3.2 Training and testing a FR deep model

After preprocessing, face images with their ground truth IDs can be used to train a FR deep model. As the progressing of computational hardware, any mainstream backbone is able to build a FR network. Different backbones with similar parameter amount have similar accuracy on FR. As a result, you can use Resnet, ResNext, SEResnet, Inception net, Densenet, etc. to form your FR system. If you need to design a FR system with limited calculation resource, Mobilenet with its variation will be good options. In addition, NAS can be used to search a better network hyper-parameter. Network distilling technology is also welcomed while building industrial FR system.

The training algorithms will be elaborated in the following sections. For testing, a face image after preprocessing can be inputted into a trained FR model to get its face embedding. Here, we list some practical tricks which will be utilized in the model training and testing steps. Firstly, training image augmentation is useful, especially for IDs with inadequate samples. Common augmentation ways, such as adding noise, blurring, modifying colors, are usually employed in FR training. However, randomly cropping should be avoided, since it will ruin face alignment results. Secondly, flipping face images horizontally is a basic operation in both of training and testing FR model. In training, flipping faces can be consider as a data augmentation process. In testing, we can put image I and its flipped mirror image I' into the model, and get their embeddings f and f'. Then their mean feature $\frac{f+f'}{2}$ can be used as the embedding of image I. This trick will further improve the accuracy of FR results.

3.3 Face recognition by comparing face embeddings

The last part of face inference pipeline is face recognition by comparing face embeddings. According to the application, it can be divided into face verification and face identification. To apply FR, a face gallery needs to be built. First we have a face ID set S, where each ID contains one (or several) face image(s). All face embeddings from the gallery images will be extracted by a trained model and saved in a database.

The protocol of face verification (FV) is: giving a face image I and a specific ID in the gallery, output a judgement whether the face I belongs to the ID. So FV is a 1:1 problem. We extract the feature of image I and calculate its similarity $s(I, ID)$ with the ID's embedding in the database. $s(I, ID)$ is usually measured by cosine similarity. If $s(I, ID)$ is larger than a threshold μ, we will output 'True' which represents image I belongs to the ID, and vice verse.

However, face identification (FI) is a 1:N problem. Giving a face image I and a face ID set S, it outputs the ID related to the face image, or 'not recognized'. Similarly, we extract the feature of I and calculate its similarity $s(I, ID), ID \in S$ with all IDs' embeddings in the database. Then, we find the maximum of all similarity values. If $\max(s(I, ID))$ is larger than a threshold μ, we output its related ID $\arg_{ID} \max(s(I, ID))$ as identification target; otherwise, we output 'not recognized', which shows that the person of image I is not in the database. Phan *et al.* [39] enriched the FI process by adopting an extra Earth Mover's Distance. They first used cosine similarity to obtain a part of the most similar faces of the query image. Then a patch-wise Earth Mover's Distance was employed to re-rank these similarities to get final identification results. In the subsection 4.5, we will introduce some methods to accelerate the process of face identification.

4 Algorithms

In this section, we will introduce FR algorithms in recent years. Based on different aspects in deep FR modeling, we divided all FR methods into several categories: designing loss function, refining embedding, FR with massive IDs, FR on uncommon images, FR pipeline acceleration, and close-set training.

4.1 Loss Function

4.1.1 Loss based on metric learning

Except from softmax based classification, FR can be also regarded as extracting face features and performing feature matching. Therefore, training FR model is a process to learn a compact euclidean feature space, where distance directly correspond to a measure of face similarity. This is the basic modeling of metric learning. In this section, we introduce some representative FR methods based on metric learning.

Inspired by the pipeline of face verification, one intuitive loss designing is judging whether two faces in a pair have identical ID or not. Han *et al.* [40] used this designing in a cross-entropy way, and the loss is:

$$L_{i,j} = -[y_{ij} \log p_{ij} + (1 - y_{ij}) \log(1 - p_{ij})] \tag{2}$$

where y_{ij} is the binary GT of whether the two compared face images i and j belong to the same identity. p_{ij} is the logits value.

Different from [40], contrastive loss [41] was proposed to directly compare the features of two face images in a metric learning way. If two images belong to a same ID, their features in the trained space should be closer to each other, and vice verses. As a result, the contrastive loss is:

$$L_{i,j} = \left\{ \begin{array}{ll} \frac{1}{2}\|f_i - f_j\|_2^2, & if\ y_{ij} = 1 \\ \frac{1}{2}max(0, m - \|f_i - f_j\|_2^2), & if\ y_{ij} = -1 \end{array} \right. \tag{3}$$

where i, j are two samples of a training pair, and f_i and f_j are their features. $\|f_i - f_j\|_2^2$ is their Euclidean distance. m is a margin for enlarging the distance of sample pairs with different IDs (negative pairs). Further more, Euclidean distance can be replaced by cosine distance, and contrastive loss become the following form:

$$L_{i,j} = \frac{1}{2}(y_{ij} - \sigma(wd + b))^2 \quad , \quad d = \frac{f_i f_j}{\|f_i\|_2 \|f_j\|_2} \tag{4}$$

where d is the cosine similarity between feature f_i and f_j, w and b are learnable scaling and shifting parameters, σ is the sigmoid function.

Different from contrastive loss, BioMetricNet [42] did not impose any specific metric on facial features. Instead, it shaped the decision space by learning a latent representation in which matching (positive) and non-matching (negative) pairs are mapped onto clearly separated and well-behaved target distributions. BioMetricNet first extracted face features of matching and non-matching pairs, and mapped them into a new space in which a decision is made. Its loss used to measure the statistics distribution of both matching and non-matching pairs in a complicated form.

