Deep Learning for Computer Vision

CNNs for Human Understanding: Faces

Vineeth N Balasubramanian

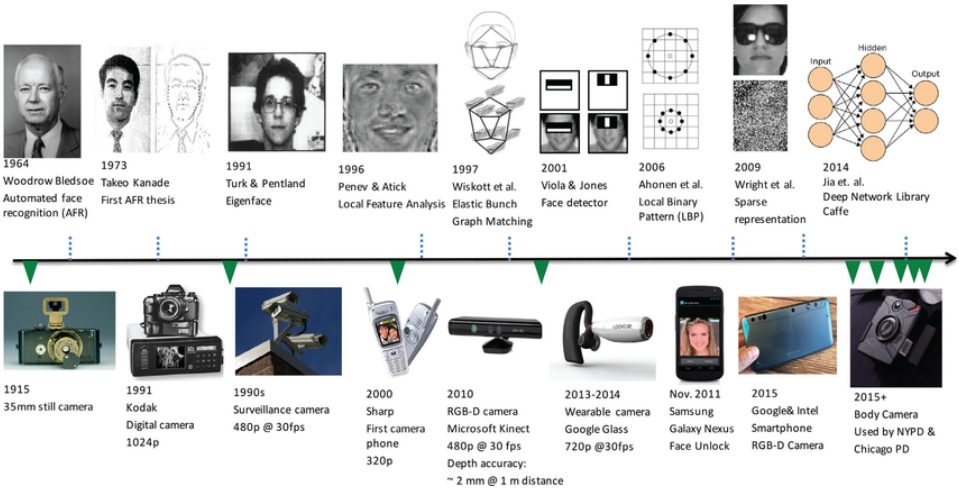
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Face Recognition

- Face Recognition (FR): Core vision task with applications in security, finance, healthcare and many other areas of social and financial importance
- Unconstrained FR challenging due to variations in lighting, occlusions, pose/alignment, expressions, etc

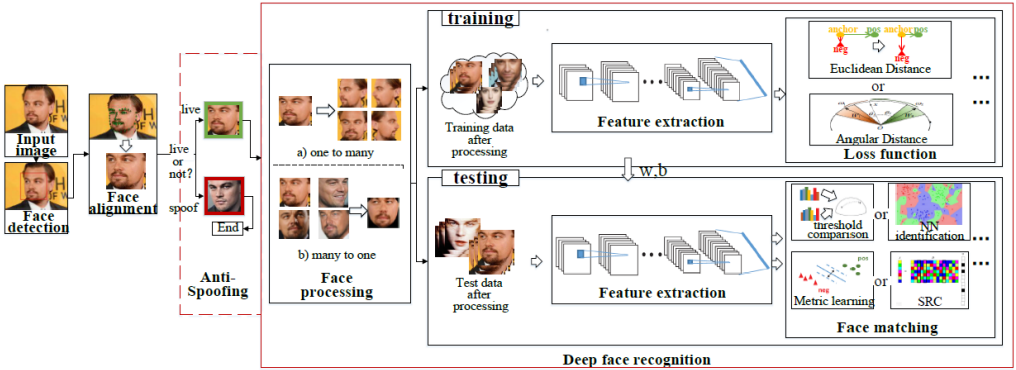
Credit: VGGFace2 Dataset

Face Recognition: History



Credit: Jain et al, 50 Years of Biometric Research: Accomplishments, Challenges, and Opportunities, PR Letters 2016

Face Recognition

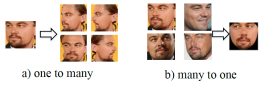


- A typical deep FR system's pipeline is shown above
- Face Recognition = Face Detection + Face Alignment + Face Matching

Credit: Wang et al, Deep Face Recognition: A Survey, 2018

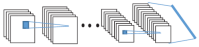
Face Recognition System: Key Components

- **Face Processing**
  - One-to-many Augmentation
  - Many-to-one Normalization
- **Deep Feature Extraction**
  - Network Architecture
  - Loss Function
- **Face Matching** by Deep Features

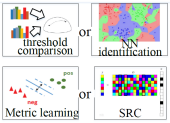


Face processing

Loss function



Feature extraction using deep conv. networks



Face matching

Credit: Wang et al, Deep Face Recognition: A Survey, 2018

Face (Pre-)Processing

- **One-to-many Augmentation**
  - Generating many patches or images of pose variations from a single image (through rotations, for e.g.)
- **Many-to-one Normalization**
  - Attempts to recover canonical view of face images from one or many images of a non-frontal view
  - Can help in preserving identity despite variations in pose, lighting, expression and background

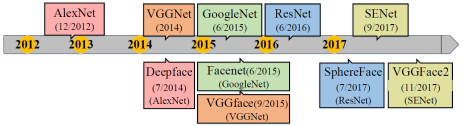
One-to-many Augmentation

Many-to-one Normalization

Credit: Wang et al, A Survey on Face Data Augmentation, Neural Computing and Applications, 2019; Qian et al, Unsupervised Face Normalization With Extreme Pose and Expression in the Wild, CVPR 2019

Network Architectures for Face Recognition

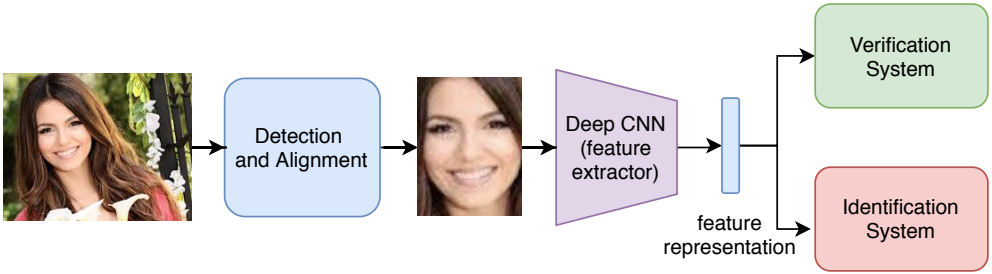
- Few special CNN architectures proposed for deep FR
- However, most successful backbone networks in deep FR shaped around then SOTA deep object classification networks



