FaceNet

Florian Schroff, Dmitry Kalenichenko, James Philbin

Google Inc.

Presentation by Ignacio Aranguren and Rahul Rana

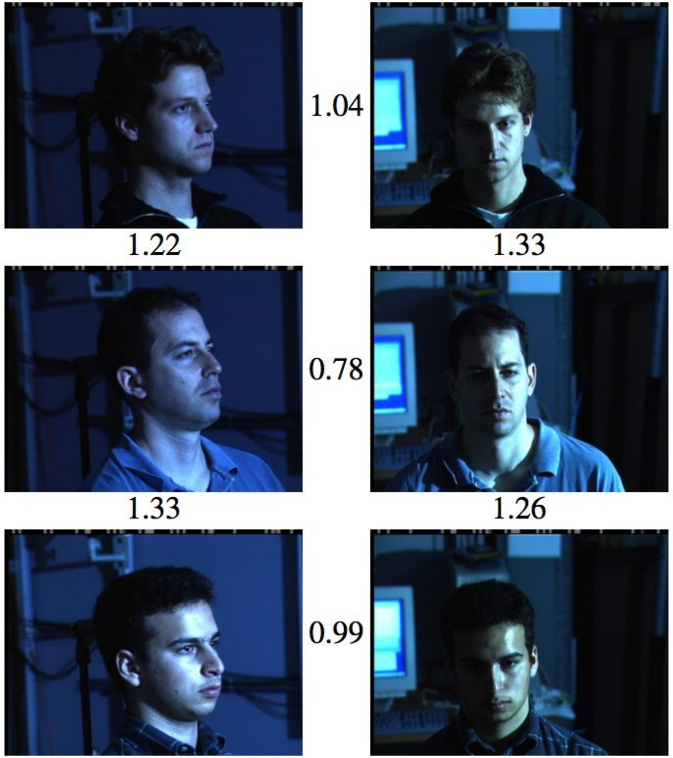
Introduction

FaceNet learns a mapping from face images to a compact Euclidean Space where distances directly correspond to a measure of face similarity. Once this is done, tasks such as face recognition, verification, and clustering are easy to do using standard techniques (using the FaceNet embeddings as features).

Uses a Deep CNN trained to optimize the embedding itself, rather than using the output of an intermediate bottleneck layer.

Training is done using triplets: one image of a face ('anchor'), another image of that same face ('positive exemplar'), and an image of a different face ('negative exemplar').

Main benefit is representational efficiency: can achieve state-of-the-art performance (record 99.63% accuracy on LFW, 95.12% on Youtube Faces DB) using only 128-bytes per face.

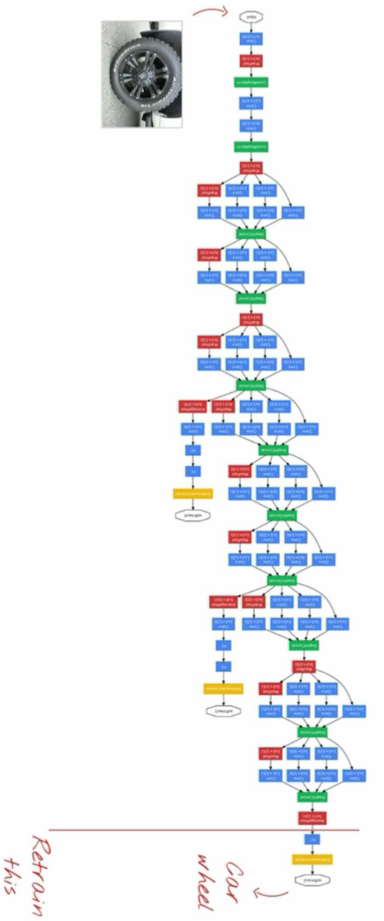


Related Work - Facial Recognition

Previous face recognition approaches based on deep networks use a classification layer trained over a set of known face identities and then take an intermediate bottleneck layer as a representation used to generalize recognition beyond the set of identities used in training. Some of these then combine the output of a CNN with PCA for dimensionality reduction and SVM for classification.