FaceNet [34] proposed the triplet loss to minimize the distance between an anchor and a positive, both of which have the same identity, and maximize the distance between the anchor and a negative of a different identity. Triplet loss was motivated in [43] in the context of nearest-neighbor classification. The insight of triplet loss, is to ensure that an image x^a (anchor) of a specific person is closer to all other images x^p (positive) of the same person than it is to any image x^n (negative) of any other person. Thus the loss is designed as:

$$L = \sum_{(x^a, x^p, x^n) \in T} \max(\|f(x^a) - f(x^p)\|_2^2 - \|f(x^a) - f(x^n)\|_2^2 + \alpha, 0) \tag{5}$$

where T is is the set of all possible triplets in the training set. $f(x)$ is the embedding of face x. α is a margin that is enforced between positive and negative pairs. In order to ensure fast convergence, it is crucial to select triplets that violate the triplet constraint of:

$$\|f(x^a) - f(x^p)\|_2^2 + \alpha < \|f(x^a) - f(x^n)\|_2^2 \tag{6}$$

This means that, given x^a , we want to select an x^p (hard positive) such that $\arg\max_{x^p} \|f(x^a) - f(x^p)\|_2^2$, and similarly x^n (hard negative) such that $\arg\min_{x^n} \|f(x^a) - f(x^n)\|_2^2$. As a result, the accuracy of the triplet loss model is highly sensitive to the training triplet sample selection.

Kang *et al.* [44] simplified both of contrastive and triplet loss and designed a new loss as:

$$\begin{aligned} L &= \lambda_1 L_t + \lambda_2 L_p + \lambda_3 L_{softmax} \\ L_t &= \sum_{(x^a, x^p, x^n) \in T} \max(0, 1 - \frac{\|f(x^a) - f(x_n)\|_2}{\|f(x^a) - f(x_p)\|_2 + m}) \\ L_p &= \sum_{(x^a, x^p, .) \in T} \|f(x^a) - f(x^p)\|_2^2 \end{aligned} \tag{7}$$

where L_t, L_p, $L_{softmax}$ are modified triplet ratio loss, pairwise loss and regular softmax loss, respectively. Factor m in L_t is a margin to enlarge the difference between positive pair and negative pair, which has similar use with α in (5).

Contrastive loss measures the difference of a image pair; triplet loss represented the relations in a triplet: anchor, positive and negative samples. Some researchers have developed more metric learning based FR losses by describing more samples.

In order to enhance the discrimination of the deeply learned features, the center loss [45] simultaneously learned a center for deep features of each class and penalized the distances between the deep features and their corresponding class centers. Center loss can be added on softmax or other FR loss to refine feature distribution while training. The form of center loss is:

$$L = L_{softmax} + \lambda L_c \quad , \quad L_c(i) = \frac{1}{2}\sum_{i=1}^{m} \|x_i - c_{y_i}\|_2^2 \tag{8}$$

where $L_{softmax}$ is softmax loss on labelled training data. i is a image sample with its face label y_i, and x_i is its deep feature. m is the number of training classes. c_{y_i} denotes the y_i-th class center of deep features. The formulation of center loss effectively characterizes the intra-class variations.

4.1.2 Larger margin loss

Face recognition is naturally treated as a classification problem, and thus using softmax as the loss to train a FR model is an intuitive consideration. Larger margin loss (also is called as angular margin based loss) is derived from softmax. Larger margin loss is a major direction in FR, since it has largely improved the performance of FR in academy and industry. The intuition is that facial images with same identities are expected to be closer in the representation space, while different identities expected to be far apart. As a result, larger margin losses encourage the intra-class compactness and penalize the similarity of different identities.

First, we give the formulation of softmax as follows:

$$L_i = -\log \frac{e^{W_{y_i} \cdot x_i + b_{y_i}}}{\sum_j e^{W_j \cdot x_j + b_j}} \tag{9}$$

where W is the weighting vector of the last fully connected layer, and W_j represents the weight of class j ; x_i and y_i are the face embedding of sample i and its ground truth ID. For class y_i, sample i is a positive sample; and for class $j(j \neq y_i)$, sample i is a negative sample. These two definitions will be used in this whole paper. Considering positive and negative samples, softmax can be rewritten as:

$$L_i = -\log \frac{e^{W_{y_i} \cdot x_i + b_{y_i}}}{e^{W_{y_i} \cdot x_i + b_{y_i}} + \sum_{j \neq y_i} e^{W_j \cdot x_j + b_j}} \tag{10}$$

L-Softmax [46] first designed margin based loss by measuring features angles. First, it omits the bias of each class b_j, and changes the inner product of features and weights $W_j \cdot x_i$ to a new form $\|W_j\| \cdot \|x_i\| \cdot \cos(\theta_j)$, where θ_j is the angle between x_i and the weight W_j of class j. In order to enlarge the margin of angels between each class, L-Softmax modifies $\cos(\theta_{y_i})$ to $\psi(\theta_{y_i})$ by narrowing down the space between decision boundary and the class centers. In specific, it has:

$$L_i = -\log \frac{e^{\left\|W_{y_i}\right\| \|x_i\| \psi(\theta_{y_i})}}{e^{\left\|W_{y_i}\right\| \|x_i\| \psi(\theta_{y_i})} + \sum_{j \neq y_i} e^{\left\|W_j\right\| \|x_i\| \cos(\theta_j)}} \tag{11}$$

$$\psi(\theta) = (-1)^k \cos(m\theta) - 2k \tag{12}$$

where m is a fix parameter which is an integer; the angle $\theta \in [0, \pi]$ has been divided in m intervals: $[\frac{k\pi}{m}, \frac{(k+1)\pi}{m}]$; k is an integer and $k \in [0, m-1]$. Therefore, it has $\psi(\theta) \leq \cos(\theta)$.