Method	Public. Time	Loss	Architecture	Number of Networks	Training Set	Accuracy±Std(%)
DeepFace [195]	2014	softmax	Alexnet	3	Facebook (4.4M,4K)	97.35±0.25
DeepID2 [187]	2014	contrastive loss	Alexnet	25	CelebFaces+ (0.2M,10K)	99.15±0.13
DeepID3 [188]	2015	contrastive loss	VGGNet-10	50	CelebFaces+ (0.2M,10K)	99.53±0.10
FaceNet [176]	2015	triplet loss	GoogleNet-24	1	Google (500M,10M)	99.63±0.09
Baidu [124]	2015	triplet loss	CNN-9	10	Baidu (1.2M,18K)	99.77
VGGface [149]	2015	triplet loss	VGGNet-16	1	VGGface (2.6M,2.6K)	98.95
light-CNN [225]	2015	softmax	light CNN	1	MS-Celeb-1M (8.4M,100K)	98.8
Center Loss [218]	2016	center loss	Lenet+-7	1	CASIA-WebFace, CACD2000, Celebrity+ (0.7M,17K)	99.28
L-softmax [126]	2016	L-softmax	VGGNet-18	1	CASIA-WebFace (0.49M,10K)	98.71
Range Loss [261]	2016	range loss	VGGNet-16	1	MS-Celeb-1M, CASIA-WebFace (5M,100K)	99.52
L2-softmax [157]	2017	L2-softmax	ResNet-101	1	MS-Celeb-1M (3.7M,58K)	99.78
Normface [206]	2017	contrastive loss	ResNet-28	1	CASIA-WebFace (0.49M,10K)	99.19
CoCo loss [130]	2017	CoCo loss	–	1	MS-Celeb-1M (3M,80K)	99.86
vMF loss [75]	2017	vMF loss	ResNet-27	1	MS-Celeb-1M (4.6M,60K)	99.58
Marginal Loss [43]	2017	marginal loss	ResNet-27	1	MS-Celeb-1M (4M,80K)	99.48
SphereFace [125]	2017	A-softmax	ResNet-64	1	CASIA-WebFace (0.49M,10K)	99.42
CCL [155]	2018	center invariant loss	ResNet-27	1	CASIA-WebFace (0.49M,10K)	99.12
AMS loss [205]	2018	AMS loss	ResNet-20	1	CASIA-WebFace (0.49M,10K)	99.12
Cosface [207]	2018	cosface	ResNet-64	1	CASIA-WebFace (0.49M,10K)	99.33
Arcface [42]	2018	arcface	ResNet-100	1	MS-Celeb-1M (3.8M,85K)	99.83
Ring loss [272]	2018	Ring loss	ResNet-64	1	MS-Celeb-1M (3.5M,31K)	99.50

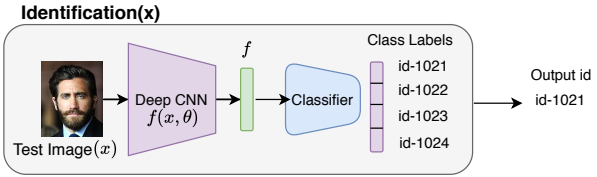
Credit: Wang et al, Deep Face Recognition: A Survey, 2018

Face Recognition: Verification and Identification



- Face recognition can be broadly divided into two tasks:
  - **Face verification**
  - **Face identification**

Face Identification



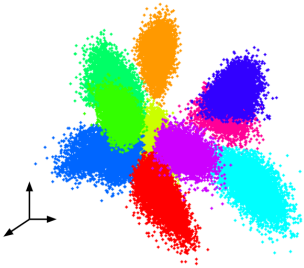
- Assign input image to person name/id from database (one-to-many matching)
- Formulated as K+1 multi-class classification (one additional class for unrecognized faces)
- **Input**: Face image
  Output: Identity class/face ID

Softmax + CrossEntropy

- Consider last linear layer W, b parameterizes the subjects i.e., maps feature embeddings on to subject labels, cross-entropy loss is

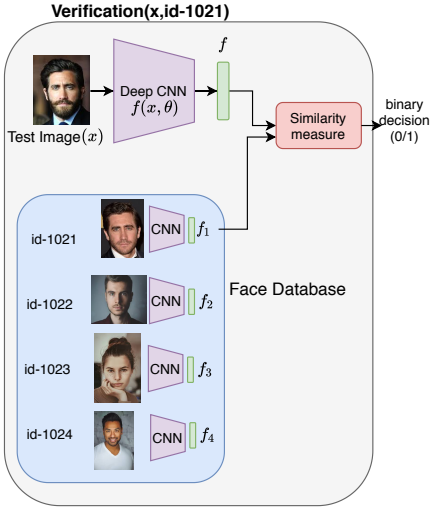
$$\mathcal{L}_{CE} = -\log\left(\frac{e^{W_i x^\intercal + b_i}}{\sum_j e^{W_j x^\intercal + b_j}}\right)$$

- Softmax+CE produces feature embeddings that geometrically looks like ellipsoids i.e., large intra-class variance
- The loss enforces good classification (nice boundaries between classes i.e., small inter-class variance) but it does not enforce small intra-class variance



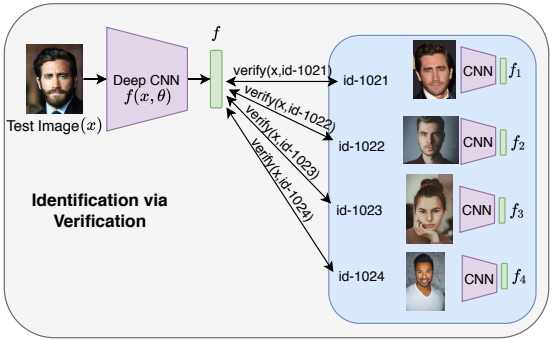
Source: Maci et al. Deep Face Recognition: A Survey, SIBGRAPI'18. [9]

Face Verification



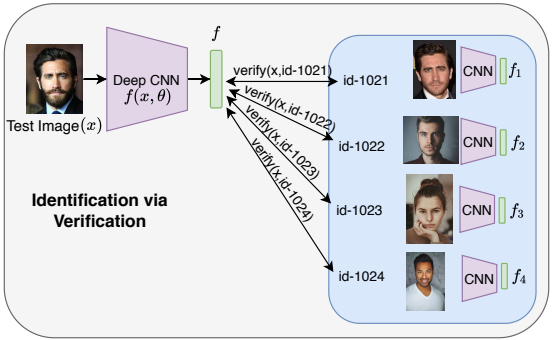
- Verifying whether two images belong to the same identity
- Ascertain whether image is of claimed person/id (one-to-one matching)
- **Input**: Face image, Face ID
  Output: Match/Not match (binary classification)

Identification via Verification



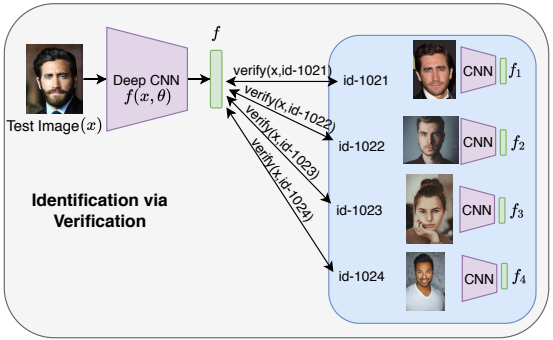
- Identification can be solved via verification approach

Identification via Verification



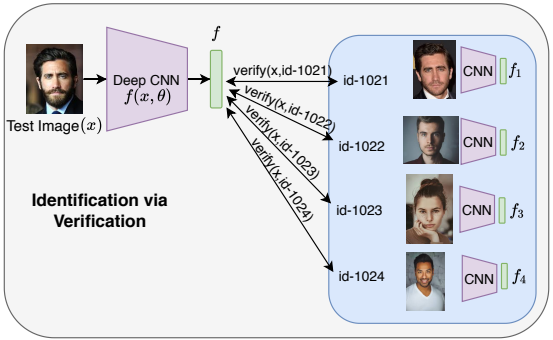
- Identification can be solved via verification approach
- Removes need to retrain model on addition of new face classes to the database $\implies$ more practical/feasible

