

# Large-scale Bisample Learning on ID vs. Spot Face Recognition

Xiangyu Zhu\* · Hao Liu\* · Zhen Lei · Hailin Shi · Fan Yang · Dong Yi · Guojun Qi · Stan Z. Li

Received: date / Accepted: date

**Abstract** In real-world face recognition applications, there is a tremendous amount of data with two images for each person. One is an ID photo for face enrollment, and the other is a probe photo captured on spot. Most existing methods are designed for training data with limited breadth (a relatively small number of classes) and sufficient depth (many samples for each class). They would meet great challenges on ID vs. Spot (IvS) data, including the under-represented intra-class variations and an excessive demand on computing devices. In this paper, we propose a deep learning based large-scale bisample learning (LBL) method for IvS face recognition. To tackle the bisample problem with only two samples for each class, a classification-verification-classification (CVC) training strategy is proposed to progressively enhance the IvS performance. Besides, a dominant prototype softmax (DP-softmax) is incorporated to make the deep learning scalable on large-scale classes. We conduct LBL on a IvS face dataset with more than two million identities. Experimental results show that the proposed method achieves superior performance to

Xiangyu Zhu, Hao Liu, Zhen Lei, Hailin Shi and Stan Z. Li  
Center for Biometrics and Security Research & National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences.

E-mail: {xiangyu.zhu,hao.liu2016,zlei,hailin.shi,szli}@nlpr.ia.ac.cn

Fan Yang  
College of Software, Beihang University.  
E-mail: fanyang@buaa.edu.cn

Dong Yi  
DAMO Academy, Alibaba Group.  
E-mail: yidong.yd@alibaba-inc.com

Guojun Qi  
HUAWEI Cloud, USA.  
E-mail: guojunq@gmail.com

\*denotes equal contribution



Fig. 1: The ID vs. Spot (IvS) data, each identity has one ID photo and one spot photo.

previous ones, validating the effectiveness of LBL on IvS face recognition.

**Keywords** Face Recognition · ID vs. Spot · Large-scale Bisample Learning · Dominant Prototype Softmax

## 1 Introduction

Face recognition has witnessed dramatic improvements in recent years, primarily due to the advances in network architectures [1, 2, 3, 4, 5], training strategies [6, 7, 8, 9, 10, 11] and a large amount of face data [12, 13, 14, 15]. Recent methods mainly focus on face recognition in the wild, where the training datasets are collected from internet by web searching engines [12] or electronic album applications [13]. Most of wild datasets like CASIA-Webface [12], Ms-Celeb-1M [14] and VGG2 [15] are well-posed, where they have a limited number of classes (less than 100,000) and adequate samples per class (more than 20). However, this is not the case in many real-world face data, like the ID vs. Spot (IvS)

face recognition, which aims to match unconstrained spot photos with constrained ID photos, see Fig. 1 for example. Compared with wild datasets, IvS datasets present threefold challenges below.

1. **Heterogeneity:** ID and spot photos are taken in different environments. The ID photos are taken in constrained environments with clean background, in frontal pose, normal illumination and neutral expression. The spot photos are taken in unconstrained environments. There are pose, lighting, expression and occlusion (e.g., glasses, haircut, scarf etc.) variations. Moreover, there may be a large age gap between ID and spot photos since ID photos are updated every 10 – 20 years. This heterogeneity increases the difficulty of IvS face recognition.
2. **Bisample Data:** Usually, IvS training data is collected by face authentication systems. When a user passes the authentication system, a pair of his photos will be recorded, one ID photo from his ID card and the other spot photo taken online. As a result, there are only two samples available for each subject. The intra-variations of classes are not well represented, making the discriminative training on bisample data a more challenging problem.
3. **Large-scale Classes:** IvS data is collected by practical systems, where there can be as many as million or even hundreds of million identities. How to perform deep learning on such massive classes with limited GPU devices is worth studying.

The above three characteristics pose great challenges for IvS face recognition. In real-world applications, the high recognition rate at low false acceptance rate is demanded. To this end, the large margins between inter-class samples and the compactness of intra-class samples in the feature space are necessary. However, since there are only two samples for each subject, it is difficult to describe the intra-variations in the training phase so that the derived feature space would not be discriminative enough. In addition, there is a huge number of classes. It is a great challenge to explore the discriminative information among these classes with limited GPU devices. Taking deep learning with softmax as an example, there need to be millions of prototypes in the GPU memory, which is infeasible for most of computing devices.

In this paper, we cast the deep learning on IvS data as a **Large-scale Bisample Learning (LBL)** problem, where the training data has a huge number of classes and each class has only one positive pair. To enhance existing training strategies to handle the LBL problem, two challenges must be resolved: The weak intra-variations caused by bisample data and the model training s-

calability caused by large-scale classes. To deal with weak intra-variations, we propose a progressive model transferring method, named Classification-Verification-Classification (CVC). We pre-train a model on web-collected data by classification and finetune it on IvS data by verification to get a good initialization. Then we perform large-scale classification to obtain the final IvS model.

To improve scalability for model training, we adopt a prototype selection strategy in the last stage of CVC to scale up softmax-like losses to any number of classes. Specifically, we observe that the gradients of softmax are dominated by a small fraction of classes and the dominant classes can be effectively identified by the class proximities. Based on this, we build a dominant queue for each class to record its similar classes, from which we can select the most dominant classes to participate in the classification. The new softmax can perform effective training with only 0.15% classes, significantly reducing the demand for computing devices.

We evaluate our method on a real-world IvS dataset and show it reaches the state-of-the-art performance with limited computing devices (4 TITANX GPU). Besides, we release a Public-IvS dataset of 1262 identities for open evaluation. Moreover, to make our work reproducible, we devise a new protocol Megaface-bisample to mimic the large-scale bisample learning task. To our knowledge, it is the first investigation into training deep neural networks on large-scale bisample face data.

## 2 Related Works

In this section, we review the deep learning based face recognition and discuss two related problems about the LBL task: (1) Learning with insufficient data and (2) Large-scale classification.

### 2.1 Deep Learning based Face Recognition

Recently there are two schemes to train deep models for face recognition: classification and verification. The classification scheme considers each identity as a unique category and classifies each sample into one of the classes. During testing, the classification layer is removed and the top-level feature is regarded as the face representation [16]. The most popular loss is softmax [16, 6, 17]. Based on that, the center loss [18] proposes to learn the class-specific feature centers to make features more compact in the embedding space. The L2-softmax [19] adds a L2-constraint on features to promote the under-represented classes. The normface [20] normalizes both features and prototypes to make the training and testing

phases closer. Recently, enhancing margins between different classes is found to be effective in improving feature discrimination, including large-margin softmax [21], A-softmax [22], GA-softmax [23] and AM-softmax [24]. Benefiting from the prototypes in the classification layer, the scheme can distinguish a sample from all the other classes, leading to fast convergence and good generalization ability [20].

On the other hand, the verification scheme optimizes distances between samples. Within a mini-batch, the contrastive loss [7] optimizes pairwise distances in the feature space to reduce intra-class distances and enlarge inter-class distances. The triplet loss [8] makes up a triplet consisting of an anchor, a positive sample and a negative sample. The loss aims to separate the positive pair from the negative pair by a distance margin. The lifted structured loss [25] considers all the pairwise distances within the mini-batch and select the best positives and negatives. The N-pairs loss [26] optimizes each positive pair against all the related negative pairs following a local softmax formulation. Besides, hard negative mining is widely adopted to remove the easy negative pairs to ensure fast convergence [8]. More recently, [27] presents a GAN-based method to deliberately generate hard triplet samples to improve the efficiency and effectiveness in training triplet losses. The performance of the verification scheme depends on the number of pairs generated in one mini-batch [25], which is determined by the batch size. However, increasing batch size, meaning that expanding GPU memory, is very expensive. To reduce the cost of GPU memory, smart sampling [28] selects valuable pairs in the data layer instead of the feature layer. The method memorizes the pairs having large losses and selects them with higher probabilities afterwards [28,9,29].

Most contemporary face recognition methods are based on wild datasets, e.g., CASIA-Webface [12], Ms-Celeb-1M [14], MF2 [13] and VGG2 [15]. These well-posed datasets have a limited number of identities and sufficient samples per identity. However, this is not the case in IvS datasets. Table 1 gives a brief comparison between wild and IvS datasets. Our CASIA-IvS has more than 2 million identities but only two samples per identity, on which existing well-studied methods cannot work well any more. Exploring IvS-specific training strategies is necessary.

## 2.2 Learning with Insufficient Data

**Low-shot learning** intends to recognize new classes by few samples [32]. Generally, low-shot learning transfers the knowledge from a well-posed source domain to the low-shot target domain. Siamese net [33] trains

a siamese CNN by same-or-different classification on the source domain and extracts the deep features for nearest neighbour matching in the target domain. MAN-N [34,35,36] memorizes the features of examples in the source domain to help predict the under-labeled classes. Model regression [37,38] directly transfers the neural network weights across domains. The L2-regularization on features [39,40,41] can prevent the network from ignoring low-shot classes. Besides, virtual sample generation [40,42] and semi-supervised samples [43] are found effective in promoting low-shot classes. Although both low-shot learning and bisample learning intend to learn a concept with insufficient samples, they differ in that low-shot learning is close-set classification where the testing samples belong to the low-shot classes, while bisample learning is open-set classification that the testing samples definitely belong to unseen classes.

**Long-tail problem** refers to the situation that only a limited number of classes appear frequently, while most of the others remain far less existing. Deep models trained on long-tailed data tend to ignore the classes in the tail. To resolve the problem, [44] retrieves more samples from the tail classes. [45] makes samples uniformly distributed by random sampling. [31] proposes a range loss to balance the rich and poor classes, where the largest intra-class distance is reduced and the shortest class-center distance is enlarged.