Similar with L-Softmax, Sphereface [47] proposed A-Softmax, which normalized each class weight W_j before calculating the loss. Therefore, the loss became:

$$L_i = -\log \frac{e^{\|x_i\| \psi(\theta_{y_i})}}{e^{\|x_i\| \psi(\theta_{y_i})} + \sum_{j \neq y_i} e^{\|x_i\| \cos(\theta_j)}} \tag{13}$$

In the training step of Sphereface, its actual loss is (1-α)*softmax + α*A-softmax, where α is a parameter in [0,1]. During training, α is changing gradually from 0 to 1. This design is based on two motivations. Firstly, directly training A-Softmax loss will lead to hard convergence, since it brutally pushes features from different ID apart. Secondly, training a softmax loss first will decrease the angle θ_{y_i} between feature i and its related weight W_{y_i}, which causes the part $\cos(m\theta)$ in A-softmax loss in a monotonous area. And it is easier to get a lower loss while gradient descending.

Besides the weight of each class, NormFace [48] proposed that face embeddings need to be normalized as well. Also, NormFace scales up the normalized face embeddings, which will alleviate the data unbalancing problem of positive and negative samples for better convergence. However, NormFace doesn't use extra margin for positive samples as Sphereface. Therefore, the NormFace loss is shown as:

$$L_i = -\log \frac{e^{s\tilde{W}_{y_i}\tilde{x}_i}}{\sum_j e^{s\tilde{W}_j\tilde{x}_j}} = -\log \frac{e^{s\cos(\theta_{y_i})}}{\sum_j e^{s\cos(\theta_j)}} \tag{14}$$

where $\tilde{W}$ and $\tilde{x}$ are the normalized class weights and face embeddings; s is a scaling up parameter and $s > 1$. To this end, both face embeddings and class weights are distributed on the hypersphere manifold, due to the normalization.

AM-Softmax [49] and CosFace [50] enlarged softmax based loss by minus a explicit margin m outside $\cos(\theta)$ as follows:

$$L_i = -\log \frac{e^{s(\cos(\theta_{y_i}) - m)}}{e^{s(\cos(\theta_{y_i}) - m)} + \sum_{j \neq y_i} e^{s\cos(\theta_j)}} \tag{15}$$

ArcFace [51] putted the angular margin m inside $\cos(\theta)$, and made the margin more explainable. The ArcFace loss is shown as:

$$L_i = -\log \frac{e^{s(cos(\theta_{y_i} + m))}}{e^{s(\cos(\theta_{y_i} + m))} + \sum_{j \neq y_i} e^{s\cos(\theta_j)}} \tag{16}$$

The aforementioned angular margin based losses (such as CosFace and ArcFace) includes sensitive hyper-parameters which can make training process unstable. P2SGrad [52] was proposed to address this challenge by directly designing the gradients for training in an adaptive manner.

AM-Softmax [49], CosFace [50] and ArcFace [51] only added angular margins of positive samples. Therefore, their decision boundaries only enforced positive samples getting closer to its centers. On the contrary, SV-Softmax [53] got the intra-class compactness by pushing the hard negative samples away from positive class centers, which was chosen by hard example mining. In SV-Softmax, a binary label I_j adaptively indicates whether a sample j is hard negative or not, which was defined as:

$$I_j = \begin{cases} 0, & \cos(\theta_{y_i}) - \cos(\theta_j) \geq 0 \\ 1, & \cos(\theta_{y_i}) - \cos(\theta_j) < 0 \end{cases} \tag{17}$$

Therefore, the loss of SV-Softmax is:

$$L_i = -\log \frac{e^{s(\cos(\theta_{y_i}))}}{e^{s(\cos(\theta_{y_i}))} + h(t, \theta_j, I_j) \sum_{j \neq y_i} e^{s\cos(\theta_j)}} \tag{18}$$

$$h(t, \theta_j, I_j) = e^{s(t-1)(\cos(\theta_j)+1)I_k} \tag{19}$$

One step forward, SV-softmax evolve to SV-X-Softmax by adding large margin loss on positive samples:

$$L_i = -\log \frac{e^{sf(m,\theta_{y_i})}}{e^{sf(m,\theta_{y_i})} + \sum_{j \neq y_i} h(t, \theta_j, I_j) e^{s\cos(\theta_j)}} \tag{20}$$

where $f(m, \theta_{y_i})$ can be any type of large margin loss above, such as CosFace or ArcFace. At the same time, I_j is changed to:

$$I_j = \begin{cases} 0, & f(m, \theta_{y_i}) - \cos(\theta_j) \geq 0 \\ 1, & f(m, \theta_{y_i}) - \cos(\theta_j) < 0 \end{cases} \tag{21}$$

SV-Softmax has also evolved to MV-Softmax [54] by re-defining $h(t, \theta_j, I_j)$.

In aforementioned methods such as CosFace [50] and ArcFace [51], features have been normalized in the loss. Ring loss [55] explicitly measured the length of features and forced them to a same length R, which is shown as:

$$L_R = \frac{\lambda}{2m} \sum_{i=1}^{m} (\|f(x_i)\|_2 - R)^2 \tag{22}$$

where $f(x_i)$ is the feature of the sample x_i. R is the target norm value which is also learned and m is the batch-size. λ is a weight for trade-off between the primary loss function. In [55], the primary loss function was set to softmax and SphereFace [47].