Approaches such as those of Zhenyao *et al* [1] and the DeepFace group at Facebook [2] first “warp” or “align” faces into a more amenable form (either ‘canonical frontal view’ or DeepFaces general 3D model) and then learn a CNN to classify each face as belonging to an identity.

The architectures explored using FaceNet are based on either the Zeiler&Fergus [3] model or Szegedy *et al*.’s *Inception* [4] model (which recently won the ImageNet competition in 2014).



Related Work - Triple Loss

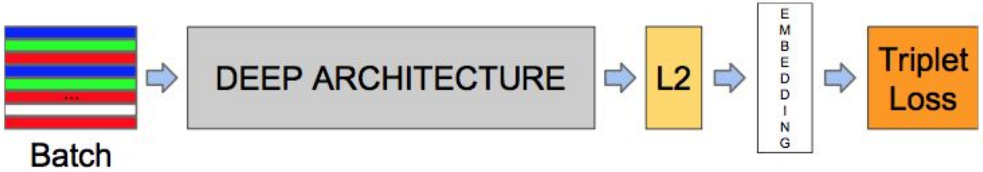
The triplet-based loss function used to learn the mapping is an adaptation of Kilian Weinberger's Large Margin Nearest Neighbor (LMNN) classifier [5] (which repeatedly pulls together images of the same person and simultaneously pushes images of any different person away) to deep neural networks.

Sun *et al*. [6] use ensembles of networks trained using a combination of classification and verification loss. The verification loss they use is similar to the triplet loss used to learn the mapping used by FaceNet in that it minimizes squared L_2 distances between images of faces from the same person and enforces a margin separating images of faces from a different person, but it's different in that only pairs of images are compared, whereas the triplet loss encourages a *relative* distance constraint by looking at three at a time.

A loss similar to FaceNet's triple loss was used by Wang *et al.* [7] for ranking images by semantic and visual similarity.

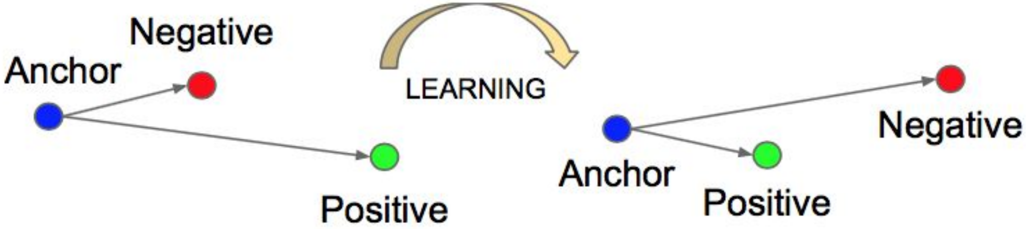
Method - Overview

Treating the CNN architecture as a blackbox, the most important part of FaceNet lies in the end-to-end learning of the system.



FaceNet looks for an embedding $f(x)$ from an image into feature space $\mathbb{R}^d$, such that the squared L_2 distance between all face images (independent of imaging conditions) of the same identity is small, whereas the distance between a pair of face images from different identities is large.

Whereas previously used losses encourage all faces of the same identity onto a single point in $\mathbb{R}^d$, the triplet loss additionally tries to enforce a margin between each pair of faces from one person (anchor and positive) to all others' faces. This margin enforces discriminability to other identities.



Method - Triplet Loss

We want to ensure that an image x_i^a of a specific person is closer to all other images x_i^p of that same person than it is to any image x_i^n of any other person *by a margin* α. That is,

$$\|x_i^a - x_i^p\|_2^2 + \alpha < \|x_i^a - x_i^n\|_2^2, \ \forall \left(x_i^a, x_i^p, x_i^n\right) \in \mathcal{T}$$

Therefore, the loss (L) is:

$$\sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha\right]_+ \qquad \alpha = 0.2$$

Of all possible triplets (N of them), many would easily satisfy the above constraint. So it'd be a waste to look at these during training (wouldn't contribute to adjusting parameters, would only slow down convergence); it's therefore important to select "hard" triplets (which would contribute to improving the model) to use in training. How do we do that?

