International Journal of Computer Vision manuscript No.
VISI-D-18-00137

Face-Specific Data Augmentation for Unconstrained Face Recognition

Iacopo Masi* · **Anh Tuấn Trần*** · **Tal Hassner*** · **Gozde Sahin** · **Gérard Medioni**

Received: February 22, 2018 / Accepted: March 14, 2019

Abstract We identify two issues as key to developing effective face recognition systems: maximizing the appearance variations of *training* images and minimizing appearance variations in *test* images. The former is required to train the system for whatever appearance variations it will ultimately encounter and is often addressed by collecting massive training sets with millions of face images. The latter involves various forms of appearance normalization for removing distracting nuisance factors at test time and making test faces easier to compare. We describe novel, efficient *face-specific* data augmentation techniques and show them to be ideally suited for *both* purposes. By using knowledge of faces, their 3D shapes, and appearances, we show the following: (a) We can artificially enrich training data for face recognition with face-specific appearance variations. (b) This synthetic training data can be efficiently produced online, thereby reducing the massive storage requirements of large-scale training sets and simplifying training for many appearance variations. Finally, (c) The same, fast data augmentation techniques can be applied at test time to reduce appearance variations and improve face representations. Together, with additional technical novelties, we describe a highly effective face recognition pipeline which, at the time of submission, obtains state-of-the-art results across multiple benchmarks.

* Denotes equal contribution.

I. Masi is with the Information Sciences Institute (ISI), USC, CA, USA E-mail: iacopo@isi.edu.
A. Tran, G. Sahin, G. Medioni are with Institute for Robotics and Intelligent Systems, USC, CA, USA E-mail: anhttran, gsahin, medioni @usc.edu
T. Hassner is with the Open University of Israel, Israel. E-mail: talhassner@gmail.com
This is a post-peer-review, pre-copyedit version of an article published in International Journal of Computer Vision. The final authenticated version is available online at dx.doi.org/10.1007/s11263-019-01178-0

Portions of this paper were previously published by Masi et al (2016b, 2017).

Keywords Face Recognition · Deep Learning · Data Augmentation

1 Introduction

Deep convolutional neural networks (CNN) have had a profound impact on the face recognition capabilities of machine vision systems. Although it is not entirely clear how CNNs work so well, these capabilities can at least partially be attributed to their complexity. Unlike simpler models previously used by older classification and recognition systems, CNNs typically consist of millions of trainable parameters combined in highly nonlinear relationships. Together, these parameters allow CNN systems to learn the complex decision rules required to discriminate between different faces of sometimes very similar people.

Fully realizing the potentials of CNN-based systems for face recognition requires that their millions of parameters are optimally assigned. To set their values, training is performed using labeled examples; here, face images with associated identity labels. Of course, the performances of trained CNN systems will vary according to the nature of the data sets used to train them. This relationship between performance and training data naturally raises the question: what is a good data set for training a face recognition CNN? Very often, the answer to this question seems to be: the bigger the data set, the better.

Realizing that the face recognition abilities of CNNs will improve by increasing their training data, many have focused efforts on harvesting and labeling large training sets of face images. Taigman et al (2014) trained a standard CNN on 4.4 million labeled Facebook faces and achieved what was, at the time, state-of-the-art performance on the Labeled

Faces in the Wild (LFW) benchmark (Huang et al, 2007). Some time later, Parkhi et al (2015) proposed the VGG-Face representation which they trained on 2.6 million faces, and Face++ proposed its Megvii System (Zhou et al, 2015) which was trained on 5 million faces. All of these are pale in comparison to the Google FaceNet (Schroff et al, 2015) which used 200 million labeled faces for training.

These data collection efforts undoubtedly helped improve face recognition accuracy. Ideally, however, effective training sets must reflect ample inter- and intra-subject appearance variations in order for a network to learn what makes faces similar or not. As we show in Fig. 1, existing sets are limited in the intra-subject appearance variations they provide (Masi et al, 2018). Moreover, collecting *more images* does not guarantee obtaining *more appearance variations*, as demonstrated in Fig. 2 for head yaw angles.

A first key question discussed in this paper is therefore:

Question 1: How can we effectively and efficiently increase the appearance variations of our face image training set?

Ostensibly, by answering this question and training a CNN on a sufficiently rich data set, the network would be robust to such appearance variations. In practice, even well-trained networks can be sensitive to changing appearances at test time. One way to mitigate these effects is to normalize the appearances of test faces, suppressing variations due to nuisance factors and making them easier to compare. The second question we address is therefore:

Question 2: How can we reduce nuisance appearance variations of face images at test time?

There is an obvious symmetry to these two questions: Where the first seeks ways of increasing the variations of appearances during training, the second seeks to reduce them at testing. We show that beyond this symmetry, both questions also share a possible solution—that is, by using the same face-specific data augmentation to answer them both.

Specifically, we offer the following contributions:

- **Fast rendering.** We explain why, under certain circumstances, rendering novel views of faces appearing in single images can be done at the same computational cost as standard 2D image warping.
- **Online augmentation in training.** We use our fast rendering during training to enrich our training set, on the fly, with intra-subject appearance variations, thus effectively training our CNN on a much larger training set.
- **Novel recognition pipeline.** We describe a novel face recognition pipeline that uses our fast rendering to produce multiple, well-aligned versions of input test images, thereby reducing differences due to pose in test face appearances. We further show that by pooling the CNN features from these synthetic images, we obtain a highly robust face image representation.

Beyond these technical contributions, we further describe a novel training set which we use to increase the benefits of our contributions:

- **The COW face set.** We offer a processed combination of the publicly available data sets: MS-Celeb-1M (Guo et al, 2016), Oxford VGGFace (Parkhi et al, 2015), and CASIA WebFace (Yi et al, 2014). In total, our set consists of ∼4 million images which we further increase to far greater numbers using our augmentation techniques.

We demonstrate the effects that different training set sizes have on accuracy—in particular, training on the CASIA WebFace collection (Yi et al, 2014) with its 495K faces and our combined COW dataset, with and without augmentations. Note that augmentations do not change the numbers of subjects available for training but rather increase their intra-class appearance variations (see Fig. 1 for more details).

Finally, we test our approach extensively, showing ablation studies and reporting results on the Labeled Faces in the Wild (LFW), YouTube Faces (YTF) benchmarks, and the recent IJB-A and IJB-B benchmarks. These results show that our fast, face-specific augmentations can often replace labor-intensive and storage-demanding harvesting and labeling of huge training sets. We note that code, CNN models, and data used in this paper will be publicly available on our project webpage.[1]

2 Related work

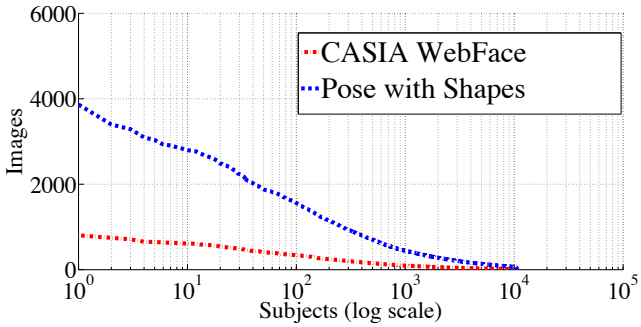
Face recognition. Face recognition is one of the central problems in computer vision and, as such, work on this problem is extensive. As with many other computer vision problems, face recognition performances sky-rocketed with the introduction of deep learning techniques, and in particular, CNNs. Though CNNs have been used for face recognition as far back as Lawrence et al (1997), only when massive amounts of data became available did their performance soar. This was originally demonstrated by the Facebook DeepFace system of Taigman et al (2014) which used an architecture not unlike the one used by Lawrence et al (1997)—but with over 4 million images used for training, they obtained far more impressive results.

Since then, CNN-based recognition systems frequently break performance barriers with some notable examples—including the Deep-ID 1-3 systems of Sun et al (2014b,a, 2015). They, and many others since, developed and trained their systems using far fewer training images at the cost of somewhat more elaborate network architectures.

[1] Available: www.openu.ac.il/home/hassner/projects/augmented_faces

Dataset	#ID	#Img	#Img/#ID
Facebook (Taigman et al, 2014)*	4,030	4.4M	1K
Google (Schroff et al, 2015)*	8M	200M	25
CASIA WebFace (Yi et al, 2014)	10,575	494,414	46
VGGFace (Parkhi et al, 2015)**	2,622	2.6M	1K
MS-Celeb-1M (Guo et al, 2016)**	100K	10M	100
MegaFace (Kemelmacher-Shlizerman et al, 2016)	690,572	1.02M	1.5
UMDFaces (Bansal et al, 2017)	8,277	367,888	45
Aug. pose+shape (Masi et al, 2016b)	10,575	1,977,656	187
Noisy COW	68,906	5,242,931	76
COW	62,955	4,069,066	64
Aug. COW, pose+shape+qual.	62,955	18,634,779	296

(a) Face set statistics



(b) Images for subjects

Fig. 1: **Face data set statistics.** (a) Comparison of our augmented dataset with other face datasets along with the average number of images per subject. (b) Our improved distribution of per-subject image numbers, obtained by augmentation (Aug.), avoids the long-tail effect of the CASIA set (Yi et al, 2014) (also shown in (a) (Masi et al, 2016b)).* Not publicly available. ** See important notes in Sec. 7 regarding the actual numbers of images and subjects used in our tests.

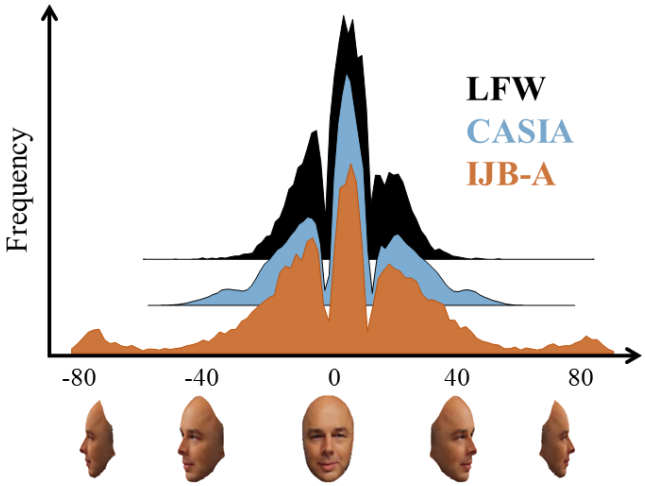


Fig. 2: **Yaw angle frequencies.** Despite being much larger than LFW, CASIA does not offer significantly more head yaw angle variations. Regardless of absolute size, both CASIA and LFW do not offer examples of the yaw angle variations in the IJB-A test set.

Though novel network architecture designs can lead to better performance, further improvement can be achieved by collecting more training data. This has been demonstrated by Google FaceNet (Schroff et al, 2015) which was developed and trained on 200 million images. Beside improving results, they also offered a fascinating analysis of the consequences of adding more data: apparently, there is a significant diminishing returns effect when training with increasing image numbers. Thus, the leap in performance obtained by going from thousands of images to millions is substantial, but increasing the numbers further provides smaller and smaller benefits. One way to explain this is that the data used by Google Facenet and others suffers from a *long tail* phenomenon (Yi et al, 2014) where most subjects in these huge datasets have too few images available for the network to learn intra-subject appearance variations.

Evaluation data sets. These methods were all evaluated on the LFW dataset, which was the *de facto standard* for measuring face recognition performances. Many of these LFW results, however, already reached near-perfect accuracy, suggesting that LFW is no longer a challenging benchmark for today's CNN-based systems. Another relevant benchmark, also frequently used to report performances, is the YouTube Faces (YTF) set (Wolf et al, 2011b). It contains unconstrained face videos rather than images, but it too is quickly being saturated.

Recently, IJB-A (Klare et al, 2015) and IJB-B (Whitelam et al, 2017) were released in order to once again push machine face recognition capabilities. They offer several novelties compared to existing sets, including *template*, rather than image-based recognition and a mix of both images and videos. With many profile views, low-quality images, and occluded faces, both are also far tougher than previous collections. Not surprisingly, dominating performance on both benchmarks are CNN methods (Chen et al, 2016).

Data augmentation. Data augmentation techniques are label-preserving transformations typically applied to training images Chatfield et al (2014). Such methods are known to improve the accuracy of CNN-based methods and prevent overfitting (Chatfield et al, 2014). Popular augmentation methods include simple, geometric transformations such as *oversampling* (multiple image translations by cropping at different offsets) (Krizhevsky et al, 2012; Levi and Hassner, 2015), *mirroring* (horizontal flipping) (Chatfield et al, 2014; Yang and Patras, 2015), rotation (Xie and Tu, 2015), and various photometric transformations (Krizhevsky et al,

2012; Simonyan and Zisserman, 2015; Eigen and Fergus, 2015).

Recently, some researchers have started applying more domain-specific augmentation techniques. One example is the work of McLaughlin et al (2015) which proposes to augment training data for a person re-identification network by replacing image backgrounds. We propose a far more elaborate, yet easily accessible means of data augmentation. Xie et al (2015) proposed a so-called task-specific data augmentation method. They, as well as Xu et al (2015), do not synthesize new data as we propose to do here, but rather offer additional means of collecting images from the Internet to improve learning in fine-grained recognition tasks. This is, of course, very different from our own approach.

View synthesis for face recognition. The idea that face images can be synthetically generated in order to aid face recognition systems is not new. To our knowledge, it was originally proposed by Hassner (2013) and then effectively used by Taigman et al (2014) and Hassner et al (2015). Contrary to us, they all produced frontal faces which are presumably better aligned and easier to compare. They did not use other transformations to synthesize images (e.g., other viewpoints). More importantly, their images were used to reduce appearance variability. Like them, we reduce test time variability, but we do so in a more effective manner.

Mokhayeri et al (2018) also used face synthesis to augment the gallery set when only a single, reference image is available. Finally, recent efforts used 3D rendering techniques to inflate the training set. Masi et al (2016b) applied this method to synthesize novel poses, expressions, and different 3D face shapes. Crosswhite et al (2017) improved their approach by increasing the size of the VGGFace set of Parkhi et al (2015) with novel 3D poses and illumination changes (Crispell et al, 2016).

GAN for face synthesis. Recently, generative adversarial nets (GAN) (Goodfellow et al, 2014) were proposed as a novel means for data augmentation and synthesis. GANs were shown to synthesize novel samples from the underlying appearance distribution through a generative model. This ability proved to be particularly effective for structured object classes such as faces.

