Loss functions for one shot learning recognition

Bar Maltzman

Software Engineer
Similarweb

Center Loss

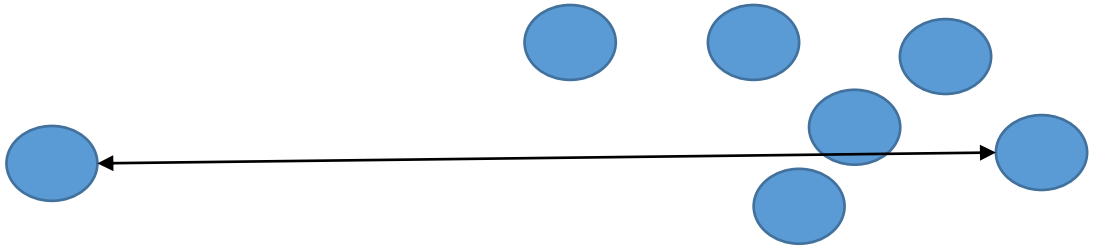
- Goal – Enhancing the Discriminative power of deeply learned features.

- But… how? –

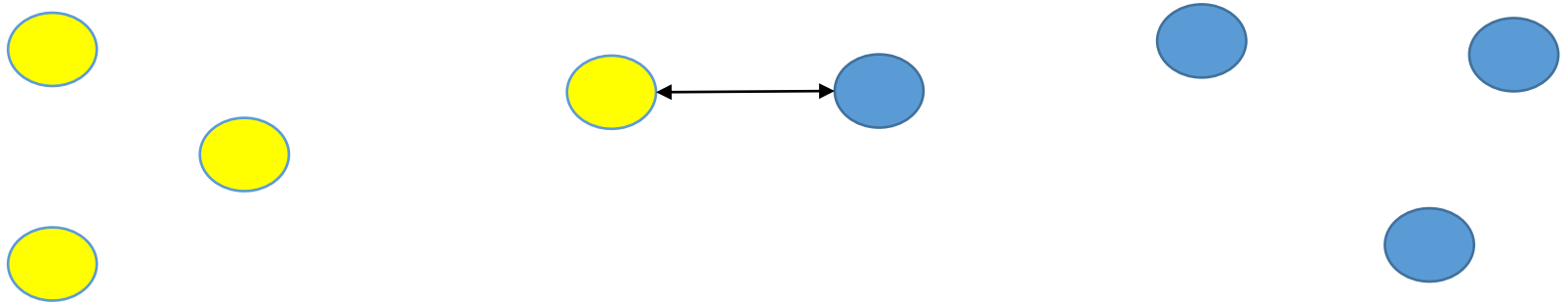
- Simultaneously learn a center of features of each class and penalizing the deep features and their corresponding class center.
- We use both center loss and softmax Loss

Definitions

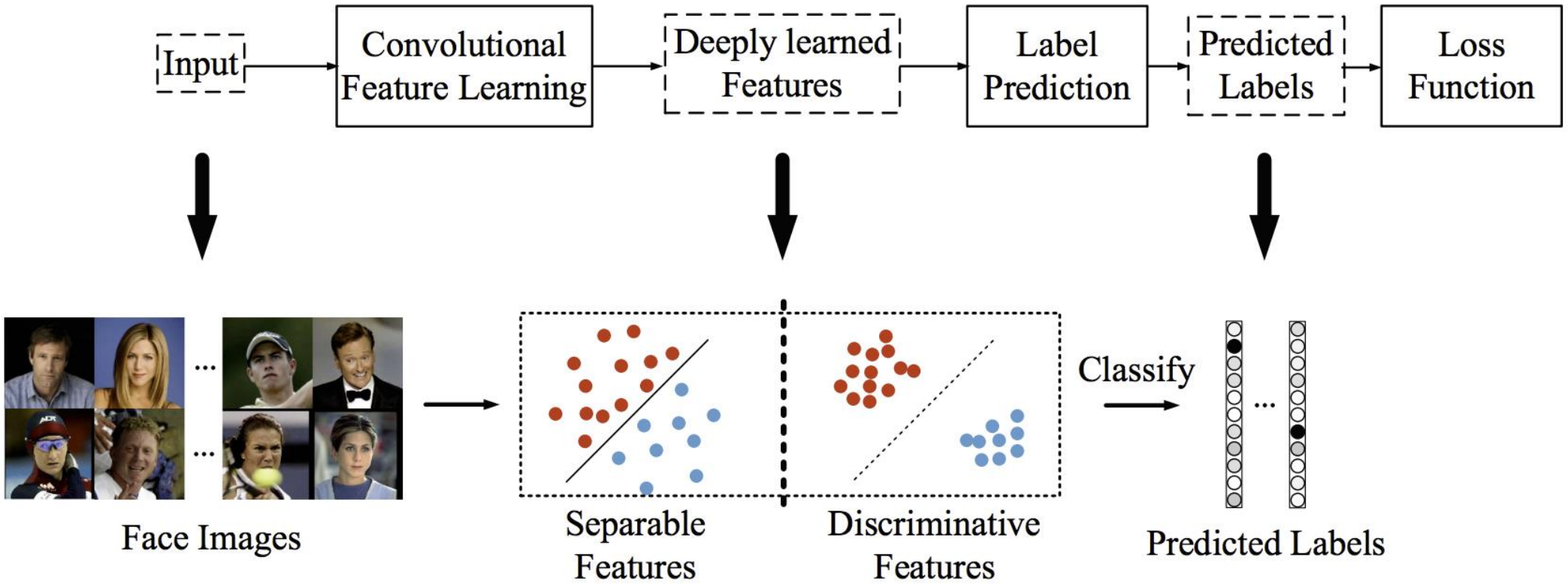
- Intra-class Variation – the largest distance between dots of the same class.



- Inter-class Difference – the smallest distance between dots of different classes.



The typical framework of convolutional neural networks.



Close-set identification

- The Classes of the possible testing sample are within the training set
- The predicted labels dominate the Performance.
- Softmax loss is able to address Classification problem

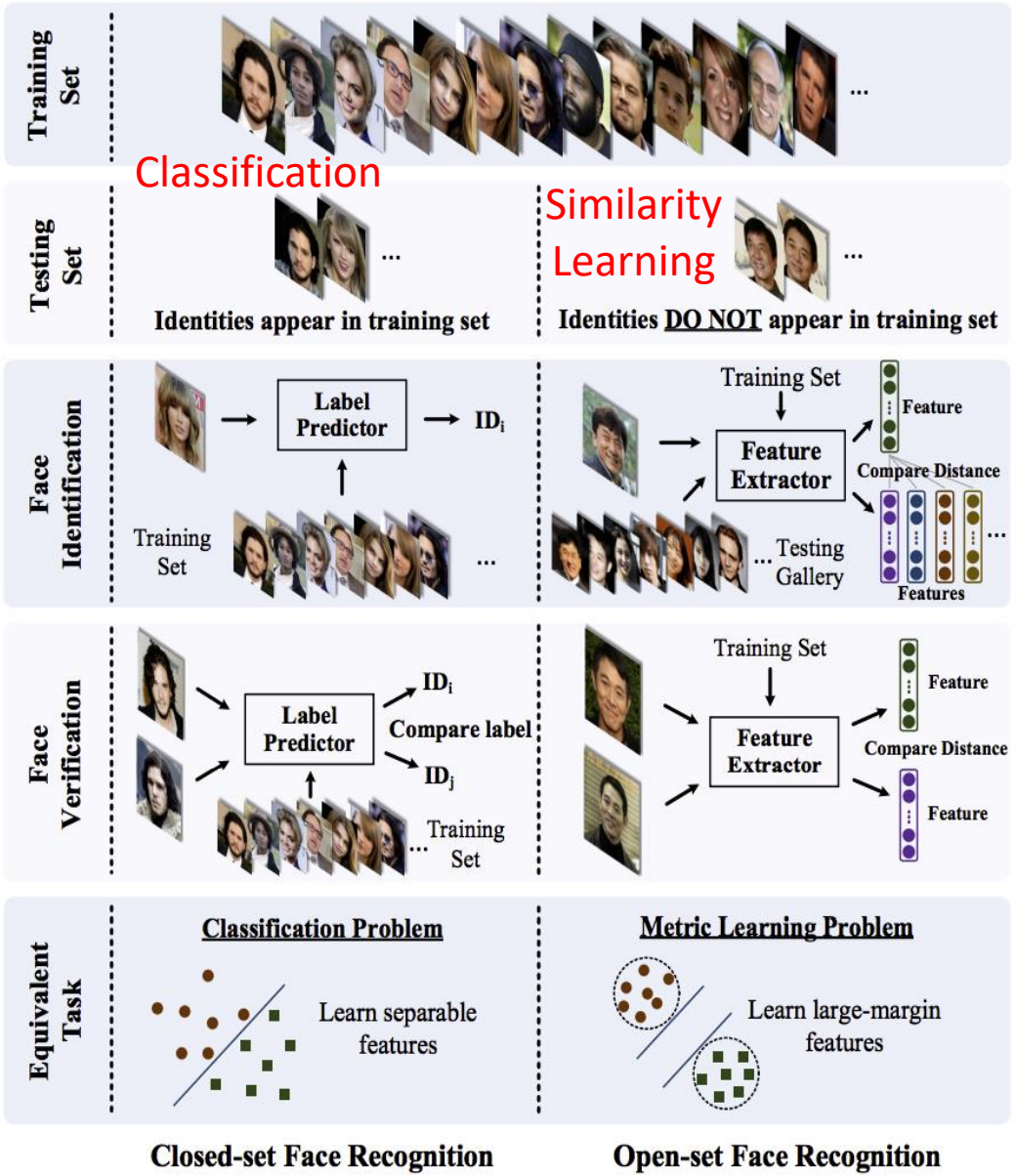
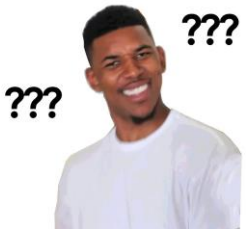


Figure 1: Comparison of open-set and closed-set face recognition.

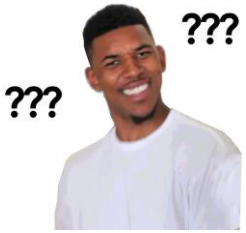
Challenges

- Constructing efficient loss function for discriminating feature learning in CNN

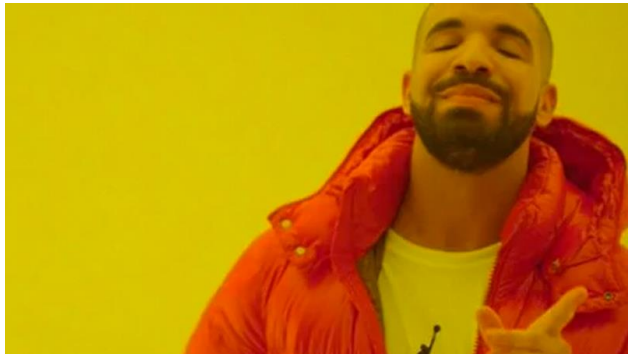
- SGD? works well on mini batch, but impractical for a huge scale of training.