Method - Triplet Selection

An idea: Given an anchor image x_i^a, select the "hardest" positive image (of the same person) as x_i^p (i.e. the one that's furthest away in the dataset) and select the "hardest" negative image (of a different person) as x_i^n (i.e. the one that's closest in the dataset). If this triplet doesn't violate condition, then none with that anchor will. (Think: if $d_- - d_+ > \boldsymbol{\alpha}$, then the condition is met.)

Problem: Infeasible to compute these argmax and argmin across the *whole* dataset. Also this might lead to poor training (considering that mislabelled and poorly imaged faces would dominate the hard positives and negatives).

To avoid this: Generate triplets online. That is, select x_i^p and x_i^n (argmax and argmin) *from a mini-batch* (not from the *entire* dataset) for x_i^a.

Batch details: They sample training data such that around 40 images are selected per identity for each mini-batch (to ensure a meaningful representation of the anchor-positive distances), and randomly sample negative faces for each mini-batch. Instead of picking the "hardest" positive for a given anchor, they used all the anchor-positive pairs within the batch while still selecting hard negatives (one to correspond to each anchor); they do this because they found this leads to a more stable and faster-converging solution.

Zeiler&Fergus-Inspired Architecture

- Consists of multiple interleaved layers of convolutions, non-linear activations, local response normalizations, and max pooling layers (with several additional 1x1xd convolutional layers throughout).
- 1x1 conv layer is inspired by the cross-channel parametric pooling.

layer	**size-in**	**size-out**	**kernel**	**param**	**FLPS**
conv1	$220\times220\times3$	$110\times110\times64$	$7\times7\times3, 2$	9K	115M
pool1	$110\times110\times64$	$55\times55\times64$	$3\times3\times64, 2$	0	
rnorm1	$55\times55\times64$	$55\times55\times64$		0	
conv2a	$55\times55\times64$	$55\times55\times64$	$1\times1\times64, 1$	4K	13M
conv2	$55\times55\times64$	$55\times55\times192$	$3\times3\times64, 1$	111K	335M
rnorm2	$55\times55\times192$	$55\times55\times192$		0	
pool2	$55\times55\times192$	$28\times28\times192$	$3\times3\times192, 2$	0	
conv3a	$28\times28\times192$	$28\times28\times192$	$1\times1\times192, 1$	37K	29M
conv3	$28\times28\times192$	$28\times28\times384$	$3\times3\times192, 1$	664K	521M
pool3	$28\times28\times384$	$14\times14\times384$	$3\times3\times384, 2$	0	
conv4a	$14\times14\times384$	$14\times14\times384$	$1\times1\times384, 1$	148K	29M
conv4	$14\times14\times384$	$14\times14\times256$	$3\times3\times384, 1$	885K	173M
conv5a	$14\times14\times256$	$14\times14\times256$	$1\times1\times256, 1$	66K	13M
conv5	$14\times14\times256$	$14\times14\times256$	$3\times3\times256, 1$	590K	116M
conv6a	$14\times14\times256$	$14\times14\times256$	$1\times1\times256, 1$	66K	13M
conv6	$14\times14\times256$	$14\times14\times256$	$3\times3\times256, 1$	590K	116M
pool4	$14\times14\times256$	$7\times7\times256$	$3\times3\times256, 2$	0	
concat	$7\times7\times256$	$7\times7\times256$		0	
fc1	$7\times7\times256$	$1\times32\times128$	maxout p=2	103M	103M
fc2	$1\times32\times128$	$1\times32\times128$	maxout p=2	34M	34M
fc7128	$1\times32\times128$	$1\times1\times128$		524K	0.5M
L2	$1\times1\times128$	$1\times1\times128$		0	
total				140M	1.6B