The GAN's generative model is trained with an adversarial loss and learns to produce novel, realistic samples which *fool* a discriminator network. This approach was used in Zheng et al (2017) to inflate a limited training set for re-identification. Although this approach showed promising results, regular GANs, such as the one proposed by Zheng et al (2017), can generate novel images but not new subjects. It further offers no explicit control over the identity of the generated sample.

For these reasons, this approach does not meet the standard definition of data augmentation (Chatfield et al, 2014). Consequently, the approach softly assigns generated samples uniformly to all training subjects when training for classification rather than assigning the synthesized samples to specific subjects. Thus, this GAN-based approach acts more as a regularizer in the loss function rather than a data augmentation technique. These observations can be appreciated qualitatively by visually inspecting the samples generated by Zheng et al (2017). The synthetic individuals they produce appear to be interpolations between existing subjects rather than novel samples.

Another promising GAN-based approach was proposed by Zhao et al (2017). They proposed to render faces similarly to us, but they then apply a GAN to improve the realism of the generated faces. Since the generator can modify a synthetic profile image to match the real, mainly frontal, face distribution in unexpected ways, the system trains with multiple loss functions in order to preserve the identity and pose of the refined face. The method reached state-of-the-art results on the IJB-A benchmark, though their method was separately tuned ten times, one for each split of the dataset, and so data augmentation was not entirely responsible for the domain shift between the train and test sets.

Discussion: GAN-based vs. graphics-based synthesis.

Classic data augmentation techniques apply procedural transformations which use domain knowledge. These methods leverage domain knowledge acquired over several decades of work on computer vision and graphics. Face images in particular offer an abundance of such domain knowledge, along with effective, well-established methods for processing and augmenting face images. By leveraging this domain knowledge, these augmentation techniques can be applied rather easily, without requiring additional training steps. Beyond simplifying their adoption, graphics-based methods inject new information into the training set, which is not reflected in the training distribution.

A drawback of these graphics-based augmentation methods, however, is the difficulty of ensuring realistic-looking synthetic images. GAN-based methods, on the other hand, learn from training data to produce images which are faithful to the underlying distribution. These images can therefore be realistic (Zheng et al, 2017). We note, however, that using these methods can result in loss of subject-specific appearances, if proper losses are not added to the training process (Zhao et al, 2017). In addition, GAN-based methods cannot be used to generate new views without retraining. Finally, GAN-based synthesis uses deep networks at test time: a forward pass through the generator network. Unlike our rendering method, GAN systems therefore require far more computation for a forward pass through their generator networks, and the use of GPU hardware.

3 Overview of our face-specific augmentations

We augment an existing seed face dataset, enriching it with substantially more per-subject appearance variations, yet without changing subject labels or losing meaningful information. Specifically, we propose efficient and effective methods for generating (synthesizing) new face images with the following face-specific appearance variations:

1. **Pose:** Simulating face image appearances across unseen (novel) 3D viewpoints.
2. **Shape:** Producing facial appearances with different underlying 3D generic face shapes.

As we explain in Sec. 5.3, in practice, we use these augmentation methods in concert with standard, generic augmentation techniques such as 2D transformations (i.e., in-plane face alignment) and image quality adjustments obtained by synthetic noise.

3.1 The importance of generic 3D face shapes

Our augmentation methods leverage the extremely efficient rendering methods described in Sec. 4. This efficiency is partly achieved by making extensive use of *generic* 3D face shapes as proxies for rendering faces to new views. We emphasize that fixed, generic shapes are used instead of attempting to fit different 3D face shapes to different faces or deforming the 3D face shapes to account for facial expressions. This approach was originally proposed by Hassner et al (2015) for the purpose of face *frontalization*. In fact, augmentation technique (1) above can be considered an extension to multiple views of their frontalization method. Conceptually, however, we use this approach very differently than they do.

It is worth noting that our use of generic face shapes contradicts previous claims regarding the importance of 3D face shape estimation for face alignment and recognition (Taigman et al, 2014). Contrary to their claims—that accurate 3D face shapes are important for preserving subject-specific appearances when generating novel views—we *ignore subject-specific 3D shapes* and instead render all faces using the same, small, fixed collection of generic 3D face shapes. As evident qualitatively in, e.g., Fig. 3, and quantitatively in our recognition results, although the use of generic shapes introduces appearance variations, barring extreme view changes, these typically have a limited effect on perceived identity and instead provide additional sources of appearance nuisances that the network should learn to ignore.

3.2 Pose variations

Given a face image, in order to generate unseen viewpoints we use a technique similar to the one proposed by Hassner et al (2015) for frontalization. We extend their work to multiple viewpoints, change the manner in which we estimate the viewpoint for the face in the input image, and expedite their rendering process.

For input image, $\mathbf{I}$, we compute the six degrees of freedom (6DoF), 3D viewpoint of the face using the FacePoseNet (FPN) of Chang et al (2017). We then use the estimated viewpoint to render the input face to novel viewpoints at fixed yaw angles: $\theta = \{0^\circ, \pm 22^\circ, \pm 40^\circ, \pm 55^\circ, \pm 75^\circ\}$. We detail this process in Sec. 4.

3.3 3D shape variations

Rather than using a single generic 3D shape or estimating it from the image directly, we instead extend our rendering by using multiple generic 3D faces. In particular, we use a set of generic 3D shapes $\mathscr{S} = \{\mathbf{S}_j\}_{j=1}^{10}$ and simply repeat the viewpoint synthesis procedure using the same efficient rendering techniques described in Sec. 4, with these ten shapes instead of only a single 3D shape.

We used 3D generic shapes from the publicly available Basel 3D face set (Paysan et al, 2009). It includes ten high-quality 3D face scans captured from different people with different face shapes and varying in gender, age, and weight. The models are further well aligned to each other, and so 3D pose estimation of an input face image is only required once; the estimated pose can be used for all ten 3D shapes as they share the same coordinate frame. In our work, we further take advantage of the correspondences between shapes when modifying these models to allow rendering full heads and backgrounds. Details on these modifications are provided in Appendix A.

3.4 Qualitative case studies

Qualitative examples of this process are provided in Fig. 3. The figure illustrates the ten generic models we use in our work (top, rendered in grayscale). We show several input images, selected to represent a variety of ethnic groups, genders, ages, and facial expressions. Alongside each input image we show our synthesized novel images, with columns representing the different 3D face shapes and rows representing different viewpoints. Subjects in these images typically remain identifiable despite the use of varying, sometimes very different, 3D shapes used for rendering. Still, each image is slightly but noticeably different from the rest, thereby introducing appearance variations to this subject's image set.

Though the use of generic 3D face shapes may be counterintuitive, and indeed contradicts claims made in previous work, the rendered views remain identifiable, and the choice

Fig. 3: **Effect of 3D shape on rendered faces for multiple poses.** Top: The ten generic 3D face shapes used for rendering. Bottom: Each input image on the left is rendered at different poses (rows) and with different 3D shapes (columns). Different shapes induce subtle appearance variations yet do not change the perceived identity of the face in the image, irrespective of the gender, expression, occlusions, age, and ethnicity of the input subject. A higher resolution version of this picture is available in the supplementary material.

of 3D shape for the rendering has a limited effect on the perceived identity. This is especially remarkable, considering the neutral expression of the 3D face shapes compared to the varying expressions of the input faces. The perceived ethnicity and age are also mostly unaffected by the choice of 3D face shape, even if it is clearly mismatched with the face image. A potential consequence is that for unconstrained face images, estimating 3D face shapes is unnecessary if the goal is to generate new views; texture encodes sufficient information for this purpose. The observation is consistent with the conclusions of (Hassner et al, 2015).

4 Face rendering at (2D) warp speed

Given a face image, we render novel views of this face from different poses (Sec. 3.2) and use different generic 3D shapes (Sec. 3.3). Existing rendering techniques, in particular methods for synthesizing new views of faces from single images, involve two standard and well-understood steps sometimes referred to in computer graphics literature as *texture mapping* and *ray casting / rasterization*.

In light of the importance of rendering as a key step in computer graphics applications, tremendous efforts were dedicated to expediting state-of-the-art rendering engines to fractions of a second, even for complex 3D scenes. Specialized computer hardware—the graphical processing unit (GPU)—was also developed for this purpose. More details on these techniques can be found in standard computer graphics textbooks (Hughes et al, 2014).

Our work is tangential to these efforts: we show that by assuming one (or a few) generic 3D shapes and fixed desired output viewpoints, a great deal of the effort required to render new facial views can be performed at preprocessing—thereby substantially reducing the online effort required to generate novel views and the complexity of the system required for rendering.

4.1 Face pose estimation and texture mapping

Texture mapping of an image $\mathbf{I}$ (i.e., a training or test image of a face viewed in unconstrained settings) onto a 3D surface $\mathbf{S} \subset \mathbb{R}^3$ is the process of assigning every 3D surface position $\mathbf{P} = (X,Y,Z) \in \mathbf{S}$ with a location $\mathbf{q} = (u,v)$ in the image (where image coordinates are often normalized to the range of $u,v \in [0,1]$). For our purposes, $\mathbf{S}$ is a 3D face shape which is assumed to be a predetermined, generic 3D face.

We assume that a bounding box around the face is provided either by the data set or by the use of a face detector such as the one proposed by Yang and Nevatia (2016) and used by our method. Texture mapping is performed by estimating the 3D pose (viewpoint) of the face in the bounding box, in the coordinate frame of the 3D face shape.

Rendering is typically agnostic to the particular method used to estimate the 3D pose. In the work described by Hassner (2013); Hassner et al (2015); Masi et al (2016b,a, 2017); Taigman et al (2014), pose estimation was performed by detecting facial landmarks using landmark detectors such as the supervised descent method of Xiong and De la Torre (2013). In our implementation, we instead use the recent FacePoseNet (FPN) (Chang et al, 2017). FPN bypasses landmark detection by regressing a 6DoF 3D face transformation $\left[\mathbf{R}^I \; \mathbf{t}^I\right]$ directly from image intensities by using a CNN. It was shown to surpass the accuracy of landmark detectors when used for alignment in face recognition systems.

We assume a fixed intrinsic camera matrix $\mathbf{K}^I$, estimating only the rotation and translation matrices $\mathbf{R}^I$ and $\mathbf{t}^I$ in the 3D model's coordinate frame. We thus obtain a perspective camera model mapping the 3D face shape $\mathbf{S}$ to the input image so that $\mathbf{q}_i \sim \mathbf{M}^I \, \mathbf{P}_i$ where $\mathbf{M}^I = \mathbf{K}^I \; \left[\mathbf{R}^I \; \mathbf{t}^I\right]$ is the estimated camera matrix for the input view. Hence, matrix $\mathbf{M}^I$ can be used to map any point on the 3D surface onto the input image, thereby providing the desired texture map.

4.2 Ray casting / rasterization

Following texture mapping, $\mathbf{S}$ is projected to the desired view, $\mathbf{J}$. To this end, the output view's camera matrix, $\mathbf{M}^J = \mathbf{K}^J \; \left[\mathbf{R}^J \; \mathbf{t}^J\right]$, is manually specified in order to set the desired output viewpoint (e.g., frontal view for face *frontalization* (Hassner et al, 2015)). This includes setting both the intrinsic camera parameters in $\mathbf{K}^J$ and the 3D rotation and translation in $\mathbf{R}^J \; \mathbf{t}^J$ in the coordinate frame of the 3D shape. The matrix $\mathbf{M}^J$ is then used to intersect the rays emanating from $\mathbf{J}$'s center of projection, passing through each of its pixels, $\mathbf{p}_i = (x_i, y_i) \in \mathbf{J}$, and the surface of $\mathbf{S}$. Each such intersection is a 3D point $\mathbf{P}_i = (X_i, Y_i, Z_i) \in \mathbf{S}$. Following the texture mapping of Sec. 4.1, these 3D points are already linked to locations $\mathbf{q}_i = (u_i, v_i)$ in the input face image, $\mathbf{I}$. Thus, the pixel $\mathbf{p}_i$ in the output image is assigned intensity values by sampling $\mathbf{I}$ at its determined $\mathbf{q}_i$.

4.3 Precomputing output projections

In a typical graphics pipeline, ray casting (Sec. 4.2) is one of the most time-consuming steps. It computes the locations of intersections between the rays passing through each output pixel and the surface of an object in the 3D scene (here, a face). This process often involves Z-buffering or binary space partitioning methods which determine visibility of the 3D shape at each output pixel (Hughes et al, 2014).

As previously mentioned, these steps can be expedited using specialized hardware, optimized code, and various approximation methods. Importantly, however, when the 3D shape and the output views are both fixed, *these steps only need to be performed once*, at preprocessing. Subsequent renders using the same shape and pose but different textures (input images) can skip this step. Hence, future rendering of these is only as computationally expensive as texture mapping—here, viewpoint estimation for the input image (Sec. 4.1) along with standard 2D image warping required to sample the input image and assign its intensities to pixels in the output view.

During preprocessing, we use a standard rendering engine to perform ray casting of our generic 3D face shape $\mathbf{S}$ onto a desired output view $\mathbf{J}$. This process is performed once, regardless of the number of images we will later warp.

We use the public code provided by Hassner et al (2014) for this purpose. For each output, we obtain the 2D pixel location $\mathbf{p}_i \in \mathbf{J}$ and the 3D coordinates $\mathbf{P}_i \in \mathbf{S}$ projected onto that pixel (i.e., the 3D location of the surface point $\mathbf{P}_i$ visible at $\mathbf{p}_i$). This information is then stored in a fixed lookup table $\mathbf{U}$, simply defined as:

$$\mathbf{U}(\mathbf{p}_i) = \mathbf{P}_i. \tag{1}$$

In practice, $\mathbf{U}$ is stored as an $N \times M \times 3$ matrix, where N and M are the dimensions of the output view and the last dimension indexes the X, Y, and Z coordinates of the 3D point on the surface of the generic 3D face model projected onto each pixel. The Z component of $\mathbf{U}$ is visualized in Fig. 4 along with the overall rendering process for two views.