Constructive loss and triplet loss? Slows convergence and Instable.



Center Loss



Basic idea:

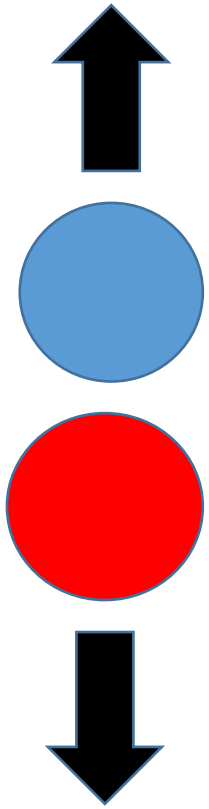
-Learn a center of a deep feature of each class.

-Simultaneously update the center and minimizes the distance between the deep features of the same class and their corresponding class center.

-CNNs trained under the joint supervision of softmax loss and center loss, with hyper parameter to balance the 2 supervision signals.

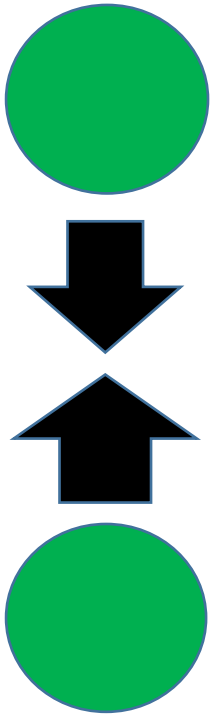
Softmax loss

$$\mathcal{L}_S = -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boldsymbol{x}_i + b_j}}$$



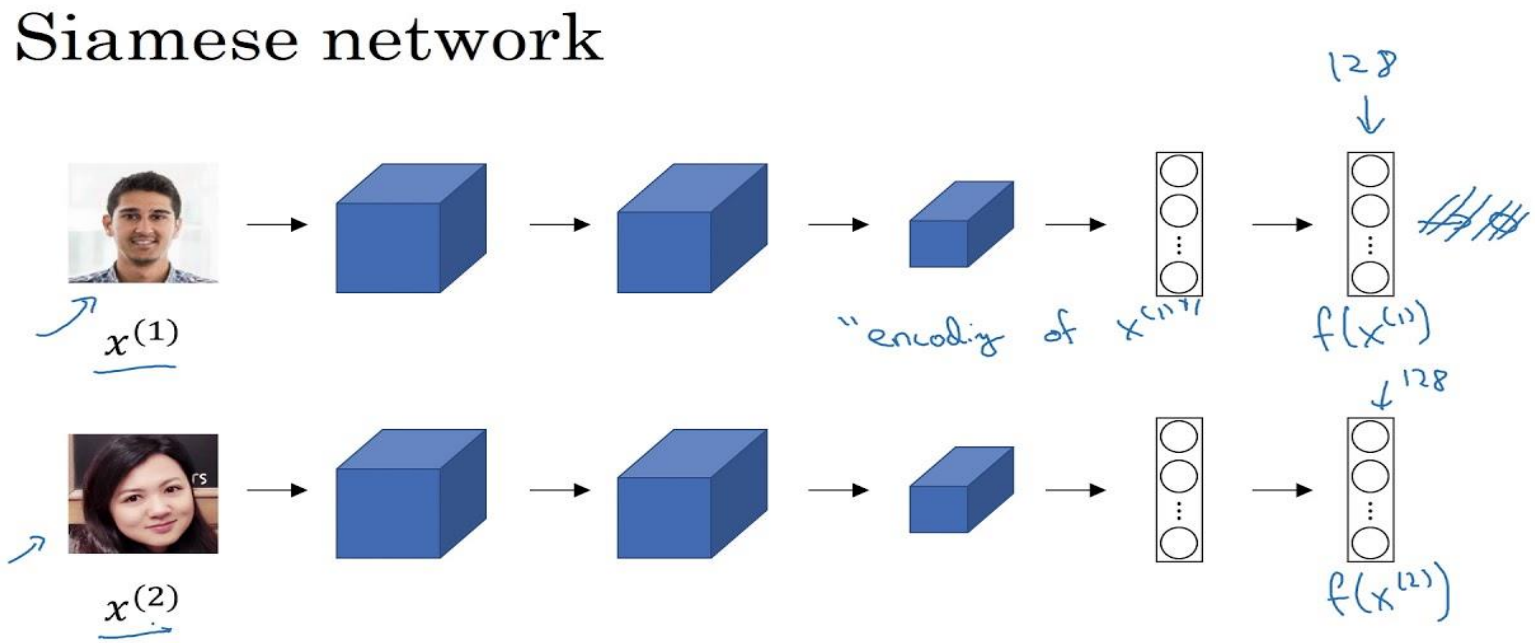
Center loss

$$\mathcal{L}_C = \frac{1}{2} \sum_{i=1}^{m} \| \boldsymbol{x}_i - \boldsymbol{c}_{y_i} \|_2^2$$



1.train siamese networks for driving the similarity metric to be small for positive pairs, and large for the negative pairs.

2. learn a nonlinear transformations and yield discriminative deep metric with a margin between positive and negative face image pairs.



[Taigman et. al., 2014. DeepFace closing the gap to human level performance]

Andrew Ng

A toy example

Performed on MINIST dataset.

Table 1. The CNNs architecture we use in toy example, called LeNets++. Some of the convolution layers are followed by max pooling. $(5, 32)_{/1,2} \times 2$ denotes 2 cascaded convolution layers with 32 filters of size 5×5, where the stride and padding are 1 and 2 respectively. $2_{/2,0}$ denotes the max-pooling layers with grid of 2×2, where the stride and padding are 2 and 0 respectively. In LeNets++, we use the Parametric Rectified Linear Unit (PReLU) [12] as the nonlinear unit.

	Stage 1		Stage 2		Stage 3		Stage 4
Layer	Conv	Pool	Conv	Pool	Conv	Pool	FC
LeNets	$(5, 20)_{/1,0}$	$2_{/2,0}$	$(5, 50)_{/1,0}$	$2_{/2,0}$			500
LeNets++	$(5, 32)_{/1,2} \times 2$	$2_{/2,0}$	$(5, 64)_{/1,2} \times 2$	$2_{/2,0}$	$(5, 128)_{/1,2} \times 2$	$2_{/2,0}$	**2**

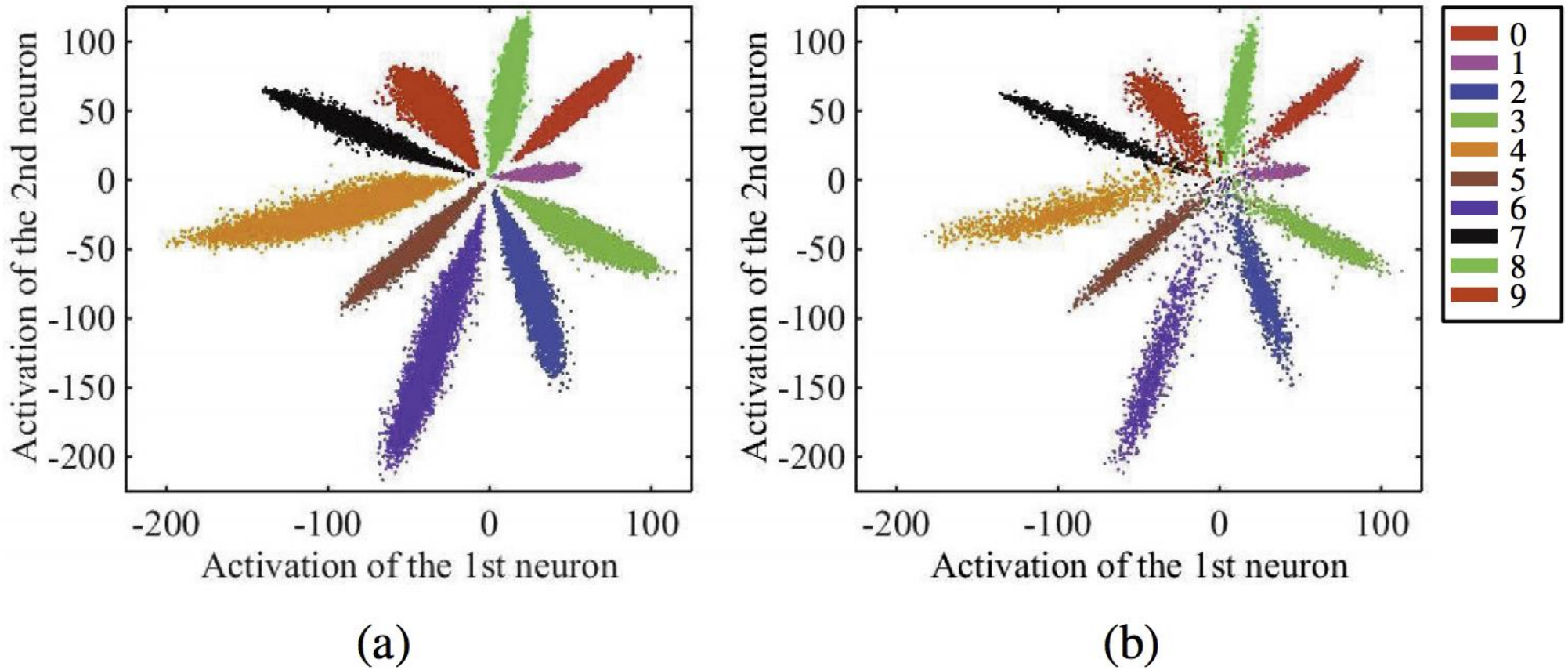


Fig. 2. The distribution of deeply learned features in (a) training set (b) testing set, both under the supervision of softmax loss, where we use 50K/10K train/test splits. The points with different colors denote features from different classes. **Best viewed in color.** (Color figure online)

the center loss can not be used directly

- Ideally, the c_yi(the y_ith class center of the deep feature) should be updated as the deep features changed. In other words, we need to take the entire training set into account and average the features of every class in each iteration, which is inefficient and even impractical.

Addressing the problem

- 1.instead of updating the centers with respect to the entire training set, an update based on a mini-batch is performed. In each iteration, the centers are computed by averaging the features of the corresponding classes
- 2. to avoid large perturbations caused by few mislabeled samples, use a scalar α to control the learning rate of the centers.