Inception-Inspired Architecture

type	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj (p)	params	FLOPS
conv1 (7×7×3, 2)	112×112×64	1							9K	119M
max pool + norm	56×56×64	0						m 3×3, 2		
inception (2)	56×56×192	2		64	192				115K	360M
norm + max pool	28×28×192	0						m 3×3, 2		
inception (3a)	28×28×256	2	64	96	128	16	32	m, 32p	164K	128M
inception (3b)	28×28×320	2	64	96	128	32	64	L_2, 64p	228K	179M
inception (3c)	14×14×640	2	0	128	256,2	32	64,2	m 3×3,2	398K	108M
inception (4a)	14×14×640	2	256	96	192	32	64	L_2, 128p	545K	107M
inception (4b)	14×14×640	2	224	112	224	32	64	L_2, 128p	595K	117M
inception (4c)	14×14×640	2	192	128	256	32	64	L_2, 128p	654K	128M
inception (4d)	14×14×640	2	160	144	288	32	64	L_2, 128p	722K	142M
inception (4e)	7×7×1024	2	0	160	256,2	64	128,2	m 3×3,2	717K	56M
inception (5a)	7×7×1024	2	384	192	384	48	128	L_2, 128p	1.6M	78M
inception (5b)	7×7×1024	2	384	192	384	48	128	m, 128p	1.6M	78M
avg pool	1×1×1024	0								
fully conn	1×1×128	1							131K	0.1M
L2 normalization	1×1×128	0								
total									7.5M	1.6B

Datasets and Evaluation

The model is evaluated on 4 different datasets & these parameters are evaluated:

$$\mathrm{TA}(d) = \{(i,j) \in \mathcal{P}_{\mathrm{same}}, \text{with } D(x_i, x_j) \le d\} \qquad \mathrm{FA}(d) = \{(i,j) \in \mathcal{P}_{\mathrm{diff}}, \text{with } D(x_i, x_j) \le d\}$$

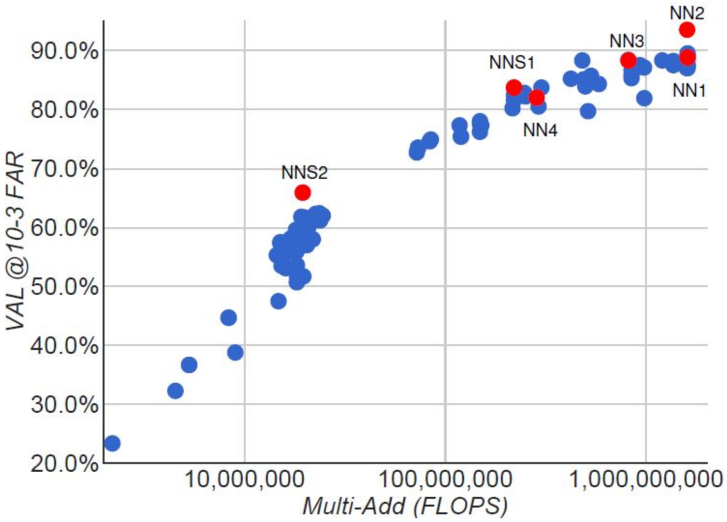
$$\mathrm{VAL}(d) = \frac{|\mathrm{TA}(d)|}{|\mathcal{P}_{\mathrm{same}}|} \qquad \mathrm{FAR}(d) = \frac{|\mathrm{FA}(d)|}{|\mathcal{P}_{\mathrm{diff}}|}$$

1. Hold-out Test Set: 1M images having the same distribution as the training set. Divided into 5 subsets. VAL and FAR are calculated on 100k x 100k image pairs.
2. Personal Photos: 12k images with FAR and VAL calculated for 12k x 12k image pairs.
3. Labeled Faces in the Wild (LFW): de-facto academic test set for face recognition. FAR and VAL are not calculated.
4. Youtube Faces DB: setup is similar to LFW, but pairs of videos instead of images are used. FAR and VAL are not calculated.