## 2.3 Large-scale Classification

Large-scale classification aims to perform classification on a vast number of classes, where the class number reaches millions or tens of millions. This task presents a great problem for deep learning: the common softmax loss can not be adopted due to the prohibitive parameter size and computation cost. The Megaface challenge [13] proposes four methods for training models on 670k identities. Model-A trains the network on random 20,000 identities via softmax. Model-B finetunes Model-A on all the 670k identities with the triplet loss. Model-C adopts rotating softmax that randomly selects 2,600 identities every 20 epoches. After each rotation the parameters in the softmax layer are randomly initialized. Model-D further triplet-finetunes Model-C on all the identities.

Beyond the computer vision, extreme multi-label learning [46] and noise contrastive estimation [47] are related to large-scale classification. **Extreme Multi-label Learning** learns a classifier to tag a sample with the most relevant label from a large label set [46]. It faces the same challenge as LBL that training a multi-class classifier is computationally prohibitive when the class number is extremely large. To tackle this problem, the tree based methods [48,49,50] learn a label hierarchy

Dataset	Identities	Samples/ID	Scenarios	Descriptions
CASIA-Webface [12]	10,575	46.7	wild	Celebrity photos by web-searching
Ms-Celeb-1M [14]	98,685/79,077	50.7/63.8	wild	Celebrity photos by web-searching
MF2 [13]	657,559/90,399	6.8/24.7	wild	User photos of electronic album
VGG2 [15]	9,131	362.6	wild	Celebrity photos by web-searching
CASIA-IvS	2,578,178	2	IvS	ID and spot photos of the masses

Table 1: Description of face recognition datasets. We clean Ms-Celeb-1M and MF2 due to their low purities [30], and cut the identities whose samples are smaller than 10 to balance the long tail distribution [31]. The numbers after / indicate the information after cleaning.

as follows: The root node contains the entire label set and a node partitioning formulation is optimized to determine which labels belong to the left child and which to the right. Nodes are recursively partitioned until each leaf contains a small number of labels. Finally a base classifier identifies the samples in only one leaf node. Although tree based methods reduce the class number for each classifier, the prediction error made at top-level cannot be corrected at lower levels due to its cascading architecture [51]. On the other hand, the embedding based methods [52, 53, 54] assume the label matrix [46], where each row is a {0, 1} label vector of a sample, is low rank and the label vectors can be projected onto a low-dimensional linear subspace. As a result, the extreme classification task can be converted to a low-dimensional regression problem. However, the low rank assumption indicates that the samples concentrate on a small number of active classes, which is not the case in IvS data where the samples are evenly distributed among identities.

In softmax, computing the probabilities requires normalizing over the entire class set, which is the major cost in large-scale classification. **Noise Contrastive Estimation (NCE)** [47] provides an approximate method to estimate the probabilistic distribution without the normalization constant. Its basic idea is training a logistic regression classifier to discriminate samples from data distribution and noise distribution, so that the density estimation is reduced to probabilistic binary classification. Although NCE has been successfully applied in language models [55, 56, 57], recent face recognition tasks [7, 16] have shown that promoting the contrast among classes is crucial in training discriminative models. Turning multi-class classification to binary logistic regression may lose inter-class information and get inferior performance.

### 3 Large-scale Bisample Learning

The proposed method contains a complete pipeline for deep learning on large-scale bisample data. We begin

by discussing of the classification and the verification schemes, showing how their pros and cons motivate the proposed methods. Then we present the way to train deep neural networks on bisample data. Finally we develop a dominant prototype softmax to perform 2-million-way classification in a scalable fasion. Fig. 2 shows an overview of our method.

#### 3.1 Problem Formulation and Motivation

Currently there are two schemes for training deep neural networks, i.e., verification and classification. The verification scheme optimizes sample-to-sample distances, such as the contrastive loss [7] and the triplet loss [8]. In each iteration, it performs local optimization within a mini-batch by making positive pairs close and negative pairs far away. Besides, the mining strategy [8] filters out easy pairs for fast convergence. On the other hand, the classification scheme regards each identity as a unique class and trains the network as a  $N$ -way classification problem, such as softmax [16] and A-softmax[22]. Compared with the verification scheme, the classification scheme performs global optimization by identifying each sample into one of the  $N$  classes.

In this paper, we motivate our method by comparing classification and verification. Interestingly, if we formulate the loss function for a whole mini-batch, we can unify the two schemes in a pair matching and weighting framework. First, the verification scheme extracts features with a neural network and makes pairs between deep features. Taking contrastive loss [7] as an example:

$$L_{ver}(\mathbf{X}) = - \sum_{j=1}^M \sum_{k=j+1}^M NM(\mathbf{x}_j^T \mathbf{x}_k, y_{jk}) \mathbf{x}_j^T \mathbf{x}_k, \quad \text{with} \quad (1)$$

$$NM(\mathbf{x}_j^T \mathbf{x}_k, y_{jk}) = \begin{cases} 1 & \text{if } y_{jk} = 1 \\ -1 & \text{if } y_{jk} = 0 \text{ and } \mathbf{x}_j^T \mathbf{x}_k \geq \tau \\ 0 & \text{if } y_{jk} = 0 \text{ and } \mathbf{x}_j^T \mathbf{x}_k < \tau \end{cases}$$

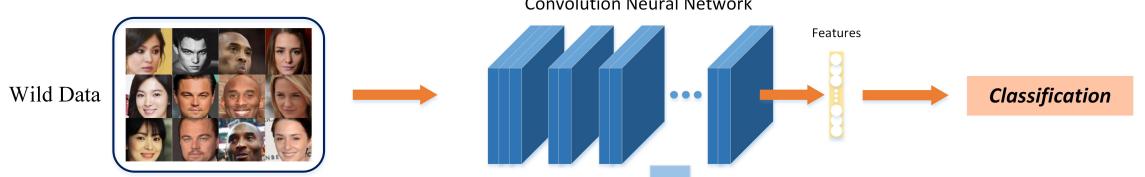
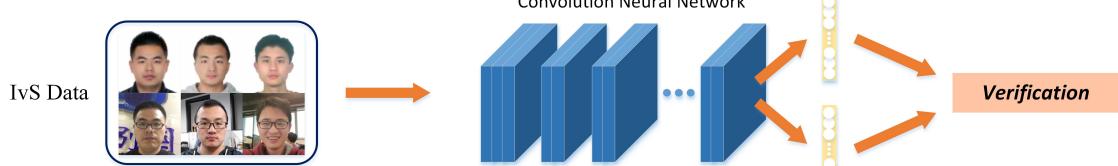
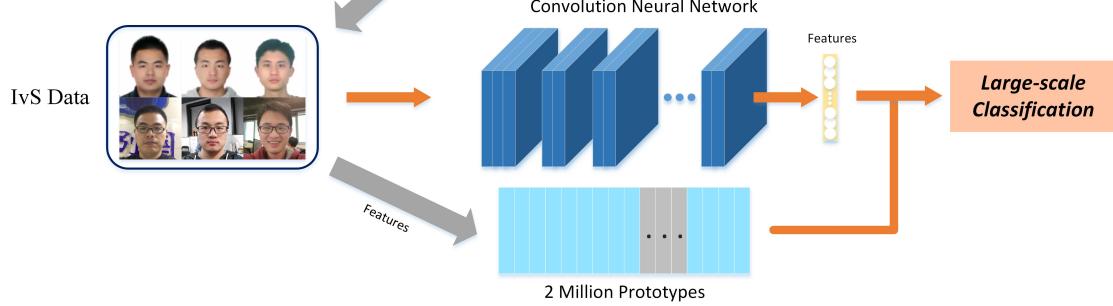
**Stage 1: Pre-learning****Stage 2: Transfer Learning****Stage 3: Fine-grained Learning**

Fig. 2: Overview of the large-scale bisample learning (LBL). LBL adopts a classification-verification-classification (CVC) training strategy, which has three stages: The first stage, pre-learning, is training the network from scratch on a wild dataset by a classification loss. The second stage, transfer learning, is finetuning the network on the IvS dataset with a verification loss. The last stage, fine-grained learning, is performing large-scale classification on the IvS dataset with a new dominant prototype softmax.

where  $\mathbf{x}$  is the  $D$  dimensional deep feature extracted by the neural network;  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_M]$  are the features in the mini-batch where  $M$  is the batch size;  $y_{jk} = 1$  if  $\mathbf{x}_j$  and  $\mathbf{x}_k$  belong to the same class and  $y_{jk} = 0$  if not;  $N(\cdot)$  is the hard negative mining that filters out easy negative pairs with a threshold  $\tau$ . We can see that the contrastive loss makes pairs within deep features  $\mathbf{X}$  and assigns  $\{0, 1\}$  weights to them.

In contrast, the classification scheme makes pairs between features and prototypes. Taking the softmax loss [7] as an example:

$$L_{cls}(\mathbf{W}, \mathbf{X}) = - \sum_{j=1}^M \log \left( \frac{e^{\mathbf{w}_{y^{(j)}}^T \mathbf{x}_j}}{\sum_{i=1}^N e^{\mathbf{w}_i^T \mathbf{x}_j}} \right), \quad (2)$$

where  $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_N]$  is the prototype matrix in the softmax layer where  $N$  is the number of classes and  $y^{(j)}$  is the label of  $\mathbf{x}_j$ . Its derivatives to a prototype  $\mathbf{w}_i$  and

a feature  $\mathbf{x}_j$  are:

$$\frac{\partial L_{cls}}{\partial \mathbf{x}_j} = - \sum_{i=1}^N (1\{y^{(j)} == i\} - p_{ij}) \mathbf{w}_i \quad (3)$$

$$\frac{\partial L_{cls}}{\partial \mathbf{w}_i} = - \sum_{j=1}^M (1\{y^{(j)} == i\} - p_{ij}) \mathbf{x}_j$$

$$\text{with } p_{ij} = \frac{e^{\mathbf{w}_i^T \mathbf{x}_j}}{\sum_{k=1}^N e^{\mathbf{w}_k^T \mathbf{x}_j}}, \quad (4)$$

where  $1(\cdot)$  is the indicator function which is 1 when the statement is true and 0 otherwise, and  $p_{ij}$  is the probability that  $\mathbf{x}_j$  belongs to the  $i$ th class. Given that network training only concerns the gradients back-propagated, we can construct a dummy softmax loss

sharing the same gradients with Equ. 2:

$$L_{dum}(\mathbf{W}, \mathbf{X}) = - \sum_{j=1}^M \sum_{i=1}^N P(\tilde{p}_{ij}, i, j) \mathbf{w}_i^T \mathbf{x}_j, \quad \text{with} \quad (5)$$

$$P(\tilde{p}_{ij}, i, j) = \begin{cases} 1 - \tilde{p}_{ij} & \text{if } y^{(j)} = i \\ -\tilde{p}_{ij} & \text{if } y^{(j)} \neq i \end{cases}$$

where  $\tilde{p}_{ij}$  is computed as  $p_{ij}$  in Equ. 4 and considered as a constant.  $L_{cls}$  and  $L_{dum}$  are equivalent in network training since they produce the same back-propagated signals. Obviously  $L_{dum}$  makes pairs between  $\mathbf{W}$  and  $\mathbf{X}$ , and assigns a weight to each pair ( $\mathbf{w}_i, \mathbf{x}_j$ ) by the probability  $p_{ij}$ . The negative pairs with higher probabilities and the positive pairs with lower probabilities have larger weights and yield louder signals during training.

