How to Train Triplet Networks with 100K Identities?

Chong Wang
Orion Star
Beijing, China
chongwang.nlpr@gmail.com

Xue Zhang
Orion Star
Beijing, China
yuannixue@126.com

Xipeng Lan
Orion Star
Beijing, China
xipeng.lan@gmail.com

Abstract

Training triplet networks with large-scale data is challenging in face recognition. Due to the number of possible triplets explodes with the number of samples, previous studies adopt the online hard negative mining(OHNM) to handle it. However, as the number of identities becomes extremely large, the training will suffer from bad local minima because effective hard triplets are difficult to be found. To solve the problem, in this paper, we propose training triplet networks with subspace learning, which splits the space of all identities into subspaces consisting of only similar identities. Combined with the batch OHNM, hard triplets can be found much easier. Experiments on the large-scale MS-Celeb-1M challenge with $100K$ *identities demonstrate that the proposed method can largely improve the performance. In addition, to deal with heavy noise and large-scale retrieval, we also make some efforts on robust noise removing and efficient image retrieval, which are used jointly with the subspace learning to obtain the state-of-the-art performance on the MS-Celeb-1M competition (without external data in* $Challenge1$*).*

1. Introduction

Triplet loss is a metric learning method that has been widely used in many applications, *e.g.*, face recognition [10,11], image retrieval [12,14] and person re-identification [1,4]. A triplet usually consists of three samples: an anchor sample, a positive one with the same class to the anchor, and a negative one with the different class. The objective of triplet loss is to learn a metric that pushes the positive pairs closer while pulls the negative pairs away, thus the samples within the same class can be nearest to each other.

One question is why we need the triplet loss? Actually, there are some alternatives such as the softmax loss, which is also popular. However, as the number of classes becomes extremely large, the fully-connected layer that connects to softmax loss will also become extremely large, thus the GPU memory cost will be unbearable with an usual batch-

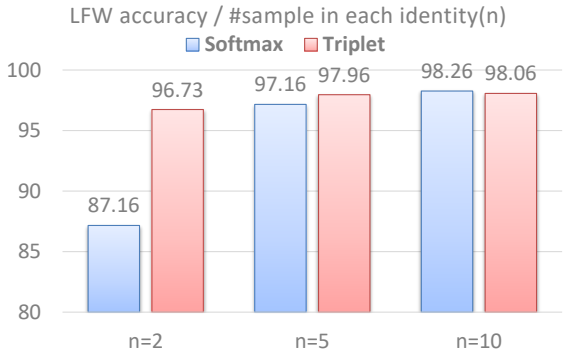

Figure 1. The LFW accuracy of the models trained with softmax and triplet loss with the number of samples in each class set to be 2, 5 and 10. The model is trained on MS-Celeb-1M [2] with $100K$ identities, and the evaluation is given by LFW [6].

size, while the small batchsize will take too long to train. Another reason is that if only a few samples are available in each class, training with softmax loss is difficult, and Fig.1 shows its influence on softmax and triplet loss with $100K$ identities. The triplet performs much better when the number of samples in each class is small($n = 2$).

Though the advantage is clear, there are some challenges to use it. One big challenge is how to train triplet models effectively with large-scale data, *e.g.*, $100K$ and $1M$ identities are common cases in face recognition. The difficulty of scaling triplet is that the number of possible triplets is cubic in the number of samples, and most triplets are too easy that cannot help training. To reduce searching space, some researchers transferred the triplet loss into softmax loss [5, 9, 15], while some proposed the Online Hard Negative Mining(OHNM) [7, 10, 13] or batch OHNM [4]. Most studies focused on the small-scale case, while FaceNet [10] used an extremely large number of identities($8M$), but it suffered from long training time(a few months).

