Identification Systems

Lecture 9

Piotr Syga

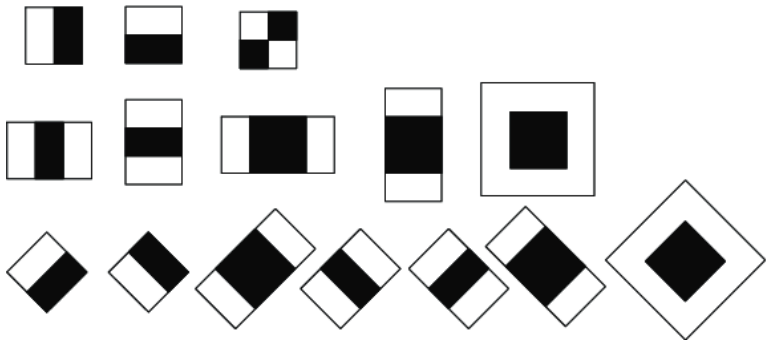
2021-05-11

Requirements

- The method should be efficient – facial image may cover relatively small part of high resolution picture.
- The method should be universal.
- We aim for robustness against scale, uneven lighting, occlusions, noisy image, compression, blurring and environmental conditions.
- We want to minimize the influence of face alignment.

Main types: sliding window or region-based localization.

Haar-like features



A single strong classifier $h(x)$ is composed as $h(x) = \text{sign}(\sum_j w_j h_j(x))$, where w_j are weak classifier weights,

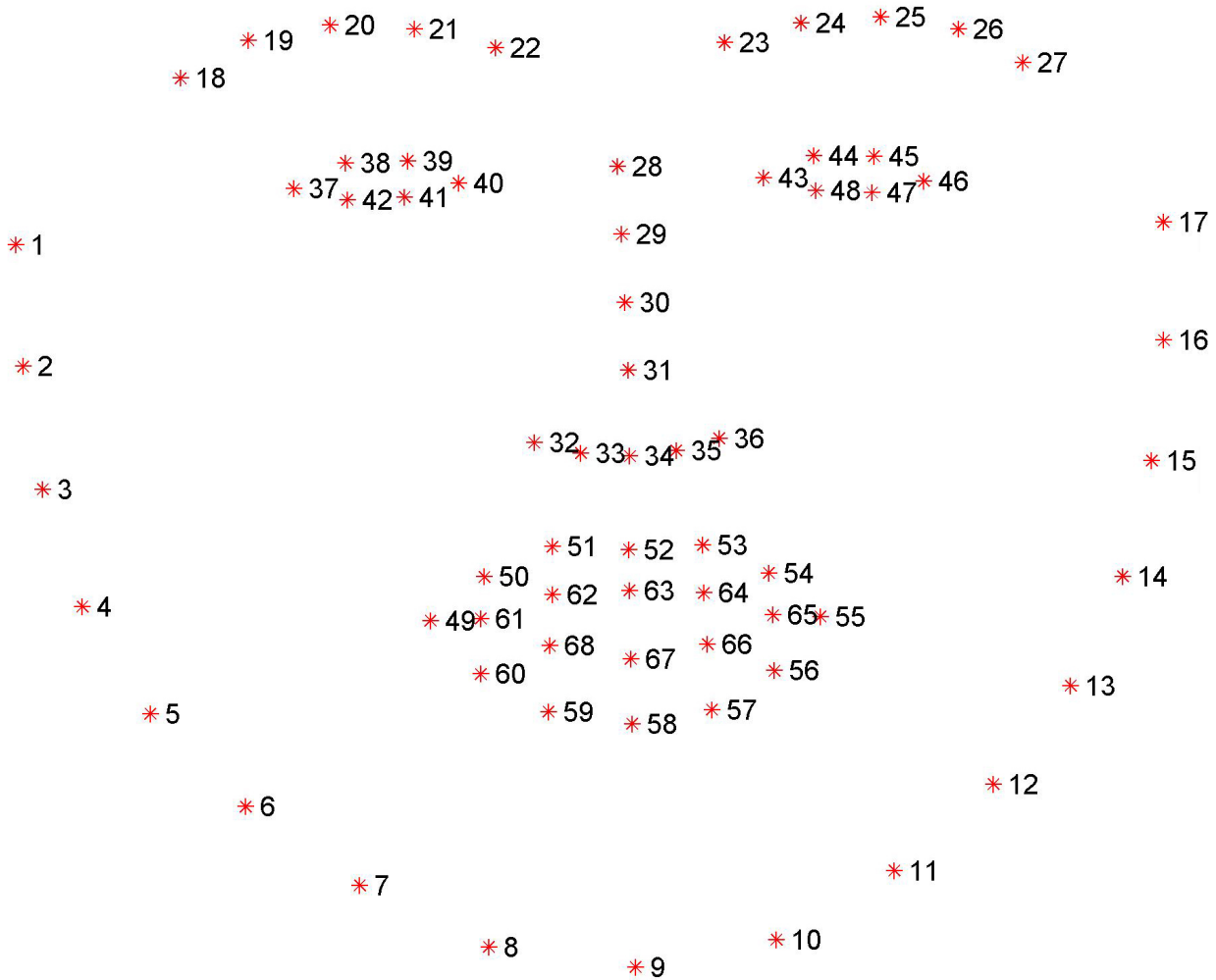
$$h_j(x) = \begin{cases} -1 & p_j f_j < p_j T_j \\ 1 & \text{oth.} \end{cases}$$