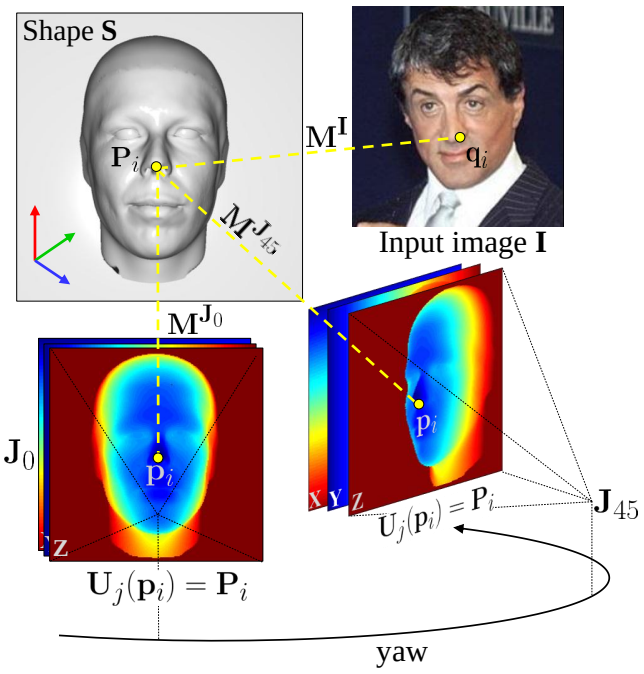


Fig. 4: **Precomputed projections and rendering process.** A generic 3D face shape $\mathbf{S}$ is used to render an input image $\mathbf{I}$ at different views $\mathbf{J}$. Matrices $\mathbf{U}_j$ (Eq. (1)) are precomputed offline and visualized as a heat map of the Z component for a frontal ($\mathbf{J}_0$) and half-profile ($\mathbf{J}_{45}$) rendered face. Each pixel stores the distance along the ray passing through it and emanating from the center of projection for the desired output, to where the ray intersects the surface of the 3D face. The rendering is achieved by precomputing all the information except for texture mapping $\mathbf{M}^I$, which happens online. The texture is then transfered to each view using $\mathbf{U}$. This process is implemented by the simple, ten-line Python code provided in Appendix B.

4.4 Rendering with precomputed projections

Given an input image $\mathbf{I}$ containing a face in unconstrained settings, we use the following simple procedure to render it to a desired new view. A simple, ten-line Python code snippet implementing this process is available in Appendix B. This code should be compared with the more involved and computationally heavier implementations required without precomputed projections.

We first estimate the 3D pose of the face, as described in Sec. 4.1. This provides a camera matrix $\mathbf{M}^I$ associating 3D points on the surface of $\mathbf{S}$ with pixels in $\mathbf{I}$. Let

$$\bar{\mathbf{q}} = \mathbf{M}^I \bar{\mathbf{U}}, \tag{2}$$

where $\bar{\mathbf{U}}$ is matrix $\mathbf{U}$ reshaped to a $4 \times (NM)$ matrix such that the columns are the 3D points in homogeneous notation, stored in $\mathbf{U}$. Matrix $\bar{\mathbf{q}}$ is then a $3 \times (NM)$ matrix with columns representing the 2D projections of these 3D points onto $\mathbf{I}$, also in homogeneous coordinates.

An output view $\mathbf{J}$ can then be produced simply by sampling image $\mathbf{I}$, interpolating its values at coordinates $\bar{\mathbf{q}}$ (following conversion to Euclidean coordinates). Sampled intensities are mapped back to the output view $\mathbf{J}$ by using the correspondence between columns in $\bar{\mathbf{q}}$, columns in $\bar{\mathbf{U}}$, and (x, y) pixel locations in $\mathbf{U}$.

4.5 Analysis of rendering runtime

The rendering process described above includes precisely the same steps as standard inverse warping (Szeliski, 2010). Compared with, for example, the 2D warping regularly performed in real time even on mobile devices, the only difference is in applying a 3×4 camera matrix transformation to homogeneous 3D coordinates, rather than a 3×3 projective transformation.

In addition to rendering, this process also includes estimating the 3D pose of the input image (Sec. 4.1). By using the FPN (Chang et al, 2017) and assuming a GPU, this process is remarkably fast and in particular, faster than most state-of-the-art landmark detection methods routinely used for both 3D and 2D face alignment.

5 Training a CNN with face-specific augmentation

Our fast rendering of Sec. 4 does not, on its own, provide improved face recognition accuracy. As we explain next, it does enable face-specific data augmentation without the penalties of the heavy processing typically associated with 3D rendering.

5.1 Network architectures and training

We experimented with using the following deep network architectures: VGG-19 (Simonyan and Zisserman, 2015) and the very deep ResNet-101 of He et al (2016). We train these networks on various face recognition training sets, with and without augmentations. The descriptions below refer to ResNet-101 as it provided superior results in our tests.

For the specific case of ResNet-101, we keep all layers $\{\mathbf{W}_k, \mathbf{b}_k\}_{k=1}^{101}$ of our networks except for the last linear classification layer which we train from scratch. This layer produces a mapping from the embedded features $\mathbf{x} \in \mathbb{R}^D$ (pool5 $\in \mathbb{R}^{2048}$) to the subject labels in our training set (e.g., the N=62,955 subjects of our COW set; see Sec. 7).

Fine-tuning is performed by minimizing the standard softmax loss:

$$\mathscr{L}(\{\mathbf{W}_k, \mathbf{b}_k\}) = -\sum_t \ln\left(\frac{e^{\mathbf{y}_l^t}}{\sum_{g=1}^N e^{\mathbf{y}_g^t}}\right) \tag{3}$$

where $\mathbf{y}^t = \mathbf{W}_{101}\mathbf{x}^{t\top} + \mathbf{b}_{101}$, l is the ground-truth index over N subjects and t indexes all training images. Eq. (3) is optimized using stochastic gradient descent (SGD) with standard ℓ_2 norm over the learned weights and momentum.

Alternative loss function. In addition to the standard softmax loss of Eq. (3), we experimented with the angular-softmax (*ASM*) of Liu et al (2017a). In this case, the last classification layer consists only of weights without the biases, and the loss is modified to normalize each column (corresponding to each training subject) in the weight matrix to have ℓ_2 norm one. That is, the classification vector $\mathbf{y}$ of Eq. (3) becomes $\mathbf{y} = \mathbf{W}_{101}\mathbf{x}^\top$ and can be written as:

$$\mathbf{y} = \|\mathbf{x}\| \cos\left(m\theta_g\right), \qquad \left\|\mathbf{W}_g\right\| = 1, \tag{4}$$

where θ is the angle between the feature vector $\mathbf{x}$ and the subject represented by the g-th column of $\mathbf{W}$, denoted by $\mathbf{W}_g$. In our implementation we used the margin $m = 2$. For further details refer to Liu et al (2017a).

Training hyper-parameters. When updating the weights, we learn the classification layer faster since it is trained from scratch while other network weights are updated with a learning rate an order of magnitude lower. Specifically, we initialize the classification layer with parameters drawn from a Gaussian distribution with zero mean and standard deviation 0.01. Bias is initialized to zero.

Since the classification layer is very large, we trained our ResNet-101 with a mini-batch of 16 augmented samples on six GPUs simultaneously, for an effective batch size of 96. The learning rate μ for the entire CNN is set to 1e-3, except the classification layer which was trained with a learning rate of $\mu \times 10$. The final learning rate is 1e-5. Learning rate is decreased by an order of magnitude when validation accuracy for the fine-tuned network saturates. Meanwhile, biases are learned twice as fast as other weights. For all other parameter settings, we use the same values as originally described by Krizhevsky et al (2012). The models fine tuned on our COW training set (Sec. 7), with and without augmentation, were trained for about five days and were initialized from weights previously obtained by training on our Noisy Aug. COW for approximately two weeks.

5.2 Training with on-the-fly data augmentation

In Sec. 4.5 we show that rendering new views for faces, using fixed, generic 3D face shapes and predetermined output views, is as fast as 2D image warping. This result allows us to apply our face-specific data augmentation techniques from Sec. 3 on-the-fly, during training, and so eliminates the storage requirements for multiple augmented versions of each training face image.

Specifically, we introduce a novel data layer to the face recognition networks described in Sec. 5.1. This layer produces augmented images on-the-fly while training. Of course, such online data augmentation has been regularly used by others since the seminal work of Krizhevsky et al (2012) and has also been applied when training deep networks for face recognition (Günther et al, 2017). To our knowledge, however, we are the first to propose face-specific data augmentation methods that involve complex 3D transformations applied *on-the-fly while training*; previous methods focused on generic augmentation techniques involving simple 2D and photometric transformations.

In practice, we employed a *hybrid* approach to on-the-fly augmentation: For each training image, we precompute the actual 3D transformations (i.e., run FPN; Sec. 4.1) and store estimated poses offline. This requires storage for only six floating point numbers per training face, representing the three rotation and three translation vectors aligning a generic 3D face with the image. During training, rendering training faces to multiple novel views using multiple generic 3D faces is performed on-the-fly.

5.3 Training with multiple synthetic and real images

Our pipeline employs a single CNN trained on both real and augmented data generated as described in Sec. 3 and loosely inspired by Masi et al (2016b, 2017). Specifically, the training uses the following multiple versions of each image:

– **2D alignment to frontal and profile views.** Training faces are aligned and warped using a simple 2D, in-plane, similarity transform to one of two ideal coordinate systems: roughly frontal facing faces (face yaw es-

timates in $[-30°,...,30°]$) are aligned with a frontal facing template; profile images (all other yaw angles) are aligned to a profile face view.
- **Novel view rendering.** Each training image is rendered from five novel views in yaw angles $\{0°,\pm22°,\pm40°,\pm55°,\pm75°\}$, as described in Sec. 3.2.
- **3D generic shape.** Synthesized views are produced by randomly selecting a single 3D generic face model from the ten in $\mathscr{S}$ as the underlying face shape (see Sec. 3.3), thereby adding shape variations.
- **Generic image quality variations.** 2D aligned images are further subjected to randomly determined quality augmentation with Gaussian blur or quantization artifacts. These are standard augmentation techniques.

On average, the approach synthesizes 4.5 novel samples given an input image. This augmented set always includes a 2D, in-plane aligned image, as well as other novel views, determined according to the estimated viewpoint. If the input face is determined to be in profile view, we do not frontalize it to avoid introducing artifacts when estimating occluded parts of the face (Masi et al, 2018).

Specifically, if a face is near frontal ($|\text{yaw}| < 15°$) we render it to all views as shown in Fig. 3; if viewed close at or near profile ($|\text{yaw}| > 40°$) we render it to the two near profile views; otherwise we render to $\{40°,55°,75°\}$. Additionally, if the face image is near frontal, we apply soft-symmetry following the approach by Hassner et al (2015).

The augmented set also contains a single-image quality-based augmentation (also aligned using 2D similarity). Later, in Sec. 8 we provide results showing the effect of our augmentation on recognition accuracy. We further evaluate the effect that the size of the underlying seed training set has on recognition.

6 Recognition with on-the-fly data augmentation

Previous methods commonly used face alignment in 2D and 3D to reduce appearance variations at test time and make face images easier to compare (Wolf et al, 2011b; Hassner, 2013; Taigman et al, 2014; Hassner et al, 2015; Masi et al, 2016b, 2018). We propose a different approach to the use of aligned faces at test-time.

6.1 Deep face recognition

Given a test face image, we first obtain a bounding box around the face with the face detector of Yang and Nevatia (2016). Our learned ResNet-101 CNN is then applied to this box and used to generate a face feature vector—the *pool5* layer output—as our face representation, $\mathbf{x}$.

We further specialize the representation $\mathbf{x}$ to the target benchmark by applying cheap, unsupervised, principal component analysis (PCA) learned from the training splits of the test benchmark. Power normalization is then applied to the PCA projected features. This step is widely used in Fisher-vector encoding schemes to improve their representation power (Sánchez et al, 2013). Finally, the similarity of two faces, $s(\mathbf{x}_1,\mathbf{x}_2)$, is their correlation score.

6.2 Pooling across augmented faces

Ideally, a deep network optimally trained for face recognition should produce an *identical face descriptor for different images of the same subject*: After all, the similarity between two generated descriptors reflects a similarity between the subject identities in the two images, and should be maximal for two images of the same subject, regardless of any nuisance factors in each image.

In practice, this rarely happens (if ever). No matter how well the network is trained, appearance variations in the *input image* affect the values of the *output descriptors* generated by the network. Thus, although different photos of the same subject under, say, different viewpoints should be assigned the same descriptor, different viewpoints change the values of the descriptors generated by the network.

We consider these descriptor value variations as *noise* and use our face-specific data augmentation to suppress it.

For a test face image, we generate multiple augmented versions of the face, using the methods detailed in Sec. 3, and extract deep features for each of these images. This process introduces nuisance variations to the input images, which are expressed as noise affecting the values of the descriptors generated by the network.

Assuming these descriptors are all noisy variations of the same subject-specific descriptors, we apply element-wise averaging (average pooling) of these descriptors for simple noise removal. The result is a single descriptor representation for each image, no matter how many augmentations were applied in practice. This process is visualized in Fig. 5. We show how this pooling affects the quality of our face representations in Sec. 8.

It is instructive to compare this process with the application of alignment and new view synthesis by previous work (Wolf et al, 2011b; Hassner, 2013; Taigman et al, 2014; Hassner et al, 2015; Masi et al, 2016b, 2018). These previous methods selected a single, *ideal* view, and aligned input faces to that coordinate system. Unsuitable views (e.g., frontal for an input profile image) or erroneous alignments (e.g., when landmark detection failed) result in corrupt descriptors and failure of recognition. Our approach mitigates these problems by pooling multiple descriptors from multiple alignments.

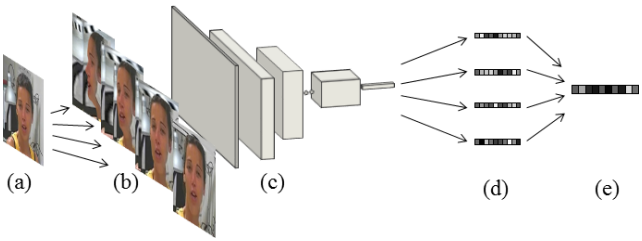


Fig. 5: **Pooling across augmented faces.** (a) Input test face photo is used to produce multiple synthetic views (b) using our face-specific augmentation method. (c) Deep features from our face recognition CNN are extracted, (d) one for each synthetic view. Finally, (e) element-wise average pooling is used to reduce the effects of the synthetically generated viewing variations on the face descriptors.