In the above methods, all of them consider all identities to sample the batch. It is widely accepted that hard triplets can help training because they can reduce the ambiguity of recognizing similar identities, and it indicates these hard triplets should better come from similar identities. However, sampling from all the identities cannot guarantee to

include the similar ones, thus it will fail in generating effective hard triplets. Especially, in the large-scale face recognition with $100K$ or $1M$ identities, the probability of sampling similar identities with an usual batchsize is very tiny, *e.g.*, the batch with the size of 1800 is randomly sampled from $8M$ identities in FaceNet [10]. Therefore, how to find the similar identities is the key to improve the training of triplet networks with large-scale data.

In this paper, we consider the case of large-scale face recognition, and propose to train triplet networks with subspace learning. The basic idea is to generate a representation for each identity, and apply clustering on all the identities to generate clusters or subspaces, wherein identities are similar in each subspace. With the batch OHNM applied in each subspace iteratively, the proposed method can easily generate more effective hard triplets. Evaluations on the large-scale MS-Celeb-1M dataset with $100K$ identities [2] show that the proposed method can largely improve performance and get more robust triplet models. Particularly, our single triplet network obtains the LFW [6] accuracy of 99.48%, which can be competitive to FaceNet's 99.63% [10] with $8M$ identities.

This subspace learning with batch OHNM is our main contribution. In addition, given the fact that the MS-Celeb-1M dataset is noisy, we also design a noise removing trick to clean the training data at the beginning, and experiments show that it is able to handle different number of noises. Furthermore, consider that the number of training images is about $5M$ after the cleaning, we use a two-layer hierarchical trick to retrieve a test image accurately and efficiently. Combined with the proposed triplet network, we have achieved the state-of-the-art performance on the MS-Celeb-1M competition, *i.e.*, $Challenge1$ without external data. The recall of the random and hard set is 75% and 60.6% respectively, in which 75% has achieved the upper limit on the random set.

2. Related Work

In this part, we introduce some previous studies on how to accelerate the training of triplet networks. One big difficulty is that the number of possible triplets scales cubically with the number of training samples. To avoid directly searching the whole space, some researchers [5, 9, 15] convert the triplet loss into a form of softmax loss. Sankaranarayanan *et al*. [9] propose to transfer the Euclidean distance between positive and negative pairs into probabilistic similarity, and they use low-dimensional embedding for fast retrieval. Similar to [9], Zhuang *et al*. [15] convert the retrieval problem into a multi-label classification problem with binary codes, which is optimized by the binary quadratic algorithm to achieve faster retrieval. To simplify the optimization, Hoffer *et al*. [5] propose a Siamese-like triplet network by transferring the retrieval problem into a 2-class classification problem. These methods have shown promising speedup, but no hard triplets are considered, which will result in inferior performance.

Inspired by the efficiency of classification, some studies [1,7,10,13] combine the advantages of classification and hard triplets. Wang *et al*. [13] use a pretrained classification model to select possible hard triplets offline, but the offline selection is fixed as the classification model will not be updated. To achieve faster training and handle variant triplets, Parkhi *et al*. [7] train a classification network that is further fine-tuned with triplet loss. They use the Online Hard Negative Mining(OHNM), wherein only the triplets violating the margin constraint are considered as the hard ones for learning. Instead of fine-tuning with only triplet loss, Chen *et al*. [1] propose to train networks jointly with softmax and triplet loss to preserve both inter-class and intra-class information, and they also adopt OHNM in training. To apply OHNM in large-scale data, Schroff *et al*. propose the FaceNet [10] that trains triplet networks with $8M$ identities, and it takes a few months to finish with a large batchsize of 1800. One limitation of OHNM is that triplets are predefined in the batch, and this will miss possible hard negative samples that contained in the batch.

To make full use of the batch, some studies [4] generate hard triplets online within the batch. Hermans *et al*. [4] propose the batch OHNM, in which negative samples are searched within the batch based on their distance to the anchor, and the top nearest ones are considered as candidate hard negative samples. In this way, more hard triplets can be found easily, and the best performance is obtained in person re-identification with 1500 identities. Due to their small scale, the probability of sampling similar identities with an usual batchsize(128 or 256) is large. However, in the large-scale case, randomly sampling similar identities will be much more difficult, thus the batch OHNM will fail. In this paper, we focus on how to find effective hard triplets in large-scale face recognition.