with polarity $p_j \in \{-1, 1\}$, f_j – an output of rectangular filter and T_j – threshold acceptance level.
The parameters w, p, f and T are learned using AdaBoost in the training stage.
Classification is made based on a cascade of length N that eliminates false positive matches – we assume that we start with a face and abort if a strong classifier determines otherwise.

Other methods

- A simple method for face localization utilizes HoG and a classifier – a linear SVM or CNN, e.g. using dlib.
- Face detection using colors and PCA.
- Face detection using HoG-like edge orientation.
- You only look once detection. (YOLO paper)

Landmarks – recalled



Facial alignment

- The face orientation both in cases of rotation and tilt may affect face detection and identification.
- A proper localization of the landmarks may help with both identification and face realignment.
- Decision whether to perform realignment and normalization is not an obvious one: matching may be performed after the normalization or pose synthesis.
- Due to bounding box rescaling before the measurement even small misidentification of landmark localization may have significant effect on the user identification if we use landmarks to detect faces.

Facial alignment

Examples of misidentified and corrected features, limited to detecting 5 key features by Zhang et al.: Facial Landmark Detection by Deep Multi-task Learning.



Eigenface

- The idea focuses on lowering the dimension of facial features.
- We translate each face to a linear combination of some n base faces.
- Using eigenvector decomposition of an image we obtain features 'ordered by their importance'.
- With PCA we lower the dimension of the vectors.

Eigenface

- We start with a base set of n face images of set $N \times N$.
- We concatenate the columns of each face so we operate on $N^2 \times 1$ vectors.
- We realign coordinate system: calculate a mean face of the set and subtract it from all the faces.
- We concatenate all the columns into a matrix A of size $N^2 \times n$ and calculate its covariance $A^T A$.
- We calculate eigenvectors and eigenvalues (with dimension reduction).
- We select k eigenvectors for $k < n$ largest eigenvalues.
- Each normalized in step 3. face is represented as linear combination of the selected eigenvectors.
- Linear coefficients are used as a user feature vector.
- During the identification we subtract the mean face from the sample face, project it into eigenspace and find a vector that minimizes the distance. A threshold may apply.

Eigenface

- Main advantages include: reduction in number of features – lowering the complexity and acceptable scaling for large databases, realtime identification after the enrollment, simple steps, does not require facial landmarks.
- Main drawbacks include: low robustness against face mimic and lighting changes as well as sensitivity to affine transforms.

The idea was enhanced in Fisherface recognition (esp. illumination invariance). Both methods are available in `OpenCV`: `face.createEigenFaceRecognizer()`, `face.createFisherFaceRecognizer()`. Comparison between the methods may be found in Belhumeur et al..