Comparing Equ. 5 and Equ. 1, we can conclude that both classification and verification follow the same pair matching and weighting framework. The only differences lie in the pairing candidates (features with prototypes vs. within features) and the weighting methods (soft weight vs. hard weight). Recent works have empirically observed that increasing the number of pairs always delivers faster convergence and better discriminative power, hence the loss functions involving more pairs are preferred. Within a mini-batch with  $M$  as the batch size and  $N$  as the class number, a classification loss makes  $N \times M$  pairs in Equ. 5 and a verification loss makes  $M(M - 1)/2$  pairs in Equ. 1. In real implementation with limited GPU memory,  $N \gg M$  always holds. For example, when training ResNet64 [21] with a TITAN-X GPU, the batch size  $M$  is about 50 and the class number  $N$  easily reaches tens or even hundreds of thousands. With more orders of magnitude pairs, **the classification scheme is expected to acquire more discriminative features**, which has been shown in the state-of-the-art methods [21, 23, 24, 58]. However, two challenges make classification infeasible on IvS data. First, the classification scheme has difficulty to converge on bisample data due to the weak intra-variations, which is demonstrated in our experiments. Second, the classification scheme suffers from weak scalability to large-scale classes due to the limited GPU memory. Directly performing 2-million-way classification with two samples per class is infeasible for current optimization methods and computing devices.

In this paper, we motivate our method to make the classification scheme feasible on large-scale bisample data. To this end, its robustness to bisample data and scalability to large-scale classes should be enhanced. First, we find the classification scheme convergent on bisample data only if it is well initialized. So that we propose a CVC training strategy to initialize the model and construct the prototypes for the classification scheme.

Second, we propose a prototype selection strategy to scale up the classification scheme to any number of classes. With the improvements, we achieve superior performance to existing methods.

### 3.2 Bisample Learning

It has been observed that when training data is insufficient, transferring knowledge from related tasks is better than directly training on the target domain [33]. Inspired by this, we regard the well-posed wild data as the source domain and the IvS data as the target domain. A classification-verification-classification (CVC) training strategy is proposed to transfer the knowledge from wild scenarios to IvS scenarios and boost the performance by large-scale classification. As shown in Fig. 2, the CVC involves three stages:

- 1. Pre-learning (Classification):** We first train the deep model on a wild dataset to get a good initialization for general face recognition. With a limited number of classes (less than 100,000), we can adopt a classification loss like softmax [16] and A-softmax [22] to perform one-vs-all optimization. The trained model performs well in wild scenarios but terribly in IvS scenarios due to the large bias [59]. Nevertheless, the model has learned basic knowledge about human faces and will not be puzzled by IvS data.
- 2. Transfer Learning (Verification):** Since the verification scheme only concerns a small number of classes and just needs two samples per class to optimize intra-class distances in each iteration. We believe verification is robust to large-scale bisample data. In this stage, we adopt the verification scheme to transfer the face knowledge from wild scenarios to IvS scenarios. Specifically, we remove the classification layer and finetune the model on the IvS dataset with a verification loss like contrastive [7] or triplet [8]. Benefiting from the initialization from the previous stage and the robustness to bisample data of the verification scheme, we can successfully optimize the loss function and provide a good initialization for the final large-scale classification.
- 3. Fine-grained Learning (Classification):** We construct a classification layer on the top of the network and conduct classification with 2 million classes on the IvS dataset. A novel dominant prototype softmax is adopted to select a small number of dominant classes to participate into the classification in each iteration. The new softmax can effectively and efficiently perform large-scale classification and further

boost the performance, finally achieves satisfactory recognition accuracy in IvS scenarios.

The key in CVC is that the knowledge transferring should be smooth. We find after the first stage, the large-scale classification has been able to converge. However, the loss descends slowly and the optimization gets stuck into a bad local optima. Considering that the verification scheme has good robustness to data distribution, we bridge the two classification stages with a verification stage, which gives a better initialization for large-scale classification and finally achieves much better performance. Although classification followed by verification [60] and the joint identification-verification [7] have been applied in training web-face models, the two schemes are applied on the same dataset. While the first two stages of CVC are applied on different datasets with different scenarios, which acts as a knowledge-transferring role.

To perform classification in the final stage of CVC, we must construct the absent classification layer, which contains the prototype for each class. Considering prototypes serve as the class proxies, to which the deep features will be optimized, we construct the prototype of a class by the features belonging to it. Specifically, we try two kinds of prototypes: ID-prototype and avg-prototype. Suppose  $\mathbf{x}_i^{id}$  and  $\mathbf{x}_i^{spot}$  are the deep features of the ID and spot photos of the  $i$ th identity, we set the ID-prototype  $\mathbf{w}_i^{id} = \mathbf{x}_i^{id}$  and the avg-prototype  $\mathbf{w}_i^{avg} = (\mathbf{x}_i^{id} + \mathbf{x}_i^{spot})/2$ . Intuitively, the ID-prototype enforces the spot feature to approach the more reliable ID feature and the avg-prototype makes the two features approach their centroid. Our experiments show that which kind of prototype is better depends on the loss function.

In the next section, we will introduce how to perform large-scale classification in the final stage of CVC.

### 3.3 Large-scale Classification

#### 3.3.1 Random Prototype Softmax

With the well initialized network and prototypes, the only problem remained is to scale up the classification scheme to massive classes. If we directly perform classification on 2 million classes, the massive prototypes will take 1/3 GPU memory (4GB of the 12GB) and dramatically increase the training time due to their numerous parameters.

We aim to improve scalability by reducing the cost of large-scale classification. As shown in Fig. 3, we select a fraction of prototypes to participate in the classification in each iteration. In the pair-matching

formulation of softmax (Equ. 5), given one mini-batch  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_M]$  where samples have different labels, all the prototypes  $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_N]$  can be divided into  $M$  positive prototypes  $\mathbf{W}_{pos}$  and the rest negative prototypes  $\mathbf{W}_{neg}$ . Each prototype in  $\mathbf{W}_{pos}$  has a mate in  $\mathbf{X}$  to make up a positive pair, while the prototypes in  $\mathbf{W}_{neg}$  do not share class with any of  $\mathbf{X}$  and only make up negative pairs. Given that  $M \ll (N - M)$ , it is unnecessary to put the whole  $\mathbf{W}_{neg}$  into GPU memory since negative pairs are redundant. Based on this, we propose a naive solution called **Random Prototype Softmax (RP-softmax)**. The RP-softmax stores the full prototype matrix  $\mathbf{W}$  in the memory. In each iteration, it first constructs a temporary prototype matrix  $\mathbf{W}_{iter} = [\mathbf{W}_{pos}, \widehat{\mathbf{W}}_{neg}]$ , where  $\widehat{\mathbf{W}}_{neg}$  has  $N_{iter} - M$  randomly selected prototypes from  $\mathbf{W}_{neg}$  and  $N_{iter}$  is the number of selected prototypes. Then  $\mathbf{W}_{iter}$  is copied into GPU for training and updated to  $\mathbf{W}_{iter}^+$ . Finally,  $\mathbf{W}_{iter}^+$  and  $\mathbf{W}$  are synchronized by replacing the selected prototypes with the updated ones. Overall, the prototype selection and updating procedure is listed in Algorithm 1.

---

#### Algorithm 1: Random Prototype Softmax

---

```

Input : Prototype matrix:  $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_N]$   

         Feature matrix:  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_M]$   

         Number of selected classes  $N_{iter}$   

Output : Updated prototype matrix  

1 Initialize selected class set  $\mathcal{L} = \emptyset$   

2 for each feature  $\mathbf{x}_j$  in  $\mathbf{X}$  do  

3   | Get the label  $y^{(j)}$  of  $\mathbf{x}_j$   

4   |  $\mathcal{L}.insert(y^{(j)})$   

5 end  

6 while  $\mathcal{L}.size < N_{iter}$  do  

7   | randomly select a label  $y^{(r)}$   

8   |  $\mathcal{L}.insert(y^{(r)})$   

9 end  

10 for  $i = 1 \dots N_{iter}$  do  

11   |  $\mathbf{W}_{iter}[i, :] = \mathbf{W}[\mathcal{L}_i, :]$ ,  $\mathcal{L}_i$  is the member of  $\mathcal{L}$   

12 end  

13 Training with  $\mathbf{W}_{iter}$  and  $\mathbf{X}$ , getting updated  $\mathbf{W}_{iter}^+$   

14 for  $i = 1 \dots N_{iter}$  do  

15   |  $\mathbf{W}[\mathcal{L}_i, :] = \mathbf{W}_{iter}^+[i, :]$   

16 end
```

---

The hyper parameter  $N_{iter}$  plays a key role in RP-softmax. Larger  $N_{iter}$  brings more negative pairs and provides richer inter-variation information. However, increasing  $N_{iter}$  is not cost free. Besides the time-consuming large matrix multiplication, the softmax layer has to get blocked until  $\mathbf{W}_{iter}$  is copied into GPU. Sometimes the waiting time exceeds the forward propagation time. Moreover, increasing  $N_{iter}$  squeezes the batch size and degrades the data-driven layers like batch-normalization.