3. Method

In this part, we will introduce how to train triplet networks with large-scale data. We first review the Online Hard Negative Mining(OHNM) and batch OHNM in training the triplet network, then we elaborate how to improve the training with subspace learning.

Let x be an image. Similar to FaceNet [10], we map the sample x to a d-dimensional embedding with $L2$-normalization, as shown in Fig.2(a), and this gives the representation $f(x) \in \mathbb{R}^d$ that satisfies $\|f(x)\|^2 = 1$. Let x_i^a, x_i^p and x_i^n be the anchor sample, positive sample and negative sample respectively, in which x_i^a and x_i^p have the same identity, while x_i^a and x_i^n come from different identities, *i.e.*, $\mathrm{I}(x_i^a) = \mathrm{I}(x_i^p)$ and $\mathrm{I}(x_i^a) \neq \mathrm{I}(x_i^n)$, wherein $I(x)$ denotes the identity of x.

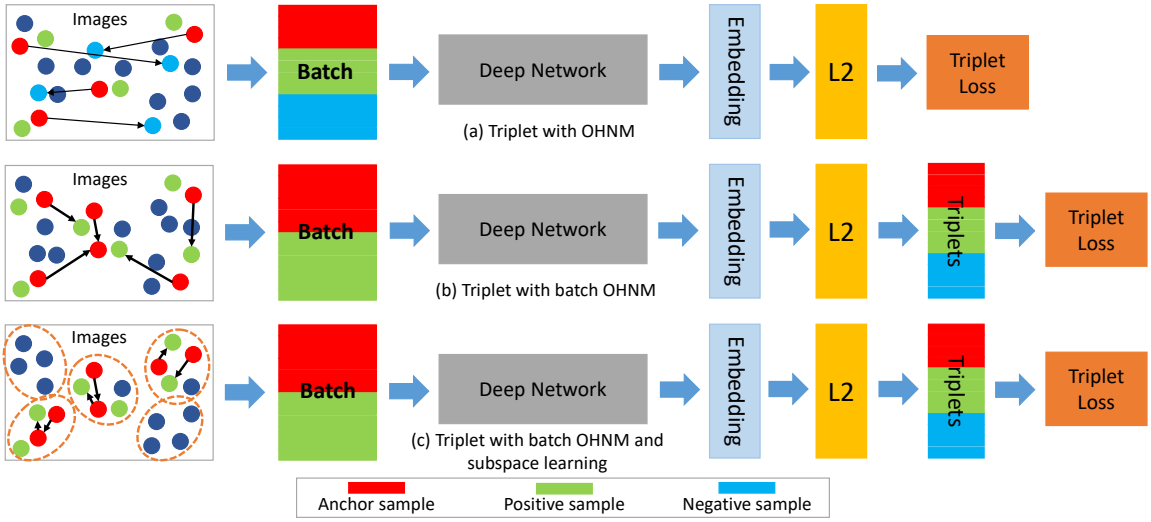

Figure 2. The pipeline of training triplet networks with three methods. (a) Triplet with OHNM; (b) Triplet with batch OHNM; (c) Triplet with batch OHNM and subspace learning.