Identification via Verification



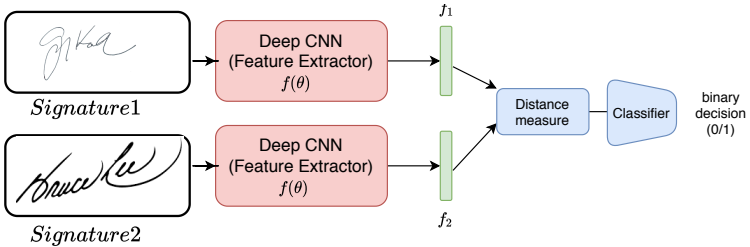
- Identification can be solved via verification approach
- Removes need to retrain model on addition of new face classes to the database $\implies$ more practical/feasible
- Identification involves multiple verification steps $\implies$ error gets amplified

Identification via Verification



- Identification can be solved via verification approach
- Removes need to retrain model on addition of new face classes to the database $\implies$ more practical/feasible
- Identification involves multiple verification steps $\implies$ error gets amplified
- Goal becomes to build an accurate/efficient verification system via learning a robust similarity metric

Verification: Siamese Networks[1]



- First proposed for signature verification in 1994
- Two replicas of same architecture parametrized with same weights working in tandem on different inputs
- Network parameters learned via some form of distance measure to extract distinctive features - an idea that is used even today

[1] Bromley et al, Signature Verification using a Siamese Time Delay Neural Network, NIPS 1994

DeepFace: Identification[2]

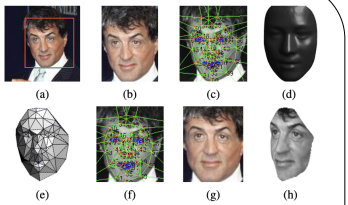


Figure 1. **Alignment pipeline.** (a) The detected face, with 6 initial fiducial points. (b) The induced 2D-aligned crop. (c) 67 fiducial points on the 2D-aligned crop with their corresponding Delaunay triangulation, we added triangles on the contour to avoid discontinuities. (d) The reference 3D shape transformed to the 2D-aligned crop image-plane. (e) Triangle visibility w.r.t. to the fitted 3D-2D camera; darker triangles are less visible. (f) The 67 fiducial points induced by the 3D model that are used to direct the piece-wise affine warpping. (g) The final frontalized crop. (h) A new view generated by the 3D model (not used in this paper).

- **Step 1:** Face localization, fiducial point detection and alignment $\implies$ frontal crop of face
- **Step 2:** Frontal crop passed for identification to deep CNN model with K-way softmax (multi-class classification)

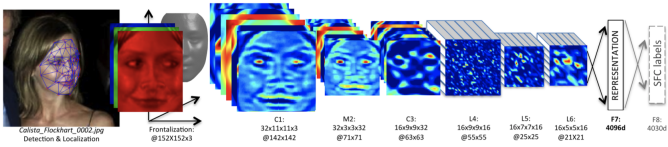
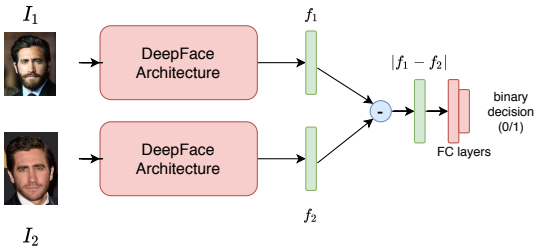


Figure 2. **Outline of the *DeepFace* architecture**. A front-end of a single convolution-pooling-convolution filtering on the rectified input, followed by three locally-connected layers and two fully-connected layers. Colors illustrate feature maps produced at each layer. The net includes more than 120 million parameters, where more than 95% come from the local and fully connected layers.

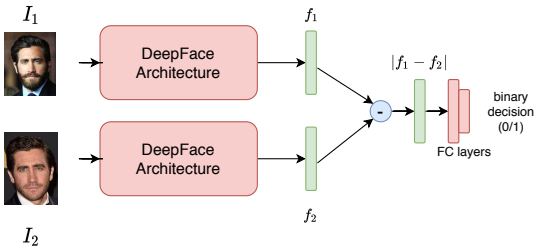
[2] Taigman et al, DeepFace: Closing the Gap to Human-Level Performance in Face Verification, CVPR 2014

DeepFace: Verification (Siamese Networks)



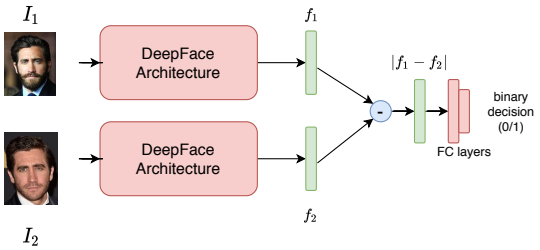
- Representation learned from identification used for verification $\implies$ freeze classification parameters

DeepFace: Verification (Siamese Networks)



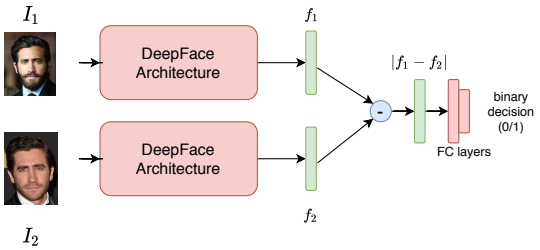
- Representation learned from identification used for verification $\implies$ freeze classification parameters
- Given a face image I, $f = G(I)$ $\implies$ penultimate layer representation for I

DeepFace: Verification (Siamese Networks)



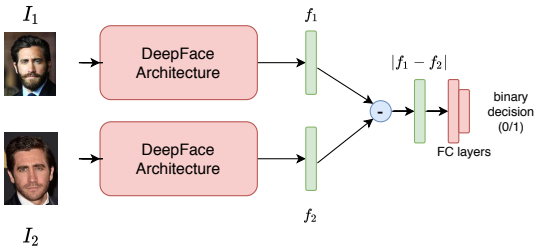
- Representation learned from identification used for verification $\implies$ freeze classification parameters
- Given a face image I, $f = G(I)$ $\implies$ penultimate layer representation for I
- Network trained by taking absolute difference between features, followed by a fully connected head

DeepFace: Verification (Siamese Networks)



- Representation learned from identification used for verification $\implies$ freeze classification parameters
- Given a face image I, $f = G(I)$ $\implies$ penultimate layer representation for I
- Network trained by taking absolute difference between features, followed by a fully connected head
- Distance induced: $d(f_1, f_2) = \sum_i \alpha_i |f_1[i] - f_2[i]|$ where α_i are trainable parameters

DeepFace: Verification (Siamese Networks)



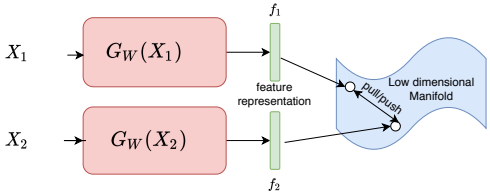
- Representation learned from identification used for verification $\implies$ freeze classification parameters
- Given a face image I, $f = G(I)$ $\implies$ penultimate layer representation for I
- Network trained by taking absolute difference between features, followed by a fully connected head
- Distance induced: $d(f_1, f_2) = \sum_i \alpha_i |f_1[i] - f_2[i]|$ where α_i are trainable parameters
- **Output**: Binary decision (same/not same)