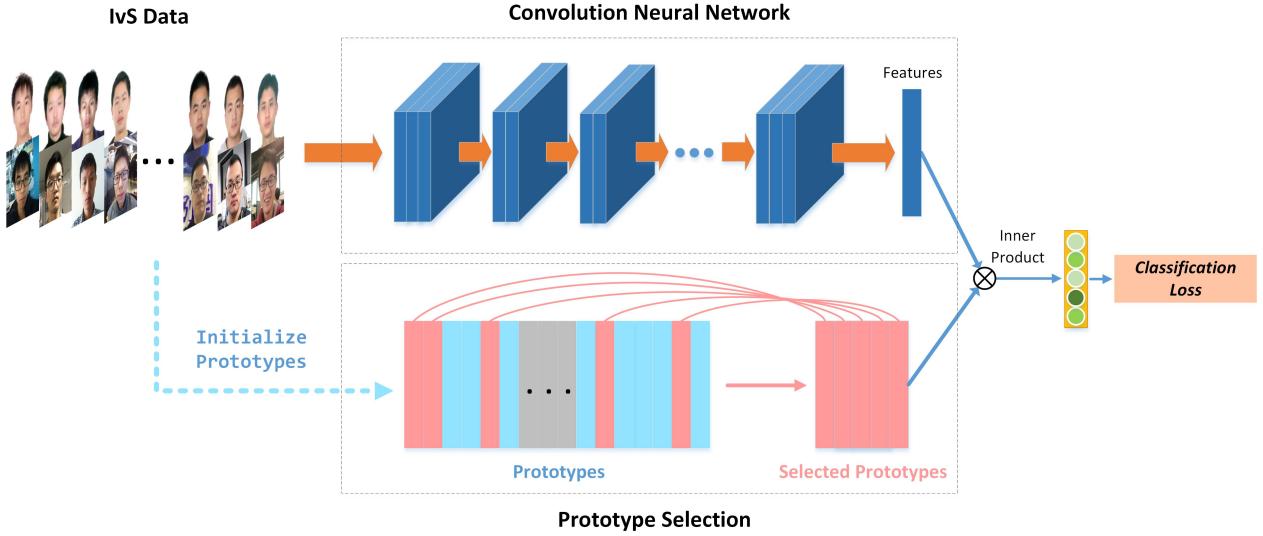


Fig. 3: Overview of large-scale classification.

As a result,  $N_{iter}$  is set empirically to balance the performance and the training time. In our experiments, with  $N_{iter} = 100,000$  the RP-softmax significantly improves the performance in IvS scenarios.

### 3.3.2 Dominant Prototype Softmax

Although RP-softmax makes it possible to perform large-scale classification, it is still inefficient due to its blind prototype selection. In this section, we show that the quality not the quantity really matters in prototype selection. We begin with the demonstration that in each iteration, only a small fraction of negative prototypes generate strong gradients.

In Equ. 3, a negative prototype  $\mathbf{w}_i$  contributes to the back-propagated gradient by  $p_{ij}\mathbf{w}_i$ , whose norm is  $p_{ij}\|\mathbf{w}_i\|$ . Usually, we restrict  $\|\mathbf{w}_i\|$  to one [22] and the norm will be  $p_{ij}$ , which can measure the impact of  $\mathbf{w}_i$  to the training process. In this paper, with a mini-batch  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_M]$ , we define the energy of a negative prototype as:

$$E_{neg}(\mathbf{w}_i) = \sum_{j=1}^M p_{ij}, \quad (6)$$

where  $p_{ij}$  is the probability that  $\mathbf{x}_j$  belongs to class  $i$ . Note that none of  $\mathbf{X}$  has the label  $i$  since  $\mathbf{w}_i$  is a negative prototype. To analyze whether the energy is concentrated on a small fraction of prototypes, we further define the top- $K$  cumulative energy as:

$$CE_K = \frac{\sum_{\mathbf{w}_i \in \mathcal{T}_K} E_{neg}(\mathbf{w}_i)}{\sum_{\mathbf{w}_i \in \mathcal{W}_{neg}} E_{neg}(\mathbf{w}_i)}, \quad (7)$$

where  $\mathcal{W}_{neg}$  is the set of negative prototypes and  $\mathcal{T}_K$  is the set of  $K$  negative prototypes with the largest energy. A large  $CE_K$  with small  $K$  denotes that the energy of negative prototypes are highly concentrated. We plot the  $CE_K$  along the training process in Fig. 4. It can be seen that in the beginning the top-5000 possesses 92.71% of energy. As the training proceeds, the energy becomes more and more concentrated. In the middle and end of the training process, the energy of top-5000 is increased to 96.09% and 98.79%. These results indicate that only a small fraction of prototypes can produce large gradients to affect training. We call these negative prototypes with large energy as **dominant prototypes**.

In real implementation, given a batch of features, how can we know the most dominant prototypes before we compute the probabilities in softmax? In this paper, we assume that if two identities have similar ID features, their prototypes and features are likely to make hard negative pairs. Based on this, we propose the **Dominant Prototype Softmax (DP-softmax)**. The basic idea is selecting prototypes from a set of dominant queues and updating the queues by the softmax predictions. The procedure is detailed as follows:

**Queue Initialization:** For each class  $i$ , we define the K-Nearest Classes  $\mathcal{NC}_K^{(i)}$  as the top- $K$  classes having the nearest ID features with  $i$ . Before training, we build an approximate nearest neighbor (ANN) graph by ID features and get the  $\mathcal{NC}_K^{(i)}$  for each class. Then we construct a dominant queue  $\mathcal{Q}_i$  and a candidate set  $\mathcal{C}_i$  for each class. The  $\mathcal{Q}_i$  is initialized by  $\mathcal{NC}_{100}^{(i)}$  and its members are sorted by the distances of ID features to  $i$ . The  $\mathcal{C}_i$  is set to  $\mathcal{NC}_{300}^{(i)}$ . Note that  $\mathcal{Q}_i \subseteq \mathcal{C}_i$ .

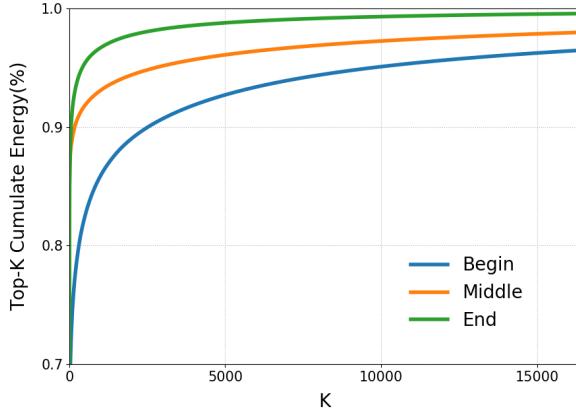


Fig. 4: The top- $K$  cumulate energy of negative prototypes ( $CE_K$ ) for a mini-batch, in the beginning, middle (100,000 iterations) and end (200,000 iterations) of the training process. The batch size is 50 and the number of classes is 2,578,178. The curves come from averaging  $CE_K$  of 5,000 mini-batches.

**Prototype Selection:** After training begins, in each iteration we need to select prototypes for the mini-batch  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_M]$ . First we select their positive prototypes  $\mathbf{W}_{pos} = [\mathbf{w}_{y^{(1)}}, \dots, \mathbf{w}_{y^{(M)}}]$  where  $y^{(j)}$  is the label of  $\mathbf{x}_j$ . Second, for each feature  $\mathbf{x}_j$  we select the prototypes of the classes in its dominant queue that  $\mathbf{W}_j^{qu} = [\mathbf{w}_q | q \in \mathcal{Q}_{y^{(j)}}]$  and the full negative prototypes are  $\widehat{\mathbf{W}}_{neg} = [\mathbf{W}_j^{qu}, \dots, \mathbf{W}_M^{qu}]$ . Thirdly, we remove the repeated prototypes and randomly select negative prototypes into  $\widehat{\mathbf{W}}_{neg}$  until a preset number is reached. Finally  $\mathbf{W}_{pos}$  and  $\widehat{\mathbf{W}}_{neg}$  constitute the temporary prototype matrix  $\mathbf{W}_{iter}$  in this iteration and are copied into GPU for training. Algorithm 2 summarizes the DP-softmax.