The gradients of $\mathcal{L}_C$ with respect to $\boldsymbol{x}_i$ and update equation of $\boldsymbol{c}_{y_i}$ are computed as:

$$\frac{\partial \mathcal{L}_C}{\partial \boldsymbol{x}_i} = \boldsymbol{x}_i - \boldsymbol{c}_{y_i} \tag{3}$$

$$\Delta \boldsymbol{c}_j = \frac{\sum_{i=1}^{m} \delta(y_i = j) \cdot (\boldsymbol{c}_j - \boldsymbol{x}_i)}{1 + \sum_{i=1}^{m} \delta(y_i = j)} \tag{4}$$

where $\delta(condition) = 1$ if the *condition* is satisfied, and $\delta(condition) = 0$ if not. α is restricted in $[0, 1]$. We adopt the joint supervision of softmax loss and center loss to train the CNNs for discriminative feature learning. The formulation is given in Eq. 5.

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_S + \lambda \mathcal{L}_C \\ &= -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boldsymbol{x}_i + b_j}} + \frac{\lambda}{2} \sum_{i=1}^{m} \| \boldsymbol{x}_i - \boldsymbol{c}_{y_i} \|_2^2 \end{aligned} \tag{5}$$

Algorithm 1. The discriminative feature learning algorithm

Input: Training data $\{\boldsymbol{x}_i\}$. Initialized parameters θ_C in convolution layers. Parameters W and $\{\boldsymbol{c}_j|j=1,2,...,n\}$ in loss layers, respectively. Hyperparameter λ, α and learning rate μ^t. The number of iteration $t \leftarrow 0$.

Output: The parameters θ_C.

while not converge **do**

  $t \leftarrow t+1$.

  Compute the joint loss by $\mathcal{L}^t = \mathcal{L}_S^t + \mathcal{L}_C^t$.

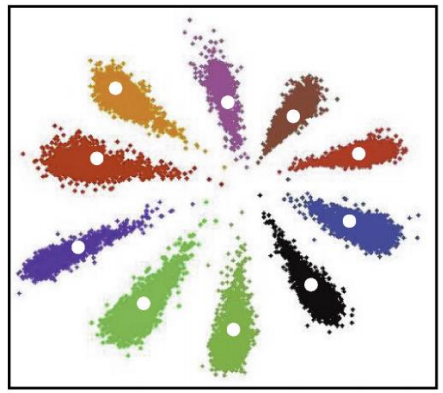
  Compute the backpropagation error $\frac{\partial \mathcal{L}^t}{\partial \boldsymbol{x}_i^t}$ for each i by $\frac{\partial \mathcal{L}^t}{\partial \boldsymbol{x}_i^t} = \frac{\partial \mathcal{L}_S^t}{\partial \boldsymbol{x}_i^t} + \lambda \cdot \frac{\partial \mathcal{L}_C^t}{\partial \boldsymbol{x}_i^t}$.

  Update the parameters W by $W^{t+1} = W^t - \mu^t \cdot \frac{\partial \mathcal{L}^t}{\partial W^t} = W^t - \mu^t \cdot \frac{\partial \mathcal{L}_S^t}{\partial W^t}$.

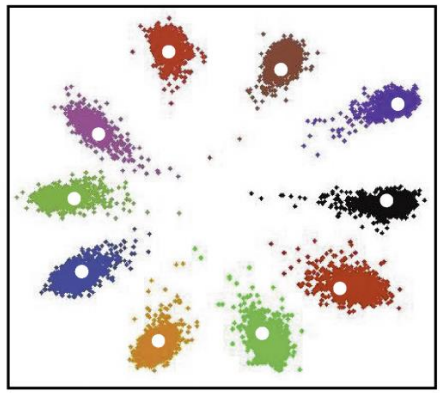
  Update the parameters $\boldsymbol{c}_j$ for each j by $\boldsymbol{c}_j^{t+1} = \boldsymbol{c}_j^t - \alpha \cdot \Delta \boldsymbol{c}_j^t$.

  Update the parameters θ_C by $\theta_C^{t+1} = \theta_C^t - \mu^t \sum_i^m \frac{\partial \mathcal{L}^t}{\partial \boldsymbol{x}_i^t} \cdot \frac{\partial \boldsymbol{x}_i^t}{\partial \theta_C^t}$.

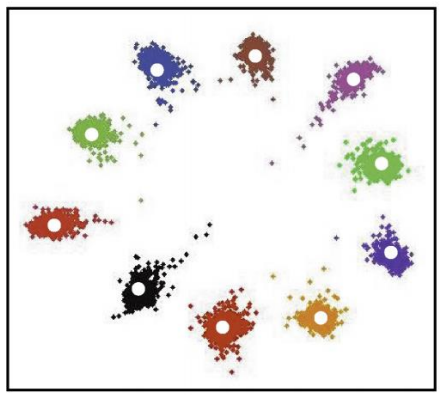
end while



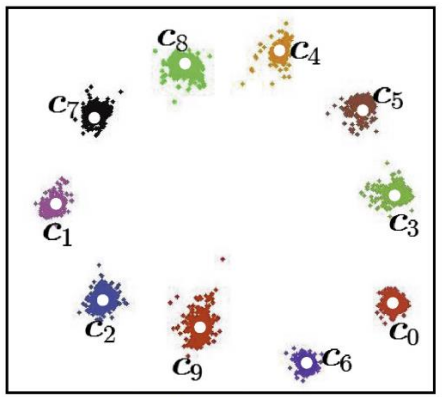
(a) $\lambda = 0.001$



(b) $\lambda = 0.01$



(c) $\lambda = 0.1$



(d) $\lambda = 1$

Experiments

C: The convolution layer

P: The max-pooling layer

LC: The local convolution layer

FC: The fully connected layer

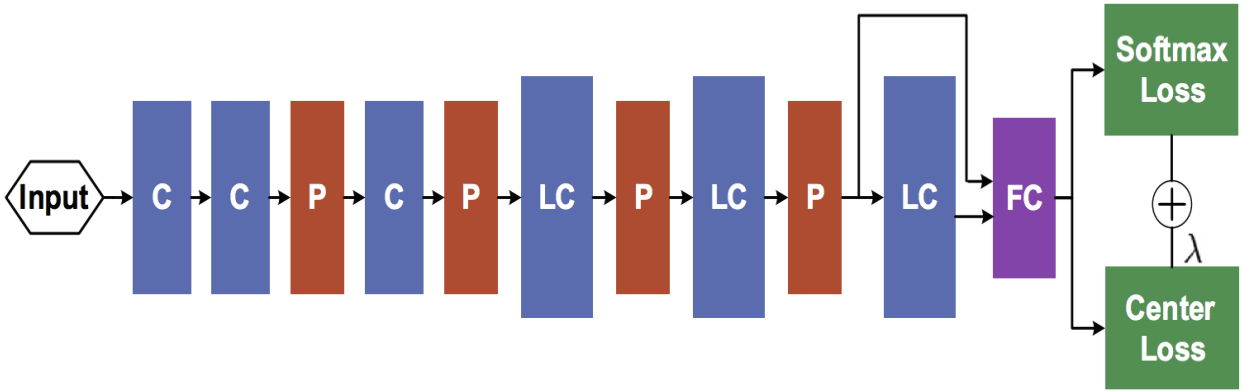
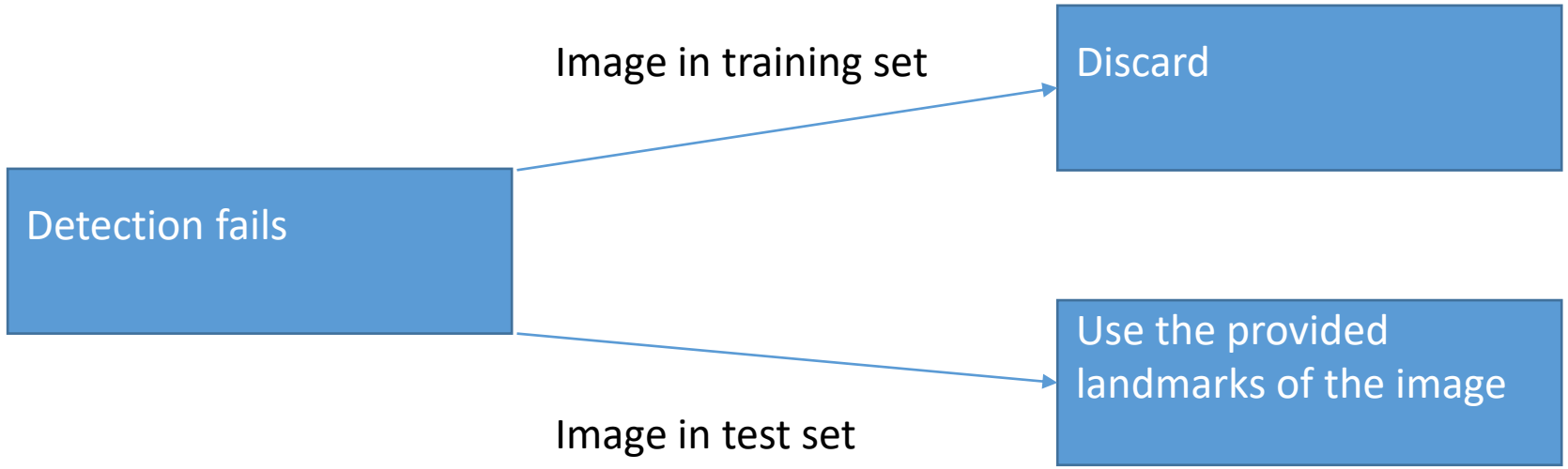


Fig. 4. The CNN architecture using for face recognition experiments. Joint supervision is adopted. The filter sizes in both convolution and local convolution layers are 3×3 with stride 1, followed by PReLU [12] nonlinear units. Weights in three local convolution layers are locally shared in the regions of 4×4, 2×2 and 1×1 respectively. The number of the feature maps are 128 for the convolution layers and 256 for the local convolution layers. The max-pooling grid is 2×2 and the stride is 2. The output of the 4th pooling layer and the 3th local convolution layer are concatenated as the input of the 1st fully connected layer. The output dimension of the fully connected layer is 512. **Best viewed in color.** (Color figure online)

Implementation Details

-Preprocessing: 5 landmarks for simplicity transformation for similarity transformation. Faces are cropped to 112X96 RGB images.



Training data

- Taken from: CASIA-WebFace, CACD2000 and Celebrity+.(considered fairly small training set).

- The experiment uses only 0.49 million training data, which are horizontally flipped for data argumentation.

Experiments

- We train three kind of models:

- Model A - Softmax loss.
- Model B - Softmax loss and Constrastive loss.
- Model C - Softmax loss and center loss.