Contrastive Loss

- Based on metric learning paradigm
- Used to learn distinctive discriminative feature representations for various downstream computer vision tasks
- **Goal**: Map the input to embedding space where the distance between points corresponds to "semantic similarity" between input points
- Various formulations: Pairwise Contrastive Loss, Ranking Loss, Triplet Loss

We will look at some of these to build verification systems

Pairwise Contrastive Loss



- Hadsell[3] et al introduced an approach to learn a mapping
  - invariant to complex transforms
  - "similar" points in input space are mapped to nearby points on a low-dimensional manifold.
  - capable of consistently mapping new points (unseen during training)

- **Objective**: Learn W such that $D_W(X_1, X_2) = ||G_W(X_1) - G_W(X_2)||_2$ approximates the "semantic similarity" of inputs

[3] Hadsell et al, Dimensionality Reduction by Learning an Invariant Mapping, CVPR 2006

Pairwise Contrastive Loss

Algorithm

The algorithm first generates the training set, then trains the machine.

Step 1: For each input sample $\vec{X}_i$, do the following:

(a) Using prior knowledge find the set of samples $\mathcal{S}_{\vec{X}_i} = \{\vec{X}_j\}_{j=1}^{p}$, such that $\vec{X}_j$ is deemed similar to $\vec{X}_i$.

(b) Pair the sample $\vec{X}_i$ with all the other training samples and label the pairs so that: $Y_{ij} = 0$ if $\vec{X}_j \in \mathcal{S}_{\vec{X}_i}$, and $Y_{ij} = 1$ otherwise.

Combine all the pairs to form the labeled training set.

Step 2: Repeat until convergence:

(a) For each pair $(\vec{X}_i, \vec{X}_j)$ in the training set, do

i. If $Y_{ij} = 0$, then update W to decrease $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

ii. If $Y_{ij} = 1$, then update W to increase $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

- Let $\mathbf{x}_1, \mathbf{x}_2 \implies$ be a pair of high-dimensional input vectors

Pairwise Contrastive Loss

Algorithm

The algorithm first generates the training set, then trains the machine.

Step 1: For each input sample $\vec{X}_i$, do the following:

(a) Using prior knowledge find the set of samples $\mathcal{S}_{\vec{X}_i} = \{\vec{X}_j\}_{j=1}^{p}$, such that $\vec{X}_j$ is deemed similar to $\vec{X}_i$.

(b) Pair the sample $\vec{X}_i$ with all the other training samples and label the pairs so that: $Y_{ij} = 0$ if $\vec{X}_j \in \mathcal{S}_{\vec{X}_i}$, and $Y_{ij} = 1$ otherwise.

Combine all the pairs to form the labeled training set.

Step 2: Repeat until convergence:

(a) For each pair $(\vec{X}_i, \vec{X}_j)$ in the training set, do

i. If $Y_{ij} = 0$, then update W to decrease $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

ii. If $Y_{ij} = 1$, then update W to increase $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

- Let $\mathbf{x}_1, \mathbf{x}_2 \implies$ be a pair of high-dimensional input vectors
- y a binary label assigned to this pair:

$$\begin{cases} y = 0 & \text{if } \mathbf{x}_1, \mathbf{x}_2 \text{ are similar} \\ y = 1; & \text{otherwise} \end{cases}$$

Pairwise Contrastive Loss

Algorithm

The algorithm first generates the training set, then trains the machine.

Step 1: For each input sample $\vec{X}_i$, do the following:

(a) Using prior knowledge find the set of samples $\mathcal{S}_{\vec{X}_i} = \{\vec{X}_j\}_{j=1}^{p}$, such that $\vec{X}_j$ is deemed similar to $\vec{X}_i$.

(b) Pair the sample $\vec{X}_i$ with all the other training samples and label the pairs so that: $Y_{ij} = 0$ if $\vec{X}_j \in \mathcal{S}_{\vec{X}_i}$, and $Y_{ij} = 1$ otherwise.

Combine all the pairs to form the labeled training set.

Step 2: Repeat until convergence:

(a) For each pair $(\vec{X}_i, \vec{X}_j)$ in the training set, do

i. If $Y_{ij} = 0$, then update W to decrease $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

ii. If $Y_{ij} = 1$, then update W to increase $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

- Let $\mathbf{x}_1, \mathbf{x}_2 \implies$ be a pair of high-dimensional input vectors
- y a binary label assigned to this pair:

$$\begin{cases} y = 0 & \text{if } \mathbf{x}_1, \mathbf{x}_2 \text{ are similar} \\ y = 1; & \text{otherwise} \end{cases}$$

- Given $(y, \mathbf{x}_1, \mathbf{x}_2)$ are labeled pairs of data, **Pairwise Contrastive Loss**, $L_{contrastive}(W, y, \mathbf{x}_1, \mathbf{x}_2)$ given by:

$$\frac{1-y}{2} D_W^2 + \frac{y}{2} \max(0, m - D_W^2)$$

where $m > 0$ is a margin which defines a radius around $G_W(\mathbf{x})$

DeepID2[4]

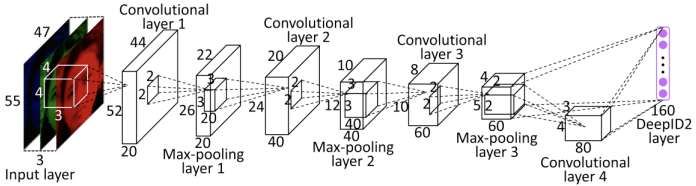


Figure 1: The ConvNet structure for DeepID2 extraction.

- Trains a deep CNN to jointly perform identification and verification

[4]Sun et al, Deep Learning Face Representation by Joint Identification-Verification, NIPS 2014

DeepID2[4]

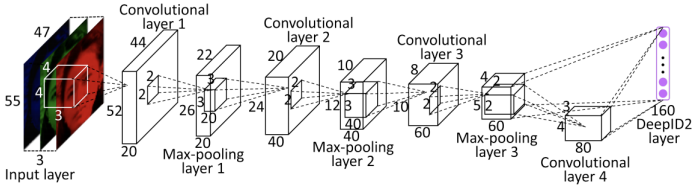


Figure 1: The ConvNet structure for DeepID2 extraction.

- Trains a deep CNN to jointly perform identification and verification
- **Identification task:** Increases inter-personal variations by pushing features from different identities apart

[4]Sun et al, Deep Learning Face Representation by Joint Identification-Verification, NIPS 2014

DeepID2[4]

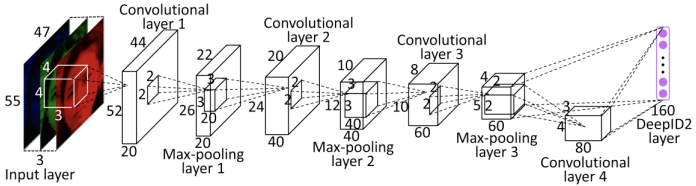


Figure 1: The ConvNet structure for DeepID2 extraction.