Experiments - Computation vs. Accuracy Trade-off

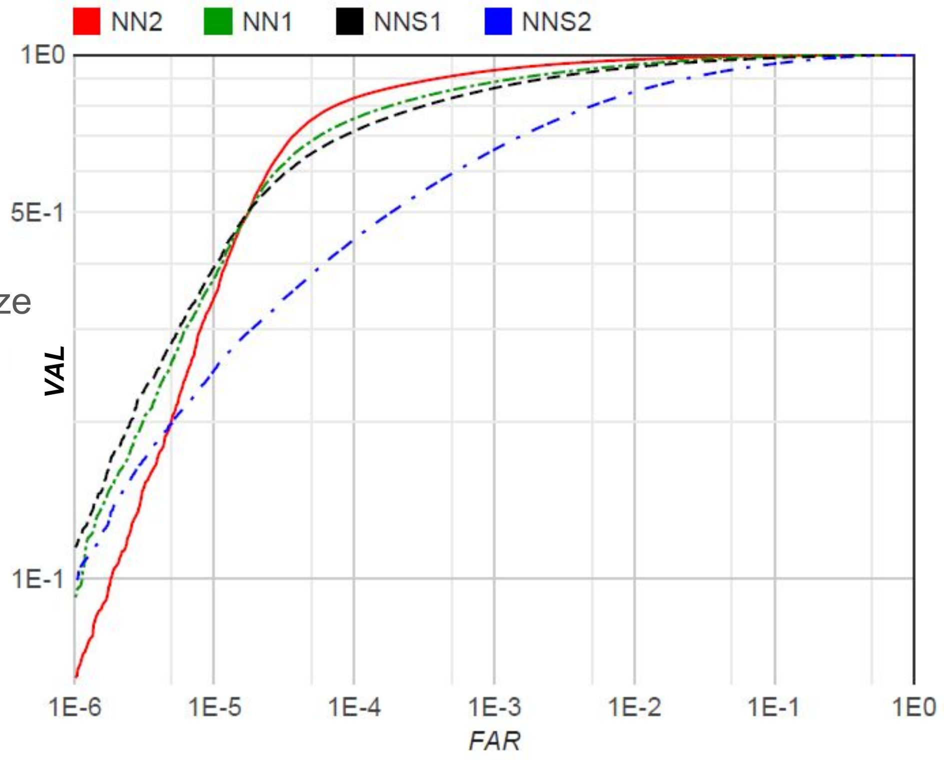
- 100M - 200M images training face thumbnails, having 8M identities are used.
- Pre-processing: detecting faces and generating a tight bound box around each face. Resized depending on the input sizes of the networks varying from 96x96 to 224x224.
- There is tradeoff b/w accuracy vs FLOPS.
- The graph shows a strong correlation between FLOPS & accuracy achieved.
- There isn't a correlation b/w accuracy vs no. of parameters.
- NN2 achieves comparable performance to NN1 with 20th of parameters but similar FLOPS.

architecture	VAL
NN1 (Zeiler&Fergus 220×220)	87.9% ± 1.9
NN2 (Inception 224×224)	89.4% ± 1.6
NN3 (Inception 160×160)	88.3% ± 1.7
NN4 (Inception 96×96)	82.0% ± 2.3
NNS1 (mini Inception 165×165)	82.4% ± 2.4
NNS2 (tiny Inception 140×116)	51.9% ± 2.9



Effect of CNN Model

- Zeiler&Fergus [3] based architectures (NN1) and GoogLeNet based Inception model [4] (NN2) differ in number of parameters by a factor of 20. But they achieve comparable performance.
- NNS2, a tiny version of NN2, having input size of 140x116 model can be run on a mobile phone at 30ms / image and be good enough for face recognition. VAL = 51.9%



Sensitivity to Image Quality

- Their models are robust to JPEG compression and perform well even at a JPEG quality of 20.

jpeg q	val-rate
10	67.3%
20	81.4%
30	83.9%
50	85.5%
70	86.1%
90	86.5%

- Performance drop is very less with 120x120 input image size and remains acceptable even at 80x80.