Softmax Loss

$$\mathcal{L}_S = -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boldsymbol{x}_i + b_j}}$$

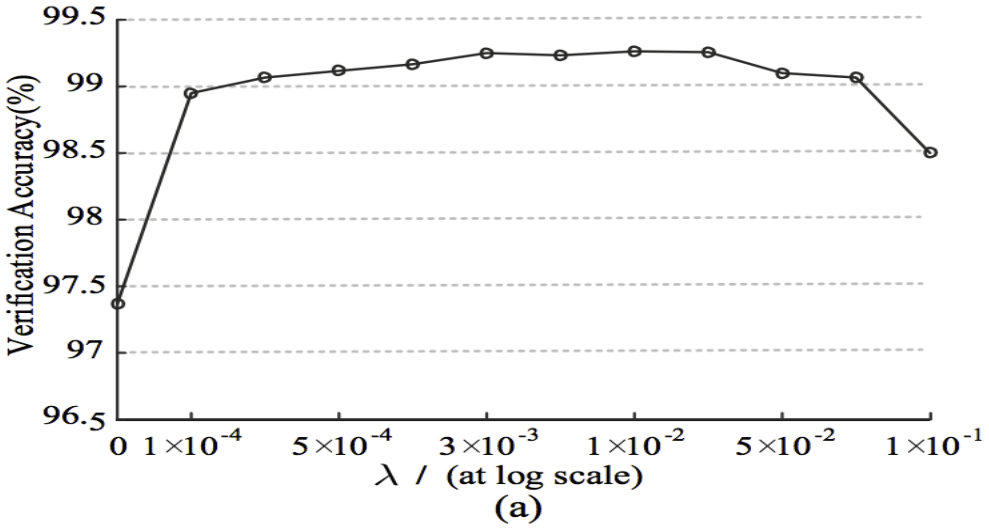
Contrastive Loss

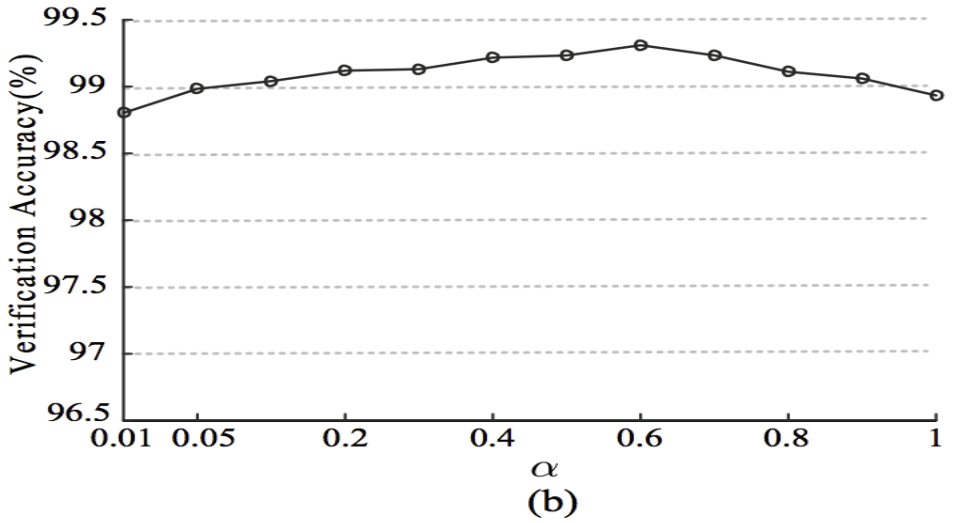
$$(1-Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{max(0, m - D_W)\}^2$$

Center Loss

$$\mathcal{L}_C = \frac{1}{2}\sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2$$

Choosing right Lambda and Alpha





Model A- where Lamdba = 0 is not a good choice!

The Results

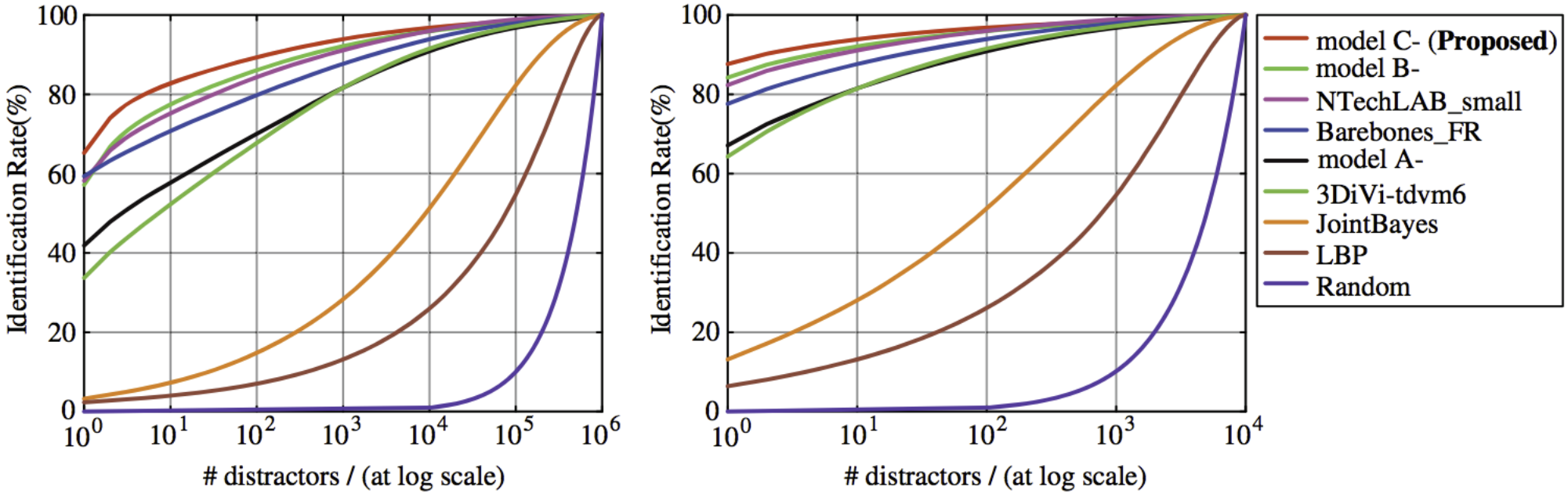
Method	Images	Networks	Acc. on LFW	Acc. on YTF
DeepFace [34]	4M	3	97.35 %	91.4 %
DeepID-2+ [32]	-	1	98.70 %	-
DeepID-2+ [32]	-	25	99.47 %	93.2 %
FaceNet [27]	200M	1	99.63 %	95.1 %
Deep FR [25]	2.6M	1	98.95 %	97.3 %
Baidu [21]	1.3M	1	99.13 %	-
model A	0.7M	**1**	97.37 %	91.1 %
model B	0.7M	**1**	99.10 %	93.8 %
model C (Proposed)	**0.7M**	**1**	**99.28 %**	**94.9 %**

LFW stands for Labeled Faces in the Wild and YTF stands for YouTube Faces.

Face Identification results

LFW stands for Labeled Faces in the Wild

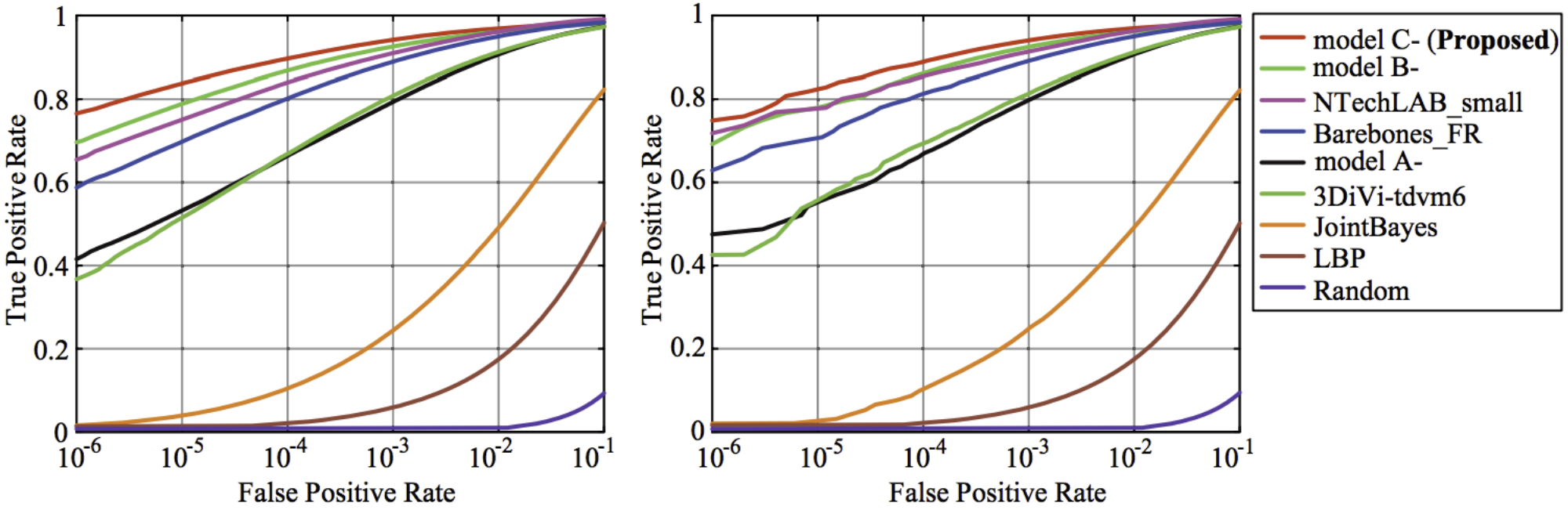
YTF stands for YouTube Faces



Face Verification results

LFW stands for Labeled Faces in the Wild

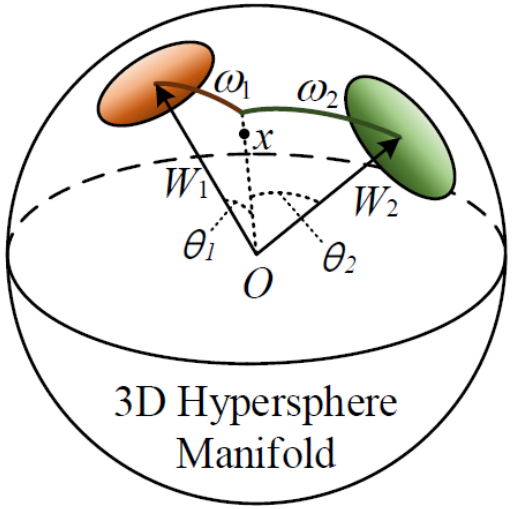
YTF stands for YouTube Faces



SphereFace

- An open set protocol.