3.1. Triplet with OHNM

Online Hard Negative Mining(OHNM) [1, 7, 10, 13] is proposed to only focus on hard triplets for training, and the triplet loss with OHNM can be formulated as minimizing the following loss

$$\sum_{i=1}^{|B|}\left[\left\|f\left(x_i^a\right)-f\left(x_i^p\right)\right\|^2-\left\|f\left(x_i^a\right)-f\left(x_i^n\right)\right\|^2+\alpha\right]_+, \quad \forall x_i^a, x_i^p, x_i^n \in T \tag{1}$$

wherein $|B|$ is the batchsize, B is the batch with $3|B|$ images sampled from the image space T, and α denotes the margin enforced between positive and negative pairs. In Eqn.1, hard triplets are the ones that violate the margin constraint. However, due to x_i^n is randomly sampled from all identities, it is difficult to generate effective hard triplets. Fig.2(a) gives an illustration, where most sampled x_i^n(light blue circles) are far away from the positive pairs. Though the hard negative ones are in the batch, they cannot be selected, *e.g.*, some dark blue circles.

3.2. Triplet with Batch OHNM

To fully exploit the batch, some researchers propose the batch OHNM [4] to explore more negative samples. Instead of sampling x_i^n from the whole image space T, batch OHNM finds x_i^n in the batch, and the loss with batch OHNM is minimized as follows

$$\sum_{i=1}^{|B|}\left[\left\|f\left(x_i^a\right)-f\left(x_i^p\right)\right\|^2-\left\|f\left(x_i^a\right)-f\left(x_i^n\right)\right\|^2+\alpha\right]_+ \quad s.t.\ x_i^n=\arg\min_x \left\|f\left(x_i^a\right)-f\left(x\right)\right\|^2,\ \mathrm{I}\left(x_i^a\right)\neq \mathrm{I}\left(x\right) \quad \forall x_i^a, x_i^p \in T,\ x\in B \tag{2}$$

Different from Eqn.1, x_i^n is selected with respect to the Euclidean distance to each anchor x_i^a. In training, we randomly sample $|B|$ different identities, wherein x_i^a and x_i^p are also randomly sampled in each identity, thus there are $2|B|-2$ possible x_i^n for each x_i^a. Batch OHNM is more advantageous not only for the harder negatives, but also more triplets can be used for training as x_i^n is no longer the input. Similar to FaceNet [10], we consider several nearest x_i^n but not the most nearest one, because it might lead to poor training as mislabelled and poorly imaged faces would dominate x_i^n. Fig.2(b) gives an illustration, in which some x_i^n are much closer to positive pairs, and more hard triplets can be used to accelerate training, *e.g.*, 6 triplets in Fig.2(b) compared to 4 triplets in Fig.2(a).

3.3. Triplet with Subspace Learning

The effectiveness of hard negative mining comes from its ability to handle ambiguity in recognizing similar identities, and this indicates the hard triplets should better be generated from similar identities. However, in OHNM and batch OHNM, all identities are used to randomly sample the batch, which cannot be guaranteed to include similar identities. Especially in the large-scale case with $100K$ identities, sampling similar identities with an usual batchsize such as 128 or 256 can be rather difficult.

To find similar identities, the basic idea is to get identity representation and cluster on all identities to generate subspaces, wherein identities can be similar. We achieve this with a classification model, which can be pretrained on a subset of the whole training set. Let $x_i^c\ (\forall i=1,...,N_c)$ be an image with the identity c, and N_c is the number of images in that identity. Then, the representation of x_i^c extracted by the classification model is denoted as $g\left(x_i^c\right)$, and the identity representation $g\left(c\right)$ is given by

$$g(c) = \sum_{i=1}^{N_c} g(x_i^c) \Big/ N_c,\ \forall c = 1, ..., C. \tag{3}$$

Then, we apply k-means clustering on all the identity representation $[g(1), ..., g(C)]$ to generate M subspaces, and each subspace is denoted as $T_m\ (\forall m = 1, ..., M)$, as the dotted circles shown in Fig2(c).