Local Binary Pattern

Idea:

T. Ojala, M. Pietikäinen, D. Harwood: *Performance evaluation of texture measures with classification based on Kullback discrimination of distributions*

- analyzing texture of an image
- presenting a visual descriptor that may be combined with others (e.g., HOG)
- we divide an image into multiple cells of constant size
- we compare the value of pixel p to all pixels in $N_8(p)$
- we interpret the values as an 8-bit integer
- we calculate and normalize the histogram for the whole cell
- we use concatenated histograms as a characteristic vector for an image
- classification: closest match, exact match, k-NN, SVM or CNN classifier

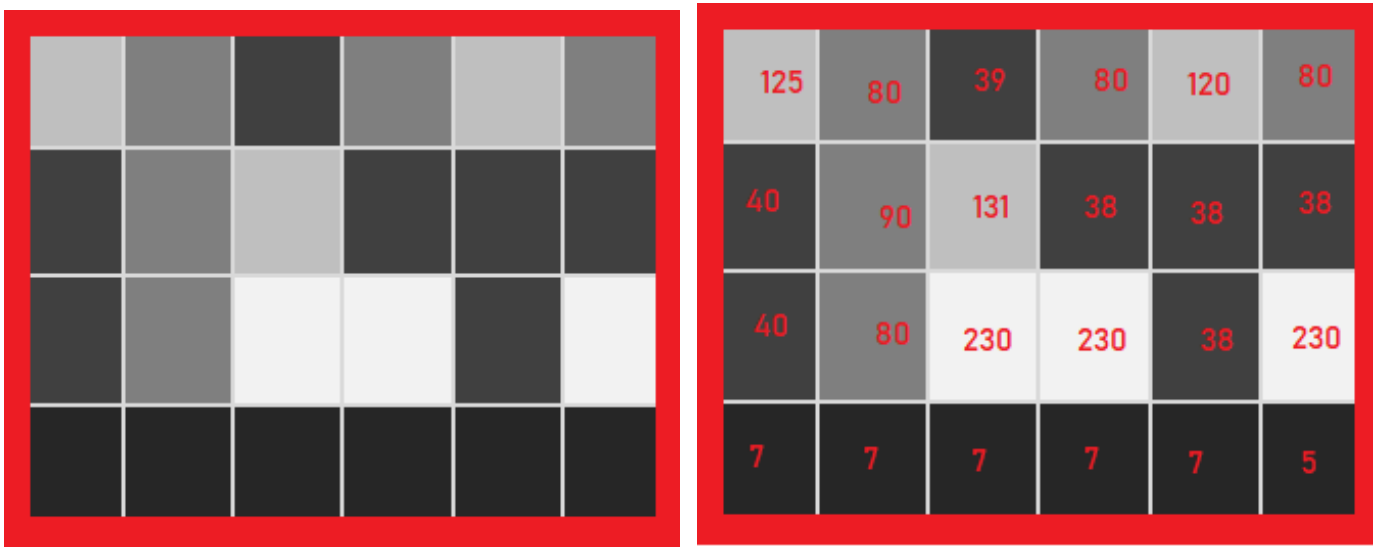
Local Binary Pattern

Cells before conversion to a single channel:



Local Binary Pattern

Selecting a cell and coding:



Local Binary Pattern

Shifting window:



Local Binary Pattern

Mapping into integer:

- in the stated order, each pixel's value is compared to the central value
- if the central value is larger we assign 0, otherwise we assign 1
- we derive a binary number, e.g. 10011000
- we could calculate the decimal value (often calculated counter clockwise) – 153 (or 25) and assign the calculated value to the pixel with the centrals' coordinates
- . . . yet it would require 256 bins making histograms sparse and sensitive to noise
- we reduce the number of bins to 59