- We propose here an angular softmax (A-softmax) loss that enables CNNs to learn angular discriminative features



Closed-set vs Open-set

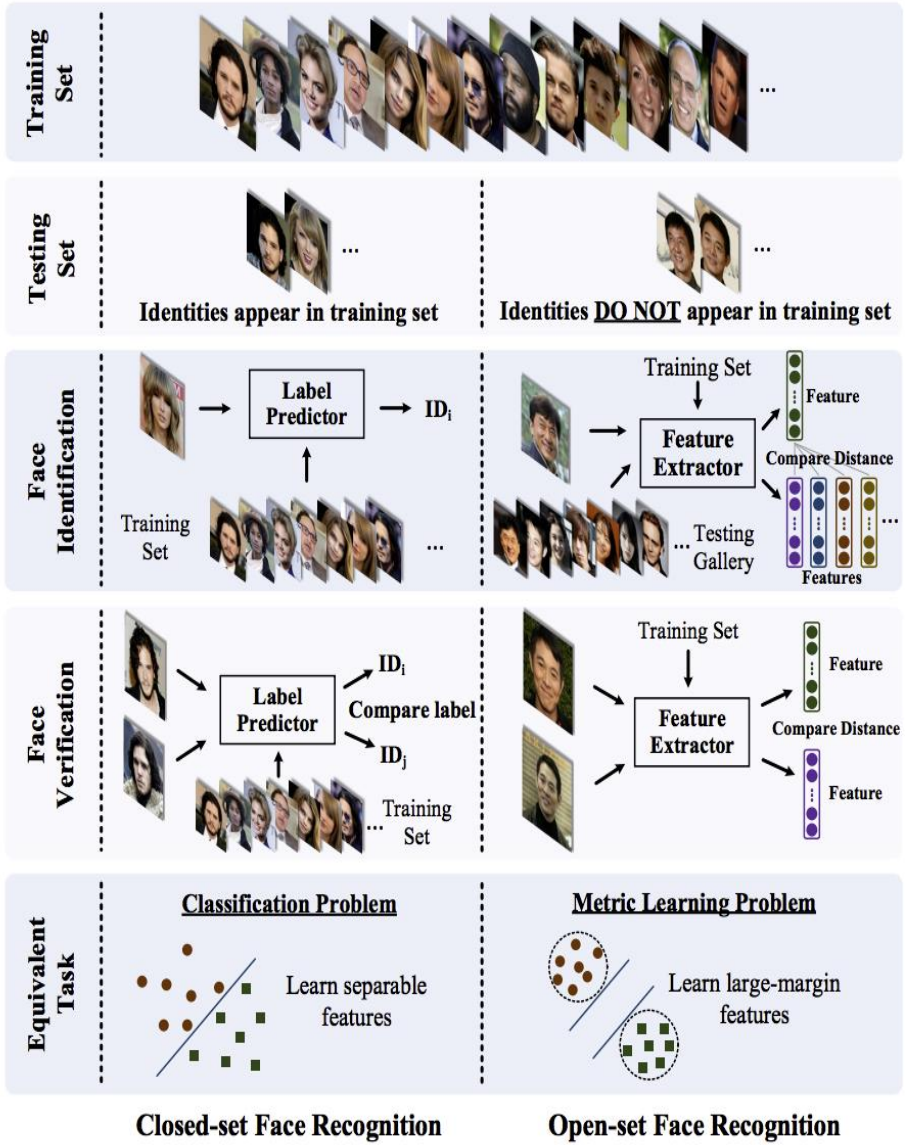


Figure 1: Comparison of open-set and closed-set face recognition.

Our mission

- Achieving maximal intra-class distance which is smaller than the minimal inter-class distance under a certain metric space.

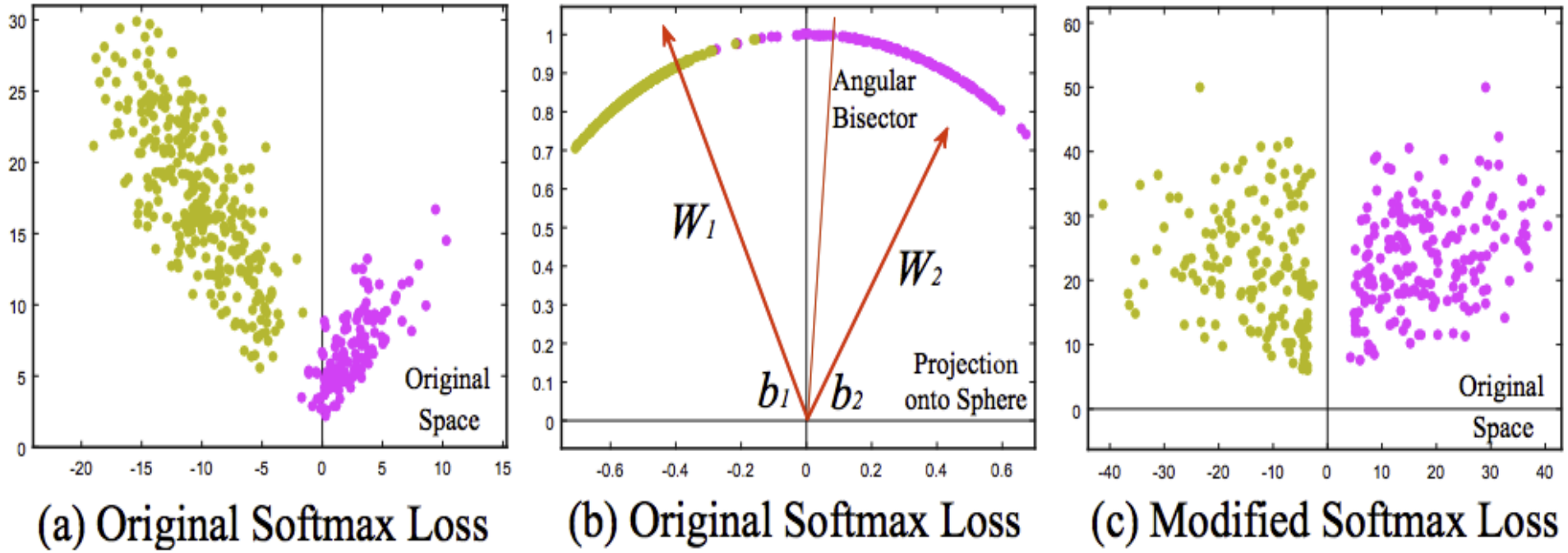
The Problem

Learning features with this criterion is difficult because of large intra-class variation and high inter-class similarity

The decision boundary in softmax loss is $(w1-w2)x+b1-b2=0$.

w_i and and b_i are weights and bias in softmax loss, respectively.
If x is a feature Vector and $||w1||=||w2||=1$ and b1=b2=0 the decision boundary becomes $||x||(\cos(\theta 1)-\cos(\theta 2))=0$. where θi is the angle Between wi and x. the new decision baundary depends only on $\theta 1$ and $\Theta 2$. modified sofmax loss is able to optimize angles, enabling the cnn to learn angular distributed features.

From Euclidean Margin to A-Softmax Loss



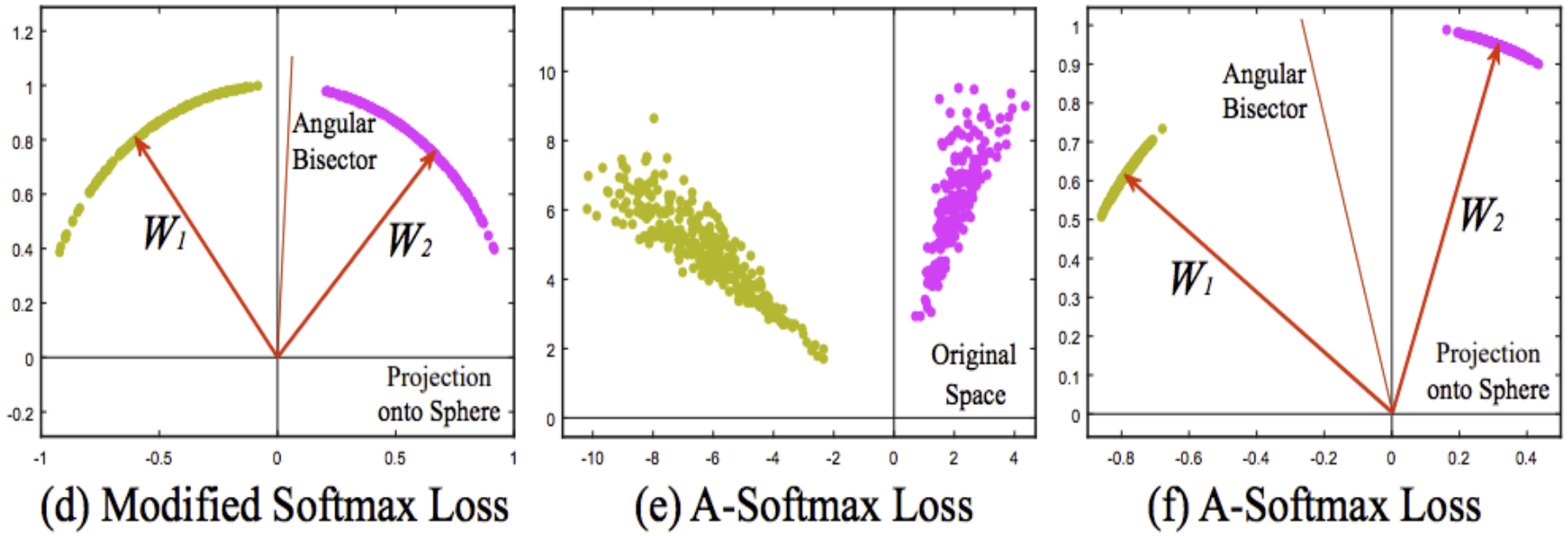
(a) Original Softmax Loss (b) Original Softmax Loss (c) Modified Softmax Loss

Angular softmax loss

- We introduce an integer m>=1 to quantitatively control the the angular the decision boundaries. In binary case, decision boundaries for class 1 becomes $||x||(\cos(m\theta 1) - \cos(\theta 2)) = 0$.

- By optimizing the A-softmax loss the decision regions become more separated and simultaneously enlarging the inter class margin and compressing the intra-class angular margin.