To accelerate training, we refer to [7] that uses the pretrained classification model as initialization. With batch OHNM applied in each subspace iteratively, the triplet loss with subspace learning can be minimized as

$$\begin{aligned} &\sum_{i=1}^{|B|} \left[\|f(x_i^a) - f(x_i^p)\|^2 - \|f(x_i^a) - f(x_i^n)\|^2 + \alpha \right]_+ \\ &s.t.\ x_i^n = \arg\min_x \|f(x_i^a) - f(x)\|^2,\ \mathrm{I}(x_i^a) \neq \mathrm{I}(x) \\ &\forall x \in B,\ x_i^a, x_i^p \in T_m,\ m = 1, ..., M \end{aligned} \tag{4}$$

Different from the single batch OHNM, the batch is randomly sampled in T_m with similar identities, thus the selected x_i^n can be much harder to generate more effective hard triplets. Fig.2(c) illustrates this process, wherein hard negative samples are very close to the positive pairs in a subspace(dotted circle). Though some time is cost in feature extraction and clustering, the subspace learning can largely reduce the searching space and training time. Particularly, similar to batch OHNM, several nearest x_i^n are considered to avoid bad local minima.

3.4. Some Discussions

In the subspace learning, the selection of identity representation and the number of subspaces are important. In this part, we give some discussions on them.

For the selection of $g(c)$ in Eqn.3, we use the average of all image representation in that identity as the identity representation. Due to the large variations in viewpoint, illumination and expression, the average operation can remove the individual difference and extract the common characteristics of an identity. Actually, $g(c)$ can be considered as a cluster center in the k-means clustering, but with only one center in each identity.

For the number of subspaces M, it cannot be too large or too small, *i.e.,* each subspace should contain a moderate number of identities. If M is too small, the whole image space only has a rough division, thus many dissimilar identities will belong to one subspace and not enough hard triplets can be found. If M is too large, many small subspaces that contain only a few identities will be generated. However, the similarity cannot be guaranteed to be effective as the identity representation is only given by the pretrained classification model, which is not reliable enough to determine the similarity. As a result, the fine space division will lead to the over-fitting, which will also give inferior performance. In our experiments, we validate that each subspace having about $10k$ identities is appropriate.

4. Cleaning and Retrieval

Except for the triplet network, how to deal with noisy data and large-scale retrieval are also challenging problems. In this part, we propose two tricks to handle them, and use the tricks on the MS-Celeb-1M competition. Finally, we give the algorithm pipeline as a short summary.

4.1. Noise Removing

In $Challenge1$ of the MS-Celeb-1M competition [2], there are many mislabelled images throughout the dataset. We observe that except for some identities that are very noisy, most identities only have a small number of noisy faces. As clean data dominates, Sukhbaatar *et al.* [8] show that CNN can be robust to a small number of noise. Based on the evidence, we propose to clean the data with three steps as follows: (1) we use all the data to train an initial classification model; (2) the initial model is used as feature extractor to remove the noise; (3) a new classification model is re-trained with the clean data. For the removing step, we adopt a simple trick that only keeps the images with the same predicted identity and labeled identity. Fig.3 illustrates the removing with three steps.

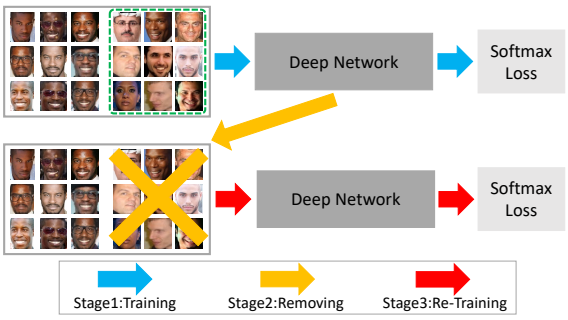

Figure 3. An illustration of the noise removing with three steps.

4.2. Large-Scale Retrieval

As the number of training samples is reduced from $8.4M$ to $5M$ after the cleaning, retrieving a test sample is still challenging. Suppose there are C identities with N samples in the training set. Directly computing the Euclidean distance to all the training samples gives the complexity of $O(N)$, but this is infeasible as N is extremely large, and it will take long even with GPU computation. To retrieve efficiently, we propose a two-layer hierarchical retrieval with the help of identity representation.