Considering a training set with noise, Hu *et al.* [56] concluded that the smaller angle θ between a sample and its related class center, the greater probability that this sample is clean. Based on this fact, a paradigm of noisy learning is proposed, which calculates the weights of samples according to θ, where samples with less noise will be distributed higher weights during training. In a training set with noisy samples, the distribution of θ normally consists of one or two Gaussian distributions. These two Gaussian distributions correspond to noisy and clean samples. For the entire distribution, left and right ends of θ distribution are defined as δ_l and δ_r. And the peaks of the two Gaussian distributions are μ_l and μ_r ($\mu_l = \mu_r$ if the distribution consists of only one Gaussian). During training, θ tends to decreasing, so δ_r can indicate the progress of the training. In the initial training stage, the θ distribution of clean and noisy samples will be mixed together. At this stage, the weights of all samples are set to the same, that is, $w_{1,i} = 1$. As the training progressing, θ of the clean sample is gradually smaller than the noisy sample. At this time, clean sample should be given a higher weight, that is, $w_{2_i} = \frac{softmax(\lambda z)}{softmax(\lambda)}$, where $z = \frac{\cos\theta - \mu_l}{\delta_r - \mu_l}$, $softmax(x) = \log(1 + e^x)$, λ is the regularization coefficient. At later training stage, the weight of semi-hard samples should be higher than that of easy samples and hard samples, that is, $w_{3,i} = e^{-(\cos\theta - \mu_r)^2/2\sigma^2}$.

In conclusion, the final loss based on AM-Softmax is

$$L_i = -w_i \log \frac{e^{s(\cos(\theta_{y_i})-m)}}{e^{s(\cos(\theta_{y_i})-m)} + \sum_{j \neq y_i} e^{s\cos(\theta_j)}} \tag{23}$$

$$w_i = \alpha(\delta_r) w_{1,i} + \beta(\delta_r) w_{2,i} + \gamma(\delta_r) w_{3,i} \tag{24}$$

where $\alpha(\delta_r)$, $\beta(\delta_r)$ and $\gamma(\delta_r)$ are parameters to reflect the training stage, which are designed empirically.

Similar with [56], the sub-center ArcFace [57] also solved the problem of noisy sample learning on the basis of ArcFace. The proposed sub-centers encourage one dominant sub-class that contains the majority of clean faces and non-dominant sub-classes that include hard or noisy faces aggregate, and thus they relax the intra-class constraint of ArcFace to improve the robustness to label noise.

In this method, Deng *et al.* modified the fully connected layer classifier with dimension $d \times n$ to a sub-classes classifier with dimension $d \times n \times K$, where d is the dimension of embedding, n is the number of identities in the training set, and K is the number of sub-centers for each identity. The sub-center ArcFace loss is shown as

$$L_i = -\log \frac{e^{s(\cos(\theta_{i,y_i}+m))}}{e^{s(\cos(\theta_{i,y_i}+m))} + \sum_{j\neq y_i} e^{s\cos(\theta_{i,j})}} \tag{25}$$

$$\theta_{i,j} = \arccos \max_k (W_{j_k}^T x_i), k = 1, \cdots, K \tag{26}$$

where $\theta_{i,j}$ is the smallest angular between embedding x_i and all sub-class weights in class j.

CurricularFace [58] adopted the idea of curriculum learning and weighted the class score of negative samples. So in the earlier stage of training CurricularFace, the loss fits easy samples. And at the later training stage, the loss fits hard samples. In specific, CurricularFace loss is :

$$L_i = -\log \frac{e^{sf(m,\theta_{y_i})}}{e^{sf(m,\theta_{y_i})} + \sum_{j\neq y_i} e^{sN(t,m,\theta_j,\theta_{y_i})}} \tag{27}$$

and

$$N(t, m, \theta_j, \theta_{y_i}) = \begin{cases} \cos\theta_j, & \cos(\theta_{y_i} + m) - \cos(\theta_j) \geq 0 \\ \cos\theta_j(t + \cos\theta_j), & \cos(\theta_{y_i} + m) - \cos(\theta_j) < 0 \end{cases} \tag{28}$$

Different from SV-Softmax, t in $N(t, \theta_j)$ from CurricularFace is dynamically updated by Exponential Moving Average (EMA), which is shown as follows:

$$t^{(k)} = \alpha r^{(k)} + (1 - \alpha) r^{(k-1)} \tag{29}$$

where $t_0 = 0$, α is the momentum parameter and set to 0.99.

NPCface [59] observed that high possibility of co-occurrence of hard positive and hard negative appeared in large-scale dataset. It means that if one sample with ground truth class i is hard positive, it has a larger chance to be a hard negative sample of class j. Therefore, NPCface emphasizes the training on both negative and positive hard cases via the collaborative margin mechanism in the softmax logits, which is shown as:

$$L_i = -\log \frac{e^{s\cos(\theta_{y_i}+\tilde{m_i})}}{e^{s\cos(\theta_{y_i}+\tilde{m_i})} + \sum_{j\neq y_i} e^{s\cos(\theta_j+\tilde{m_j})}} \tag{30}$$

$$\tilde{m_i} = \begin{cases} m_0 + \frac{\sum_{j\neq y_i}(M_{i,j}\cos(\theta_j))}{\sum_{j\neq y_i} M_{i,j}} m_1, & \text{if } \sum_{j\neq y_i} M_{i,j} \neq 0 \\ m_0, & \text{if } \sum_{j\neq y_i} M_{i,j} = 0 \end{cases} \tag{31}$$

where $M_{i,j}$ is a binary indicator function, which represents whether sample i is a hard negative sample of class j.