From Euclidian Margin to A-Softmax Loss



(d) Modified Softmax Loss (e) A-Softmax Loss (f) A-Softmax Loss

Deep Hypersphere Embedding

- Posterior probabilities obtained by softmax loss function (binary class example):

$$p_1 = \frac{\exp(\boldsymbol{W}_1^T \boldsymbol{x} + b_1)}{\exp(\boldsymbol{W}_1^T \boldsymbol{x} + b_1) + \exp(\boldsymbol{W}_2^T \boldsymbol{x} + b_2)} \quad (1)$$

$$p_2 = \frac{\exp(\boldsymbol{W}_2^T \boldsymbol{x} + b_2)}{\exp(\boldsymbol{W}_1^T \boldsymbol{x} + b_1) + \exp(\boldsymbol{W}_2^T \boldsymbol{x} + b_2)} \quad (2)$$

- Predicted labels will be assigned to class 1 if p1> p2, else, if p1<p2 assigned to class 2.

$$L = \frac{1}{N}\sum_i L_i = \frac{1}{N}\sum_i -\log\Big(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\Big) \tag{3}$$

where f_j denotes the j-th element ($j \in [1, K]$, K is the class number) of the class score vector $\boldsymbol{f}$, and N is the number of training samples. In CNNs, $\boldsymbol{f}$ is usually the output of a fully connected layer $\boldsymbol{W}$, so $f_j = \boldsymbol{W}_j^T \boldsymbol{x}_i + b_j$ and $f_{y_i} = \boldsymbol{W}_{y_i}^T \boldsymbol{x}_i + b_{y_i}$ where $\boldsymbol{x}_i$, $\boldsymbol{W}_j$, $\boldsymbol{W}_{y_i}$ are the i-th training sample, the j-th and y_i-th column of $\boldsymbol{W}$ respectively. We further reformulate L_i in Eq. (3) as

$$\begin{aligned} L_i &= -\log\Big(\frac{e^{\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_j e^{\boldsymbol{W}_j^T \boldsymbol{x}_i + b_j}}\Big) \\ &= -\log\Big(\frac{e^{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\cos(\theta_{y_i,i}) + b_{y_i}}}{\sum_j e^{\|\boldsymbol{W}_j\|\|\boldsymbol{x}_i\|\cos(\theta_{j,i}) + b_j}}\Big) \end{aligned} \tag{4}$$

in which $\theta_{j,i}(0 \leq \theta_{j,i} \leq \pi)$ is the angle between vector $\boldsymbol{W}_j$ and $\boldsymbol{x}_i$. As analyzed above, we first normalize $\|\boldsymbol{W}_j\| = 1, \forall j$ in each iteration and zero the biases. Then we have the modified softmax loss:

$$L_{\text{modified}} = \frac{1}{N} \sum_i -\log \left(\frac{e^{\|\boldsymbol{x}_i\| \cos(\theta_{y_i,i})}}{\sum_j e^{\|\boldsymbol{x}_i\| \cos(\theta_{j,i})}} \right) \tag{5}$$

Softmax Loss vs Modified Loss vs A-softmax Loss

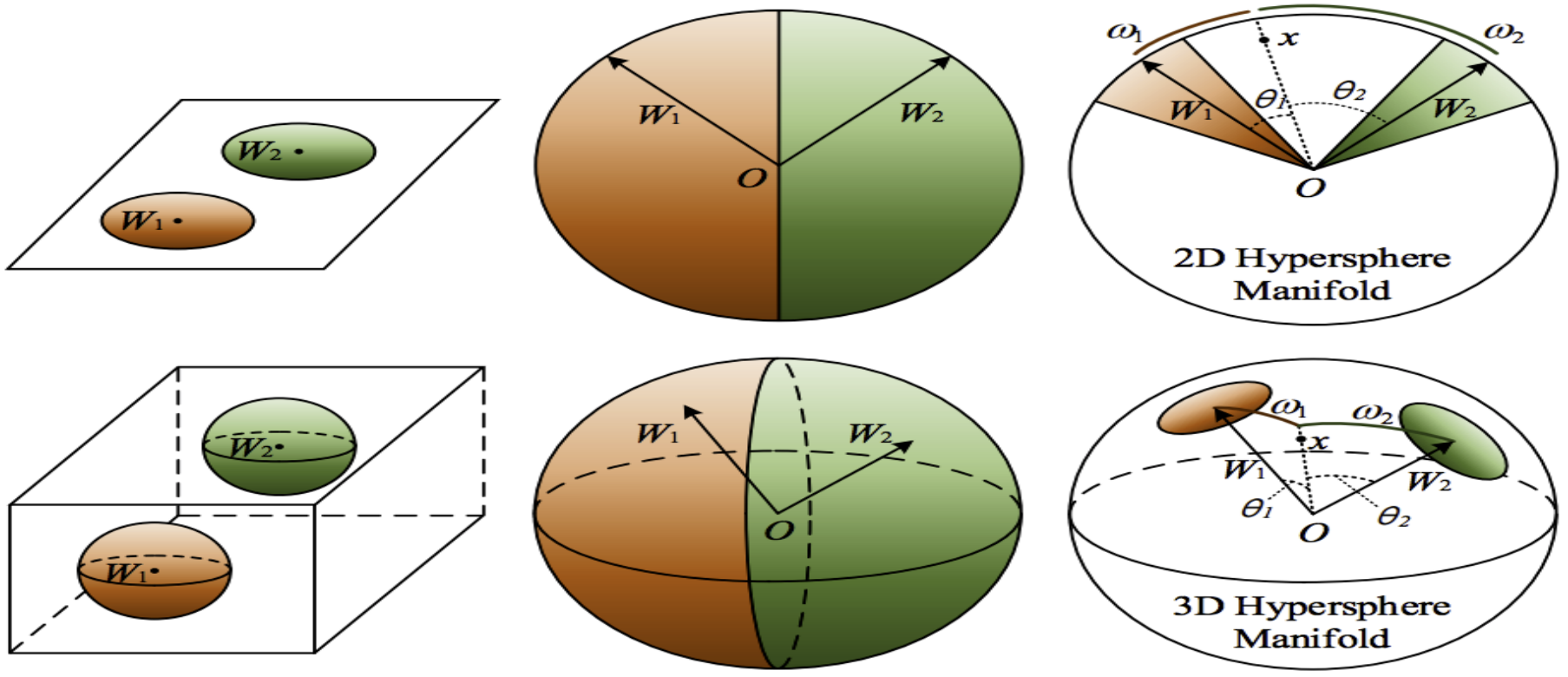


Figure 3: Geometry Interpretation of Euclidean margin loss (e.g. contrastive loss, triplet loss, center loss, etc.), modified softmax loss and A-Softmax loss. The first row is 2D feature constraint, and the second row is 3D feature constraint. The orange region indicates the discriminative constraint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

$$L_{\text{ang}} = \frac{1}{N}\sum_{i} -\log\Big(\frac{e^{\|\boldsymbol{x}_i\|\cos(m\theta_{y_i,i})}}{e^{\|\boldsymbol{x}_i\|\cos(m\theta_{y_i,i})} + \sum_{j\neq y_i} e^{\|\boldsymbol{x}_i\|\cos(\theta_{j,i})}}\Big) \tag{6}$$

where $\theta_{y_i,i}$ has to be in the range of $[0, \frac{\pi}{m}]$. In order to get rid of this restriction and make it optimizable in CNNs, we expand the definition range of $\cos(\theta_{y_i,i})$ by generalizing it to a monotonically decreasing angle function $\psi(\theta_{y_i,i})$ which should be equal to $\cos(\theta_{y_i,i})$ in $[0, \frac{\pi}{m}]$. Therefore, our proposed A-Softmax loss is formulated as:

$$L_{\text{ang}} = \frac{1}{N}\sum_{i} -\log\Big(\frac{e^{\|\boldsymbol{x}_i\|\psi(\theta_{y_i,i})}}{e^{\|\boldsymbol{x}_i\|\psi(\theta_{y_i,i})} + \sum_{j\neq y_i} e^{\|\boldsymbol{x}_i\|\cos(\theta_{j,i})}}\Big) \tag{7}$$

in which we define $\psi(\theta_{y_i,i}) = (-1)^k\cos(m\theta_{y_i,i}) - 2k$, $\theta_{y_i,i} \in [\frac{k\pi}{m}, \frac{(k+1)\pi}{m}]$ and $k \in [0, m-1]$. $m \geq 1$ is an integer that controls the size of angular margin. When $m = 1$, it becomes the modified softmax loss.

Properties of A- Softmax Loss

- Definition: m_min is the minimal value such that while m>m_min, A-Softmax loss defines a learning task where the maximal inta-class angular feature distance is constrain to be smaller than the minimal inter-class angular feature distance.

Why angular margin

- Faces lie on the manifold and angular margin links to discriminativeness on the manifold.

- Features learned by the original softmax loss have intrinsic angular distribution.

Comparisons with other existing losses

- Constructive loss and Triplet loss and Center loss only only impose Euclidian margin.

- Constructive loss and Triplet loss suffer from expansion when constituting the pairs/triplets from training set, while Angular Margin requires no sample mining and imposes discriminative constrain to our entire mini-batches.

How to train Sphereface features?

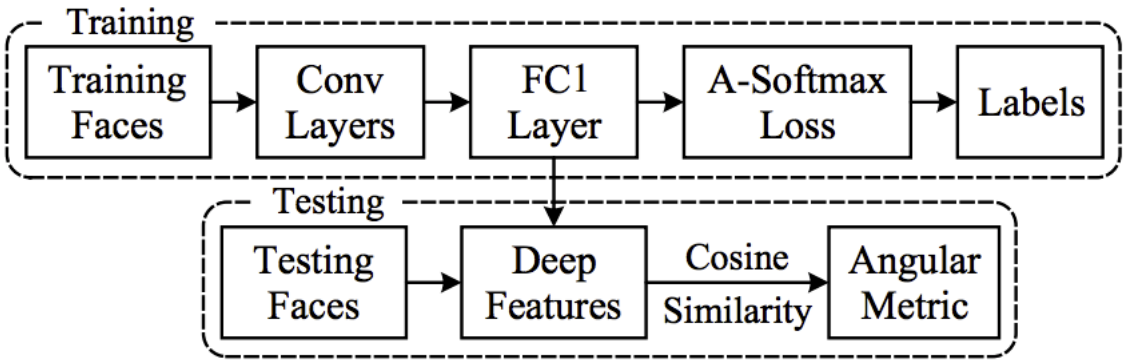


Figure 4: Training and Extracting *SphereFace* features.

Training Data

- We use 0.49 million face images belonging 10,500 individuals. These face images are horizontally flipped for data arguments. (to compare, other datasets including: DeepFace, VGGFace, and FaceNet use 2,4 and 200 million images, respectively).

Testing

The final representation of a testing face is obtained by concatenating its original face features and its horizontally flipped features.