- Trains a deep CNN to jointly perform identification and verification
- **Identification task:** Increases inter-personal variations by pushing features from different identities apart
- **Verification task:** Reduces intra-personal variations by pulling features from same identity together

[4]Sun et al, Deep Learning Face Representation by Joint Identification-Verification, NIPS 2014

DeepID2

- **Cross-entropy loss** for training identification parameters θ_{id}
- **Pairwise contrastive loss** for learning verification parameters θ_{ve}

Identification Loss

$-\sum_{i=1}^{n} p_i \log \hat{p}_i = -\log \hat{p}_t$

f: DeepID2 feature vector
t: target class
θ_{id}: softmax layer parameters
p_i: target probability distribution
$\hat{p}_i$: is predicted probability distribution

Verification Loss

$L_{contrastive}(\theta_{ve}, y_{ij}, f_i, f_j)$

f_i,f_j: DeepID2 feature vectors for face images under comparison
θ_{ve}: verification loss parameters
$y_{ij} = 1 \implies f_i$ and f_j are from same identity
$y_{ij} = -1 \implies$ different identities

DeepID2: Algorithm

Table 1: The DeepID2 learning algorithm.

input: training set $\chi = \{(x_i, l_i)\}$, initialized parameters θ_c, θ_{id}, and θ_{ve}, hyperparameter λ, learning rate $\eta(t)$, $t \leftarrow 0$

while not converge **do**

$t \leftarrow t + 1$ sample two training samples (x_i, l_i) and (x_j, l_j) from χ

$f_i = \text{Conv}(x_i, \theta_c)$ and $f_j = \text{Conv}(x_j, \theta_c)$

$\nabla\theta_{id} = \frac{\partial \text{Ident}(f_i, l_i, \theta_{id})}{\partial \theta_{id}} + \frac{\partial \text{Ident}(f_j, l_j, \theta_{id})}{\partial \theta_{id}}$

$\nabla\theta_{ve} = \lambda \cdot \frac{\partial \text{Verif}(f_i, f_j, y_{ij}, \theta_{ve})}{\partial \theta_{ve}}$, where $y_{ij} = 1$ if $l_i = l_j$, and $y_{ij} = -1$ otherwise.

$\nabla f_i = \frac{\partial \text{Ident}(f_i, l_i, \theta_{id})}{\partial f_i} + \lambda \cdot \frac{\partial \text{Verif}(f_i, f_j, y_{ij}, \theta_{ve})}{\partial f_i}$

$\nabla f_j = \frac{\partial \text{Ident}(f_j, l_j, \theta_{id})}{\partial f_j} + \lambda \cdot \frac{\partial \text{Verif}(f_i, f_j, y_{ij}, \theta_{ve})}{\partial f_j}$

$\nabla\theta_c = \nabla f_i \cdot \frac{\partial \text{Conv}(x_i, \theta_c)}{\partial \theta_c} + \nabla f_j \cdot \frac{\partial \text{Conv}(x_j, \theta_c)}{\partial \theta_c}$

update $\theta_{id} = \theta_{id} - \eta(t) \cdot \theta_{id}$, $\theta_{ve} = \theta_{ve} - \eta(t) \cdot \theta_{ve}$, and $\theta_c = \theta_c - \eta(t) \cdot \theta_c$.

end while

output θ_c

- θ_{id} and θ_{ve}: separately updated via respective objectives; θ_c (backbone CNN parameters): trained via weighted contribution from identification and verification objectives
- enables backbone network to extract inter-personal features (via identification signal), and intra-personal features (via verification signal)

FaceNet[5]

Figure 2. **Model structure.** Our network consists of a batch input layer and a deep CNN followed by L_2 normalization, which results in the face embedding. This is followed by the triplet loss during training.

- Previous methods use bottleneck representation from pre-trained classification model. Do you see any problems?

[5] Schroff et al, FaceNet: A Unified Embedding for Face Recognition and Clustering, CVPR 2015

FaceNet[5]

Figure 2. **Model structure.** Our network consists of a batch input layer and a deep CNN followed by L_2 normalization, which results in the face embedding. This is followed by the triplet loss during training.

- Previous methods use bottleneck representation from pre-trained classification model. Do you see any problems?
  - Indirectness
  - Possible ineffectiveness; one has to hope that bottleneck representation will generalize to new faces
- FaceNet outputs compact 128-D embedding using a **triplet loss function** (commonly used today)

[5] Schroff et al, FaceNet: A Unified Embedding for Face Recognition and Clustering, CVPR 2015

FaceNet: Triplet Loss

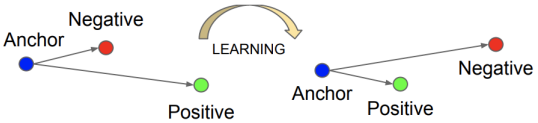


Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

- Alternate formulation of contrastive loss; considers triplet of inputs (namely anchor, positive, negative) instead of input pairs

FaceNet: Triplet Loss

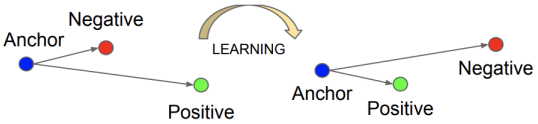


Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

- Alternate formulation of contrastive loss; considers triplet of inputs (namely anchor, positive, negative) instead of input pairs
- **Goal**: Ensure distance between anchor (x_a) and negative sample (x_n) representation is at least "margin" more than distance between anchor and positive sample (x_p)

FaceNet: Triplet Loss

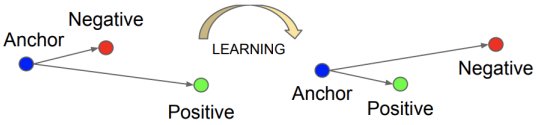


Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

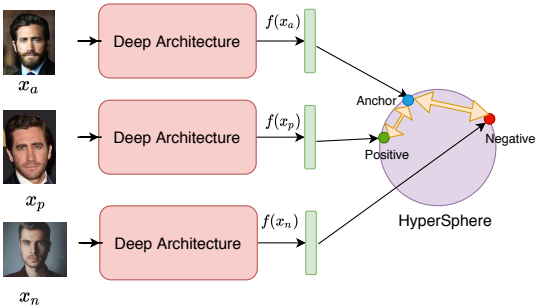
- Alternate formulation of contrastive loss; considers triplet of inputs (namely anchor, positive, negative) instead of input pairs
- **Goal**: Ensure distance between anchor (x_a) and negative sample (x_n) representation is at least "margin" more than distance between anchor and positive sample (x_p)
- Triplet Constraint:

$$||f(x_a) - f(x_p)||_2^2 + \alpha < ||f(x_a) - f(x_n)||_2^2$$

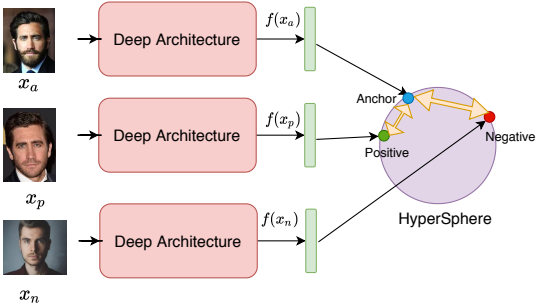
where α is margin

FaceNet: Triplet Loss Formulation

- Let $f(x)$: representation/embedding on d-dimensional hypersphere s.t. $||f(x)||_2 = 1$