Given a test sample, the basic idea is to determine its identity at first, then the training images in that identity are used for retrieval, as shown in Fig.4. In this way, the complexity can be reduced to only $O(C + N/C)$, wherein N/C is the average number of samples in each identity. Since there are about $100K$ identities, *i.e.*, $C \ll N$, this

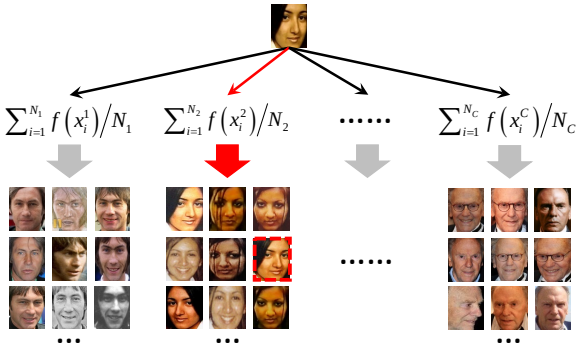

Figure 4. An illustration of the proposed two-layer hierarchical retrieval process with identity representation.

hierarchical trick can largely accelerate the retrieval. Different from the identity representation in Eqn.3 that uses a pretrained classification model, we adopt the triplet network for representation, which is given by

$$f(c) = \sum_{i=1}^{N_c} f(x_i^c) \Big/ N_c, \ \forall c = 1, ..., C. \qquad (5)$$

4.3. Algorithm Pipeline

As a short summary, we give the pipeline of the triplet with subspace learning in Alg.1. The pipeline mainly contains three parts: (1) Initialization, which is also the noise removing process; (2) Training, which trains triplet networks with subspace learning; (3) Testing, which is the two-layer hierarchical retrieval.

Algorithm 1 The pipeline of triplet with subspace learning.

Initilization:
1: Set the margin α and the number of subspaces M;
2: Train an initial classification model on all data or subset;
3: Remove noisy images by the initial model;
4: Re-train a classification network on clean data;
Training:
5: Extract image representation $g(x_i^c)$ by the new classification model;
6: Generate identity representation $g(c)$ for all identities;
7: Generate M subspaces on all identities with k-means clustering;
8: Train triplet networks with subspace learning and batch OHNM;
Testing:
9: Extract image representation $f(x_i^c)$ for all images;
10: Generate identity representation $f(c)$ for all identities;
11: Given a test sample, use the two-layer hierarchical retrieval;

5. Experimental Evaluation

In this section, we give the evaluation of the proposed method. We first introduce the experimental setup in detail, then show the main results of training triplet networks, data cleaning and large-scale retrieval.

5.1. Experimental Settings

Database: We use two datasets in experiments, including MS-Celeb-1M [2] and LFW [6]. The MS-Celeb-1M consists of three parts: training set, development set and test set. Firstly, in the training set, there are 99891 identities with about $8.4M$ images, while data cleaning reduces the number to $5M$. Then, in the development set, there are two subsets based on different recognition difficulty: random set(easy samples) and hard set(hard samples), each of which has 500 images for model selection. Finally, in the test set, there are $50K$ images with only 75% identities contained in the training set, and it is used for final evaluation. For fair comparison with other methods, we also give results on LFW, wherein the test set has 6000 image pairs with each pair having the same identity or not.

Evaluation: For the evaluation of MS-Celeb-1M [2], assume there are N images in the development or test set. If an algorithm recognizes M images among which C images are correct, the precision and coverage will be calculated as $Precision = C/M$ and $Coverage = M/N$ respectively. By varying the confidence threshold, the coverage when $precision = 0.95 \ or \ 0.99$ can be determined. For the evaluation of LFW [6], the accuracy is given by how many pairs are correct in the 6000 pairs.