Effect of m

- Larger m leads to a larger margin

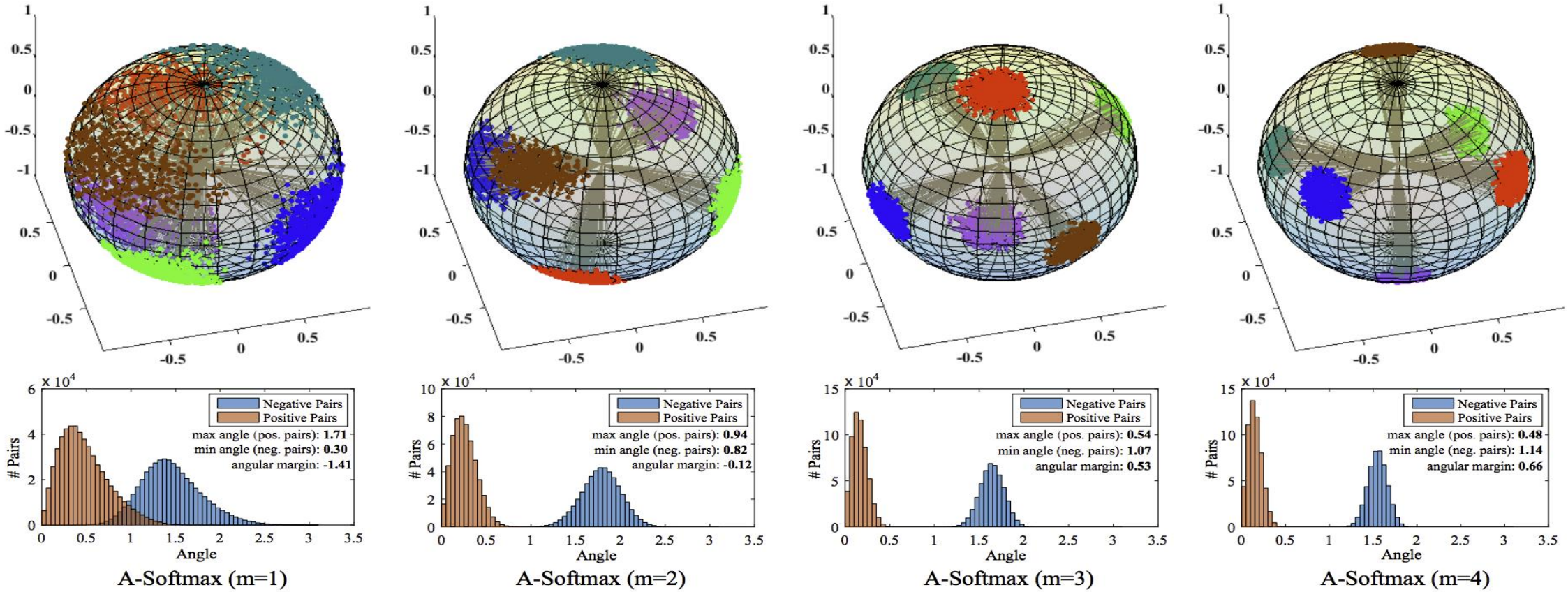


Figure 5: Visualization of features learned with different m. The first row shows the 3D features projected on the unit sphere. The projected points are the intersection points of the feature vectors and the unit sphere. The second row shows the angle distribution of both positive pairs and negative pairs (we choose class 1 and class 2 from the subset to construct positive and negative pairs). Orange area indicates positive pairs while blue indicates negative pairs. All angles are represented in radian. Note that, this visualization experiment uses a 6-class subset of the CASIA-WebFace dataset.

Experiment results

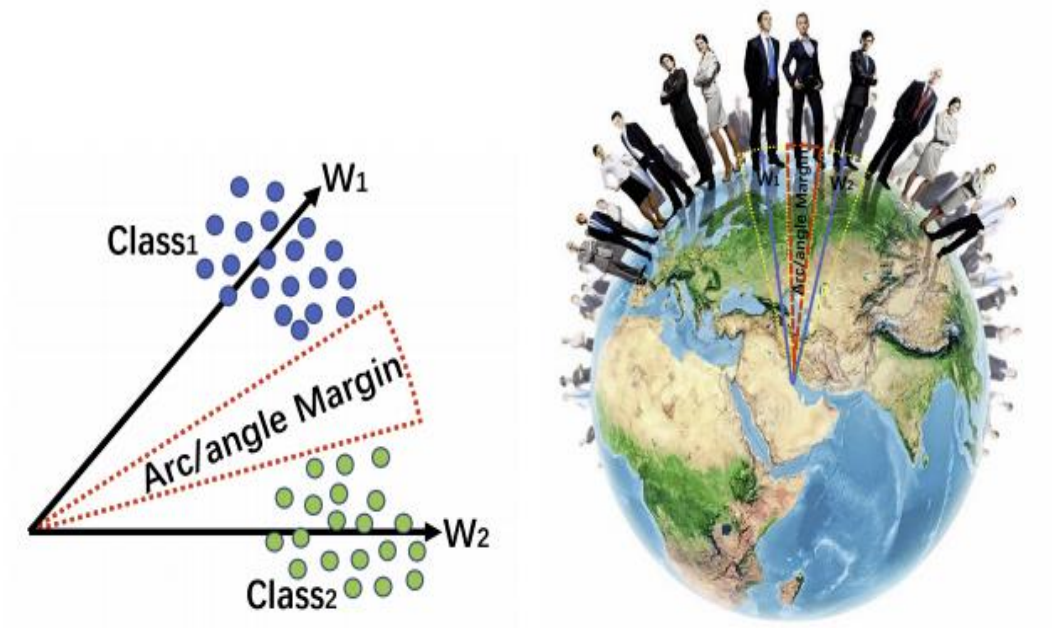
Dataset	Original	m=1	m=2	m=3	m=4
LFW	97.88	97.90	98.40	99.25	**99.42**
YTF	93.1	93.2	93.8	94.4	**95.0**

Table 3: Accuracy(%) comparison of different m (A-Softmax loss) and original softmax loss on LFW and YTF dataset.

ArcFace: Additive Angular Margin Loss for Deep Face Recognition

- Has a better geometrical interpretation than supervision signals we had so far.
- Arcface cos(theta + m) maximize decision boundary in angular space.
- Compared to multiplicative angular margin and additive cosine margin, Arcface can obtain more deep features

Enbedding feature vector such that features of the same person have smaller distance and features of different people have a considerable distance.



(a) ArcFace (b) Geodesic Correspondence

Figure 1. Geometrical interpretation of ArcFace. (a) Blue and green points represent embedding features from two different classes. ArcFace can directly impose angular (arc) margin between classes. (b) We show an intuitive correspondence between angle and arc margin. The angular margin of ArcFace corresponds to arc margin (geodesic distance) on the hypersphere surface.

Attributes

- 1. face recognition models from industry perform better than models from the academia, due to orders of magnitude difference in the training data scale.

- 2. high capacity deep convolutional network can obtain better performance.

Design of the loss function

- Angular:

Works by adding multiplicative angular constraint to each identity to improve feature discrimination.

Due to non monotonicity of the cosine function a piece-wise function is applied to ensure monotonicity.

Softmax loss is combined to ensure convergence.

Additive cosine margin moves angular margin to cosine space.

The most widely used classification loss function, Softmax loss, is presented as follows:

$$L_1 = -\frac{1}{m}\sum_{i=1}^{m}\log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}}, \tag{1}$$

We fix bias bj=0, and transform target logit as follows:

$$W_j^T x_i = \|W_j\|\,\|x_i\|\cos\theta_j, \tag{2}$$

Following [23, 43, 45], we fix $\|W_j\| = 1$ by L2 normalisation, which makes the predictions only depend on the angle between the feature vector and the weight.

$$L_2 = -\frac{1}{m}\sum_{i=1}^{m}\log\frac{e^{\|x_i\|\cos(\theta_{y_i})}}{e^{\|x_i\|\cos(\theta_{y_i})} + \sum_{j=1, j\neq y_i}^{n} e^{\|x_i\|\cos\theta_j}}. \tag{3}$$

In the experiments of SphereFace, L2 weight normalisation only improves little on performance.

Multiplicative Angular Margin

$$L_3 = -\frac{1}{m}\sum_{i=1}^{m}\log\frac{e^{\|x_i\|\cos(m\theta_{y_i})}}{e^{\|x_i\|\cos(m\theta_{y_i})}+\sum_{j=1,j\neq y_i}^{n}e^{\|x_i\|\cos\theta_j}}, \tag{4}$$

where $\theta_{y_i} \in [0, \pi/m]$. In order to remove this restriction, $\cos(m\theta_{y_i})$ is substituted by a piece-wise monotonic function $\psi(\theta_{y_i})$. The SphereFace is formulated as:

$$L_4 = -\frac{1}{m}\sum_{i=1}^{m}\log\frac{e^{\|x_i\|\psi(\theta_{y_i})}}{e^{\|x_i\|\psi(\theta_{y_i})}+\sum_{j=1,j\neq y_i}^{n}e^{\|x_i\|\cos\theta_j}}, \tag{5}$$

where $\psi(\theta_{y_i}) = (-1)^k\cos(m\theta_{y_i}) - 2k, \theta_{y_i} \in \left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m}\right], k \in [0, m-1]$, $m \geqslant 1$ is the integer

$$\psi(\theta_{y_i}) = \frac{(-1)^k \cos(m\theta_{y_i}) - 2k + \lambda \cos(\theta_{y_i})}{1+\lambda}. \quad (6)$$

$$L_5 = -\frac{1}{m}\sum_{i=1}^{m} \log \frac{e^{s\psi(\theta_{y_i})}}{e^{s\psi(\theta_{y_i})} + \sum_{j=1, j\neq y_i}^{n} e^{s\cos\theta_j}}. \quad (7)$$

m is removed outside the cos(theta) and we get:

$$L_6 = -\frac{1}{m}\sum_{i=1}^{m} \log \frac{e^{s(\cos(\theta_{y_i})-m)}}{e^{s(\cos(\theta_{y_i})-m)} + \sum_{j=1, j\neq y_i}^{n} e^{s\cos\theta_j}}. \quad (8)$$

Additive angular margin

We add an angular margin m within $\cos\theta$. Since $\cos(\theta + m)$ is lower than $\cos(\theta)$ when $\theta \in [0, \pi - m]$, the constraint is more stringent for classification. We define the proposed ArcFace as:

$$L_7 = -\frac{1}{m}\sum_{i=1}^{m}\log\frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))}+\sum_{j=1,j\neq y_i}^{n}e^{s\cos\theta_j}}, \tag{9}$$

subject to

$$W_j = \frac{W_j}{\|W_j\|}, x_i = \frac{x_i}{\|x_i\|}, \cos\theta_j = W_j^T x_i. \tag{10}$$

If we expand the proposed additive angular margin $\cos(\theta + m)$, we get $\cos(\theta + m) = \cos\theta\cos m - \sin\theta\sin m$. Compared to the additive cosine margin $\cos(\theta) - m$ proposed in [44, 43], the proposed ArcFace is similar but the margin is dynamic due to $\sin\theta$.

In Figure 2, we illustrate the proposed ArcFace, and the angular margin corresponds to the arc margin. Compared to SphereFace and CosineFace, our method has the best geometric interpretation.