FaceNet: Triplet Loss Formulation

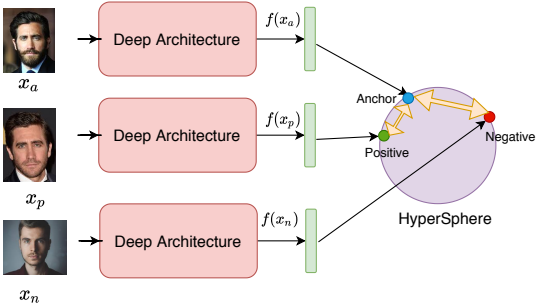


- Let $f(x)$: representation/embedding on d-dimensional hypersphere s.t. $||f(x)||_2 = 1$
- **Goal**: train θ to ensure that for all triplets x_a(anchor), x_p(positive), x_n(negative):

$$||f(x_a) - f(x_p)||_2^2 + \alpha < ||f(x_a) - f(x_n)||_2^2$$

where α is margin

FaceNet: Triplet Loss Formulation



- Let $f(x)$: representation/embedding on d-dimensional hypersphere s.t. $||f(x)||_2 = 1$
- **Goal**: train θ to ensure that for all triplets x_a(anchor), x_p(positive), x_n(negative):

$$||f(x_a) - f(x_p)||_2^2 + \alpha < ||f(x_a) - f(x_n)||_2^2$$

where α is margin

- Achieved by training parameters θ to minimize:

$$L_{triplet} = \sum_i [||f(x_a^i) - f(x_p^i)||_2^2 - ||f(x_a^i) - f(x_n^i)||_2^2 + \alpha]$$

Triplet Selection Strategy

- How to choose triplets?

Triplet Selection Strategy

- How to choose triplets?
- "Easy" triplets can:
  - inhibit learning; model does not strive to learn parameters capturing distinctive features
  - have slower convergence
- Important to select triplets that violate triplet constraint initially, enabling the model to "work hard" to satisfy the constraint

Hard Triplet Mining

- Selecting "hard" triplets $\implies$ choose x_p^i (hard positive): $\arg\max_{x_p^i} ||f(x_a^i) - f(x_p^i)||_2^2$
  x_n^i (hard negative): $\arg\min_{x_n^i} ||f(x_a^i) - f(x_n^i)||_2^2$

Hard Triplet Mining

- Selecting "hard" triplets $\implies$ choose x_p^i (hard positive): $\arg\max_{x_p^i} ||f(x_a^i) - f(x_p^i)||_2^2$
  x_n^i (hard negative): $\arg\min_{x_n^i} ||f(x_a^i) - f(x_n^i)||_2^2$
- FaceNet uses online triplet sampling; hard positive/negatives sampled from every mini-batch

Hard Triplet Mining

- Selecting "hard" triplets $\implies$ choose x_p^i (hard positive): $\arg\max_{x_p^i} ||f(x_a^i) - f(x_p^i)||_2^2$
  x_n^i (hard negative): $\arg\min_{x_n^i} ||f(x_a^i) - f(x_n^i)||_2^2$
- FaceNet uses online triplet sampling; hard positive/negatives sampled from every mini-batch
- **Problem**: Very hard negatives early on $\implies$ bad local minima initially, potential training/model collapse

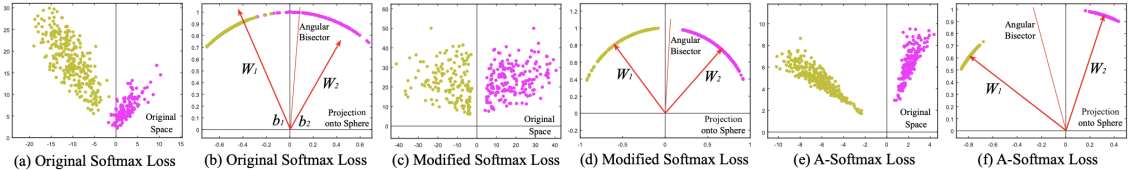
Hard Triplet Mining

- Selecting "hard" triplets $\implies$ choose x_p^i (hard positive): $\arg\max_{x_p^i} ||f(x_a^i) - f(x_p^i)||_2^2$
  x_n^i (hard negative): $\arg\min_{x_n^i} ||f(x_a^i) - f(x_n^i)||_2^2$
- FaceNet uses online triplet sampling; hard positive/negatives sampled from every mini-batch
- **Problem**: Very hard negatives early on $\implies$ bad local minima initially, potential training/model collapse
- **Solution**: "Semi-hard" negatives $\implies$ negatives that lie inside margin α; how?

Hard Triplet Mining

- Selecting "hard" triplets $\implies$ choose x_p^i (hard positive): $\arg\max_{x_p^i} ||f(x_a^i) - f(x_p^i)||_2^2$
  x_n^i (hard negative): $\arg\min_{x_n^i} ||f(x_a^i) - f(x_n^i)||_2^2$
- FaceNet uses online triplet sampling; hard positive/negatives sampled from every mini-batch
- **Problem**: Very hard negatives early on $\implies$ bad local minima initially, potential training/model collapse
- **Solution**: "Semi-hard" negatives $\implies$ negatives that lie inside margin α; how? Choose x_n^i (semi-hard):$||f(x_a) - f(x_p)||_2^2 < ||f(x_a) - f(x_n)||_2^2$

Improving over Triplet Loss: Angular Softmax Loss[6]



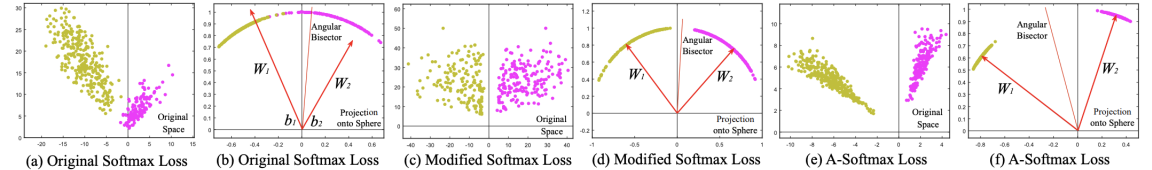
(a) Original Softmax Loss (b) Original Softmax Loss (c) Modified Softmax Loss (d) Modified Softmax Loss (e) A-Softmax Loss (f) A-Softmax Loss

Figure 2: Comparison among softmax loss, modified softmax loss and A-Softmax loss. In this toy experiment, we construct a CNN to learn 2-D features on a subset of the CASIA face dataset. In specific, we set the output dimension of FC1 layer as 2 and visualize the learned features. Yellow dots represent the first class face features, while purple dots represent the second class face features. One can see that features learned by the original softmax loss can not be classified simply via angles, while modified softmax loss can. Our A-Softmax loss can further increase the angular margin of learned features.