Pre-processing: In training, we use the same preprocessing in all the networks. Given an training image, we first resize it to 256×256, then a sub-image with 224×224 is randomly cropped and flipped. Particularly, we use no mean subtraction or image whitening, as we put a batch normalization layer right after the input data to learn the normalization parameters. In the testing phase, both training and testing images are resized to 224×224 and flipped, then the average of the original and flipped image representation is considered as the final representation.

Network and Training: To learn the large number of identities, we use the popular ResNet-50 [3] network, which is deep enough to handle our problem. We use a single machine with 4 Titan X in training, and the batchsize is set to be 336 and 160 in the classification and triplet network respectively. Particularly, the Nesterov Accelerated Gradient(NAG) is adopted for optimization, which is found to converge much quickly than SGD.

Parameter Setup: For the learning rate, 0.1 is used for the classification network that is trained from scratch; while for the triplet network, 0.01 is used to fine-tune the classification model. The training of both networks ends with the rate of 0.0001, and 20 epochs are used in each rate to fully converge. Then, for the number of subspaces(M in Eqn.4), we set $M = 10$ to include about $10K$ identities in each subspace. Finally, we set the margin α by cross-validation, *i.e.*, $\alpha = 0.4$ throughout the experiments.

5.2. Results of Triplet

In this part, we give an evaluation of the triplet network with subspace learning and batch OHNM. Table.5.2 shows the LFW accuracy of the networks trained with softmax

loss and triplet loss. Particularly, all the triplet networks are initialized with the pretrained classification model, and the ones with "+*Softmax*" are fine-tuned jointly with softmax loss and triplet loss.

Method	LFW Acc(%)
Softmax (Baseline, $100K$)	99.36
Triplet+Batch OHNM ($100K$)	98.98
Triplet+Batch OHNM+Random-Subspace ($100K$)	99.08
Triplet+Batch OHNM+Subspace-5 ($100K$)	99.25
Triplet+Batch OHNM+Subspace-10 ($100K$)	**99.38**
Triplet+Batch OHNM+Subspace-20 ($100K$)	99.33
Triplet+Batch OHNM + Softmax ($100K$)	99.33
Triplet+Batch OHNM+Random-Subspace + Softmax ($100K$)	99.35
Triplet+Batch OHNM+Subspace-5 + Softmax ($100K$)	99.38
Triplet+Batch OHNM+Subspace-10 + Softmax ($100K$)	**99.48**
Triplet+Batch OHNM+Subspace-20 + Softmax ($100K$)	99.41
FaceNet [10] (Triplet+OHNM, $8M$)	99.63

Table 1. The LFW accuracy of the baseline classification network and different triplet networks.

We have five main observations. Firstly, compared to the classification model, the performance of the triplet network with OHNM has decreased a lot, *e.g.*, from 99.36% to 98.98%. Due to our single machine can only hold small batchsize, the chance of sampling similar identities from $100K$ identities with batch OHNM is very tiny. As a result, the network cannot generate enough hard triplets, and it will be trained to fit the semi-hard triplets, which will result in the drop of performance.

Secondly, the performance of the triplet network with subspace learning increases a little, *e.g.,* from 99.36% to 99.38%. With enough hard triplets generated, the model is able to overcome the ambiguity in recognizing similar identities. However, the improvement is not obvious as we observe the softmax loss increases a lot in training, which indicates that training with only triplet loss will harm the identity information.

Thirdly, we also compare the triplet network with random subspace, named as *Triplet+Batch OHNM+Random-Subspace*, which divides the whole image space randomly. Table.5.2 shows that *Random-Subspace* has obtained a slight improvement over the single *Batch OHNM*, *e.g.*, from 98.98% to 99.08%, but it is still far below the subspace learning. This result implies that some similar or semi-similar identities may be luckily put together, and the searching space can be reduced a little. However, compared to the cluster based subspace learning, not enough hard triplets can be generated.