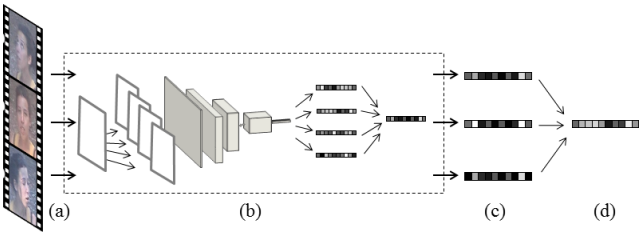


Fig. 6: **Pooling across video frames for recognition.** (a) Frames from an input face video. (b) Each frame is processed as described in Sec. 6.2 (see also Fig. 5), producing (c) one robust descriptor per frame. Finally, (d) Element-wise average pooling of the per-frame descriptors is used to produce a single robust descriptor for the entire video.

6.3 Video and media pooling

The same rationale described in Sec. 6.2 is applied when processing multiple frames in a face video and multiple media sources in the same template. Specifically, we apply the same average pooling to the per-frame face descriptors of Sec. 6.2. This process, illustrated in Fig. 6, provides a single descriptor per video sequence, regardless of the number of frames the video originally contained.

In some cases, multiple images and videos represent a subject. An example is the templates in the recent IARPA Janus benchmarks for unconstrained face recognition (Klare et al, 2015; Whitelam et al, 2017). We process the medias contained in a single template, producing per-media representations. These representations are then average pooled to obtain a single feature embedding for the entire template.

6.4 Pooling with confidence

Similar *template pooling* schemes (sometimes also referred to as *media pooling*) were proposed by others (Sankaranarayanan et al, 2016a; Ranjan et al, 2017; Masi et al, 2017; Crosswhite et al, 2017). Unlike previous work, however, we pool already averaged representations (Sec. 6.2 and Sec. 6.3). We additionally found that better results are obtained by weighted averaging using some measure of confidence in the quality of the input face image.

We use the confidence reported by the standard face detector of Yang and Nevatia (2016) to weigh the importance of each image in the final representation. Each video in a template is pooled separately (Sec. 6.3) with the descriptors of its individual frames weighed by this confidence. Following this step, a video is represented as a single feature vector, no matter how many frames it originally contained. The confidence for this single feature vector associated with the entire video is taken as the average confidence across all the frames in the video.

The entire template is then average-pooled (Sec. 6.3), but by weighted averaging of the features representing any videos and single images in the template. The result is a single feature representation for the entire template, no matter how many images and/or videos it originally contained.

An important consequence of this pooling scheme is that storage and retrieval of template representations are extremely efficient, as they are both constants—independent of the number of images, videos, or video durations the template originally contained. Comparing two templates is also efficient: The similarity of two templates is simply taken to be the correlation of their two pooled features. The effects that pooling and weighing have on recognition accuracies are evaluated in Sec. 8.

6.5 Computational efficiency

As explained in Sec. 4.5, generating new views requires the same computation as 2D image warping, performed routinely by standard augmentation methods (e.g., the Oversampling option of the Caffe framework (Jia et al, 2014)). Sec. 6.2 explains how we pool multiple deep embeddings generated by augmented versions of the same image. An image is therefore represented using a single deep embedding, regardless of the number of augmentations. No additional computation is therefore required for pairwise matching of multiple augmentations. The only overhead required when performing online augmentation is for extracting deep embeddings for each augmented image. This effort equates to one forward pass through the embedding network per augmented image.

7 The COW dataset

As demonstrated by Masi et al (2016b), face-specific augmentation techniques can be used as effective substitutes

for huge training sets. In their work, they matched recognition performance of systems trained on millions of images with the system they trained using only the ∼0.5 million images in the CASIA WebFace collection, but augmented using their face-specific augmentation techniques.

Their results, however, do not imply that the same augmentation methods cannot improve performances when far more images are available in the seed training set. Moreover, their methods only increase *intra-class* variations but not *inter-class*. To explore the effects of a larger seed training set, we experimented with far more training images than the numbers used by Masi et al (2016b).

Specifically, we used a collection consisting of images from three of the largest, publicly available face data sets—MSCeleb, Oxford VGG Face, and CASIA-WebFace—which we refer to collectively as our COW data set. Importantly, COW is not a simple union of the three collections. The three collections were processed to avoid duplicate subjects, test set subjects, and mislabeled images. We next describe COW and how it was assembled.

7.1 Elements of the COW face collection

The data sets which provided COW images are:

- **CASIA-WebFace** (Yi et al, 2014) contains ∼0.5M images of around 10K subjects. CASIA images are filtered, cropped, and resized such that faces appear roughly aligned for scale and translation.
- **Oxford VGG Face** (Parkhi et al, 2015) was downloaded from Internet search engines and contains about 2.6M images, varying in size and quality, of roughly 2.6K subjects. It is known to contain substantial label noise and was not manually curated as also stated by the authors.
- **MS-Celeb-1M** (Guo et al, 2016) has around 10M images of 100K celebrities. Each subject has 100 images retrieved by the Bing search engine using the celebrity's name without filtering retrieved results. This set also contains a lot of label noise and was not curated. In our implementation, rather than using images provided by MS-Celeb-1M, we downloaded the original images from their sources. By having the original images, we were able to process the unaligned images with their backgrounds. Unfortunately, we found many of the links to the sources of the images in MS-Celeb-1M to be broken. Coupled with the steps described below, a far smaller portion of MS-Celeb-1M is actually included in COW.

7.2 Duplicate and test subject discovery

COW images were processed as follows. First, we performed string matching to find overlapping subjects included in more than one of the original three sets. This step is essential to ensure that images of the same subject do not appear under multiple subject labels due to differences in subject naming conventions used by the three collections. To this end, for every pair of data sets, we generated a list of candidate overlapping subjects based on string similarity of subject names. This list was then manually filtered through visual inspection.

We then combined image sets from data set pairs into a single set, using a single subject label for any such duplicates. Importantly, the resulting set of subjects has some overlap with the subjects in the test sets of the benchmarks later used in our experiments (Sec. 8). *Any subjects also included in our test sets were removed from COW*. Note that at this point COW still contains a substantial number of mislabeled images. These images are handled next.

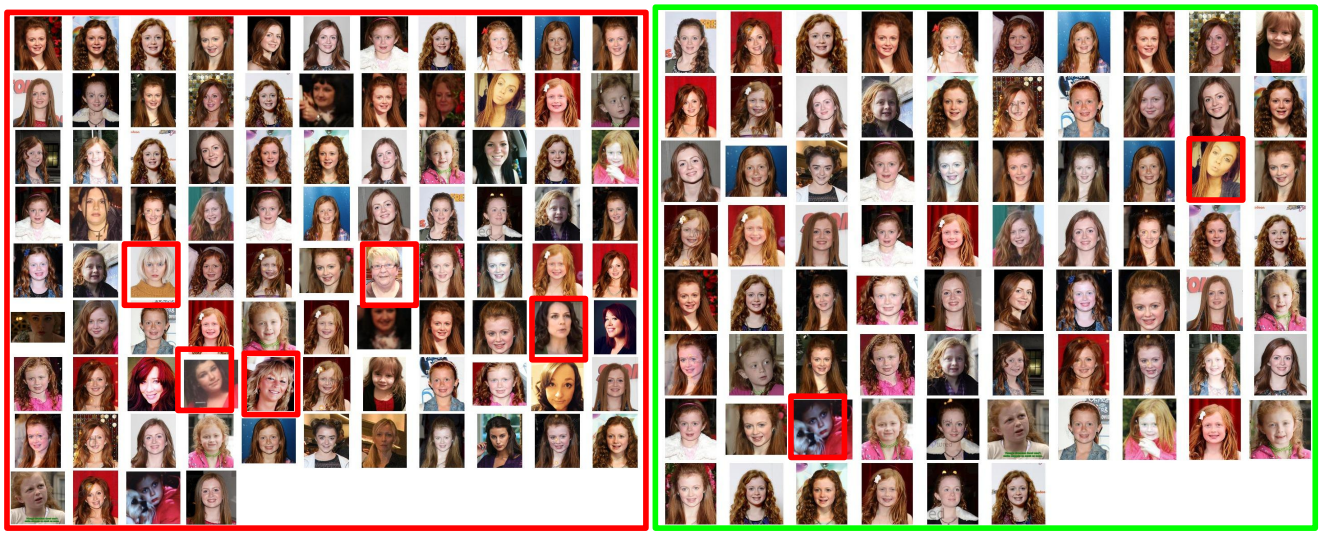
7.3 Filtering mislabeled faces

Both MS-Celeb-1M (Guo et al, 2016) and Oxford VGG Face (Parkhi et al, 2015) have many mislabeled images (images assigned with wrong subject names), non-face, or very poor quality images. These mislabeled images are outliers in the gradient-descent optimization method employed for training our networks. Such images can slow the training or cause convergence to bad local minima.

To mitigate this problem, we applied a simple yet effective two-step training process. First, we use the images in this *noisy* COW set to train a face recognition CNN with standard cross-entropy loss and ℓ_2 regularization on the weights. Although noisy COW includes label noise, the CNN was able to converge and generalize. This convergence is not surprising as it was demonstrated in the past that CNNs converge even when training labels are noisy, provided that the training set is dominated by inliers (Xie et al, 2016). We then use this trained network to refine COW by removing outliers present in the training set, in an approach similar to the one described by Liu et al (2017b).

Specifically, we used the trained network to generate feature representations for the entire training set. We then computed the probability of each image belonging to its assigned label. If this probability is smaller than threshold $\theta =$ 5e-4, then the input image is removed from the set. We selected a threshold value which was slightly higher than the random probability: $1/62K \sim$2e$-$5. The probability threshold was chosen to be conservative in the images retained in the final set. We can afford to be conservative in light of the large numbers of images and subjects in the combined COW set. In fact, when a subject has less than five remaining images, the subject is altogether removed from the set.

In total we removed ∼22% of the images from the entire collection. We estimate that MS-Celeb-1M contained roughly 25-30% label noise that was filtered by our system,



(a)

(b)

Fig. 7: **The effect of filtering mislabeled images for COW.** Incorrect labels are visualized by a red frame. (a) A subject with many mislabeled images (left) which are mostly removed by our filtering process (right). (b) A subject with few mislabeled images (left) retains most of its challenging images while still removing incorrect labels (right).

while Oxford VGGFace contained about 5% noise. Only a handful of images were removed from the CASIA WebFace collection, which is not surprising: CASIA is the only one of the three that was manually curated.

A qualitative example of this noise removal process is provided in Fig. 7. Our algorithm detected and removed several obvious mislabeled faces. Following cleaning of non-faces and mislabeled images, the COW collection used in our tests contained roughly *4M images of 63K individuals*. We will release code and data files to allow others to reproduce our COW set.

8 Experiments

We tested our approach extensively on the recently released IARPA Janus Benchmarks A (IJB-A) (Klare et al, 2015) and B (IJB-B) (Whitelam et al, 2017) as well as older benchmarks: the Labeled Faces in the Wild (LFW) (Huang et al, 2007) and YouTube Faces (YTF) (Wolf et al, 2011a). We report state-of-the-art performances on these benchmarks at the time of submission, as well as ablation studies measuring the influence of our contributions. Finally, we also report the storage and complexity gains of our face-specific

Training data	Ver. (TAR@FAR)			Ide. (Rec.Rate)		
	0.1%	1%	10%	Rank-1	Rank-5	Rank-10
	VGG-19 Masi et al (2016b)					
CASIA	.550±.025	.750±.021	.893±.091	.778±.013	.895±.009	.926±.006
Aug. CASIA +*pose*	.679±.054	.863±.018	.955±—	.880±.012	.947±.008	.966±.006
Aug. CASIA +*pose*,+*shape*	.692±.048	.878±.015	.961±—	.889±.010	.956±.007	.971±.005
	ResNet-101					
COW	.845±.074	.923±.022	.974±.010	.946±.009	.971±.006	.977±.004
Aug. COW +*pose*	.906±.010	.953±.007	.982±.003	.957±.006	.977±.004	.981±.003
Aug. COW +*pose*,+*shape*	**.911±.010**	**.958±.005**	**.984±.002**	**.962±.004**	**.980±.003**	**.984±.003**

Table 1: **Effects of multiple poses and shapes on IJB-A,** for verification (ROC) and identification (CMC).

data augmentation layer. We note that since our paper was submitted there has been further progress in recognition accuracy and we refer to relevant papers for additional details (Ranjan et al, 2018; Yin et al, 2018; Cao et al, 2018; Shen et al, 2018; Crosswhite et al, 2018; Chang et al, 2019; Wen et al, 2019; Neves and Proença, 2019; Rashedi et al, 2019).

8.1 IJB-A recognition and verification tests

IJB-A (Klare et al, 2015) is a publicly available benchmark released by NIST[2] to elevate the challenges of unconstrained face identification and verification. Faces in IJB-A are often viewed under extreme pose, expression, and illumination variations. Other challenges include occluded faces or faces viewed in very poor resolution. IJB-A provides two test protocols: face verification (1:1) and face identification (1:N). Unlike most other face recognition benchmarks, IJB-A represents subjects using templates, which contain mixtures of still images and video frames from heterogeneous sources.

Effects of multiple poses and shapes. We start by assessing what the contribution is of the different face-specific transformations introduced in Sec. 3. The augmentation that has the largest impact on generalization is out-of-plane rotations, while rendering with different 3D face shapes produces subtle variations in the rendered images (see Fig. 3), and contributes less to the final result.

Tab. 1 also confirms these observations for our larger COW training set. When training the network using synthetic profile views, rendered with a single fixed generic shape, the system opens a remarkable accuracy gap compared to a baseline that is trained with only 2D in-plane aligned images. Nevertheless, the best performance is obtained by also adding 3D face shape variations.

These results are consistent with those reported by Masi et al (2016b). The improved accuracy on IJB-A with a network trained on pose augmented images can be explained when considering the distributions of yaw angles in Fig. 2: The IJB-A benchmark has many faces in extreme, near profile views. It is therefore no surprise that introducing many

[2] IJB-A data and splits are available under request at http://www.nist.gov/itl/iad/ig/facechallenges.cfm

Training data	Ver. (TAR@FAR)			Ide. (Rec.Rate)		
	0.1%	1%	10%	Rank-1	Rank-5	Rank-10
	VGG-19 Masi et al (2016b, 2017)					
CASIA	.550±.025	.750±.021	.893±.091	.778±.013	.895±.009	.926±.006
Aug. CASIA	.750±.029	.888±.011	.965±.004	.925±.013	.966±.006	.974±.004
	ResNet-101					
Noisy COW	.806±.041	.925±.011	.981±.003	.950±.008	.979±.004	**.984±.005**
COW	.845±.074	.923±.022	.974±.010	.946±.009	.971±.006	.977±.004
Aug. Noisy COW	.870±.017	.946±.009	.981±.003	.957±.006	.978±.003	.983±.002
Aug. COW	**.911±.010**	**.958±.005**	**.984±.002**	**.962±.004**	**.980±.003**	**.984±.003**

Table 2: **Training set ablation on IJB-A,** for verification (ROC) and identification (CMC) when using different training sets, with or without our augmentations. Note: These results were produced with our complete pipeline, and so should be compared with the best results of Tables 4 and 5.