**Queue Updating:** After training in each iteration, we can update the dominant queues by the predictions of softmax. For a feature  $\mathbf{x}_j$ , its highest activated class  $h$  provides valuable information: First if  $h = y^{(j)}$  then it is a successful prediction and there is nothing to update. Second if  $h \neq y^{(j)}$  but  $h \in \mathcal{Q}_{y^{(j)}}$ , then this is a mis-prediction but the wrong-matched class is still in the dominant queue. Hence we need not to update  $\mathcal{Q}_{y^{(j)}}$ . Thirdly, if  $h \neq y^{(j)}$  and  $h \notin \mathcal{Q}_{y^{(j)}}$  but  $h \in \mathcal{C}_{y^{(j)}}$ , it means the class neighborhood has changed as the training proceeds. Therefore, we push  $h$  into  $\mathcal{Q}_{y^{(j)}}$  and pop the class that is the most dissimilar to  $y^{(j)}$ . Finally if  $h \neq y^{(j)}$  and  $h$  is not in  $\mathcal{Q}_{y^{(j)}}$  or  $\mathcal{C}_{y^{(j)}}$ , it means  $h$  and  $y^{(j)}$  have dissimilar ID features in the beginning but become close at this time. This case is mostly caused by the mislabelled or low-quality spot photo of  $h$  which

---

**Algorithm 2: Dominant Prototype Softmax**


---

```

Input : Prototype matrix:  $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_N]$ 
          Feature matrix:  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_M]$ 
          Number of selected classes  $N_{iter}$ 
          Dominant Queues:  $\mathcal{Q}_i, i = 1, \dots, N$ 
Output : Updated prototype matrix
1 Initialize selected class set  $\mathcal{L} = \emptyset$ 
2 for each feature  $\mathbf{x}_j$  in  $\mathbf{X}$  do
3   Get the label  $y^{(j)}$  of  $\mathbf{x}_j$ 
4    $\mathcal{L}.insert(y^{(j)})$ 
5   for each class  $k$  in  $\mathcal{Q}_{y^{(j)}}$  do
6      $\mathcal{L}.insert(k)$ 
7   end
8 end
9 while  $\mathcal{L}.size < N_{iter}$  do
10  | randomly select a label  $y^{(r)}$ 
11  |  $\mathcal{L}.insert(y^{(r)})$ 
12 end
13 for  $i = 1 \dots N_{iter}$  do
14  |  $\mathbf{W}_{iter}[i, :] = \mathbf{W}[\mathcal{L}_i, :]$ ,  $\mathcal{L}_i$  is the member of  $\mathcal{L}$ 
15 end
16 Training with  $\mathbf{W}_{iter}$  and  $\mathbf{X}$ , getting updated  $\mathbf{W}_{iter}^+$ 
17 for  $i = 1 \dots N_{iter}$  do
18  |  $\mathbf{W}[\mathcal{L}_i, :] = \mathbf{W}_{iter}^+[i, :]$ 
19 end

```

---

misdirecits its prototype, as shown in Fig. 5. Therefore, we do not update  $\mathcal{Q}_{y^{(j)}}$  since  $h$  is a noisy label.

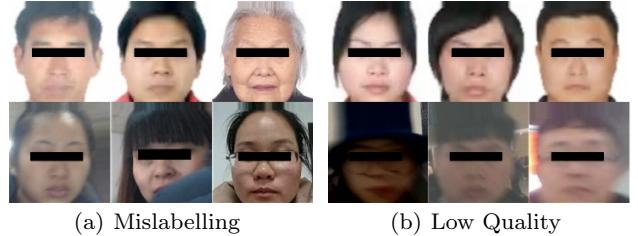


Fig. 5: The refused mis-predicted class. When a mispredicted class is refused to enter the dominant queue, there are always something wrong in its spot photo, including (a) mislabelling and (b) low quality.

The whole prototype selecting and queue updating operations can be done in real time. Compared with the RP-softmax, the DP-softmax significantly improves the quality and reduce the quantity of prototypes, leading to faster training and better performance.

Since the prototypes are saved in memory, which can easily hold tens of millions of prototypes, the dominant prototype selection scales up the classification scheme to any number of classes. Besides, when new training data come, the prototype matrix  $\mathbf{W}$  can be extended by the ID features of the new identities. Then the network can be finetuned on the whole training data.

## 4 Experiments

In this section, the proposed large-scale bisample learning (LBL) is systematically evaluated. We first analyze the CVC training strategy. Then we explore how different prototype selection methods affect the final performance. Finally we conduct comparison experiments on three datasets including CASIA-IvS-Test, Public-IvS and Megaface-bisample.

### 4.1 Datasets

**Ms-Celeb-1M:** The Ms-Celeb-1M [14] is one of the largest wild dataset containing 98,685 celebrities and 10 million images. The list of [30] is adopted to clean the noisy labels, resulting in 79,077 identities and 5 million images.

**CASIA-IvS:** The CASIA-IvS dataset is collected for IvS face recognition. The training set CASIA-IvS-Train contains 2,578,178 identities, each having two images. One image is the ID photo from the ID card, which is taken with uniform background, in frontal pose, normal illumination and neutral expression. The other is the spot photo taken by on-site devices, with variations in pose, expression, illumination, occlusion and resolution, as shown in Fig. 6. The test set CASIA-IvS-Test contains 4,000 identities and 8,000 images, which are checked manually to clean the noisy labels and ensure there is no identity overlap between training and test sets. During testing, all the ID photos and spot photos are paired, generating 4,000 positive pairs and nearly 16 million negative pairs.



Fig. 6: Example images in CASIA-IvS.

**Public-IvS:** An IvS test dataset is released for open evaluation. We found some public characters, such as politicians, teachers and researchers, had their ID photos on BaiduBaike [61] and official pages. We recorded their names and collected their spot photos on the web. Afterwards, we cleaned the dataset manually and removed the profile-view images. The final Public-IvS dataset has 1,262 identities and 5,507 images, each

identity having one ID photo and 1 to 10 spot photos. There are 4,871 positive pairs and nearly 6 million negative pairs. Fig. 7 shows some images in Public-IvS. Although Public-IvS is not a strictly IvS dataset since the spot photos are collected from the web, experiments on Public-IvS have consistent results with the real-world CASIA-IvS-Test.



Fig. 7: Example images in Public-IvS.

### 4.2 Experimental Settings

**Preprocessing** We detect faces by the FaceBox [62] detector and localize 5 landmarks (two eyes, nose tip and two mouth corners) by a simple 6-layer CNN [63]. All the faces are normalized by similarity transformation and cropped to  $120 \times 120$  RGB images.

**CNN Architecture** For the sake of fairness, all the CNN models in the experiments follow the same ResNet64 architecture [22]. It has four residual blocks and gets a 512-dimensional feature vector by average pooling. The learning rate begins with 0.001 and is divided by 10 when the loss does not decrease. All the networks are trained on 4 TITANX GPUs parallelly and the batch size is set to occupy all the GPU memory. Specifically, the batch size is 66 in the verification scheme and about 50 in the classification scheme.

**Training Setup** There are three stages in the CVC training strategy: pre-learning by classification on wild data, transfer learning by verification on IvS data and fine-grained learning by large-scale classification on IvS data. In the first stage, we train model from scratch by the A-Softmax loss [22] on the Ms-Celeb-1M. In the second stage, we finetune the model on CASIA-IvS-Train with the triplet loss [8]. The triplet loss is modified by N-pairs batch construction [26], online hard-negative mining [8] and anchor swapping [64]. In the third stage, we adopt the proposed DP-softmax to finetune the model on CASIA-IvS-Train. If not specified, there are two samples for each class in a mini-batch; the classification layer in the third stage is initialized by the ID-prototypes; softmax provides the probabilities and

A-softmax provides the gradients. In DP-softmax the sizes of dominant queues and candidate sets are 100 and 300, respectively.

**Evaluation Setup** For each image, we extract features from both the original image and the flipped one and concatenate them as the final representation. The score is measured by the cosine distance of two features. We evaluate all the networks with ROC curves. The verification rate (VR) at low false acceptance rate (FAR) is preferred since in real application false acceptance gives higher risks than false rejection.

### 4.3 Bisample Training

#### 4.3.1 Classification-Verification-Classification (CVC)

To illustrate the effectiveness of CVC, we show the intermediate results in Table 2. After the first stage, the C $\#\#$  is a well trained model in wild scenarios, with 99.53% on LFW [65] and 90.38% at FAR=  $10^{-6}$  on Megaface challenge [13]. However the state-of-the-art face model cannot work well on CASIA-IvS-Test, indicating the large bias between the two scenarios. Second, after being finetuned on CASIA-IvS-Train with the triplet loss, the CV $\#$  achieves much better performance, indicating the knowledge is successfully transferred from wild scenarios to IvS scenarios. Finally, the large-scale classification on CASIA-IvS-Train further improves the performance and reaches 91.92% at FAR=  $10^{-5}$ .

To further analyze the impact of each stage, we perform an ablation study by removing some stages. First, in C $\#\#$  we directly perform large-scale classification on IvS data without any initialization and find the loss does not decrease after 200,000 iterations. Second, we try to train model from scratch by the triplet loss on IvS data. Since the learning task is challenging without any initialization, we begin without hard-negative mining and slightly increase the ratio of hard negatives. The training converges but the model C $\#V\#$  has a bad result. Thirdly, we pre-train the model on wild data and directly finetune it on IvS data by large-scale classification. The training successfully converges but the resultant C $\#C$  is worse than the complete CVC. Finally, after pre-training on wild data, we perform joint verification and large-scale classification on IvS data, yielding the C(VC) model, which is also inferior than the complete CVC. From the results we can conclude that: (1) Comparing C $\#\#$ , C $\#C$  and CVC, a good initialization is crucial for the large-scale classification on bisample data. (2) Comparing C $\#C$ , CV $\#$  and CVC, the verification scheme has higher scalability than the classification scheme when dealing with large-scale bisample data, but it cannot get satisfactory performance independently. (3) Comparing

C $\#C$ , C(VC) and CVC, the smoothness is important in knowledge transferring and it is better to bridge the two classification stages with a verification stage.

There are some interesting phenomena we have observed in CVC learning. First, we find that the wild performance in the first stage does not affect the final IvS performance much. We begin with two pre-trained models with different wild performance (98.0% on LFW with triplet loss and 99.53% with A-softmax) and find their final IvS performances differ slightly (91.23% vs. 91.92% at FAR=  $10^{-5}$  on IvS). Second, we find the model cannot keep its high wild performance after being finetuned on IvS data. We evaluate models on both CASIA-IvS-Test and LFW [66], shown in Table 3. After each stage of CVC, the IvS performance is improved at the cost of degenerated wild performance. We further train our model on the joint data from both scenarios and find the wild performance is greatly improved with slight drop in IvS. This joint training is a good strategy when both scenarios are concerned.

Methods	CASIA-IvS-Test (VR@FAR= $10^{-5}$ )	LFW-BLUFER (VR@FAR= $10^{-5}$ )
C $\#\#$	51.90	94.23
CV $\#$	83.23	86.38
CVC	91.92	80.71
CVC $^+$	89.96	90.81

Table 3: The performances in wild scenarios (LFW-BLUFER protocol [66]) and IvS scenarios (CASIA-IvS-Test) after each stage CVC. The CVC $^+$  means the final large-scale classification stage is performed on the joint data from both Ms-Celeb-1M and CASIA-IvS-Train.

