110-2-one-shot-facial-2020-7-27.pptx

CTI One Corporation

Date: July 27, 2020
Project Lead: Harry Li, Ph.D.

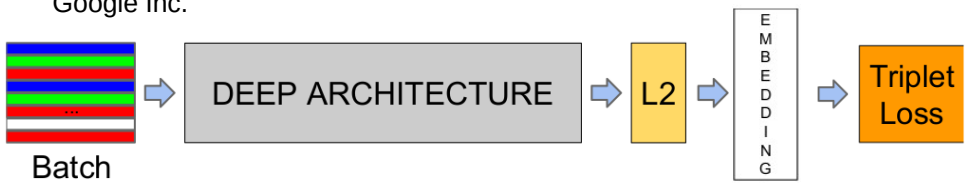
Team Member:

FaceNet from Google

A Unified Embedding for Face Recognition and Clustering

Florian Schroff Dmitry Kalenichenko James Philbin
fschroff@google.com dkalenichenko@google.com
jphilbin@google.com
Google Inc.

FaceNet learns the mapping from the images and creates embeddings rather than using any output layer for recognition



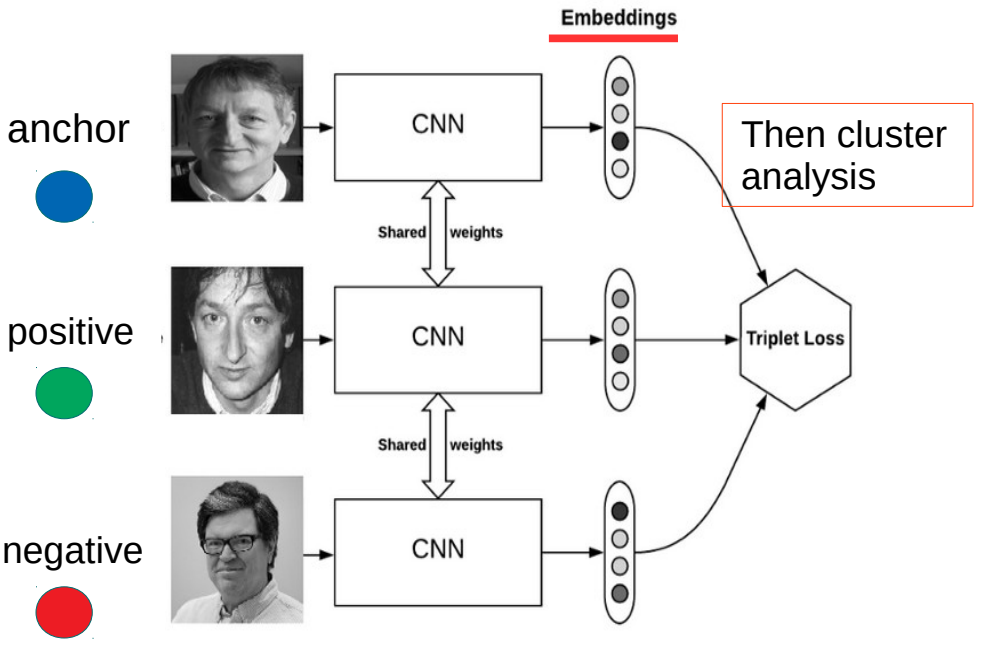
Block 1: Batch input layer

Block 2: deep CNN

Block 3: L2 distance e.g., normalization

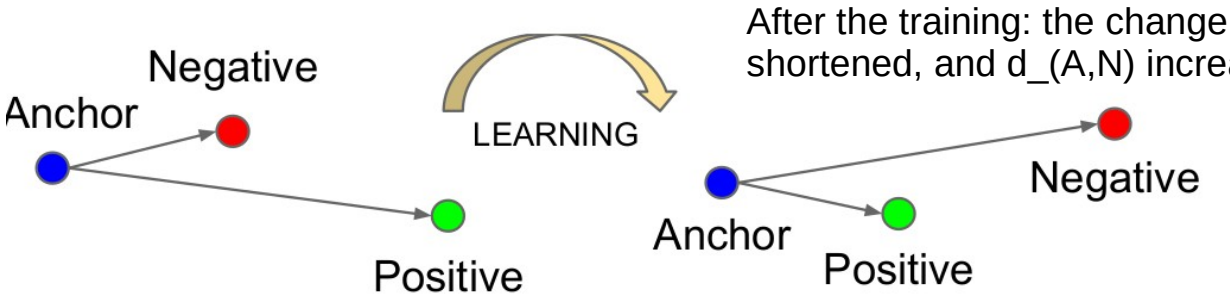
L2, e.g, the Euclidean norm, a positive distance, calculated as the square root of the sum of the squared vector values.

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



Then cluster analysis

Triplet Loss Function



After the training: the change of the distance d_(A,P) shortened, and d_(A,N) increased

The objective:

$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2 \qquad \ldots (1)$$

$$\forall\, (f(x_i^a), f(x_i^p), f(x_i^n)) \in \mathcal{T} \qquad \ldots (2)$$

The loss function to be minimized:

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

$$\ldots (3)$$

CNN1 Architecture

Table 1. NN1.
This table show the structure of our
Zeiler&Fergus [22] based model with 1×1 convolutions inspired by [9].
The input and output sizes are described
in rows × cols × #f ilters.
The kernel is specified as
rows × cols, stride and the maxout [6] pooling size as p = 2.

layer	size-in	size-out	kernel	param	FLPS
conv1	220×220×3	110×110×64	7×7×3, 2	9K	115M
pool1	110×110×64	55×55×64	3×3×64, 2	0	
rnorm1	55×55×64	55×55×64		0	
conv2a	55×55×64	55×55×64	1×1×64, 1	4K	13M