Testing method	Ver. (TAR@FAR)			Ide. (Rec.Rate)		
	0.1%	1%	10%	Rank-1	Rank-5	Rank-10
	(i) Scale, rotation, translation, flipping					
Single 2D aug.	.833±.020	.919±.013	.973±.005	.939±.008	.969±.005	.976±.005
2D alignment	.847±.018	.932±.012	.979±.004	.945±.008	.975±.003	.981±.003
Multiple 2D aug.	.864±.016	.941±.009	.980±.004	.950±.008	.976±.004	.983±.004
2D align.+2D aug.	.862±.016	.939±.010	.980±.005	.951±.007	.976±.005	.981±.004
	(ii) 2D in-plane alignment					
2D alignment	.845±.074	.923±.022	.974±.010	.946±.009	.971±.006	.977±.004
	(iii) Face-specific aug.					
2D alignment	.891±.024	.945±.008	.981±.003	.956±.005	.975±.004	.979±.005
2D align.+3D aug.	**.911±.010**	**.958±.005**	**.984±.002**	**.962±.004**	**.980±.003**	**.984±.003**

Table 3: **IJB-A results with different training and testing augmentation methods,** for verification (ROC) and identification (CMC). Different augmentation methods are tested using the same exact COW training set.

(synthetic) profile views to the training set improves the network's ability to process such extreme viewpoints. This elevated accuracy also suggests that even the large COW set has an unbalanced distribution of frontal vs. profile images.

Training set ablation studies. Results on IJB-A with different training sets are shown in Tab. 2. Results with augmented training data were obtained using our face-specific data augmentation layer (Sec. 5.2).

Tab. 2 clearly demonstrates the substantial impact of our face-specific augmentation on the quality of the trained face recognition CNN. The largest leap in performance is gained by our augmentation technique applied to the relatively small CASIA WebFace set.

The effect of our augmentation is meaningful even when applied to the much larger COW training set, which includes millions of training images. This improvement suggests that *even a huge image collection does not sufficiently capture the appearance variations of the test domain*. These deficiencies of the original seed training set may be due to the low numbers of images per subject (low intra-class variations; Fig. 1) or from a bias towards frontal poses (Masi et al, 2018).

An additional curious observation is that cleaning the COW data set does not overwhelmingly improve performance. A similar observation was made by Parkhi et al

Method	IJB-A 1:1 TAR@0.1%	IJB-A 1:1 TAR@1%	IJB-A 1:1 TAR@10%	IJB-A 1:N Rank-1	IJB-A 1:N Rank-5	IJB-A 1:N Rank-10
2D in-plane only						
Aug. COW	0.866±0.019	0.938±0.008	0.977±0.003	0.952±0.008	0.976±0.002	0.981±0.003
Aug. COW, +*ASM*	0.864±0.020	0.938±0.010	0.977±0.004	0.953±0.009	0.974±0.005	0.979±0.003
Aug. COW, +*ASM*, +*FPN*	0.867±0.021	0.937±0.010	0.978±0.004	0.954±0.006	0.975±0.005	0.981±0.003
Aug. COW, +*ASM*, +*FPN*,+*Pool.Con.*	0.891±0.024	0.945±0.008	0.981±0.003	0.956±0.005	0.975±0.004	0.979±0.005
2D in-plane and 3D Syn.						
Aug. COW, +*Pool.Syn.*	0.878±0.016	0.948±0.007	0.981±0.003	0.957±0.002	0.979±0.003	0.983±0.003
Aug. COW, +*Pool.Syn.*, +*ASM*	0.879±0.018	0.945±0.007	0.981±0.003	0.957±0.005	0.977±0.002	0.982±0.003
Aug. COW, +*Pool.Syn.*, +*ASM*, +*FPN*	0.888±0.019	0.953±0.006	0.983±0.003	0.961±0.005	0.979±0.004	0.983±0.003
Aug. COW, +*Pool.Syn.*, +*ASM*, +*FPN*,+*Pool.Con.*	**0.911±0.010**	**0.958±0.005**	**0.984±0.002**	**0.962±0.004**	**0.980±0.003**	**0.984±0.003**

Table 4: **Ablation studies on IJB-A of our method's design.** Evaluating the effect on recognition results on IJB-A of different design choices made in developing our method. Ablations are split in two: upper part reports results using 2D in-plane aligned images only; bottom part, the combination of 2D images and 3D synthesized.

(2015) who reported a drop in accuracy when training their deep models after cleaning mislabeled images from their VGGFace set.

Effects of standard data augmentation. Tab. 3 provides results on the IJB-A set comparing different, standard augmentation methods with our proposed face-specific methods. Each category reports results for a network trained with a certain augmentation approach. The training is carried out on the COW set and is performed on-the-fly for all methods.

In our experiments we compare the following baselines: (1) Randomly sampling and warping the face for scale, 2D in-plane rotation, translation and flipping, (2) aligning the images with a 2D similarity transform, and (3) our face-specific data augmentation. Table rows offer results comparing different applications of augmentation at test time. Note that faces were aligned at test time, regardless of the alignment used at training, following Schroff et al (2015); Parkhi et al (2015) who reported the benefits of such alignment.

Fig. 8 offers a qualitative comparison of the variations introduced by each method. Quantitative results are available in Tab. 3. These results show that networks trained with standard data augmentation methods generalize less compared to ours. This is noticeable in particular when comparing only *2D alignment* for different networks. Our proposed method has a +5% improvement in the TAR at FAR=0.1% with respect to a network trained with just in-plane alignment or regular 2D augmentation methods.

Ablation studies of method components. Tab. 4 examines the effects of the different design choices in our recognition method, reporting the effects on recognition accuracy of the following components: Training with augmented COW (Aug. COW); the use of angular-softmax (+*ASM*) of Liu et al (2017a) as the network loss function in Eq. (4) instead of the classic softmax loss based on cross-entropy of Eq. (3), Sec. 5.1; the impact of using FPN of Chang et al (2017) (+*FPN*) as a more robust alternative to face alignment with facial landmark detection (Sec. 4.1); and finally, pooling across synthesized images (+*Pool.Syn.*, see Sec. 6.2) and the use of confidence for weighted pooling (*Pool.Con.*), as described in Sec. 6.4.

The baseline without +*Pool.Syn.* is evaluated as follows: the entire pipeline remains unchanged, but we avoid synthesizing the images at test time with the 3D renderer. Hence, the face recognition processes only 2D aligned images with a similarity transform, without performing pooling across synthetic images but performing the same video and media pooling of Sec. 6.3.

The effect of our proposed pooling of features produced from synthesized images (Sec. 6.2) is evident in Tab. 4 (+*Pool.Syn.*) and is particularly evident in the 1:1 verification results when compared to a pipeline not using pooling (upper part of Tab. 4). In fact, pooling appears to contribute one of the largest performance gains. Another substantial gain comes from the use of the landmark-free alignment method, FPN, appearing as +*FPN*. This gain is noteworthy considering that the baseline result reported by Masi et al (2017) used a state-of-the-art facial landmark detector (Baltrusaitis et al, 2013). Changing the loss function appears to have a smaller effect on results (+*ASM*).

Finally, low resolution images may mislead the recognition process by being easier to match with other images. Such images are referred to in the biometric menagerie terminology of Yager and Dunstone (2010) as *lambs* and *wolves*. Our template weighing scheme of Sec. 6.4 effectively reduces their effect on recognition results; weighing the features based on face detection confidences (+*Pool.Con.*) improves results compared to standard averaging methods (Masi et al, 2017; Ranjan et al, 2017; Crosswhite et al, 2017), yet it is much simpler than methods which learn feature aggregation (Yang et al, 2017).

Ablation studies of PCA and PN. Fig. 9 shows the contributions of using PCA basis and the non-linear power normalization (PN) operator when applied to raw CNN fea-

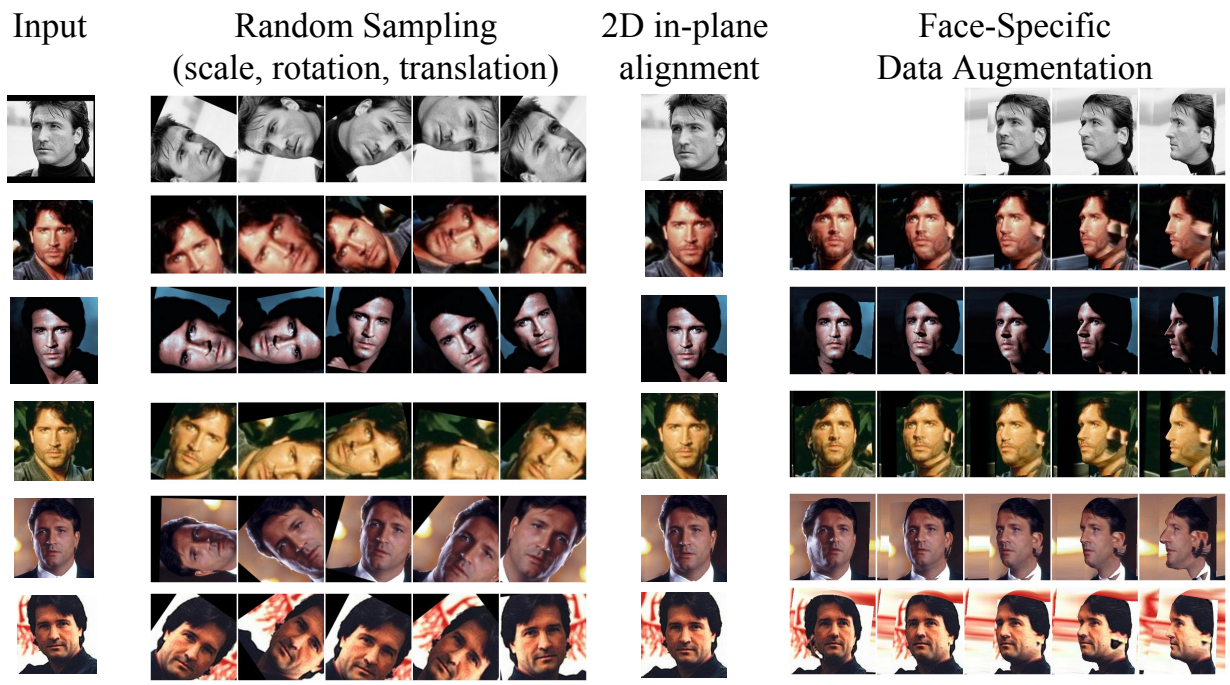


Fig. 8: **Comparison between different augmentation methods.** From left to right: the input image; output produced respectively by warping the image by random sampling for scale, rotation and translation; with a 2D transformation; with face-specific augmentation. Note that in this latter case, we get better aligned faces to the same coordinate system, generate meaningful novel samples, and greatly improve out-of-plane pose variability for subjects. It is often hardest to ensure that face training sets actually provide these variations. By comparison, 2D methods produce samples that are visually correlated with each other, offering limited viewing variations.

tures. Performance for 1:1 verification is reported in Fig. 9a and for 1:N close-set identification in Fig. 9b.

Linear PCA basis appears to have a minimal effect compared to using the raw feature encoding extracted from the network. In particular, TARs improve while the recognition rate at first rank drops slightly. A larger improvement is gained by the non-linear PN, applied after projecting the feature points with PCA. Additional improvement is obtained by repeating power normalization after each pooling step. Note how PCA and PN help improve both TAR at very low FAR and the recall at first ranks.

Comparison with the state-of-the-art. We provide a comprehensive list of published results on IJB-A in Tab. 5. Our results (also appearing in Tab. 4) outperform published state-of-the-art results at the time our paper was submitted, including, in particular, the results reported by Wang et al (2015) using *seven deep networks*, with a commercial system used to fuse their outputs.

It is interesting to compare our results to those reported by Chen et al (2016) and Sankaranarayanan et al (2016b). Both of these previous methods fine-tuned their deep networks on the ten training splits of IJB-A in order to better adjust to the test set domain, but at a substantial computational cost. We believe that by training our networks with

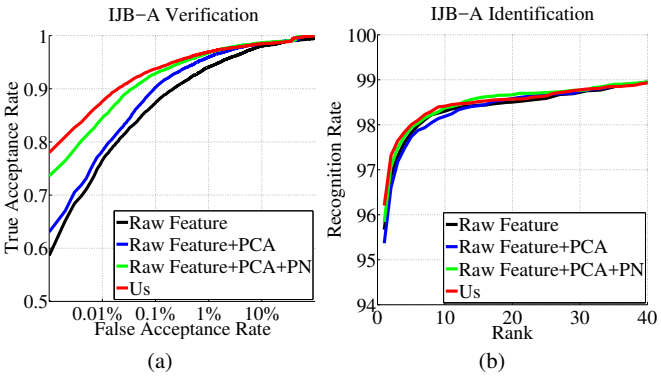


Fig. 9: **Ablation studies for PCA and PN.** (a) ROC curves for 1:1 IJB-A Verification; (b) CMC curves for 1:N close-set identification. See text for more details.

augmented training data we avoid having to similarly specialize our networks to the test data. Compared to methods which, unlike ours, did not tune networks on the target set, our method reached an even greater performance margin.

Recently, a method very similar to ours was proposed by Zhao et al (2017). Their approach also augmented for pose, adding refinement of the synthetic images using an

adversarial loss. They report state-of-the-art performance on IJB-A, though their approach is also trained on IJB-A training splits. Furthermore, unlike our method, their method cannot be used for on-the-fly data augmentation due to computational constraints.

8.2 IJB-B recognition and verification tests

IJB-B (Whitelam et al, 2017) is a newly released benchmark that extends IJB-A by providing additional subjects and more challenging viewing conditions. Compared to IJB-A, IJB-B consists of 1,845 individuals for a total of 21,798 still images and 55,026 frames, extracted from about 7K videos. Unlike IJB-A, IJB-B does not offer training splits. We therefore compute our PCA basis on a held-out set of our COW training set (see Sec. 6.1). Thus, even unsupervised feature specialization is not permitted on IJB-B.