UniformFace [60] proposed that above large margin losses did not consider the distribution of all training classes. With the prior that faces lie on a hypersphere manifold, UniformFace imposed an equidistributed constraint by uniformly spreading the class centers on the manifold, so that the minimum distance between class centers can be maximized through completely exploitation of the feature space. Therefore, the loss is :

$$L = L_{lml} + L_u \quad , \quad L_u = \frac{\lambda}{M(M-1)} \sum_{j_1=1}^{M} \sum_{j_2\neq j_1} \frac{1}{d(\boldsymbol{c}_{j_1}, \boldsymbol{c}_{j_2})} \tag{32}$$

where L_{lml} can be any large margin loss above, such as CosFace or ArcFace; and L_u is a uniform loss, which enforces the distribution of all training classes more uniform. M is the number of classes in L_u, and $\boldsymbol{c}_i$ is the center of class i, λ is a weight. As the class centers c_j are continuously changing during the training, the entire training set is require to update c_j in each iteration, which is not applicable in practice. Therefore, UniformFace updated the centers on each mini-batch by a newly designed step as follows:

$$\Delta \boldsymbol{c}_j = \frac{\sum_{i=1}^{n} \delta(y_i = j) \cdot (\boldsymbol{c}_j - \boldsymbol{x}_j)}{1 + \sum_{i=1}^{n} \delta(y_i = j)} \tag{33}$$

where n is the number of samples in a mini-batch, x_j is a embedding in this batch, $\delta(.) = 1$ if the condition is true and $\delta(.) = 0$ otherwise.

Similar with [60], Zhao *et al.* [61] also considered the distribution of each face class in the feature space. They brought in the concept of 'inter-class separability' in large-margin based loss and put forward RegularFace.

The inter-class separability of the i-th identity class: Sep_i is defined as the largest cosine similarity between class i and other centers with different IDs, where:

$$Sep_i = \max_{j\neq i} \cos(\varphi_{i,j}) = \max_{j\neq i} \frac{W_i \cdot W_j}{\|W_i\| \cdot \|W_j\|} \tag{34}$$

where W_i is the i-th column of W which represents the weight vector for ID i. RegularFace pointed out that different ids should be uniformly distributed in the feature space, as a result, the mean and variance of Sep_i will be small. Therefore RegularFace jointly supervised the FR model with angular softmax loss and newly designed exclusive regularization. The overall loss function is:

$$L = L_s + \lambda \mathcal{L}_r(W) \quad , \quad \mathcal{L}_r(W) = \frac{1}{C}\sum_i \max_{j \neq i} \frac{W_i \cdot W_j}{\|W_i\| \cdot \|W_j\|} \tag{35}$$

where C is the number of classes. λ is the weight coefficient. L_s is the softmax-based classification loss function or other large margin loss [51, 50], and $\mathcal{L}_r(\theta, W)$ is an exclusive regularization, which provides extra inter-class separability and encourages uniform distribution of each class.

Above large margin loss based classification framework only consider the distance representation between samples and class centers (prototypes). VPL [62] proposed a sample-to-sample distance representation and embedded it into the original sample-to-prototype classification framework. In VPL, both sample-to-sample and sample-to-prototype distance is replaced by the distance between sample to its variational prototype. In specific, the loss of VPL is:

$$L_i = -\log \frac{e^{s\cos(\tilde{\theta}_{y_i}+m)}}{e^{s\cos(\tilde{\theta}_{y_i}+m)} + \sum_{j\neq y_i} e^{s\cos(\tilde{\theta}_{y_i})}} \tag{36}$$

where $\tilde{\theta}_j$ is the angle between the feature x_i and the variational prototype $\tilde{W}_j$. The definition of m and s are similar with ArcFace [51]: m is the additive angular margin set as 0.5, and s is the feature scale parameter set as 64. Variational prototype $\tilde{W}_j$ is iteratively updated in training step as:

$$\tilde{W}_j = \lambda_1 W_j + \lambda_2 M_j \tag{37}$$

where W_j is the prototype of class j, namely the weight of class j. M_j is the feature centers of all embeddings in the mini batch which are belongs to class j. λ_1 and λ_2 are their weights.

UIR [63] trained the face feature extractor in semi-supervised way and modified original softmax loss by adding more unlabelled data. UIR design a loss λL_{uir} to measure the penalty on unlabelled training data, which dose not belong to any ID of labelled data. UIR assumed that, inference the unlabelled data on a well-trained face classifier, their logits scores of each class should be as equal as possible. Therefore, the assumption can be abstracted as the following optimization problem:

$$\max p_1 \cdot p_2 \cdot \ldots \cdot p_n, \; s.t. \sum_1^n p_i = 1 \tag{38}$$

where n is the number of IDs in the training set. As a result, changing this optimization in a loss way, we have the UIR loss follows:

$$L_i = L_{softmax} + \lambda L_{uir} \quad , \quad L_{uir} = -\sum_{i=1}^{n} \log(p_i) \tag{39}$$

where $L_{softmax}$ is softmax loss on labelled training data, and it can be replaced by other large margin loss.

AdaCos [64] took CosFace [50] as a benchmark and proposed new method to automatically modify the value of margin and scale hyper-parameter: m and s as the training iteration goes.

AdaCos observed that when s is too small, the positive probability of one sample i after margin based loss is also relatively small, even if the angular distance $\theta(i, y_i)$ between i and its class center is small enough. On the contrary, when s is too large, the positive probability of sample i is reaching 1, even if the angular distance $\theta(i, y_i)$ is still large.

As a result, changing the value of s during for a proper value will be good at FR model convergence. The loss of AdaCos is similar with CosFace. But its scale parameter s is changing as follows:

$$s^{(t)} = \begin{cases} \sqrt{2}\log(N-1), & t = 0 \\ \frac{\log B_{avg}^{(t)}}{\cos(\min(\pi/4, \theta_{med}^{(t)}))}, & t > 0 \end{cases} \tag{40}$$

where N is the number of training IDs; t is the iteration times. $\theta_{med}^{(t)}$ is the median of all corresponding classes' angles, $\theta_{i,y_i}^{(t)}$ from the mini-batch at the t-th iteration. $B_{avg}^{(t)}$ is the average of $B_i^{(t)}$, which is:

$$B_{avg}^{(t)} = \frac{1}{\|N^{(t)}\|} \sum_{i \in N^{(t)}} B_i^{(t)} \quad , \quad B_i^{(t)} = \sum_{k \neq y_i} e^{s^{(t-1)}\cos(\theta_{i,k})} \tag{41}$$

where $B_i^{(t)}$ is the total loss of all negative samples related to ID i. $N^{(t)}$ is the set of IDs in the mini-batch at t-th iteration, and $\|N^{(t)}\|$ is its number. The analysis of $s^{(t)}$ in detail can be check in the article of AdaCos [64], which will not be elaborated here.