conv2	55×55×64	55×55×192	3×3×64, 1	111K	335M
rnorm2	55×55×192	55×55×192		0	
pool2	55×55×192	28×28×192	3×3×192, 2	0	
conv3a	28×28×192	28×28×192	1×1×192, 1	37K	29M
conv3	28×28×192	28×28×384	3×3×192, 1	664K	521M
pool3	28×28×384	14×14×384	3×3×384, 2	0	
conv4a	14×14×384	14×14×384	1×1×384, 1	148K	29M
conv4	14×14×384	14×14×256	3×3×384, 1	885K	173M
conv5a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv5	14×14×256	14×14×256	3×3×256, 1	590K	116M
conv6a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv6	14×14×256	14×14×256	3×3×256, 1	590K	116M
pool4	14×14×256	7×7×256	3×3×256, 2	0	
concat	7×7×256	7×7×256		0	
fc1	7×7×256	1×32×128	maxout p=2	103M	103M
fc2	1×32×128	1×32×128	maxout p=2	34M	34M
fc7128	1×32×128	1×1×128		524K	0.5M
L2	1×1×128	1×1×128		0	
total				140M	1.6B

One way to reduce the training time is to normalize the activities of the neurons.

https://arxiv.org/abs/1607.06450

CNN2 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

CNN2 Architecture

Details of the NN2 Inception architecture. This model uses L 2 pooling instead of max pooling (m), where specified. The pooling is always 3×3 (aside from the final average pooling) and in parallel to the convolutional modules inside each Inception module. If there is a dimensionality reduction after the pooling it is denoted with p. 1×1, 3×3, and 5×5 pooling are then concatenated to get the final output.

"(Inception Layer) is a combination of all those layers (namely, 1×1 Convolutional layer, 3×3 Convolutional layer, 5×5 Convolutional layer) with their output filter banks concatenated into a single output vector forming the input of the next stage."

https://www.analyticsvidhya.com/blog/2018/10/understanding-inception-network-from-scratch/