2	DeepFace [29]	4M	3	97.35
3	Fusion [30]	500M	5	98.37
4	DeepID-2,3		200	99.47
5	FaceNet [17]	200M	1	98.87
6	FaceNet [17] + Alignment	200M	1	99.63
7	Ours	2.6M	1	98.95

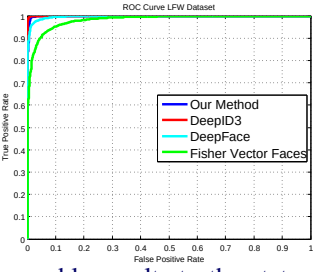

Table 5: **LFW unrestricted setting.** Left: we achieve comparable results to the state of the art whilst requiring less data (than DeepFace and FaceNet) and using a simpler network architecture (than DeepID-2,3). Note, DeepID3 results are for the test set with label errors corrected – which has not been done by any other method. Right: ROC curves.

No.	Method	Images	Networks	100%- EER	Acc.
1	Video Fisher Vector Faces [15]	-	-	87.7	83.8
2	DeepFace [29]	4M	1	91.4	91.4
3	DeepID-2,2+,3		200	-	93.2
4	FaceNet [17] + Alignment	200M	1	-	95.1
5	Ours ($K = 100$)	2.6M	1	92.8	91.6
6	Ours ($K = 100$) + Embedding learning	2.6M	1	97.4	97.3

Table 6: **Results on the Youtube Faces Dataset, unrestricted setting**. The value of K indicates the number of faces used to represent each video.

the performance on the YTF dataset. We achieve the state of the art performance using our triplet loss embedding method.

Conclusions

In this work we have made two contributions: first, we have designed a procedure that is able to assemble a large scale dataset, with small label noise, whilst minimising the amount of manual annotation involved. One of the key ideas was to use weaker classifiers to rank the data presented to the annotators. This procedure has been developed for faces, but is evidently suitable for other object classes as well as fine grained tasks. The second contribution was to show that a deep CNN, without any embellishments but with appropriate training, can achieve results comparable to the state of the art. Again, this is a conclusion that may be applicable to many other tasks.

Acknowledgements. This research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract number 2014-14071600010. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright annotation thereon.
Authors would like thank Yujie Zhong for helping with the near duplicate removal stage of the dataset building.

References

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

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

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

[4] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun. Bayesian face revisited: A joint formulation. In *Proc. ECCV*, pages 566–579, 2012.

[5] R. G. Cinbis, J. J. Verbeek, and C. Schmid. Unsupervised metric learning for face identification in TV video. In *Proc. ICCV*, pages 1559–1566, 2011.

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

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

[8] 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.

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

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

[11] 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.

[12] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[13] C. Lu and X. Tang. Surpassing human-level face verification performance on lfw with gaussianface. *AAAI*, 2015.

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

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

[16] 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.

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

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

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

[20] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman. Fisher Vector Faces in the Wild. In *Proc. BMVC.*, 2013.

[21] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *IEEE PAMI*, 2014.

[22] J. Sivic, M. Everingham, and A. Zisserman. Person spotting: Video shot retrieval for face sets. In *Proc. CIVR*, 2005.

[23] J. Sivic, M. Everingham, and A. Zisserman. "Who are you?" – learning person specific classifiers from video. In *Proc. CVPR*, 2009.

[24] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. *NIPS*, 2014.

[25] Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10,000 classes. In *Proc. CVPR*, 2014.

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

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

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

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

[30] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Web-scale training for face identification. In *Proc. CVPR*, 2015.

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

[32] L. Wolf, Tal. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *Proc. CVPR*, 2011.

[33] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. *NIPS*, 2014.