- Features learned by softmax loss have intrinsic angular distribution $\Longrightarrow$ Euclidean margin-based losses incompatible with softmax loss

[6] Liu et al, SphereFace: Deep Hypersphere Embedding for Face Recognition, CVPR 2017

Improving over Triplet Loss: Angular Softmax Loss[6]



(a) Original Softmax Loss (b) Original Softmax Loss (c) Modified Softmax Loss (d) Modified Softmax Loss (e) A-Softmax Loss (f) A-Softmax Loss

Figure 2: Comparison among softmax loss, modified softmax loss and A-Softmax loss. In this toy experiment, we construct a CNN to learn 2-D features on a subset of the CASIA face dataset. In specific, we set the output dimension of FC1 layer as 2 and visualize the learned features. Yellow dots represent the first class face features, while purple dots represent the second class face features. One can see that features learned by the original softmax loss can not be classified simply via angles, while modified softmax loss can. Our A-Softmax loss can further increase the angular margin of learned features.

- Features learned by softmax loss have intrinsic angular distribution $\Longrightarrow$ Euclidean margin-based losses incompatible with softmax loss
- **A-Softmax loss** hence uses angle between feature and classification layer vector, both in training and test

[6] Liu et al, SphereFace: Deep Hypersphere Embedding for Face Recognition, CVPR 2017

A-Softmax Loss

- **Softmax Loss:**

$$\mathcal{L}_{softmax} = \frac{1}{N}\sum_i -\log\left(\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_j e^{W_j^T x_i + b_j}}\right) = \frac{1}{N}\sum_i -\log\left(\frac{e^{||W_{y_i}||||x_i||\cos(\theta_{y_i,i}) + b_{y_i}}}{\sum_j e^{||W_j||||x_i||\cos(\theta_{j,i}) + b_j}}\right)$$

- **Modified Softmax Loss** ($||W_1|| = ||W_2|| = 1,\ b_1 = b_2 = 0$):

$$\mathcal{L}_{modified} = \frac{1}{N}\sum_i -\log\left(\frac{e^{||x_i||\cos(\theta_{y_i,i})}}{\sum_j e^{||x_i||\cos(\theta_{j,i})}}\right)$$

- **A-Softmax Loss:**

$$\mathcal{L}_{ang} = \frac{1}{N}\sum_i -\log\left(\frac{e^{||x_i||\cos(m\theta_{y_i,i})}}{e^{||x_i||\cos(m\theta_{y_i,i})} + \sum_{j\neq y_i} e^{||x_i||\cos(\theta_{j,i})}}\right)$$

A-Softmax Loss: Binary Classification Analysis

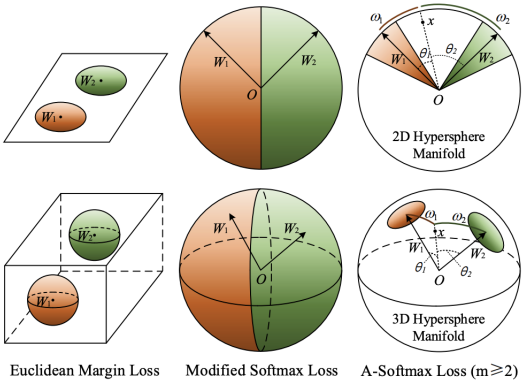


Figure 3: Geometry Interpretation of Euclidean margin loss (e.g. contrastive loss, triplet loss, center loss, etc.), modified softmax loss and A-Softmax loss. The first row is 2D feature constraint, and the second row is 3D feature constraint. The orange region indicates the discriminative constraint for class 1, while the green region is for class 2.

Let W_i, b_i be weights and bias in softmax loss:

- **Softmax** decision boundary $\Longrightarrow$ $(W_1 - W_2)x + b1 - b2 = 0$

A-Softmax Loss: Binary Classification Analysis

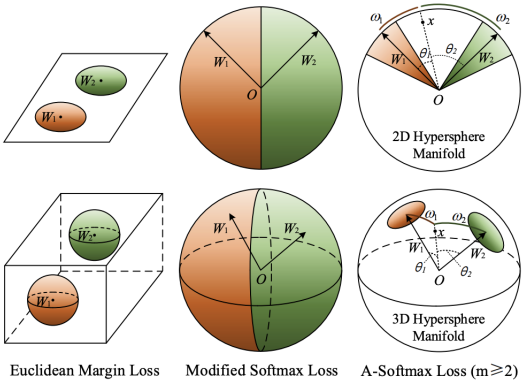


Figure 3: Geometry Interpretation of Euclidean margin loss (e.g. contrastive loss, triplet loss, center loss, etc.), modified softmax loss and A-Softmax loss. The first row is 2D feature constraint, and the second row is 3D feature constraint. The orange region indicates the discriminative constraint for class 1, while the green region is for class 2.

Let W_i, b_i be weights and bias in softmax loss:

- **Softmax** decision boundary $\implies$ $(W_1 - W_2)x + b1 - b2 = 0$
- Constrain $||W_1|| = ||W_2|| = 1$, $b_1 = b_2 = 0$;
  Modified Softmax decision boundary
  $\implies ||x||(\cos(\theta_1) - \cos(\theta_2)) = 0$

A-Softmax Loss: Binary Classification Analysis

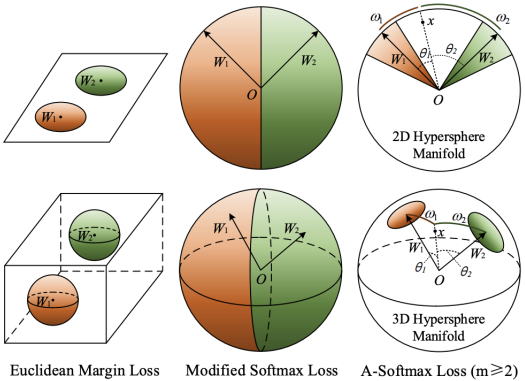


Figure 3: Geometry Interpretation of Euclidean margin loss (e.g. contrastive loss, triplet loss, center loss, etc.), modified softmax loss and A-Softmax loss. The first row is 2D feature constraint, and the second row is 3D feature constraint. The orange region indicates the discriminative constraint for class 1, while the green region is for class 2.