#### 4.3.2 Prototype Construction

As introduced in Section 3.2, there are two ways to construct the prototypes in large scale classification: The ID-prototype is the feature of the ID photo and the avg-prototype is the average vector of all the features in this class. The way to construct prototypes depends on the loss function involved. We select the most representative softmax [16] and the state-of-the-art A-softmax [22] in this experiment. Table 4 shows the performances with different losses and prototypes.

When softmax is adopted, the model initialized by avg-prototypes almost converges in the beginning and the loss only produces small gradients. If we replace avg-prototypes with ID-prototypes, the softmax loss will have a larger initial loss and end up with better results. When A-softmax is adopted, the angular margin keeps the initial loss large enough and the two prototypes

Method	Procedure			Performance		
	Classification (A-Soft on MS)	Verification (Triplet on IvS)	Classification (DP-Soft on IvS)	VR@FAR= $10^{-3}$	VR@FAR= $10^{-4}$	VR@FAR= $10^{-5}$
C##	✓			85.31	69.11	51.90
CV#	✓	✓		96.41	91.39	83.23
CVC	✓	✓	✓	97.70	96.17	91.92
#C			✓	not converge		
#V#		✓		79.39	58.90	38.33
C#C	✓		✓	94.36	85.35	72.35
C(VC)	✓	✓	✓	96.43	91.75	82.80

Table 2: The intermediate results after each stage in the CVC training strategy. The performance is evaluated by the verification rate, VR(%), on CASIA-IvS-Test. In each stage, we indicate the loss function and the training data, where A-Soft refers to A-softmax, Triplet refers to triplet loss, DP-Soft refers to DP-softmax, MS refers to Ms-Celeb-1M and IvS refers to CASIA-IvS-Train. The “#” in method names indicates the corresponding stage is skipped.

Method	FAR= $10^{-3}$	FAR= $10^{-4}$	FAR= $10^{-5}$
softmax (avg)	96.94	93.55	87.13
softmax (ID)	97.31	94.91	89.55
A-softmax (avg)	97.35	95.40	90.30
A-softmax (ID)	97.43	95.40	90.34

Table 4: The comparison of different prototype construction methods with different loss functions on CASIA-IvS-Test, evaluated by VR(%) at different FAR. The prototypes are randomly selected.

end up with close performances. In our experiments, we prefer ID-prototypes and only adopt avg-prototypes when there is no ID photo like the mimic experiments in Sec. 4.7.

#### 4.4 Large-scale Classification

In large-scale classification, we need to select a fraction of prototypes each time. In Sec. 3.3 we introduce two methods for prototype selection: one is to select prototypes randomly and the other is to select the dominant prototypes.

##### 4.4.1 Random Prototype Softmax

In random prototype softmax (RP-softmax), we can increase the involved classes at a small cost of batch size due to the tiny memory cost of a single prototype. We evaluate the RP-softmax with 20k, 50k and 100k prototypes respectively in Table 5 and find more prototypes always come with better performance.

However, increasing the number of prototypes is not cost free. More prototypes increase the overhead of computing softmax and copying prototypes in GPUs. In Fig. 8, we show the time costs and GPU-util percent with different prototype numbers. When prototypes

Method	FAR= $10^{-3}$	FAR= $10^{-4}$	FAR= $10^{-5}$
RPS(20k)	97.19	94.49	87.84
RPS(50k)	97.19	94.71	88.57
RPS(100k)	97.43	95.40	90.34

Table 5: The performance of RP-softmax (RPS) on CASIA-IvS-Test, evaluated by VR(%) at different FAR. The values in the brackets are the numbers of prototypes.

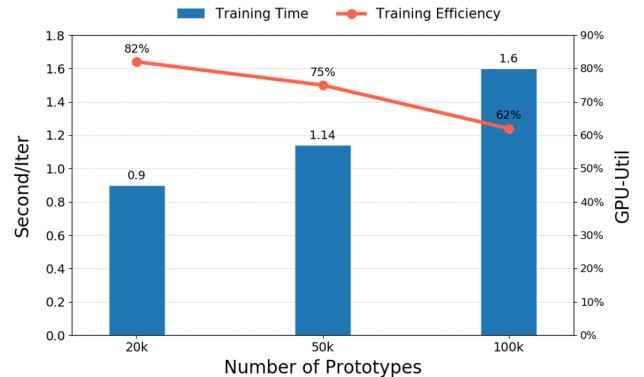


Fig. 8: The total training time (forward and backward propagation) of one mini-batch and the GPU-util percent with different prototype numbers. Low GPU-util percent means the GPU is blocked to wait for prototype copying.

increase from 20k to 100k, the training time increases by 78% and the GPU-util percent drops from 82% to 62%. We further try 300k prototypes and find the GPU-util percent drops to 48%, which means most time is spent on waiting for prototype copying.

##### 4.4.2 Dominant Prototype Softmax

To improve performance and training efficiency simultaneously, we select the dominant prototypes instead

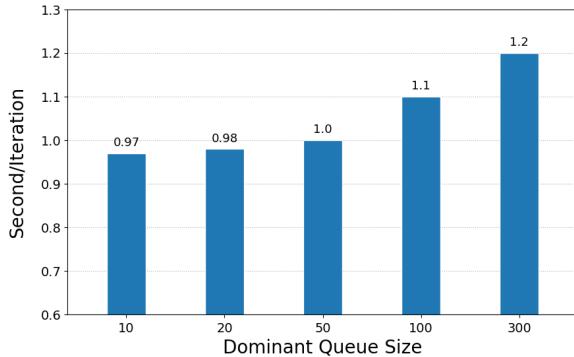


Fig. 9: The total training time (forward and backward propagation) of one mini-batch with different dominant queue size.

of the random prototypes. In DP-softmax we maintain a dominant queue for each class to store their similar classes, where the queue size  $q$  is an important parameter that impacts both performance and training time. Table 6 shows the performances with different queue sizes and Fig. 9 shows the corresponding training time. We can see that the performance increases as the queue size increases, but quickly saturates when  $q$  reaches 100 with only 3,000 prototypes. Considering both performance and efficiency we set  $q = 100$  in our implementation. Compared with RP-softmax with 100,000 prototypes, DP-softmax achieves better performance (91.92% vs. 90.30% at  $\text{FAR}=10^{-5}$ ) with much lower training time (1.1s vs. 1.6s per iteration).

Method	$\text{FAR}=10^{-3}$	$\text{FAR}=10^{-4}$	$\text{FAR}=10^{-5}$
$\text{DPS}_{10}(0.3k)$	96.62	92.48	85.12
$\text{DPS}_{20}(0.6k)$	96.77	93.69	86.57
$\text{DPS}_{50}(1.5k)$	97.16	94.37	88.29
$\text{DPS}_{100}(3.0k)$	97.70	96.17	91.92
$\text{DPS}_{300}(10.0k)$	97.72	96.27	92.01

Table 6: The performances of DP-softmax (DPS) with different queue sizes on CASIA-IvS-Test, evaluated by VR(%) at different FAR.  $\text{DPS}_q(N_{\text{iter}})$  indicates the queue size is  $q$  and the number of prototypes is  $N_{\text{iter}}$ . Note that there are two samples per class in the mini-batch, there are at most  $Mq/2$  dominant prototypes.

In Table 7, we also compare the performances with and without queue updating, which demonstrates the effectiveness of queue updating.

#### 4.4.3 Softmax Formulation

Large-scale classification mainly involves a prototype selection strategy, which can be combined with any soft-

Method	$\text{FAR}=10^{-3}$	$\text{FAR}=10^{-4}$	$\text{FAR}=10^{-5}$
w/o update	97.54	95.58	90.77
update	97.70	96.17	91.92

Table 7: The performances of DP-softmax with and without queue updating, evaluated by VR(%) at different FAR.

max formulation. Besides the traditional softmax [16], the state-of-the-art A-softmax [22] and AM-softmax [24] can also be adopted. Table 8 shows the results with different softmax formulations. We can see that A-softmax and AM-softmax have improved performance by introducing the margins and A-softmax has the best results.

Method	$\text{FAR}=10^{-3}$	$\text{FAR}=10^{-4}$	$\text{FAR}=10^{-5}$
DPS+softmax	97.31	94.91	89.55
DPS+AM-softmax	97.60	95.59	90.73
DPS+A-softmax	97.70	96.17	91.92

Table 8: The performances of adopting different softmax formulations in large-scale classification, evaluated by VR(%) at different FAR on CASIA-IvS-Test. The dominant prototype selection (DPS) is adopted.

#### 4.5 Identity Volume

It has been repeatedly observed that more data always delivers better performance [8, 67]. Does the blessing of data still exist in IvS face recognition? To study this, we randomly sample a subset of 100k, 500k and 2M identities from CASIA-IvS-Train and train the model, respectively. As shown in Fig. 10, the performance grows logarithmically as identities increase, which is consistent with [67]. We believe more identities provide more information about intra- and inter-variations, which delivers more discriminative features. Besides, it is suggested that the model can be further improved with more IvS data.

#### 4.6 Comparison Experiments

In order to compare our method with the state of the arts, we choose several methods feasible on large-scale bisample data, including **Contrastive** [7], **Triplet** [8], **Lifted Struct** [25], **N-pairs** [26] and the Model A-D in Megaface challenge [13] (**MF-A** to **MF-D**). We also evaluate the large-scale classification methods in language models including Noise Contrastive Estimation (**NCE**) [47] and Hierarchical Softmax (**H-softmax**) [50].

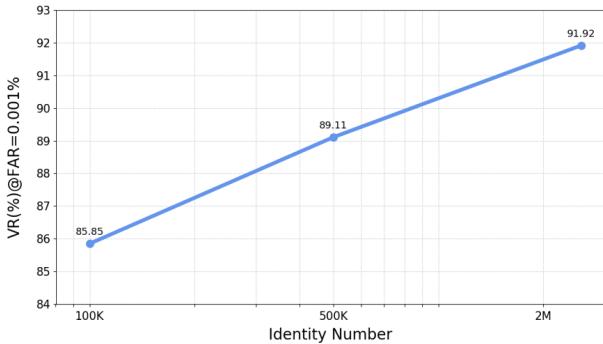


Fig. 10: The performances under different identity volume, evaluated by VR(%) at  $\text{FAR}=10^{-5}$ .