Tab. 6 provides an ablation study of the proposed system on IJB-B and should be compared with the similar study performed on IJB-A and reported in Tab. 4. Ablation results on IJB-B follow the same pattern as those reported for IJB-A, supporting the design choices made in developing our approach. Tab. 7 additionally provides a comparison with the published results on IJB-B. Because IJB-B was only recently released, far fewer baseline results are available. Importantly, despite being far larger and more challenging than IJB-A, our results on this set remain quite high (e.g., recognition rate at rank-1 is higher than 90%).

8.3 LFW verification tests

LFW (Huang et al, 2007) has been the de facto standard for measuring face recognition performances for a decade now. To test our approach, we follow the standard protocol for *unrestricted, labeled outside data* and report the mean classification accuracy as well as the 100% - equal error rate (EER). We prefer to use 100% - EER in general because it is independent of a classification threshold. We report verification accuracy for comparison with previous methods.

Fig. 10a provides ROC curves for our two face-specific augmentation techniques. The green dashed curve represents our baseline—the CNN trained on 2D in-plane aligned images. The ROC improves considerably when training also includes our generated novel views and 3D shapes. Indeed, the 100% - EER improves by +1.67%. By adding 3D face shape augmentation, performance improves even further, reaching a 100% - EER rate of 98.00% (red dashed curve).

When COW is used as the seed training set, augmentation provides only a slight improvement (see green solid curve vs. red solid curve). This limited contribution is partly due to LFW containing faces which are mostly near-frontal, and so rendering to extreme poses is less necessary. Nevertheless, augmentation does improve, suggesting that its contribution goes beyond introducing pose variations. More importantly, however, results reported for COW without augmentation are already near-perfect, and so augmentation has very little room for improvement.

Finally, a comparison of our results on LFW to those reported by methods trained on millions of images (Taigman et al, 2014, 2015; Parkhi et al, 2015; Schroff et al, 2015) shows that with the initial set of less than 500K publicly available images (Yi et al, 2014), our method surpasses those of Taigman et al (2014) and Parkhi et al (2015) (without their metric learning, which was not applied here). Our method and the state-of-the-art achieve error rates smaller than 1%. FaceNet (Schroff et al, 2015), in particular, obtains a slightly higher performance than we do. The small gain in accuracy reported by FaceNet comes at the price of using a private collection of 200 million labeled face images.

8.4 YTF face video verification results

Along with LFW, YTF (Wolf et al, 2011a) is one of the more popular face verification benchmarks. YTF contains face videos, rather than images or templates, and uses a 1:1 verification test protocol similar to LFW. It provides ten splits, each with 500 video pairs to be compared (250 *same*, 250 *not-same*). Results are reported as the average accuracy across all splits.

Tab. 8 lists previously published results on YTF. To our knowledge, state-of-the-art performance on YTF at the time of submission was reported by Parkhi et al (2015) using a single CNN trained on the 2.6M images of the VGG Face set and then improved by supervised triplet-loss embedding on the YTF training splits. All other methods, including our own, did not perform supervised training on YTF videos, which would explain why no result since has reported the same performance. Without this metric, learning their approach obtained a far lower accuracy of 91.6%. This drop in performance emphasizes the importance of supervised training on the target domain. We include this result in Tab. 8 for completeness, despite its use of additional information compared to all other methods.

Without use of YTF training data and labels, our method outperforms all other previously reported results. This includes the very recent approach of Yang et al (2017), which was explicitly designed for video-based face recognition (Kim et al, 2018) but uses different frame pooling techniques than we do.

	IJB-A 1:1			IJB-A 1:N		
Method	TAR@0.1%	TAR@1%	TAR@10%	Rank-1	Rank-5	Rank-10
OpenBR (Klontz et al, 2013)	0.104±——	0.236±——	–	0.246±——	0.375±——	–
LSFS (Wang et al, 2015)	0.514±0.060	0.733±0.034	0.895±0.013	0.820±0.024	0.929±0.013	–
GOTS (Klare et al, 2015)	0.198±——	0.406±——	–	0.443±——	0.595±——	–
VGG-Face (Parkhi et al, 2015)	–	0.805±0.030	–	0.913±0.011	–	0.981±0.005
DCNN +*fusion* (Chen et al, 2016)	–	0.838±0.042	0.967±0.009	0.903±0.012	0.965±0.008	0.977±0.007
B-CNN (Chowdhury et al, 2016)	–	–	–	0.588±0.020	0.796±0.017	–
Triplet Embedding (Sankaranarayanan et al, 2016a)	0.813±0.02	0.900±0.010	0.964±0.005	0.932±0.01	–	0.977±0.005
Pooling faces (Hassner et al, 2016)	–	0.309±——	0.631±——	0.846±——	0.933±——	0.951±——
Deep Multi-Pose (AbdAlmageed et al, 2016)	–	0.787±——	0.911±——	0.846±——	0.927±——	0.947±——
Pose-Aware Models (Masi et al, 2016a)	0.652±0.037	0.826±0.018	–	0.840±0.012	0.925±0.008	0.946±0.007
Aug. (Masi et al, 2016b)	0.725±——	0.886±——	–	0.906±——	0.962±——	0.977±——
Aug. Pose, Light (Crispell et al, 2016)	–	–	–	0.915±0.012	0.968±0.006	0.980±0.004
Aug. Pose, Light +*temp. adapt.* (Crispell et al, 2016)*	–	–	–	0.944±0.009	**0.981±0.005**	**0.989±0.003**
3DMM-CNN (Tran et al, 2017)	–	0.600±0.056	0.870±0.015	0.762±0.018	0.897±0.01	0.929±0.01
Rapid Synthesis (Masi et al, 2017)	0.750±0.029	0.888±0.011	0.965±0.004	0.925±0.013	0.966±0.007	0.974±0.005
All-in-one (Ranjan et al, 2017)	0.823±0.02	0.922±0.010	0.976±0.004	0.947±0.008	–	0.988±0.003
VGG-Face +*temp. adapt.* (Crosswhite et al, 2017)*	0.836±0.027	0.939±0.013	0.979±0.004	0.928±0.010	0.977±0.004	0.986±0.003
Investigating Nuisance Factors (Ferrari et al, 2017)	0.759±0.041	0.896±0.016	–	0.910±0.014	–	0.983±0.003
FPN (Chang et al, 2017)	0.852±——	0.901±——	–	0.914±——	0.930±——	0.938±——
NAN (Yang et al, 2017)	0.860±0.012	0.933±0.009	0.979±0.004	0.954±0.007	0.978±0.004	0.984±0.003
NAN +*media-pool* (Yang et al, 2017)	0.881±0.011	0.941±0.008	0.978±0.003	0.958±0.005	0.980±0.005	0.986±0.003
DCNN +*metric* (Chen et al, 2015) (f.t.)	–	0.787±0.043	0.947±0.011	0.852±0.018	0.937±0.010	0.954±0.007
Triplet Similarity (Sankaranarayanan et al, 2016b) (f.t.)	0.590±0.050	0.790±0.030	0.945±0.002	0.880±0.015	0.950±0.007	0.974±0.005
DA-GAN Zhao et al (2017) (f.t.)	**0.930±0.005**	**0.976±0.007**	**0.991±0.003**	**0.971±0.007**	**0.989±0.003**	–
Us	**0.911±0.010**	**0.958±0.005**	**0.984±0.002**	**0.962±0.004**	0.980±0.003	0.984±0.003

Table 5: **State-of-the-art performances on IJB-A.** Comparing previously published verification (ROC) and identification (CMC) results at different cut-off points of the evaluation metrics with our own performance. Our method surpasses previous state-of-the-art at the time of submission, with wide margins. Note: f.t. denotes fine-tuning a deep network for each training split. A network trained once with our augmented data achieves results on par, without this effort.* Methods denoted with +*temp. adapt.* used a computationally expensive approach to matching, which may not scale well to large data sets.

	IJB-B 1:1			IJB-B 1:N		
Method	TAR@0.1%	TAR@1%	TAR@10%	Rank-1	Rank-5	Rank-10
Aug. COW	0.885	0.959	0.990	0.883	0.942	0.958
Aug. COW +*Pool.Syn.*	0.899	0.963	0.990	0.895	0.948	0.963
Aug. COW, +*Pool.Syn.*, +*ASM*	0.900	0.959	0.989	0.895	0.945	0.958
Aug. COW, +*Pool.Syn.*, +*ASM*, +*FPN*	0.909	0.962	0.990	0.904	0.950	0.961
Aug. COW, +*Pool.Syn.*, +*ASM*, +*FPN*,+*Pool.Con.*	**0.920**	**0.966**	**0.991**	**0.916**	**0.956**	**0.967**

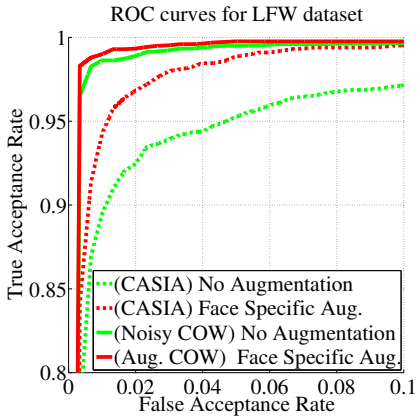
Table 6: **Ablation studies on IJB-B of our method's design.** Evaluating the effect on recognition results on IJB-B of different design choices made in developing our method. See text for further details. Note that IJB-B, unlike IJB-A, only has one test split. Unlike Tab. 4, results are therefore reported here without ± standard deviations.

8.5 Rendering speed, storage, and convergence analysis

Rendering speed. We compare the rendering of runtimes required by our approach with the existing face rendering techniques of Hassner et al (2015); Masi et al (2016b, 2013, 2016a). Runtimes for all methods were measured on an Intel Core i7-4820K CPU @ 3.70GHz (4 cores), 32GB RAM and nVidia Titan X GPU. In addition to runtimes, we report other relevant aspects of these rendering systems: their support of rendering multiple poses (rather than, e.g., only frontalization); use of multiple 3D shapes; implementation programming language; dependency on OpenGL; and finally, public availability.

For a fair comparison, we measure the average time (Avg. Rend. Time) required by each method for rendering alone; time required for pose estimation (and/or landmark detection) was excluded, although we note that the FPN used in our system is much faster than pose estimation based on facial landmark detection performed by others. Speeds were measured averaging over $2,000$ rendered views.

Our report is summarized in Tab. 9. Our method, despite being implemented using high-level, unoptimized Python code, can generate a rendered view in 14.27ms. The recent method of Masi et al (2016b), which uses optimized and compiled OpenGL code, requires more time (46.12ms) due to the repeated ray casting it performs—which we instead



(a) Ablation Study

Method	Real	Synth	Net	Acc. (%)	100% - EER
Fisher Vector Faces (Parkhi et al, 2014)	–	–	–	93.00±—	93.10±—
DeepFace (Taigman et al, 2014)	4M	–	3	97.35±—	–
Fusion (Taigman et al, 2015)	500M	–	5	98.37±—	–
FaceNet (Schroff et al, 2015)	200M	–	1	98.87±—	–
FaceNet +*alignment* (Schroff et al, 2015)	200M	–	1	**99.63±—**	–
VGG Face (Parkhi et al, 2015)	2.6M	–	1	–	97.27±—
VGG Face +*triplet* (Parkhi et al, 2015)	2.6M	–	1	98.95±—	**99.13±—** *
VGG-19 Masi et al (2016b)					
Us, (CASIA) no aug.	495K	–	1	95.31±0.84	95.26±0.97
Us, (CASIA) aug. pose	495K	2M	1	97.01±0.57	96.93±0.79
Us, (CASIA) aug. pose, shape	495K	2.4M	1	98.06±0.59	98.00±0.60
ResNet-101					
Us, (Noisy COW) no aug.	5.2M	–	1	98.80±0.63	98.80±0.68
Us, (Aug. COW) pose, shape, qual.	4M	18.6M	1	99.07±0.44	99.00±0.52

(b) Results for methods trained on millions of images

Fig. 10: **LFW verification results.** (a) Break-down of the influence of different training data augmentation methods. (b) Performance comparison with state-of-the-art methods, showing the numbers of real (original) and synthesized training images, number of CNNs used by each system, accuracy and 100%-EER. * The only system which performed supervised training on LFW data and labels.

	IJB-B 1:1 (TAR@FAR)			**IJB-B 1:N** (Rec. Rate)		
Method	0.1%	1%	10%	R1	R5	R10
Whitelam et al (2017)	0.33	0.60	0.78	0.42	0.57	0.62
Whitelam et al (2017)	0.72	0.860	0.94	0.78	0.86	0.89
Chang et al (2017)	0.916	0.965	–	0.911	0.953	0.965
Us	**0.920**	**0.966**	**0.991**	**0.916**	**0.956**	**0.967**

Table 7: **State-of-the-art performances on IJB-B.** Comparing previously published verification (ROC) and identification (CMC) results at different cutoff points of the evaluation metrics with our own performance. Our method surpasses previous state-of-the-art at the time of submission with wide margins.

Method	Accuracy (%)	AUC (%)
LM3L (Hu et al, 2014b)	81.3±1.2	89.3±—
DDML (combined) (Hu et al, 2014a)	82.3±1.5	90.1±—
EigenPEP (Li et al, 2014)	84.8±1.4	92.6±—
DeepFace-single (Taigman et al, 2014)	91.4±1.1	96.3±—
DeepID2+ (Sun et al, 2014c)	93.2±0.2	–
FaceNet (Schroff et al, 2015)	95.1±0.4	–
VGG-Face (Parkhi et al, 2015)	91.6±—	–
VGG-Face +*triplet* (Parkhi et al, 2015)	**97.3**±— *	–
Center Loss (Wen et al, 2016)	94.9±—	–
3DMM-CNN (Tran et al, 2017)	88.80±2.21	95.37±1.43
NAN (Yang et al, 2017)	95.72±0.64	**98.8±—**
Us	**95.86±1.07**	97.8±1.17

Table 8: **State-of-the-art performance evaluation on YTF.** Comparing previously published verification performance.* This is the only system which performed supervised training on YTF data and labels.

perform only once at preprocessing. Overall, our method has a speed-up of about ×3.7.