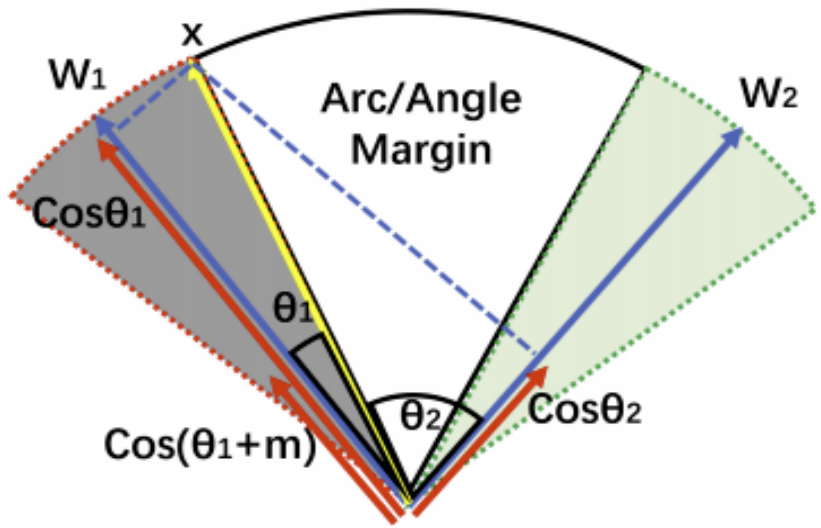


Figure 2. Geometrical interpretation of ArcFace. Different colour areas represent feature spaces from distinct classes. ArcFace can not only compress the feature regions but also correspond to the geodesic distance on the hypersphere surface.

Comparison under binary case:

Loss Functions	Decision Boundaries
Softmax	$(W_1 - W_2)\,x + b_1 - b_2 = 0$
W-Norm Softmax	$\|x\|\,(\cos\theta_1 - \cos\theta_2) = 0$
SphereFace [23]	$\|x\|\,(\cos m\theta_1 - \cos\theta_2) = 0$
F-Norm SphereFace	$s(\cos m\theta_1 - \cos\theta_2) = 0$
CosineFace [44, 43]	$s(\cos\theta_1 - m - \cos\theta_2) = 0$
ArcFace	$s(\cos(\theta_1 + m) - \cos\theta_2) = 0$

Table 1. Decision boundaries for class 1 under binary classification case. Note that, θ_i is the angle between W_i and x, s is the hypersphere radius, and m is the margin.

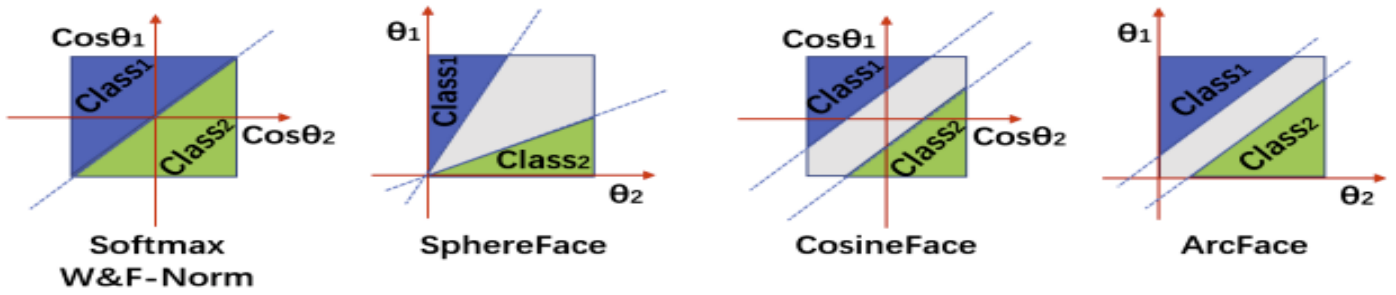
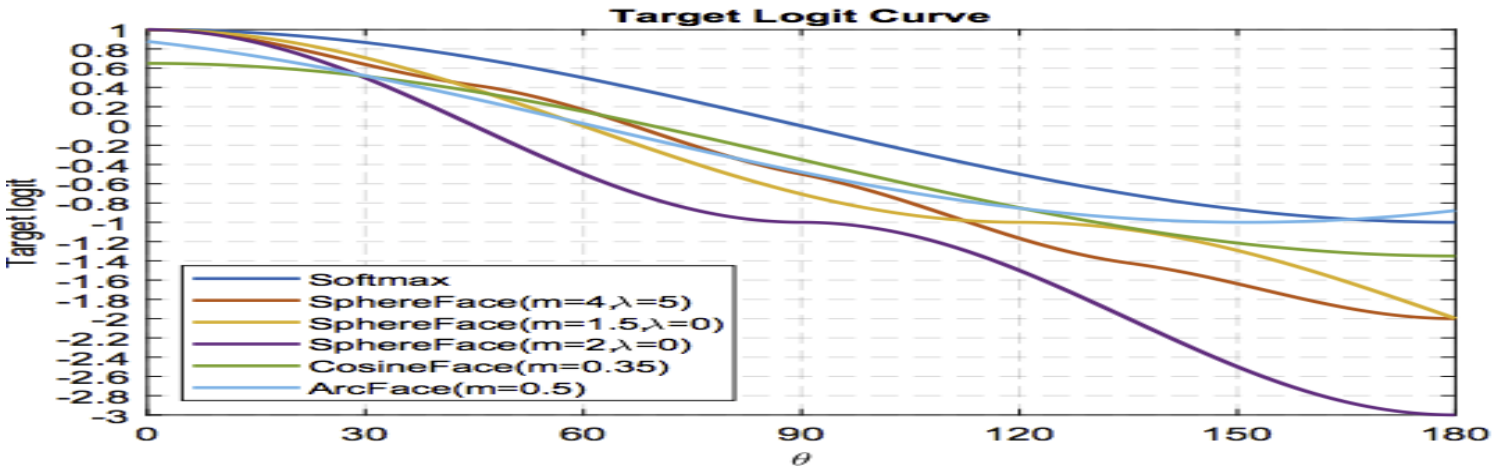
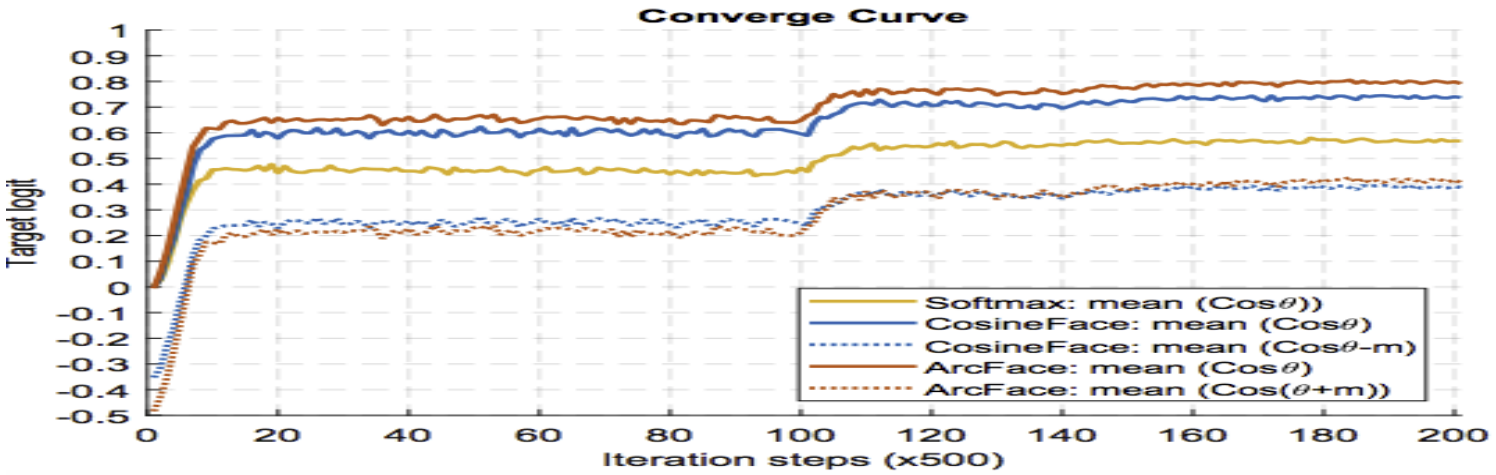


Figure 3. Decision margins of different loss functions under binary classification case. The dashed line represents the decision boundary, and the grey areas are the decision margins.

Target Logit Analysis



(a) Target Logit Curves



(b) Target Logit Converge Curves

Data-VGG2 and MS-Celeb-1M

(a) Alec Baldwin

(b) Distractors Noise

Figure 6. Noisy face image examples from the MegaFace distractors. (a) is used for annotators to learn the identity from the FaceScrub dataset. (b) shows the selected overlap faces from the MegaFace distractors.

Output setting

- How the embedding settings effect the model performance?

- Option A- Use global pooling layer(GP).
- Option-B: Use one fully connected (FC) layer after GP.
- Option-C: Use FC-Batch Normalization (BN) after GP.
- Option-D: Use FC-BN-Parametric Rectified Linear Unit (PReLu) after GP.
- • Option-E: Use BN-Dropout -FC-BN after the last convolutional layer.

Results

Methods	Rank1@10^6	VR@FAR10^{-6}	Rank1@10^6 (R)	VR@FAR10^{-6} (R)
Softmax	78.89	94.95	91.43	94.95
Softmax-pretrain,Triplet-finetune	80.6	94.65	94.08	95.03
Softmax-pretrain@VGG2, Triplet-finetune	78.87	95.43	93.96	95.07
SphereFace(m=4, λ=5)	82.95	97.66	97.43	97.66
CosineFace(m=0.35)	82.75	98.41	98.33	98.41
ArcFace(m=0.4)	82.29	98.20	98.10	97.83
ArcFace(m=0.5)	**83.27**	**98.48**	**98.36**	**98.48**

Table 9. Identification and verification results of different methods on MegaFace Challenge1 (LResNet100E-IR@MS1M). "Rank 1" refers to the rank-1 face identification accuracy and "VR" refers to face verification TAR (True Accepted Rate) at 10^{-6} FAR (False Accepted Rate). (R) denotes the refined version of MegaFace dataset.

References:

- Yandong Wen, Kaipeng Zhang, Zhifeng Li1, and Yu Qiao, Discriminative Feature Learning Approach for Deep Face Recognition, 2018.
- Weiyang Liu Yandong Wen Zhiding Yu Ming Li Bhiksha Raj Le Song, SphereFace: Deep Hypersphere Embedding for Face Recognition, 2018.
- Jiankang Deng, Jia Guo, Stefanos Zafeiriou, ArcFace: Additive Angular Margin Loss for Deep Face Recognition, 2018