Let W_i, b_i be weights and bias in softmax loss:

- **Softmax** decision boundary $\implies$
  $(W_1 - W_2)x + b1 - b2 = 0$
- Constrain $||W_1|| = ||W_2|| = 1$,
  $b_1 = b_2 = 0$;
  Modified Softmax decision boundary
  $\implies ||x||(\cos(\theta_1) - \cos(\theta_2)) = 0$
- Margin $m \geq 1$: quantitatively controls the size of angular margin
  A-Softmax decision boundary $\implies$
  class 1: $||x||(\cos(m\theta_1) - \cos(\theta_2)) = 0$
  class 2: $||x||(\cos(\theta_1) - \cos(m\theta_2)) = 0$

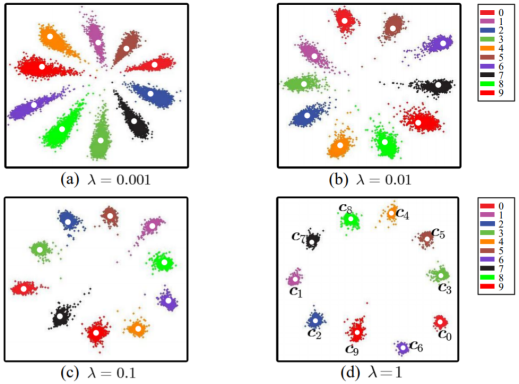
Other Loss Functions used: CenterLoss[7]

- Intended to augment Softmax+CE to also reduce intra-class variance

$$\mathcal{L}_{new} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{CL}$$

$$= \mathcal{L}_{CE} + \boxed{\frac{\lambda}{2} \sum_{i=1}^{m} ||x_i - c_{y_i}||_2^2}$$

- Added term minimizes intra-class distance between sample x and y_ith class centroid c_{y_i} of deep features
- Centroid is updated online during learning



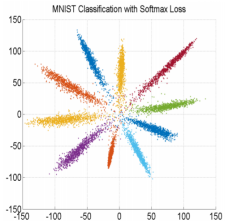
(a) $\lambda = 0.001$ (b) $\lambda = 0.01$ (c) $\lambda = 0.1$ (d) $\lambda = 1$

[7] Wen et al, Discriminative Feature Learning Approach for Deep Face Recognition, ECCV 2016

L2-Softmax[8]

- Enforces L2-norm of features to be fixed for every face image by adding an L2 constraint to feature descriptor such that it lies on hypersphere of fixed radius (α)
- Features with **high L2 norm** are **easy** to classify for network (spatially placed at boundaries in feature space)
- Features with **low L2 norm** are **difficult** to classify (spatially located near origin)

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \mathcal{L}_{CE} \\ \text{subject to} \quad & ||f(x_i)||_2 = \alpha, i = 1, \ldots, m. \end{aligned}$$



[8]Ranjan et al, L2-constrained Softmax Loss for Discriminative Face Verification, 2017

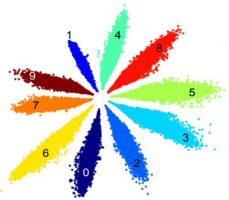
RingLoss[9]

- While L2-Softmax normalizes features with a *projection* method, **RingLoss** enforces feature normalizations directly in loss function; radius R is learned in training process

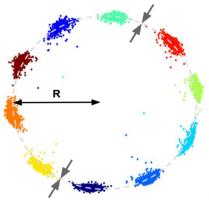
$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \mathcal{L}_{CE} \\ \text{subject to} \quad & ||f(x_i)||_2 = \mathbf{R} \end{aligned}$$

- Implemented as:

$$\mathcal{L}_R = \frac{\lambda}{2m} \sum_{i=1}^{m} (||f(x_i)||_2 - R)^2$$



(a) Features trained using Softmax



(b) Features trained using Ring loss

[9] Zheng et al, Ring Loss: Convex Feature Normalization for FaceRecognition, CVPR 2018

Other Recent Efforts

- **CosFace**[10]: proposes a large margin-based cosine loss (LMCL) for face recognition
- **UniformFace**[11]: proposes uniform loss to learn equidistributed representations to fully exploit feature space
- **RegularFace**[12]: proposes exclusive regularization to explicitly enlarge angular distance between different identities
- **GroupFace**[13]: proposes to utilize multiple group-aware representations, simultaneously, to improve quality of embedding
- **CurricularFace**[14]: proposes Adaptive Curriculum Learning loss to adaptively adjust relative importance of easy and hard samples during different training stages

[10] Wang et al, CosFace: Large Margin Cosine Loss for Deep Face Recognition, CVPR 2018

[11] Duan et al, UniformFace: Learning Deep Equidistributed Representation for Face Recognition, CVPR 2019

[12] Zhao et al, RegularFace: Deep Face Recognition via Exclusive Regularization, CVPR 2019

[13] Kim et al, GroupFace: Learning Latent Groups and Constructing Group-based Representations for Face Recognition, CVPR 2020

[14] Huang et al, CurricularFace: Adaptive Curriculum Learning Loss for Deep Face Recognition, CVPR 2020

Face Recognition: Closed-Set vs Open-Set

- **Closed-Set Loss Functions** (all subjects known)
  - **Softmax + CrossEntropy**
  - **Center Loss** (reduce intra-class variability)
  - **L2-Softmax** (reduce intra-class variability)
  - **RingLoss** (reduce intra-class variability)
  - **Angular Softmax** (increase margin between subjects)
- **Open-Set Loss Functions** (all subjects not known)
  - **Double Margin Contrastive Loss** (Homework!)
  - **Triplet Loss**

Source: Masi et al, Deep Face Recognition: A Survey, SIBGRAPI 2018

Homework

Readings

- Raúl Gómez, Understanding Ranking Loss, Contrastive Loss, Margin Loss, Triplet Loss, Hinge Loss and all those confusing names
- A detailed tutorial on deep face recognition by Masi et al
- (Optional) Adrian Rosebrock, Face recognition with OpenCV, Python, and deep learning
- (Optional) Wang et al, Deep Face Recognition: A Survey, Neurocomputing 2018

Question

What is double-margin contrastive loss, and how can it be used for open-set face recognition? (*Hint:* Read the tutorial above!)

References I

R. Hadsell, S. Chopra, and Y. LeCun. "Dimensionality Reduction by Learning an Invariant Mapping". In: *CVPR* (2006).

Gary B. Huang, Manjunath Narayana, and Erik Learned-Miller. "Towards Unconstrained Face Recognition". In: *Computer Vision and Pattern Recognition Workshops* (2008).

Yi Sun, Xiaogang Wang, and Xiaoou Tang. "Deep Learning Face Representation by Joint Identification-Verification". In: *NIPS* (2014).

Yaniv Taigman, Ming Yang, and Marc'Aurelio Ranzato. "DeepFace: Closing the Gap to Human-Level Performance in Face Verification". In: *CVPR* (2014).

Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering". In: *CVPR* (2015).

Y. Wen et al. "A Discriminative Feature Learning Approach for Deep Face Recognition". In: *ECCV*. 2016.

References II

Weiyang Liu et al. "SphereFace: Deep Hypersphere Embedding for Face Recognition". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 6738–6746.

Rajeev Ranjan, Carlos Castillo, and Rama Chellappa. "L2-constrained Softmax Loss for Discriminative Face Verification". In: (Mar. 2017).

Iacopo Masi et al. "Deep Face Recognition: a Survey". In: *SIBGRAPI - Conference on Graphics, Patterns and Images*. 2018.

Mei Wang and Weihong Deng. "Deep Face Recognition: A Survey". In: *ArXiv* abs/1804.06655 (2018).

Yutong Zheng, Dipan K. Pal, and Marios Savvides. "Ring Loss: Convex Feature Normalization for Face Recognition". In: *CVPR* (2018), pp. 5089–5097.

Yichen Qian, Weihong Deng, and Jiani Hu. "Unsupervised Face Normalization With Extreme Pose and Expression in the Wild". In: *CVPR* (2019), pp. 9843–9850.

Xiang Wang, Kai Wang, and Shiguo Lian. *A Survey on Face Data Augmentation*. Apr. 2019.