To appreciate the significance of this, it is possible to synthesize novel views using multiple 3D shapes simultaneously while training the CNN, following Sec. 5.2, without any significant training time overhead. In addition, rendering on-the-fly also provides a non-negligible boost in terms of space complexity.

Storage requirements. The entire COW dataset stored with JPG compression requires around 150GB of disk space. Applying many (possibly infinite) augmentations to such a huge image collection could easily inflate this size to unrealistic storage requirements if all the synthetically produced images were saved on the disk. In general, the saving in storage complexity is proportional to the number of augmentations performed. Using our augmentation methods, saving is about ×4.5. Note that some overhead is still required for storing additional metadata information (precomputed 3D poses). We found that the metadata stored on the disk requires less than 10% of the original COW size.

Training convergence analysis. We study the convergence of training a baseline model with in-plane aligned images with / without face-specific data augmentation. These two experiments are performed on the CASIA WebFace set (Yi et al, 2014), and, more importantly, aside from the use of data augmentation, use identical settings during training.

Fig. 11b shows that by synthesizing more novel samples, our training takes more time to converge: there is a difference of about twenty epochs between the baseline and our model until the validation plateaus. Note that our training converges to a higher loss value compared to the baseline method (Fig. 11a). This translates into less overfitting and better generalization, as evident from Fig. 11b. Although the baseline model is quicker to fit to the training data, it leads to a wider gap between training and validation. This gap is reduced when training using face-specific augmentation.

Method	Avg. Rend. Time (ms)	Multi-Pose	Multi-Shape	Implement.	Avoid OpenGL	Pub. Aval.
Masi et al (2013)	–	Yes	No	Blender	No	No
Hassner et al (2015)	66.33±09.51	No	No	MATLAB	Yes	Yes
Masi et al (2016a)	46.12±18.45	Yes	No	Optimized C++	No	No
Masi et al (2016b)	46.12±18.45	Yes	Yes	Optimized C++	No	No
Our renderer	**14.27±01.98**	Yes	Yes	Python	Yes	Yes

Table 9: **Face rendering comparison.** Overview of runtimes and other properties of recent face rendering techniques.

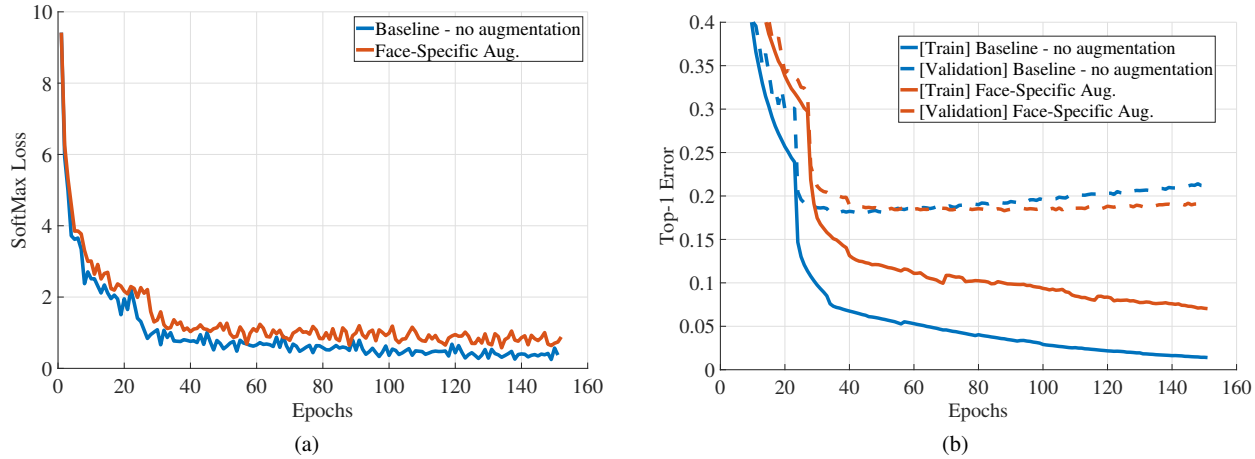


Fig. 11: **Model convergence analysis.** (a) Convergence of the Soft-Max cross-entropy loss function for the baseline and for the model trained with face-specific augmentation. (b) Top-1 error rate for both training and validation sets for the base model and for the face-specific data augmented model.

8.6 Results summary

The results throughout this section clearly show that synthesizing training images using domain tools and knowledge leads to a dramatic increase in recognition accuracy. This may be attributed to the potential of domain-specific augmentation to infuse training data with important intra-subject appearance variations—the very variations that seem hardest to obtain by simply downloading more images. As a bonus, it is a more accessible means of increasing training set sizes than downloading and labeling millions of additional faces. A final plus is that synthesizing complex 3D transformation can be done seamlessly while training, without requiring storage for synthetically generated image variations. The benefits of this augmentation are also evident at test time, where pooled representations of synthesized images appear significantly better than the single descriptors of the original images.

9 Conclusions

This paper makes several important contributions. First, we show how domain-specific data augmentation can be used to efficiently generate (synthesize) valuable additional data, on-the-fly, to train effective face recognition systems, as an alternative to expensive data collection and labeling. Second, we describe a face recognition pipeline with several novel details. In particular, one of those novel details is our pipeline's use of the very same face-specific data augmentation techniques as a means of obtaining more robust face representations which reduce the effects of appearance nuisances. Finally, our extensive analysis shows that although there is certainly a benefit to downloading increasingly larger training sets (see the proposed COW set), much of this effort can be substituted by simply synthesizing more face images and automatically removing mislabeled training samples.

There are several compelling directions for extending this work. Primarily, the underlying idea of domain-specific data augmentation can be extended in more ways (more facial transformations) to provide additional intra-subject appearance variations. Appealing potential augmentation techniques, not used here, are occlusions (Nirkin et al, 2018), facial age synthesis (Kemelmacher-Shlizerman et al, 2014), facial hair manipulations (Nguyen et al, 2008), facial details (Tran et al, 2018), or expressions (Chang et al, 2018). Finally, beyond faces, there may be other domains where such an approach is relevant and where the introduction of synthetically generated training data can help mitigate the many problems of data collection for CNN training.

Acknowledgements The authors wish to thank Jongmoo Choi for his help in this project. This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA 2014-14071600011. 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. Moreover, we gratefully acknowledge the support of NVIDIA Corporation with the donation of the NVIDIA Titan X GPU used for this research.

References

AbdAlmageed W, Wu Y, Rawls S, Harel S, Hassner T, Masi I, Choi J, Leksut J, Kim J, Natarajan P, Nevatia R, Medioni G (2016) Face recognition using deep multi-pose representations. In: Winter Conf. on App. of Comput. Vision 18

Baltrusaitis T, Robinson P, Morency LP (2013) Constrained local neural fields for robust facial landmark detection in the wild. In: Proc. Int. Conf. Comput. Vision Workshops 15

Bansal A, Nanduri A, Castillo CD, Ranjan R, Chellappa R (2017) UMDFaces: An annotated face dataset for training deep networks. In: Int. Joint Conf. on Biometrics 3

Cao K, Rong Y, Li C, Tang X, Change Loy C (2018) Pose-robust face recognition via deep residual equivariant mapping. In: Proc. Conf. Comput. Vision Pattern Recognition, pp 5187–5196 14

Chang F, Tran A, Hassner T, Masi I, Nevatia R, Medioni G (2017) FacePoseNet: Making a case for landmark-free face alignment. In: 7th IEEE International Workshop on Analysis and Modeling of Faces and Gestures, ICCV Workshops 5, 7, 8, 15, 18, 19

Chang F, Tran A, Hassner T, Masi I, Nevatia R, Medioni G (2019) Deep, landmark-free FAME: Face alignment, modeling, and expression estimation. Int J Comput Vision 14

Chang FJ, Tran AT, Hassner T, Masi I, Nevatia R, Medioni G (2018) Expnet: Landmark-free, deep, 3d facial expressions. In: Automatic Face and Gesture Recognition, pp 122–129 20

Chatfield K, Simonyan K, Vedaldi A, Zisserman A (2014) Return of the devil in the details: Delving deep into convolutional nets. In: Proc. British Mach. Vision Conf. 3, 4

Chen JC, Ranjan R, Kumar A, Chen CH, Patel VM, Chellappa R (2015) An end-to-end system for unconstrained face verification with deep convolutional neural networks. In: Proc. Conf. Comput. Vision Pattern Recognition Workshops, pp 118–126 18

Chen JC, Patel VM, Chellappa R (2016) Unconstrained face verification using deep cnn features. In: Winter Conf. on App. of Comput. Vision 3, 16, 18

Chowdhury AR, Lin TY, Maji S, Learned-Miller E (2016) One-to-many face recognition with bilinear cnns. In: Winter Conf. on App. of Comput. Vision, IEEE, pp 1–9 18

Crispell DE, Biris O, Crosswhite N, Byrne J, Mundy JL (2016) Dataset augmentation for pose and lighting invariant face recognition. In: Applied Imagery Pattern Recognition Workshop (AIPR) 4, 18

Crosswhite N, Byrne J, Stauffer C, Parkhi O, Cao Q, Zisserman A (2017) Template adaptation for face verification and identification. In: Int. Conf. on Automatic Face and Gesture Recognition 4, 11, 15, 18

Crosswhite N, Byrne J, Stauffer C, Parkhi O, Cao Q, Zisserman A (2018) Template adaptation for face verification and identification. Image and Vision Computing 79:35–48 14

Eigen D, Fergus R (2015) Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: Proc. Int. Conf. Comput. Vision, pp 2650–2658 4

Ferrari C, Lisanti G, Berretti S, Del Bimbo A (2017) Investigating nuisance factors in face recognition with dcnn representation. In: Proc. Conf. Comput. Vision Pattern Recognition Workshops 18

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Neural Inform. Process. Syst. 4

Günther M, Rozsa A, Boult TE (2017) Affact-alignment free facial attribute classification technique. In: Int. Conf. on Automatic Face and Gesture Recognition 9

Guo Y, Zhang L, Hu Y, He X, Gao J (2016) MS-Celeb-1M: challenge of recognizing one million celebrities in the real world. Electronic Imaging 2016(11) 2, 3, 12

Hassner T (2013) Viewing real-world faces in 3d. In: Proc. Int. Conf. Comput. Vision, pp 3607–3614 4, 7, 10

Hassner T, Assif L, Wolf L (2014) When standard RANSAC is not enough: cross-media visual matching with hypothesis relevancy. Machine Vision and Applications 25(4):971–983, available: www.openu.ac.il/home/hassner/projects/poses 8

Hassner T, Harel S, Paz E, Enbar R (2015) Effective face frontalization in unconstrained images. In: Proc. Conf. Comput. Vision Pattern Recognition 4, 5, 6, 7, 10, 18, 20, 24

Hassner T, Masi I, Kim J, Choi J, Harel S, Natarajan P, Medioni G (2016) Pooling faces: Template based face recognition with pooled face images. In: Proc. Conf. Comput. Vision Pattern Recognition Workshops 18

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proc. Conf. Comput. Vision Pattern Recognition 9

Hu J, Lu J, Tan YP (2014a) Discriminative deep metric learning for face verification in the wild. In: Proc. Conf. Comput. Vision Pattern Recognition, pp 1875–1882 19

Hu J, Lu J, Yuan J, Tan YP (2014b) Large margin multi-metric learning for face and kinship verification in the wild. In: Asian Conf. Comput. Vision, Springer, pp 252–267 19

Huang GB, Ramesh M, Berg T, Learned-Miller E (2007) Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Tech. Rep. 07-49, UMass, Amherst 2, 13, 17

Hughes JF, Van Dam A, Foley JD, Feiner SK (2014) Computer graphics: principles and practice. Pearson Education 7

Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T (2014) Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:14085093 11

Kemelmacher-Shlizerman I, Suwajanakorn S, Seitz SM (2014) Illumination-aware age progression. In: Proc. Conf. Comput. Vision Pattern Recognition, IEEE, pp 3334–3341 20

Kemelmacher-Shlizerman I, Seitz SM, Miller D, Brossard E (2016) The MegaFace benchmark: 1 million faces for recognition at scale. In: Proc. Conf. Comput. Vision Pattern Recognition 3

Kim K, Yang Z, Masi I, Nevatia R, Medioni G (2018) Face and body association for video-based face recognition. In: Winter Conf. on App. of Comput. Vision, pp 39–48 17

Klare BF, Klein B, Taborsky E, Blanton A, Cheney J, Allen K, Grother P, Mah A, Burge M, Jain AK (2015) Pushing the frontiers of unconstrained face detection and recognition: IARPA Janus Benchmark A. In: Proc. Conf. Comput. Vision Pattern Recognition, pp 1931–1939 3, 11, 13, 14, 18

Klontz J, Klare B, Klum S, Taborsky E, Burge M, Jain AK (2013) Open source biometric recognition. In: Int. Conf. on Biometrics: Theory, Applications and Systems 18

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Neural Inform. Process. Syst., pp 1097–1105 3, 9

Lawrence S, Giles CL, Tsoi AC, Back AD (1997) Face recognition: A convolutional neural-network approach. Trans Neural Networks 8(1):98–113 2

Levi G, Hassner T (2015) Age and gender classification using convolutional neural networks. In: Proc. Conf. Comput. Vision Pattern Recognition Workshops, URL http://www.openu.ac.il/home/hassner/projects/cnn_agegender 3

Li H, Hua G, Shen X, Lin Z, Brandt J (2014) Eigen-pep for video face recognition. In: Asian Conf. Comput. Vision, Springer, pp 17–33 19

Liu W, Wen Y, Yu Z, Li M, Raj B, Song L (2017a) Sphereface: Deep hypersphere embedding for face recognition. In: Proc. Conf. Comput. Vision Pattern Recognition 9, 15

Liu X, Li S, Kan M, Shan S, Chen X (2017b) Self-error-correcting convolutional neural network for learning with noisy labels. In: Int. Conf. on Automatic Face and Gesture Recognition, IEEE, pp 111–117 12

Masi I, Lisanti G, Bagdanov A, Pala P, Del Bimbo A (2013) Using 3D models to recognize 2D faces in the wild. In: Proc. Conf. Comput. Vision Pattern Recognition Workshops 18, 20

Masi I, Rawls S, Medioni G, Natarajan P (2016a) Pose-Aware Face Recognition in the Wild. In: Proc. Conf. Comput. Vision Pattern Recognition 7, 18, 20