For fair comparison, all the methods adopt the ResNet64 architecture and their models are pretrained on Ms-Celeb-1M. In our implementation, for contrastive, each sample is paired with all the other ones in a mini-batch and the negative pairs are filtered by hard negative mining. For triplet, we adopt N-pairs batch construction [26] and anchor swapping [64] to construct the most triplets. Besides, online hard mining [8] is performed to remove easy triplets. For N-pairs, we adopt the N-pair-mc loss to optimize each positive pair against all the related negative pairs and use the hard negative class mining to generate mini-batches with similar classes. For lifted struct, we directly use the released codes. For MF-A, we train the model with softmax on randomly selected 100,000 classes. Then we finetune MF-A on the full data with the triplet loss as MF-B. For MF-C, we adopt the rotating softmax where 20,000 random classes are selected in each epoch. Then we adopt the same triplet finetuning strategy to get MF-D. For NCE and H-softmax, directly training with the two losses cannot converge. In our implementation, we first train the models by the triplet loss and initialize the prototypes by the deep features as our LBL, making the training convergent. For our methods, we first provide a naive baseline to perform softmax on IvS data named LBL(softmax), where in the final stage of CVC we train model only on 100,000 classes (which are the most classes affordable by the machine) and the classes do not change as training proceeds. Besides, we report LBL with RP-softmax and DP-softmax.

Table. 9 shows the performances on the real-world CASIA-IvS-Test and the open Public-IvS. Fig. 11(a) and Fig. 11(b) show the corresponding ROC curves. During implementation, we find MF-A cannot achieve satisfactory performance since only a small part of data can be used. MF-C is hard to converge since the rotating softmax randomly initializes the prototypes periodically. After finetuned by the triplet loss on all

the data, the models (MF-B and MF-D) still fail to get satisfactory performances due to the poor initializations. As for our method LBL, we can see Public-IvS shows consistent results with CASIA-IvS-Test where our methods perform best. Besides, LBL significantly outperforms other methods on IvS data, especially at low FAR. The improvement at  $\text{FAR}=10^{-5}$  is 84.16% to 91.92% on CASIA-IvS-Test and 88.63% to 93.62% on Public-IvS. The DP-softmax further improves the RP-softmax and achieves the best performance. LBL also achieves better recognition rates than the large scale classification methods like NCE and H-softmax.

#### 4.7 Mimic Experiments on Megaface-bisample

To make our work reproducible, we mimic the large-scale bisample challenge on the open MF2 [13] dataset and propose a new protocol Megaface-bisample. The MF2 contains 657,559 identities which are much more than other datasets. We split MF2 into two subsets, MF2-thick and MF2-mini. The MF2-thick contains the identities having more than 15 samples, which is used to simulate the well-posed dataset for pre-learning. The MF2-mini contains two randomly selected samples for each identity, which is used to simulate the bisample data. As for testing, we follow the BLUFR protocol [66] on LFW [65]. In summary, MF2-thick, MF2-mini and LFW-BLUFR simulate Ms-Celeb-1M, CASIA-IvS-Train and CASIA-IvS-Test, respectively. Specifically, MF2-thick has 46,000 identities and 34.8 samples per identity and MF2-mini has cleaned 649,790 identities and 2 samples per identity, whose image list will be released. As well known, MF2 has few celebrities and we have tried our best to ensure there is no identity overlap between MF2 and LFW. Although Megaface-bisample is not IvS data, it shares the same challenges: the weak intra-variations and model training scalability, as IvS data. Since there is no ID photo in MF2, we initialize the classification layer with avg-prototypes and construct the  $\mathcal{NC}_K$  by avg-prototypes instead of ID features.

First, to verify the effectiveness of the simulation, we re-implement the experiments of Table. 2 about the CVC training strategy. As shown in Fig. 12, there is significant improvement after each stage. Besides, we try to train model from scratch on MF2-mini and find the training quickly falls into bad local optima. Since the results are consistent with the ones on CASIA-IvS, we believe Megaface-bisample can well simulate our task.

On Megaface-bisample we also compare our methods with the state of the arts in Table. 10, whose ROC curves are shown in Fig. 11(c). The proposed LBL still

Method	CASIA-IvS-Test			Public-IvS		
	FAR= $10^{-3}$	FAR= $10^{-4}$	FAR= $10^{-5}$	FAR= $10^{-3}$	FAR= $10^{-4}$	FAR= $10^{-5}$
Contrastive [7]	96.25	91.17	81.39	96.52	91.71	84.54
Triplet [8]	96.41	91.39	83.23	97.72	94.11	87.47
Lifted Struct [25]	96.42	92.25	83.53	98.03	94.56	<b>88.63</b>
N-pairs [26]	96.45	92.13	83.96	<b>98.23</b>	<b>94.57</b>	86.49
NCE [47](Triplet-init)	96.30	91.18	82.62	97.90	93.93	87.27
H-softmax [50](Triplet-init)	<b>96.50</b>	<b>92.36</b>	<b>84.16</b>	98.01	94.54	87.45
MF-A [13]	51.61	33.82	20.67	49.42	28.40	14.99
MF-B [13]	75.24	53.85	35.09	66.68	44.40	28.66
MF-C [13]	51.02	31.11	15.05	43.41	24.20	12.55
MF-D [13]	75.04	52.46	31.84	64.68	42.85	25.23
LBL(softmax)	97.01	93.69	86.68	98.38	95.49	89.63
LBL(RP-softmax)	97.43	95.40	90.34	98.44	96.29	91.99
LBL(DP-softmax)	<b>97.70</b>	<b>96.17</b>	<b>91.92</b>	<b>98.83</b>	<b>97.21</b>	<b>93.62</b>

Table 9: The performances of the state of the arts, evaluated by the VR(%) at different FAR. The models are trained on CASIA-IvS-Train and evaluated on CASIA-IvS-Test and Public-IvS, with our method and the best baseline highlighted.

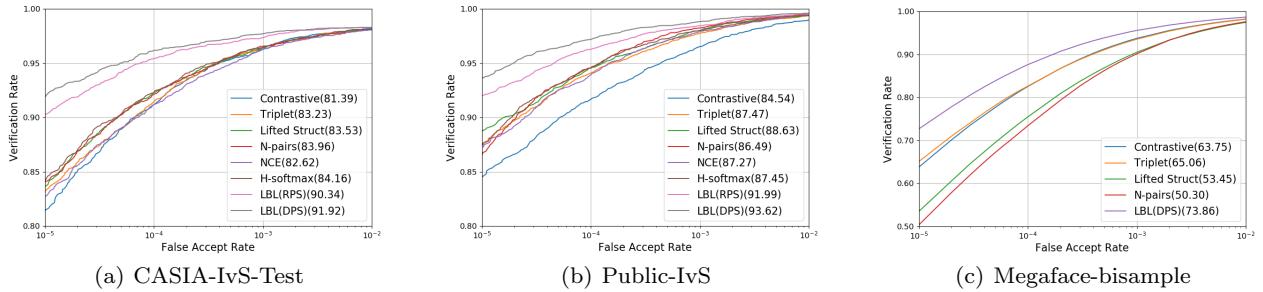


Fig. 11: Comparison of ROC curves on CASIA-IvS-Test, Public-IvS and Megaface-bisample. The values in the brackets are the VR(%) at FAR= $10^{-5}$ .

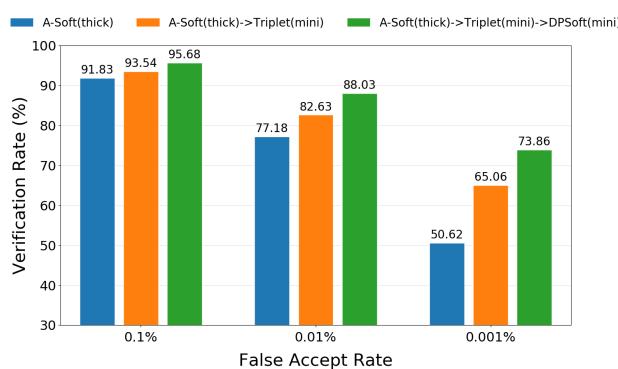


Fig. 12: The intermediate results of CVC, following the Megaface-bisample protocol.

consistently outperforms the other methods and the improvement at FAR =  $10^{-5}$  is over 8 percent.

Methods	LFW-BLUFR		
	FAR= $10^{-3}$	FAR= $10^{-4}$	FAR= $10^{-5}$
Contrastive [7]	<b>93.74</b>	82.53	63.75
Triplet [8]	93.54	<b>82.63</b>	<b>65.06</b>
Lifted Struct [25]	90.50	75.46	53.45
N-pairs [26]	90.16	73.40	50.30
LBL(DP-softmax)	<b>95.68</b>	<b>88.03</b>	<b>73.86</b>

Table 10: The verification rates, VR(%), at different false acceptance rates (FAR) on LFW-BLUFR following the Megaface-bisample, with the top-2 results highlighted.

## 5 Conclusion

This paper proposes a large-scale bisample learning (LBL) method to train deep neural networks on ID vs. Spot (IvS) face data. Specifically, we develop a Classification-Verification-Classification (CVC) bisample training strategy that first transfers the knowledge from wild scenarios to IvS scenarios and then boosts the performance by large-scale classification. We also propose a dominant prototype softmax (DP-softmax)

to perform 2-million classification, which is used in the final stage of CVC. The DP-softmax diligently selects the dominant prototypes for each mini-batch, which improves the performance and reduces the training cost simultaneously. Experiments on a large real-world dataset show the proposed LBL significantly improves the IvS face recognition and the DP-softmax can perform effective classification with only 0.15% of classes. Besides, we also release a Public-IvS dataset for open IvS evaluation and a new protocol Megaface-bisample to mimic the large-scale bisample learning task.