#pixels	val-rate
1,600	37.8%
6,400	79.5%
14,400	84.5%
25,600	85.7%
65,536	86.4%

Embedding Dimensionality

- They experimented with a lot of dimensionalities and chose 128-D, as it was the best performing.
- It was expected that the larger dimensionalities would perform better, but it could also mean that they require more training.
- During training a 128-D float vector is used which is quantized to 128-byte vector without loss of accuracy.
- Smaller embedding dimensions could be employed on mobile devices, with minor loss of accuracy.

#dims	VAL
64	$86.8\% \pm 1.7$
128	$87.9\% \pm 1.9$
256	$87.7\% \pm 1.9$
512	$85.6\% \pm 2.0$

Amount of Training Data

- Experiments were also conducted with number of training samples.
- Smaller model with input size of 96x96 was employed for this analysis. It has same architecture as NN2 but without the 5x5 conv. in the inception module.

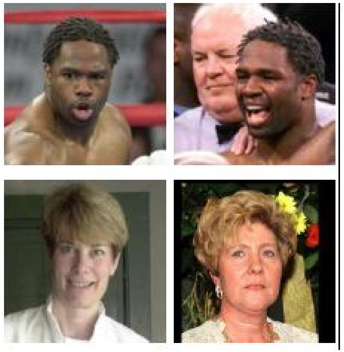
#training images	VAL
2,600,000	76.3%
26,000,000	85.1%
52,000,000	85.1%
260,000,000	86.2%

- Using only 10s of millions of images gives really good results, but with 100s of millions of images, the improvement starts to taper.

Performance on LFW

- The optimal threshold used for L_2 distance calculation is 1.242..
- The input data is pre-processed in 2 ways:
  a. Fixed center crop of the LFW provided thumbnails.
  b. Face detection using proprietary detector. If that does not align, then LFW alignment is used.
- The accuracy achieved with a is 98.87%, while with b is 99.63% (state-of-the-art)

False Accept



False Reject

Performance on Youtube Faces DB

- Average similarity of all pairs of faces in the first 100 frames that are detected by their proprietary face detector, are used.
- Classification accuracy achieved is 95.12% (state-of-the-art).
- Using first 1000 frames, accuracy achieved is 95.18%, not an improvement.
- Previous efforts DeepId2+ (Sun *et al.*) had achieved 93.2%.

Face Clustering

The compact embeddings are used to cluster photos of people with the same identity, using agglomerative clustering.

Incredibly, it is invariant to occlusion, lighting, pose and even age.

Summary and Conclusions

Innovation: Triplet Loss adapted to deep neural networks, used to map images to low-dimensional space.

Value:

1. state-of-the-art face recognition performance using only 128-bytes per face.
2. Minimal alignment required on the input dataset (tight crop around the face area), unlike DeepFace (FAIR) which performs 3D alignment.

Future Scope:

1. Understand the error cases and improve the model further.
2. Reduce the model size and computational requirements.
3. Improve the long-training time by varying curriculum learning & mining offline.

Works Cited

[1] Z. Zhu, P. Luo, X. Wang, and X. Tang. Recover canonical- view faces in the wild with deep neural networks. *CoRR*, abs/1404.3543, 2014. 2

[2] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *IEEE Conf. on CVPR*, 2014. 1, 2, 5, 8

[3] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. 2, 4, 6

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.2,4,5,6,9

[5] K.Q. Weinberger, J.Blitzer,and L.K.Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*. MIT Press, 2006. 2, 3

[6] Y. Sun, X. Wang, and X. Tang. Deeply learned face representations are sparse, selective, and robust. *CoRR*, abs/1412.1265, 2014. 1, 2, 5, 8

[7] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. *CoRR*, abs/1404.4661, 2014. 2