Local Binary Pattern

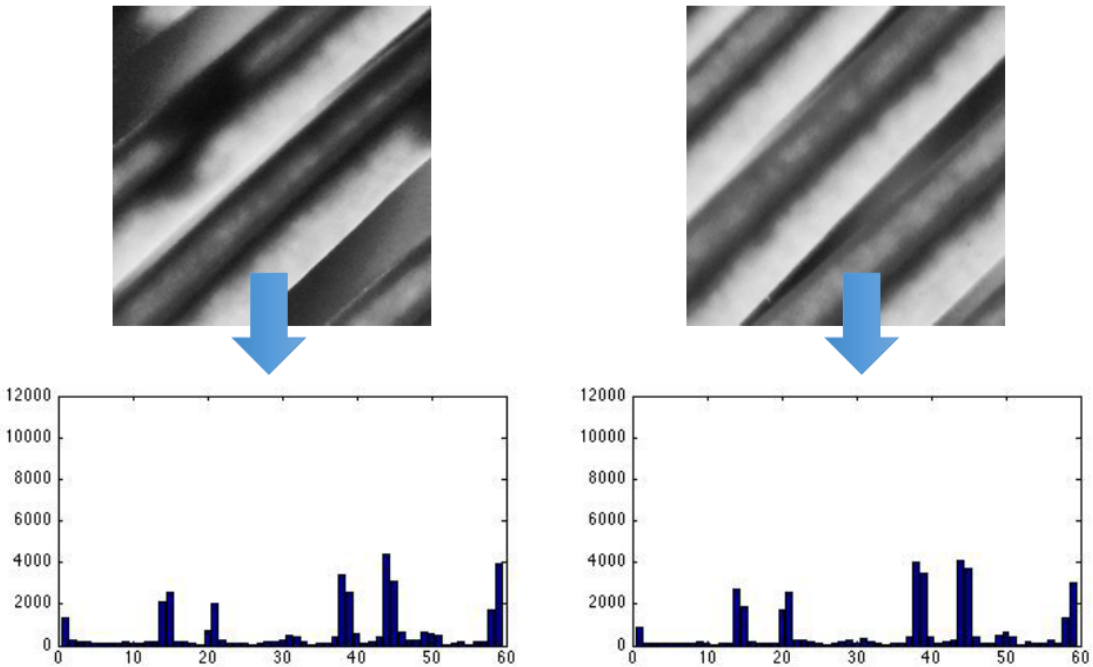
Uniform pattern

We call a binary pattern uniform if contains at most two bitwise transitions between 0s and 1s in cyclic reading.

- there are 58 uniform patterns
- uniform patterns cover almost 90% of the patterns (*Ojala et al.: Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns.*)
- hence we create unique bins for the uniform patterns and a separate for all non-uniform patterns
- each cell is represented by a sequence of 59 digits denoting the number of pixels in the cell that fall into given bin
- we normalize the histogram as in HOG and a concatenation of all histograms of an image represents its descriptor
- we utilize the fact that similar textures have similar histograms

Local Binary Pattern

Robustness against noise and intensity changes:



LBP in Facial Recognition

- We enroll a database o N users with M facial images for each subject.
- Each image is divided into $k = 2^n$ cells for LBP histogram.
- We arrange the data into $N \times M$ rows and $59 \times k$ columns.
- Given a biometric sample S, we calculate the histogram for S.
- We calculate the difference between S and each of the rows.
- We assume that the minimal difference will be responsible for the legitimate user.

LBP in Facial Recognition

Problems:

- even though LBP is robust against value changes it is vulnerable against rotations – various head poses and tilts (may be mitigated by appropriate selection of M images)
- facial hair and accessories change the texture
- proper bounding box and uniform background is required for reliable accuracy
- make up, skin condition, haircut change or spot-focused light influence the calculated distances

Neural Networks – idea

- deep networks with series of specialized layers, often based on previous facial recognition methods
- first layers responsible for reducing the amount of the information
- convolutional layers responsible for feature extraction
- sequence of extractors, with layers extracting features from the results of previous ones
- dropouts for reducing the influence of noise and specific features that may easily overfit
- determining user representation in the latent layers
- nonlinearity introduced by (possibly various) activation functions
- fully connected layers for feature propagation and classification

Example of face recognition network: InsightFace

FaceNet

F. Schroff, D. Kalenichenko, J. Philbin: *FaceNet: A Unified Embedding for Face Recognition and Clustering*

- mapping of images into Euclidean space $\mathbb{R}^d$
- introducing of triplet loss (match and non-match patches)
- compact representation: 128 bytes per face
- efficient identification

FaceNet

Triplet loss:

- motivated by k-NN classification
- learned representation aims to put subjects anchor so that it is closer to all positive samples (points representing other images of the same subjects) than to any of the negative samples, namely $\forall_{(x_i^a, x_i^p, x_i^n)} \|x_i^a - x_i^p\|_2^2 + \alpha < \|x_i^a - x_i^n\|_2^2$
- the loss that is minimized is $\mathcal{L} = \sum_i \|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha - \|f(x_i^a) - f(x_i^n)\|_2^2$, for the image mapping $f(\cdot)$, such that $\|f(x)\|_2 = 1$.
- note that many trivial (non-informative) anchors may be selected

FaceNet

Triplet Selection:

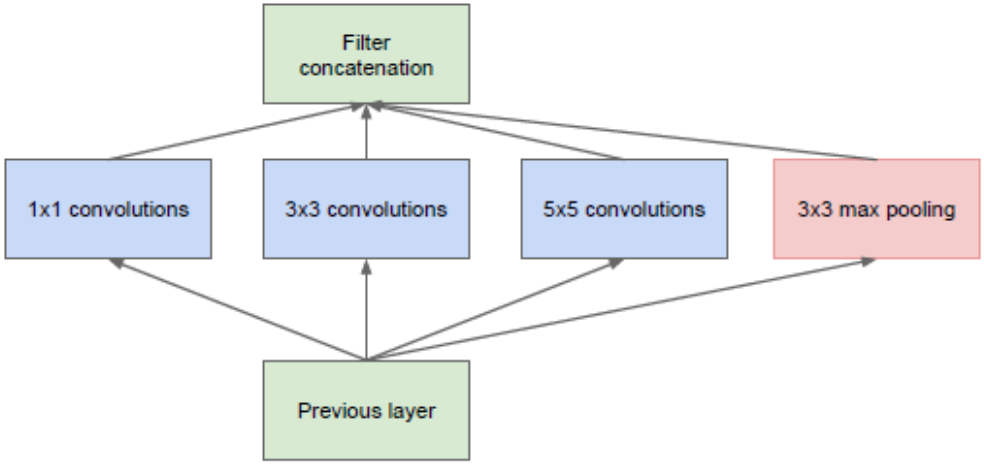
- as many non-informative anchors that satisfy the constraint may be found, we need to properly select points to quicken the convergence
- we aim at finding hard positive and negative, i.e., $\text{argmax}_{x_i^p} \|f(x_i^a) - f(x_i^p)\|_2^2$ and $\text{argmin}_{x_i^n} \|f(x_i^a) - f(x_i^n)\|_2^2$
- finding true argmax and argmin is infeasable for the whole set – we use subset of the data every n steps in an offline manner or from a mini-batch during online training (mini-batch size of 40 is used by the authors)
- anchor-positive pairs are more stable
- for mitigating the effect of local optima in the early stages, semi-hard (without the α thresholding) negative are chosen

FaceNet

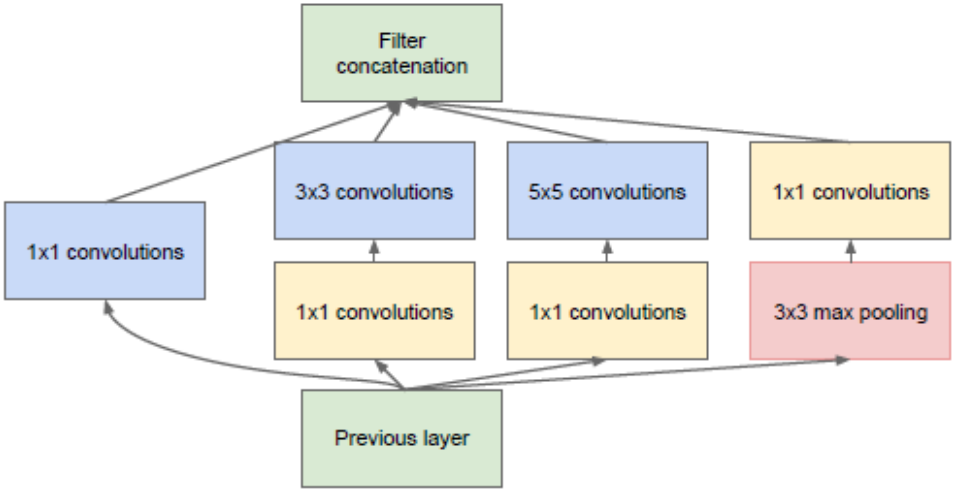
The networks:

- SGD is used for optimization and ReLU activation function
- the triplet threshold is set to $\alpha = 0.2$
- two models are presented, first based on Zeiler&Fergus network with $1 \times 1 \times 4$ convolutional layers before each $3 \times 3 \times X$ convolution, resulting in 22 layers and 140M parameters, the second – based on GoogLeNet Inception (mainly L_2 pooling instead of maxpooling), resulting in 17 layers (11 of those being inception layers) and 7.5M parameters

FaceNet



(a) Inception module, naïve version



(b) Inception module with dimensionality reduction

FaceNet

Basic Accuracy results at FAR=10^{-3}:

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

UniformFace

Y. Duan, J. Li, J. Zhou: *UniformFace: Learning Deep Equidistributed Representation for FaceRecognition*

- focusing on holistic features instead of maximizing inter-class and reducing intra-class distances
- spreading facial features of each subject on a hypersphere uniformly spreading the class centers
- minimum distance between class centers can be maximized through complete exploitation of the feature space
- approach similar to k-Means algorithm

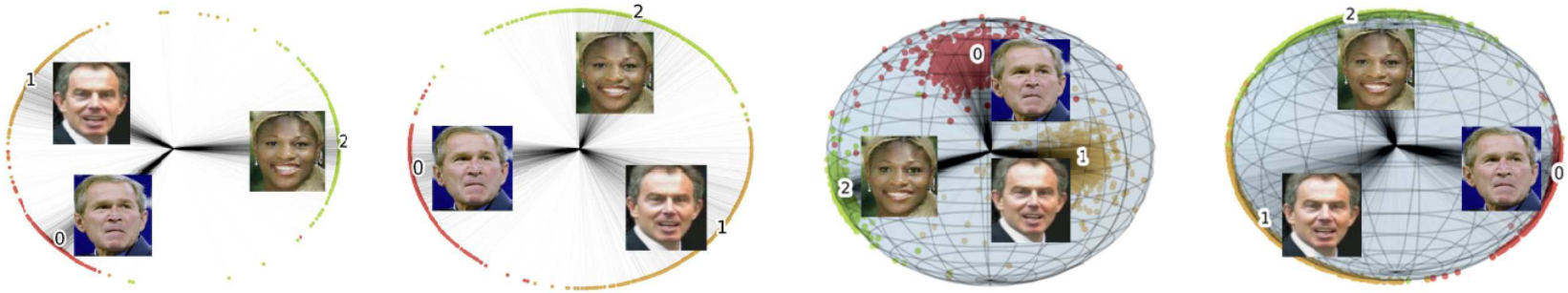
UniformFace

The loss function:

- modification of soft-max loss function
  $(\mathcal{L}_s = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{w_{y_i}^T x_i + b_{y_i}}}{\sum_j^M e^{w_j^T x_i + b_j}})$
- substituting Euclidean metric by cosine distance and introducing angular manipulation margin m:
  $\mathcal{L}_{a-s} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{\|x_i\|\psi(\theta_{y_i,i})}}{e^{\|x_i\|\psi(\theta_{y_i,i})}+\sum_{j\neq y_i} e^{\|x_i\|\cos(\theta_{j,i})}}$, where
  $\psi(x) = (-1)^k\cos(mx) - 2k$ and $x \in [\frac{k\pi}{m}, \frac{(k+1)\pi}{m}]$, $k \in [0, m-1]$
- utilizing properties of charges separating on a sphere, to achieve equiseparated centers, the authors use additional factor to loss function that is derived from Coulomb's law:
  $\mathcal{L}_u = \frac{\lambda}{M(M-1)}\sum_{j=1}^{M}\sum_{\hat{j}\neq j}\frac{1}{d(c_j, c_{\hat{j}})}$ and derive $\mathcal{L} = \mathcal{L}_{a-s} + \mathcal{L}_u$
  (SphereFace abandons the $\mathcal{L}_u$ term)

UniformFace

Idea of the distribution:



(a) SphereFace-2D (b) UniformFace-2D (c) SphereFace-3D (d) UniformFace-3D

UniformFace

Accuracy at TAR=10^{-2}, at FAR=10^{-3}, Top-1 and Top-5 classification on IJB-A dataset

SphereFace	92.3 ± 1.6	88.4 ± 4.2	93.2 ± 1.3	96.5 ± 1.1
UniformFace	$\mathbf{96.9 \pm 0.8}$	$\mathbf{92.3 \pm 1.7}$	$\mathbf{97.9 \pm 0.5}$	$\mathbf{98.8 \pm 0.2}$

Verification rate on YFT and LFW datasets.

SphereFace	96.1	99.5
UniformFace	**97.7**	**99.8**

Note: Similar aproach may be found in K. Zhao, J. Xu, M.-M. Cheng: *RegularFace: Deep Face Recognition via Exclusive Regularization*, where authors focus on maximizing inter-class distances via regularization