## 6 Acknowledgments

This work was supported by the Chinese National Natural Science Foundation Projects #61876178, #61806196, the National Key Research and Development Plan (Grant No.2016YFC0801002), and AuthenMetric R&D Funds. Zhen Lei is the corresponding author.

## References

1. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *International Conference on Neural Information Processing Systems*, 2012, pp. 1097–1105.
2. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
3. K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
4. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” *computer vision and pattern recognition*, pp. 2818–2826, 2016.
5. K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *computer vision and pattern recognition*, pp. 770–778, 2016.
6. Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “Deepface: Closing the gap to human-level performance in face verification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1701–1708.
7. Y. Sun, Y. Chen, X. Wang, and X. Tang, “Deep learning face representation by joint identification-verification,” in *Advances in neural information processing systems*, 2014, pp. 1988–1996.
8. F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.
9. E. Smirnov, A. Melnikov, S. Novoselov, E. Luckyanets, and G. Lavrentyeva, “Doppelganger mining for face representation learning,” in *International Conference on Computer Vision*, 2017.
10. C. Huang, C. C. Loy, and X. Tang, “Local similarity-aware deep feature embedding,” in *Advances in Neural Information Processing Systems*, 2016, pp. 1262–1270.
11. W. Chen, X. Chen, J. Zhang, and K. Huang, “Beyond triplet loss: a deep quadruplet network for person re-identification,” in *The Conference on Computer Vision and Pattern Recognition*, 2017.
12. D. Yi, Z. Lei, S. Liao, and S. Z. Li, “Learning face representation from scratch,” *arXiv: Computer Vision and Pattern Recognition*, 2014.
13. A. Nech and I. Kemelmacher-Shlizerman, “Level playing field for million scale face recognition,” *arXiv preprint arXiv:1705.00393*, 2017.
14. Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, “Ms-celeb-1m: A dataset and benchmark for large-scale face recognition,” in *European Conference on Computer Vision*. Springer, 2016, pp. 87–102.
15. Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, “Vggface2: A dataset for recognising faces across pose and age,” *arXiv preprint arXiv:1710.08092*, 2017.
16. Y. Sun, X. Wang, and X. Tang, “Deep learning face representation from predicting 10,000 classes,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1891–1898.
17. Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “Web-scale training for face identification,” *arXiv preprint arXiv:1406.5266*, 2014.
18. Y. Wen, K. Zhang, Z. Li, and Y. Qiao, “A discriminative feature learning approach for deep face recognition,” in *European Conference on Computer Vision*. Springer, 2016, pp. 499–515.
19. R. Ranjan, C. D. Castillo, and R. Chellappa, “L2-constrained softmax loss for discriminative face verification,” *arXiv preprint arXiv:1703.09507*, 2017.
20. F. Wang, X. Xiang, J. Cheng, and A. L. Yuille, “Normface:  $l_2$  hypersphere embedding for face verification,” *arXiv preprint arXiv:1704.06369*, 2017.
21. W. Liu, Y. Wen, Z. Yu, and M. Yang, “Large-margin softmax loss for convolutional neural networks.” in *ICML*, 2016, pp. 507–516.
22. W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, “Sphereface: Deep hypersphere embedding for face recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.
23. W. Liu, Y.-M. Zhang, X. Li, Z. Yu, B. Dai, T. Zhao, and L. Song, “Deep hyperspherical learning,” in *Advances in Neural Information Processing Systems*, 2017, pp. 3953–3963.
24. F. Wang, W. Liu, H. Liu, and J. Cheng, “Additive margin softmax for face verification,” *Signal Processing Letters, IEEE*, vol. 25, pp. 926–930, 2018.
25. H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, “Deep metric learning via lifted structured feature embedding,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4004–4012.
26. K. Sohn, “Improved deep metric learning with multi-class n-pair loss objective,” in *Advances in Neural Information Processing Systems*, 2016, pp. 1857–1865.
27. Y. Zhao, Z. Jin, G. Qi, H. Lu, and X. Hua, “A principled approach to hard triplet generation via adversarial nets,” in *European Conference on Computer Vision*, 2018.
28. V. B. Kumar, B. Harwood, G. Carneiro, I. Reid, and T. Drummond, “Smart mining for deep metric learning,” *arXiv preprint arXiv:1704.01285*, 2017.
29. C. Wang, X. Zhang, and X. Lan, “How to train triplet networks with 100k identities?” *arXiv preprint arXiv:1709.02940*, 2017.

30. X. Wu, R. He, Z. Sun, and T. Tan, “A light cnn for deep face representation with noisy labels,” *arXiv preprint arXiv:1511.02683*, 2015.
31. X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao, “Range loss for deep face recognition with long-tailed training data,” in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
32. L. Feifei, R. Fergus, and P. Perona, “One-shot learning of object categories,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.
33. G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition,” in *ICML Deep Learning Workshop*, vol. 2, 2015.
34. A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. P. Lillicrap, “One-shot learning with memory-augmented neural networks,” *arXiv: Learning*, 2016.
35. J. Weston, S. Chopra, and A. Bordes, “Memory networks,” *arXiv preprint arXiv:1410.8916*, 2014.
36. O. Vinyals, C. Blundell, T. P. Lillicrap, K. Kavukcuoglu, and D. Wierstra, “Matching networks for one shot learning,” *neural information processing systems*, pp. 3630–3638, 2016.
37. Y. X. Wang and M. Hebert, *Learning to Learn: Model Regression Networks for Easy Small Sample Learning*. Springer International Publishing, 2016.
38. L. Bertinetto, J. F. Henriques, J. Valmadre, P. H. S. Torr, and A. Vedaldi, “Learning feed-forward one-shot learners,” *neural information processing systems*, pp. 523–531, 2016.
39. Y. Guo and L. Zhang, “One-shot face recognition by promoting underrepresented classes,” *arXiv preprint arXiv:1707.05574*, 2017.
40. B. Hariharan and R. Girshick, “Low-shot visual recognition by shrinking and hallucinating features,” *arXiv preprint arXiv:1606.02819*, 2016.
41. Z. Chen, H. Zhao, X. Liu, and W. Xu, “An analysis of feature regularization for low-shot learning,” 2016.
42. J. Choe, S. Park, K. Kim, J. Hyun Park, D. Kim, and H. Shim, “Face generation for low-shot learning using generative adversarial networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1940–1948.
43. Z. Xu, L. Zhu, and Y. Yang, “Few-shot object recognition from machine-labeled web images,” *arXiv preprint arXiv:1612.06152*, 2016.
44. J. Yang, B. Price, S. Cohen, and M. H. Yang, “Context driven scene parsing with attention to rare classes,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3294–3301.
45. W. Ouyang, X. Wang, C. Zhang, and X. Yang, “Factors in finetuning deep model for object detection with long-tail distribution,” in *Computer Vision and Pattern Recognition*, 2016, pp. 864–873.
46. D. J. Hsu, S. M. Kakade, J. Langford, and T. Zhang, “Multi-label prediction via compressed sensing,” in *Advances in neural information processing systems*, 2009, pp. 772–780.
47. M. Gutmann and A. Hyvärinen, “Noise-contrastive estimation: A new estimation principle for unnormalized statistical models,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 297–304.
48. A. Choromanska, A. Agarwal, and J. Langford, “Extreme multi class classification,” in *NIPS Workshop: eXtreme Classification*, submitted, 2013.
49. Y. Prabhu and M. Varma, “Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 263–272.
50. Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.
51. R. Babbar and B. Schölkopf, “Dismec: distributed sparse machines for extreme multi-label classification,” in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 721–729.
52. K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain, “Sparse local embeddings for extreme multi-label classification,” in *Advances in Neural Information Processing Systems*, 2015, pp. 730–738.
53. Y. Tagami, “Annxml: Approximate nearest neighbor search for extreme multi-label classification,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 455–464.
54. C. Xu, D. Tao, and C. Xu, “Robust extreme multi-label learning,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1275–1284.
55. A. Mnih and Y. W. Teh, “A fast and simple algorithm for training neural probabilistic language models,” *arXiv preprint arXiv:1206.6426*, 2012.
56. A. Mnih and K. Kavukcuoglu, “Learning word embeddings efficiently with noise-contrastive estimation,” in *Advances in neural information processing systems*, 2013, pp. 2265–2273.
57. A. Vaswani, Y. Zhao, V. Fossum, and D. Chiang, “Decoding with large-scale neural language models improves translation,” in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1387–1392.
58. H. Wang, Y. Wang, Z. Zhou, X. Ji, and W. Liu, “Cosface: Large margin cosine loss for deep face recognition,” in *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.
59. E. Zhou, Z. Cao, and Q. Yin, “Naive-deep face recognition: Touching the limit of lfw benchmark or not?” *arXiv preprint arXiv:1501.04690*, 2015.
60. O. M. Parkhi, A. Vedaldi, A. Zisserman *et al.*, “Deep face recognition.” in *BMVC*, vol. 1, no. 3, 2015, p. 6.
61. <http://baike.baidu.com/>.
62. S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li, “Faceboxes: a cpu real-time face detector with high accuracy,” *arXiv preprint arXiv:1708.05234*, 2017.
63. Z.-H. Feng, J. Kittler, M. Awais, P. Huber, and X.-J. Wu, “Wing loss for robust facial landmark localisation with convolutional neural networks,” *arXiv preprint arXiv:1711.06753*, 2017.
64. V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk, “Learning local feature descriptors with triplets and shallow convolutional neural networks,” in *British Machine Vision Conference*, 2016, pp. 119.1–119.11.
65. G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, “E.: Labeled faces in the wild: A database for studying face recognition in unconstrained environments,” 2007.
66. S. Liao, Z. Lei, D. Yi, and S. Z. Li, “A benchmark study of large-scale unconstrained face recognition,” in *IEEE International Joint Conference on Biometrics*, 2014, pp. 1–8.
67. C. Sun, A. Shrivastava, S. Singh, and A. Gupta, “Revisiting unreasonable effectiveness of data in deep learning era,” in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 843–852.