Similar with AdaCos [64], Fair loss [65] also chose to adaptively modify the value of m in each iteration, however it is controlled by reinforcement learning. In detail, Fair loss is shown as follows:

$$L_i = -\log \frac{P_{y_i}^*(m_i(t), x_i)}{P_{y_i}^*(m_i(t), x_i) + \sum_{j\neq y_i} P_j(x_i)} \tag{42}$$

where $P_j(x_i) = e^{s\cos((\theta_j)}$ is the same with (16) and (15). The specific formulation of $P^*_{y_i}(m_i(t), x_i)$ can be different according to different large margin loss. Based on CosFace [50] or ArcFace [51], P^* can be formulated as follows:

$$P^*_{y_i}(m_i(t), x_i) = e^{s(\cos(\theta_{y_i}) - m_i(t))} \quad , or \quad P^*_{y_i}(m_i(t), x_i) = e^{s(\cos(\theta_{y_i} + m_i(t)))} \tag{43}$$

Then the problem of finding an appropriate margin adaptive strategy is formulated as a Markov Decision Process (MDP), described by (S, A, T, R) as the states, actions, transitions and rewards. An agent is trained to adjust the margin in every state based on enforcement learning.

Representative large margin methods such as CosFace [50] and ArcFace have an implicit assumption that all the classes possess sufficient samples to describe its distribution, so that a manually set margin is enough to equally squeeze each intra-class variations. Therefore, they set same values of margin m and scale s for all classes. In practice, data imbalance of different face IDs widely exists in

mainstream training datasets. For those IDs with rich samples and large intra-class variations, the space spanned by existing training samples can represent the real distribution.

But for the IDs with less samples, its features will be pushed to a smaller hyper space if we give it a same margin as ID with more samples. AdaptiveFace [66] proposed a modified AdaM-Softmax loss.

by adjusting the margins for different classes adaptively. AdaM-Softmax modified from CosFace is shown as follows:

$$L_{ada} = -\log \frac{e^{s(\cos(\theta_{y_i} + m_{y_i}))}}{e^{s(\cos(\theta_{y_i} + m_{y_i}))} + \sum_{j \neq y_i} e^{s\cos(\theta_j)}} \tag{44}$$

where the m_{y_i} is the margin corresponding to class y_i and it is learnable. Intuitively, a larger m is preferred to reduce the intra-class variations. Therefore, the final loss function of AdaptiveFace is:

$$L = L_{ada} + \lambda(-\frac{1}{N}\sum_i m_i) \tag{45}$$

where N is the number of IDs in the training set; and λ is positive and controls the strength of the margin constraint. As the training goes by, AdaM-Softmax can adaptively allocate large margins to poor classes and allocate small margins to rich classes.

Huang *et al.* [67] thought the aforementioned large margin loss is usually fail on hard samples. As a result, they adopted ArcFace to construct a teacher distribution from easy samples and a student distribution from hard samples. Then a Distribution Distillation Loss (DDL) is proposed to constrain the student distribution to approximate the teacher distribution. DDL can lead to smaller overlap between the positive and negative pairs in the student distribution, which is similar with histogram loss [68]. Ustinova *et al.* [68] first obtained the histogram of similarity set of positive pairs $\mathcal{S}^+$ and that of negative pairs $\mathcal{S}^-$ in a batch as their distribution. Then histogram loss is formulated by calculating the probability that the similarity of $\mathcal{S}^-$ is greater than the one of $\mathcal{S}^+$ through a discretized integral. In DDL, Huang *et al.* divided the training dataset into hard samples $\mathcal{H}$ and easy samples $\mathcal{E}$, and constructed positive and negative pairs in sets $\mathcal{H}$ and $\mathcal{E}$, respectively. Then calculated the distribution of positive pairs similarity H_r^+ and the distribution of negative pairs similarity H_r^-, and calculated the KL divergence of $\mathcal{E}$ (as teacher) and $\mathcal{H}$ (as student) on the positive/negative pairs similarity distribution as loss, namely

$$\begin{aligned} \mathcal{L}_{KL} &= \lambda_1 \mathbb{D}_{KL}(P^+ || Q^+) + \lambda_2 \mathbb{D}_{KL}(P^- || Q^-) \\ &= \lambda_1 \sum_s P^+(s) \log \frac{P^+(s)}{Q^+(s)} + \lambda_2 \sum_s P^-(s) \log \frac{P^-(s)}{Q^-(s)} \end{aligned} \tag{46}$$

where λ_1 and λ_2 are the weight coefficients. P^+ and P^- are the similarity distributions of positive and negative pairs in $\mathcal{E}$. Q^+ and Q^- are the similarity distributions of positive and negative pairs in $\mathcal{H}$. $\mathcal{L}_{KL}$ may make the distribution of teachers close to the distribution of students, so the author proposes order loss to add a constraint, namely

$$\mathcal{L}_{order} = -\lambda_3 \sum_{(i,j)\in(p,q)} (\mathbb{E}[\mathcal{S}_i^+] - \mathbb{E}[\mathcal{S}_j^-]) \tag{47}$$

where $\mathcal{S}_p^+$ and $\mathcal{S}_p^-$ represent the positive pairs and negative pairs of the teacher, respectively, and $\mathcal{S}_q^+$ and $\mathcal{S}_q^-$ represent the student's positive pairs and negative pairs. The final form of DDL is the sum of $\mathcal{L}_{KL}$, $\mathcal{L}_{order}$ and $L_{ArcFace}$.