Masi I, Trần AT, Hassner T, Leksut JT, Medioni G (2016b) Do we really need to collect millions of faces for effective face recognition? In: European Conf. Comput. Vision, Springer, pp 579–596 1, 3, 4, 7, 9, 10, 11, 12, 14, 18, 19, 20, 24

Masi I, Hassner T, Trần AT, , Medioni G (2017) Rapid synthesis of massive face sets for improved face recognition. In: Int. Conf. on Automatic Face and Gesture Recognition 1, 7, 9, 11, 14, 15, 18

Masi I, Chang FJ, Choi J, Harel S, Kim J, Kim K, Leksut J, Rawls S, Wu Y, Hassner T, AbdAlmageed W, Medioni G, Morency LP, Natarajan P, Nevatia R (2018) Learning pose-aware models for pose-invariant face recognition in the wild. Trans Pattern Anal Mach Intell PP(99):1–1 2, 10, 14

McLaughlin N, Martinez Del Rincon J, Miller P (2015) Data-augmentation for reducing dataset bias in person re-identification. In: Int. Conf. Advanced Video and Signal Based Surveillance, IEEE 4

Mokhayeri F, Granger E, Bilodeau GA (2018) Domain-specific face synthesis for video face recognition from a single sample per person. arXiv preprint arXiv:180101974 4

Neves J, Proença H (2019) a leopard cannot change its spots: Improving face recognition using 3d-based caricatures. IEEE Transactions on Information Forensics and Security 14(1):151–161 14

Nguyen MH, Lalonde JF, Efros AA, De la Torre F (2008) Image-based shaving. Computer Graphics Forum 27(2):627–635 20

Nirkin Y, Masi I, Tran A, Hassner T, Medioni G (2018) On face segmentation, face swapping, and face perception. In: Int. Conf. on Automatic Face and Gesture Recognition 20

Parkhi OM, Simonyan K, Vedaldi A, Zisserman A (2014) A compact and discriminative face track descriptor. In: Proc. Conf. Comput. Vision Pattern Recognition 19

Parkhi OM, Vedaldi A, Zisserman A (2015) Deep face recognition. In: Proc. British Mach. Vision Conf. 2, 3, 4, 12, 14, 15, 17, 18, 19

Paysan P, Knothe R, Amberg B, Romdhani S, Vetter T (2009) A 3d face model for pose and illumination invariant face recognition. In: Advanced Video and Signal Based Surveillance, 2009. AVSS '09. Sixth IEEE International Conference on, pp 296–301 5, 24

Ranjan R, Sankaranarayanan S, Castillo CD, Chellappa R (2017) An all-in-one convolutional neural network for face analysis. In: Int. Conf. on Automatic Face and Gesture Recognition, IEEE, pp 17–24 11, 15, 18

Ranjan R, Bansal A, Zheng J, Xu H, Gleason J, Lu B, Nanduri A, Chen J, Castillo CD, Chellappa R (2018) A fast and accurate system for face detection, identification, and verification. CoRR abs/1809.07586, URL http://arxiv.org/abs/1809.07586 14

Rashedi E, Barati E, Nokleby M, Chen Xw (2019) stream loss: Convnet learning for face verification using unlabeled videos in the wild. Neurocomputing 329:311–319 14

Sánchez J, Perronnin F, Mensink T, Verbeek J (2013) Image classification with the fisher vector: Theory and practice. Int J Comput Vision 105(3):222–245 10

Sankaranarayanan S, Alavi A, Castillo C, Chellappa R (2016a) Triplet probabilistic embedding for face verification and clustering. In: Int. Conf. on Biometrics: Theory, Applications and Systems 11, 18

Sankaranarayanan S, Alavi A, Chellappa R (2016b) Triplet similarity embedding for face verification. arxiv preprint arXiv:1602.03418 16, 18

Schroff F, Kalenichenko D, Philbin J (2015) Facenet: A unified embedding for face recognition and clustering. In: Proc. Conf. Comput. Vision Pattern Recognition, pp 815–823 2, 3, 15, 17, 19

Shen Y, Luo P, Yan J, Wang X, Tang X (2018) Faceidgan: Learning a symmetry three-player gan for identity-preserving face synthesis. In: Proc. Conf. Comput. Vision Pattern Recognition, pp 821–830 14

Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: Int. Conf. on Learning Representations 4, 9

Sun Y, Chen Y, Wang X, Tang X (2014a) Deep learning face representation by joint identification-verification. In: Neural Inform. Process. Syst., pp 1988–1996 2

Sun Y, Wang X, Tang X (2014b) Deep learning face representation from predicting 10,000 classes. In: Proc. Conf. Comput. Vision Pattern Recognition, IEEE 2

Sun Y, Wang X, Tang X (2014c) Deeply learned face representations are sparse, selective, and robust. arXiv preprint arXiv:14121265 19

Sun Y, Liang D, Wang X, Tang X (2015) Deepid3: Face recognition with very deep neural networks. arXiv preprint arXiv:150200873 2

Szeliski R (2010) Computer vision: algorithms and applications. Springer Science & Business Media 8

Taigman Y, Yang M, Ranzato M, Wolf L (2014) Deepface: Closing the gap to human-level performance in face verification. In: Proc. Conf. Comput. Vision Pattern Recognition, IEEE, pp 1701–1708 1, 2, 3, 4, 5, 7, 10, 17, 19, 24

Taigman Y, Yang M, Ranzato M, Wolf L (2015) Web-scale training for face identification. In: Proc. Conf. Comput. Vision Pattern Recognition 17, 19

Tran AT, Hassner T, Masi I, Medioni G (2017) Regressing robust and discriminative 3d morphable models with a very deep neural network. In: Proc. Conf. Comput. Vision Pattern Recognition 18, 19

Tran AT, Hassner T, Masi I, Paz E, Nirkin Y, Medioni G (2018) Extreme 3d face reconstruction: Seeing through occlusions. In: Proc. Conf. Comput. Vision Pattern Recognition 20

Wang D, Otto C, Jain AK (2015) Face search at scale: 80 million gallery. arXiv preprint arXiv:1507.07242, URL http://arxiv.org/abs/1507.07242 16, 18

Wen Y, Zhang K, Li Z, Qiao Y (2016) A discriminative feature learning approach for deep face recognition. In: European Conf. Comput. Vision, Springer, pp 499–515 19

Wen Y, Zhang K, Li Z, Qiao Y (2019) A comprehensive study on center loss for deep face recognition. Int J Comput Vision pp 1–16 14

Whitelam C, Taborsky E, Blanton A, Maze B, Adams J, Miller T, Kalka N, Jain AK, Duncan JA, Allen K, Cheney J, Grother P (2017) Iarpa janus benchmark-b face dataset. In: Proc. Conf. Comput. Vision Pattern Recognition Workshops 3, 11, 13, 17, 19

Wolf L, Hassner T, Maoz I (2011a) Face recognition in unconstrained videos with matched background similarity. In: Proc. Conf. Comput. Vision Pattern Recognition, IEEE, pp 529–534 13, 17

Wolf L, Hassner T, Taigman Y (2011b) Effective unconstrained face recognition by combining multiple descriptors and learned background statistics. Trans Pattern Anal Mach Intell 33(10):1978–1990 3, 10

Xie L, Wang J, Wei Z, Wang M, Tian Q (2016) DisturbLabel: Regularizing CNN on the loss layer. In: Proc. Conf. Comput. Vision Pattern Recognition 12

Xie S, Tu Z (2015) Holistically-nested edge detection. In: Proc. Int. Conf. Comput. Vision 3

Xie S, Yang T, Wang X, Lin Y (2015) Hyper-class augmented and regularized deep learning for fine-grained image classification. In: Proc. Conf. Comput. Vision Pattern Recognition, pp 2645–2654 4

Xiong X, De la Torre F (2013) Supervised descent method and its applications to face alignment. In: Proc. Conf. Comput. Vision Pattern Recognition, IEEE 7

Xu Z, Huang S, Zhang Y, Tao D (2015) Augmenting strong supervision using web data for fine-grained categorization. In: Proc. Int. Conf. Comput. Vision, pp 2524–2532 4

Yager N, Dunstone T (2010) The biometric menagerie. Trans Pattern Anal Mach Intell 32(2):220–230 15

Yang H, Patras I (2015) Mirror, mirror on the wall, tell me, is the error small? In: Proc. Conf. Comput. Vision Pattern Recognition 3

Yang J, Ren P, Zhang D, Chen D, Wen F, Li H, Hua G (2017) Neural aggregation network for video face recognition. In: Proc. Conf. Comput. Vision Pattern Recognition 15, 17, 18, 19

Yang Z, Nevatia R (2016) A multi-scale cascade fully convolutional network face detector. In: ICPR, pp 633–638 7, 10, 11

Yi D, Lei Z, Liao S, Li SZ (2014) Learning face representation from scratch. arXiv preprint arXiv:14117923 Available: http://www.cbsr.ia.ac.cn/english/CASIA-WebFace-Database.html 2, 3, 12, 17, 19

Yin X, Yu X, Sohn K, Liu X, Chandraker M (2018) Feature transfer learning for deep face recognition with long-tail data. CoRR abs/1803.09014, URL http://arxiv.org/abs/1803.09014 14

Zhao J, Xiong L, Karlekar Jayashree P, Li J, Zhao F, Wang Z, Sugiri Pranata P, Shengmei Shen P, Yan S, Feng J (2017) Dual-agent GANs for photorealistic and identity preserving profile face synthesis. In: Neural Inform. Process. Syst., pp 66–76 4, 16, 18

Zheng Z, Zheng L, Yang Y (2017) Unlabeled samples generated by GAN improve the person re-identification baseline in vitro. In: Proc. Int. Conf. Comput. Vision 4

Zhou E, Cao Z, Yin Q (2015) Naive-deep face recognition: Touching the limit of LFW benchmark or not? arXiv preprint arXiv:1501.04690, URL http://arxiv.org/abs/1501.04690 2

Zhu X, Lei Z, Liu X, Shi H, Li S (2016) Face alignment across large poses: A 3d solution. In: Proc. IEEE Computer Vision and Pattern Recognition, Las Vegas, NV 24

A Preparing generic 3D heads and backgrounds

When rendering novel views of faces, similarly to Masi et al (2016b), we use ten generic 3D face shapes $\mathscr{S} = \{\mathbf{S}_1, ..., \mathbf{S}_{10}\}$ from the Basel Face Set (Paysan et al, 2009) to model head shape variations. These 3D faces are all aligned with each other and so can easily be manipulated and segmented: Only one shape needs to be manipulated and any modifications to this shape can easily be transfered to all others. Thus, selecting the 3D eye regions in these models to avoid cross-eyed results (Hassner et al, 2015) only needs to be done once.

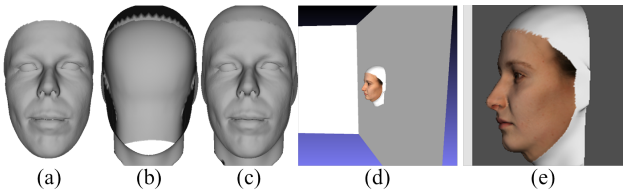


Fig. 12: **Preparing generic 3D models.** Head added to a generic 3D face along with two planes for background.

These models only represent the facial region of the head. Masi et al (2016b) therefore rendered only partial head views, without backgrounds, and this is presumably why Taigman et al (2014) and Hassner et al (2015) only used tight bounding boxes around the face center.

To allow rendering of the entire head and background, we leverage the alignment of these heads to modify them by completing the head shapes and adding a background plane. This is performed by stitching the ten 3D models to an additional, generic 3D structure containing head, ears, and neck and adding a plane representing a flat background.

The process of combining 3D faces to 3D heads is described in Fig. 12. We use the generic 3D head of Zhu et al (2016). We remove its facial region and exchange it with the 3D faces of Paysan et al (2009). To allow blending with different 3D faces (e.g., Fig. 12(a)), varying in sizes and shapes, we maintain an overlap belt with radius r = 2cm (Fig. 12(b)). Given an input 3D face (Fig. 12(a)), we merge it onto the head model using soft boundary blending. We first detect points on the overlap region of the face model. Each point $\mathbf{X}$ is then assigned a soft blending weight w:

$$w = \frac{1}{2} - \frac{1}{2}\cos\left(\frac{\pi d}{r}\right), \tag{5}$$

where d is the distance to the boundary. Next, $\mathbf{X}$ is adjusted to the new 3D position $\mathbf{X}'$ by:

$$\mathbf{X}' = w\mathbf{X} + (1-w)\mathbf{P}_X, \tag{6}$$

where $\mathbf{P}_X$ is the closest 3D point from the head. The result is a complete 3D face model (Fig. 12(c)). Ostensibly, an alternative to this method would be to scan new models, with complete 3D heads. Besides requiring less labor for scanning and alignment, the method described above was selected in order to minimize the differences between our recognition system and the one used by Masi et al (2016b), including the use of the same 3D face shapes from in their system.

To additionally preserve the background, we simply add two planes to the 3D model: one positioned just behind the head and another, perpendicular plane, on its right. This second plane is used to represent the background when the input face is rendered to a profile view, in which case the first plane is mapped to a line. Fig. 12(d) shows the models we produced using one of the 3D face shapes of Paysan et al (2009). Fig. 12(e) shows the rendered view of this generic face from the profile pose used by our system.

B Fast 3D rendering snippet code

Given an input image $\mathbf{I}$ containing a face in unconstrained settings, we use the following simple procedure to render it to a desired new view using $\mathbf{U}$. The code in Fig. 13 for a Python code example explains the process.

```
import numpy as np
import cv2
U_bar = np.vstack((U, \
        np.ones((1, threedee.shape[1]))))
q_bar = MI * U_hat
q = np.divide(q_bar[0:2, :], \
        np.tile(q_bar[2, :], (2,1)))
#idx are indices of q inside the image
q=q[idx]
synth = warpImg(I, N, M, q, idx)
def warpImg(I, N, M, q, idx):
  J = np.zeros((N*M, 3))
  #fast warping
  pixels = cv2.remap(I, \
  np.asarray( q[0,:] ).astype('float32'),\
  np.asarray( q[1,:] ).astype('float32'),\
  cv2.INTER_LINEAR)
  #copy the interpolated pixel back ( Eq. (1) )
  J[idx,:] = pixels
  J = J.reshape(( N, M, 3), order='F')
  J = J.astype('uint8')
  return J
```

Fig. 13: **Python code snippet for 3D rendering** at 2D image warping speed.