In equation (14), (15), (16), (18), (20), (27) and (30), all face embeddings x and class weights W are normalized and thus distributed on the hypersphere manifold. And Ring loss [55] explicitly set a constrain where the length of embeddings should be as same as possible. However, many papers have demonstrated that the magnitude of face embedding can measure the quality of the given face. It can be proven that the magnitude of the feature embedding monotonically increases if the subject is more likely to be recognized. As a result, MagFace [69] introduced an adaptive mechanism to learn a well-structured within-class feature distributions by pulling easy samples to class centers with larger magnitudes while pushing hard samples away and shirking their magnitudes. In specific, the loss of MagFace is:

$$L_i = -\log \frac{e^{s\cos(\theta_{y_i} + m(a_i))}}{e^{s\cos(\theta_{y_i} + m(a_i))} + \sum_{j \neq y_i} e^{s\cos(\theta_j)}} + \lambda_g g(a_i) \tag{48}$$

where a_i is the magnitude of face feature of sample i without normalization. $m(a_i)$ represents a magnitude-aware angular margin of positive sample i, which is monotonically increasing. $g(a_i)$ is a regularizer and designed as a monotonically decreasing convex function. $m(a_i)$ and $g(a_i)$ simultaneously enforce direction and magnitude of face embedding, and λ_g is a parameter balancing these two factors. In detail, magnitude a_i of a high quality face image is large, and $m(a_i)$ enforces the embedding x_i closer to class center W_i by giving a larger margin; and $g(a_i)$ gives a smaller penalty if a_i is larger.

Circle loss [70] analyzed that, the traditional loss functions (such as triplet and softmax loss) are all optimizing $(s_n - s_p)$ distance, where s_n is inter-class similarity and s_p is intra-class similarity. This symmetrical optimization has two problems: inflexible optimization and fuzzy convergence state. Based on this two facts, Sun *et al.* proposed Circle loss, where greater penalties are given to similarity scores that are far from the optimal results.

Similar with MagFace, Kim *et al.* [71] also emphasize misclassified samples should be adjusted according to their image quality. They proposed AdaFace to adaptively control gradient changing during back propagation. Kim *et al.* assume that hard samples should be emphasized when the image quality is high, and vice versa. As a result, AdaFace is designed as:

$$L_i = -\log \frac{e^{s\cos(\theta_{y_i}+g_{\text{angle}})-g_{\text{add}}}}{e^{s\cos(\theta_{y_i}+g_{\text{angle}})-g_{\text{add}}} + \sum_{j\neq y_i} e^{s\cos(\theta_j)}} \tag{49}$$

$$g_{\text{angle}} = -m\cdot\widehat{\|z_i\|},\quad g_{\text{add}} = m\cdot\widehat{\|z_i\|} + m,\quad \widehat{\|z_i\|} = \lfloor \frac{\|z_i\| - \mu_z}{\sigma_z/h} \rceil_{-1}^{1} \tag{50}$$

where $\|z_i\|$ measures the quality of face i, and $\widehat{\|z_i\|}$ is normalized quality by using batch statistics μ_z and σ_z with a factor h. Similar with Arcface and CosFace, m represents the angular margin. AdaFace can be treated as the generalization of Arcface and CosFace: when $\widehat{\|z_i\|} = -1$, function (49) becomes ArcFace; when $\widehat{\|z_i\|} = 0$, it becomes CosFace.

At the end of this subsection, we introduce a FR model quantization method, which was specially designed for large margin based loss. Traditionally, the quantization error for feature $\boldsymbol{f}_i$ is defined as follows:

$$\text{QE}(\boldsymbol{f}_i) = \frac{1}{d}\sum_{l=1}^{d}(\boldsymbol{f}_i^l - Q(\boldsymbol{f}_i^l))^2 \tag{51}$$

where $\boldsymbol{f}_i$ and $Q(\boldsymbol{f}_i)$ denote a full precision (FP) feature and its quantization. d and the superscript l represent the length of features and the l-th dimension. Wu *et al.* [72] redefined the quantization error (QE) of face feature as the angle between its FP feature and its quantized feature :

$$\text{AQE}(\boldsymbol{f}_i) = \arccos\left(\left\langle \frac{\boldsymbol{f}_i}{\|\boldsymbol{f}_i\|_2}, \frac{\tilde{Q}(\boldsymbol{f}_i)}{\left\|\tilde{Q}(\boldsymbol{f}_i)\right\|_2}\right\rangle\right) \tag{52}$$

Wu *et al.* [72] believed that for each sample, quantization error AQE can be divided into two parts: error caused by the category center of the sample after quantization (class error), and error caused by the sample deviating from the category center due to quantization (individual error), namely

$$\text{AQE}(\boldsymbol{f}_i) = \text{AQE}(\boldsymbol{c}_{y_i}) + \mathcal{I}(\boldsymbol{f}_i) \tag{53}$$

The former term (class error of class y_i) will not affect the degree of compactness within the class, while the latter term (individual error) will. Therefore, the individual error should be mainly optimized. They introduced the individual error into CosFace as an additive angular margin, named rotation consistent margin (RCM):

$$\mathcal{L}_i = -\log \frac{\exp(s\cdot\cos(\theta_{i,j} + \delta(j=y_i)\cdot m + \delta(j=y_i)\cdot\lambda\theta_Q)}{\sum_j \exp(s\cdot\cos(\theta_{i,j} + \delta(j=y_i)\cdot m + \delta(j=y_i)\cdot\lambda\theta_Q)} \tag{54}$$

$$\theta_Q = \|\mathcal{I}(\boldsymbol{f}_i)\| = \|\text{AQE}(\boldsymbol{f}_i) - \text{AQE}(\boldsymbol{c}_{y_i})\| \tag{55}$$

where $\delta(j = y_i)$ is the indicative function, where $j = y_i$ gives its value 1, otherwise 0.

4.1.3 FR in unbalanced training data

Large-scale face datasets usually exhibit a massive number of classes with unbalanced distribution. Features with non-dominate IDs are compressed into a small area in the hypersphere, leading to training problems. Therefore, for different data unbalance phenomena, different methods were proposed.

The first data unbalance phenomenon is long tail distributed, which widely exists in the mainstream training set, such as MS-Celeb-1M. In MS-Celeb-1M dataset, the number of face images per person falls drastically, and only a small part of persons have large number of images. Zhang *et al.* [73] set an experiment to show that including all tail data in training can not help to obtained a better FR model by contrastive loss [41], triplet loss [34], and center loss [45]. Therefore, the loss needs to be delicately designed.

Inspired by contrastive loss, range loss [73] was designed to penalize intra-personal variations especially for infrequent extreme deviated value, while enlarge the inter-personal differences simultaneously. The range loss is shown as:

$$L = L_{softmax} + \alpha L_{R_{intra}} + \beta L_{R_{inter}} \tag{56}$$

where α and β are two weights, $L_{R_{intra}}$ denotes the intra-class loss and $L_{R_{inter}}$ represents the inter-class loss. $L_{R_{intra}}$ penalizes the maximum harmonic range within each class:

$$L_{R_{intra}} = \sum_{i \in I} L^i_{R_{intra}} = \sum_{i \in I} \frac{k}{\sum_{j=1}^{k} \frac{1}{D_j}} \tag{57}$$

where I denotes the complete set of identities in current mini-batch, and D_j is the j-th largest Euclidean distance between all samples with ID i in this mini-batch. Equivalently, the overall cost is the harmonic mean of the first k-largest ranges within each class, and k is set to 2 in the experiment. $L_{R_{inter}}$ represents the inter-class loss that

$$L_{R_{inter}} = \max(M - D_{center}, 0) = \max(M - \|\overline{x}_Q - \overline{x}_R\|_2^2, 0) \tag{58}$$

where D_{Center} is the shortest distance between the centers of two classes, and M is the max optimization margin of D_{Center}. Q and R are the two nearest classes within the current mini-batch, while $\overline{x}_Q$ and $\overline{x}_R$ represents their centers.

Zhong *et al.* [74] first adopted a noise resistance (NR) loss based on large margin loss to train on head data, which is shown as follows:

$$L_{NASB}(i) = -[\alpha(P_{y_{ip}}) \log(P_{y_i}) + \beta(P_{y_i}) \log(P_{y_{ip}})] \tag{59}$$

where y_i is the training label and y_{ip} is the current predict label. $P_{y_{ip}}$ is the predict probability of training label class and P_{y_i} is that of the current predict class. Namely:

$$y_{ip} = \arg\max_{y_i} \frac{e^{W_j^T x_i + b_j}}{\sum_k e^{W_k^T x_i + b_k}} \tag{60}$$

$$P_{y_i} = \frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_k e^{W_k^T x_i + b_k}} \;,\; P_{y_{ip}} = \frac{e^{W_{y_{ip}}^T x_i + b_{y_{ip}}}}{\sum_k e^{W_k^T x_i + b_k}} \tag{61}$$

$\alpha(P)$ and $\beta(P)$ control the degree of combination:

$$\alpha(P) = \begin{cases} \rho, & P > t \\ 0, & P \le 0 \end{cases} \;,\; \beta(P) = \begin{cases} 1-\rho, & P > t \\ 0, & P \le 0 \end{cases} \tag{62}$$

The NR loss (59) can be further modified to CosFace [50] or ArcFace [51] forms. After a relatively discriminative model have been learned on the head data by L_{NASB}, center-dispersed loss is employed to deal with the tail data. It extracts features of tail identities using the base model supervised by head data; then add the tail data gradually in an iterative way and disperse these identities in the feature space so that we can take full advantage of their modest but indispensable information. To be more specifically, Center-dispersed Loss can be formulated as:

$$L_{CD} = \min \frac{1}{m(m-1)} \sum_{1 \le i < j \le m} S_{i,j}^2 \;,\; S_{i,j} = (\frac{C_i}{\|C_i\|})^T (\frac{C_i}{\|C_i\|}) \tag{63}$$

where $S_{i,j}$ is the similarity between identity i and j in mini-batch, and the most hard m identities are mined from a candidate bag to construct a tail data mini-batch for efficiency. C_i and C_j represent normalized features centers of identity i and j, which can be relatively robust even to moderate noise.

The second data unbalance phenomenon is shallow data. In many real-world scenarios of FR, the training dataset is limited in depth, i.e. only small number of samples are available for most IDs. By applying softmax loss or its angular modification loss (such as CosFace [50]) on shallow training data, results are damaged by the model degeneration and over-fitting issues. Its essential reason consists in feature space collapse [75].

Li *et al.* [76] proposed a concept of virtual class to give the unlabeled data a virtual identity in mini-batch, and treated these virtual classes as negative classes. Since the unlabeled data is shallow such that it is hard to find samples from the same identity in a mini-batch, each unlabeled feature can be a substitute to represent the centroid of its virtual class. As a result, by adding a virtual class term into the large margin based loss (such as ArcFace [51]), the loss has:

$$L_i = -\log \frac{e^{s(cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))} + \sum_{j \ne y_i} e^{s\cos(\theta_j)} + \sum_{u=1}^{U} e^{s\cos(\theta_u)}} \tag{64}$$

where U is the number of unlabeled shallow data in mini-batch, and θ_u is the angular between embedding x_i and x_u, which is also the centroid of virtual class u. For the purpose of exploiting more potential of the unlabeled shallow data, a feature generator was designed to output more enhanced feature from unlabeled data. More details about the generator can be check in [76].