Fourthly, we evaluate the influence of the number of subspaces M. Three values are tested: 5, 10 and 20, and their corresponding models are denoted as $Subspace-5/10/20$ respectively. Table.5.2 shows that setting $M = 10$ achieves the best performance, *i.e.,* 99.38% for *Triplet+Batch OHNM+Subspace-10*, and this result is just as expected. As analyzed in Sec.3.4, small M can only give rough subspaces, each of which contains many dissimilar identities; large M will give too fine division, but this division is only given by the baseline *Softmax* that is not reliable enough, thus over-fitting may happen.

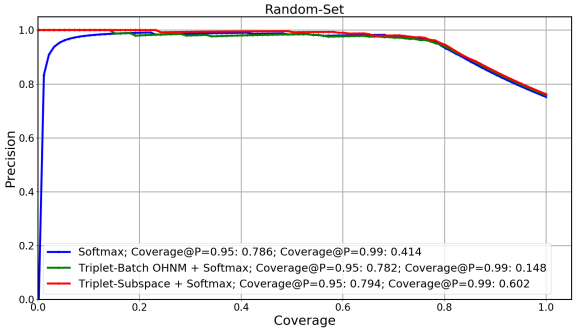

Figure 5. The performance of single classification or triplet models on the random set, which belongs to the development set of $Challenge1$ without external data. Best viewed in color.

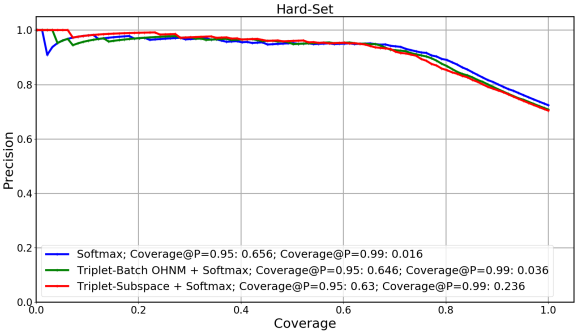

Figure 6. The performance of single classification or triplet models on the hard set, which belongs to the development set of $Challenge1$ without external data. Best viewed in color.

Finally, compared to the networks trained with only triplet loss, the one trained with additional softmax loss obtains promising improvements, *e.g.*, 99.38% to 99.48% in subspace learning, which can be competitive to FaceNet's 99.63% [10] with $8M$ identities. The joint training with softmax and triplet loss is more advantageous because softmax loss can preserve inter-class information while triplet loss can preserve intra-class information, thus the jointly trained model can be more robust.

Fig.5 and Fig.6 show the performance of classification and triplet networks on the random and hard set respectively, and the performance is given by the coverage under the precision of 0.95 and 0.99. It can be observed that *Triplet-Subspace+Softmax* obtains a large improvement over the one with only batch OHNM, especially the coverage under the precision of 0.99, *e.g.*, from 0.148 to 0.602 in the random set and from 0.036 to 0.236 in the hard set. Besides, the performance in the random set is much higher than the one in the hard set, and this is reasonable as the samples in the hard set are more difficult.

5.3. Results of Multi-Model

To give the best performance, we use two models for evaluation: the baseline classification model(*Softmax* in Table.5.2), and the jointly trained triplet model (*Triplet+Softmax* in Table.5.2). We adopt a simple combination that averages the image representation of both models, thus the final representation for the image and identity is given as

$$\frac{g\left(x_i^c\right)+f\left(x_i^c\right)}{2}, \quad \frac{g\left(c\right)+f\left(c\right)}{2}. \tag{6}$$

Fig.7 and Fig.8 show the performance of multi-models on the random and hard set respectively, and the performance is given by the coverage under the precision of 0.95 and 0.99. Particularly, *Multi-Batch OHNM* is the combination of *Softmax* and *Triplet+Batch OHNM+Softmax*, while *Multi-Subspace* consists of *Softmax* and *Triplet+Batch OHNM+Subspace-10+Softmax*.

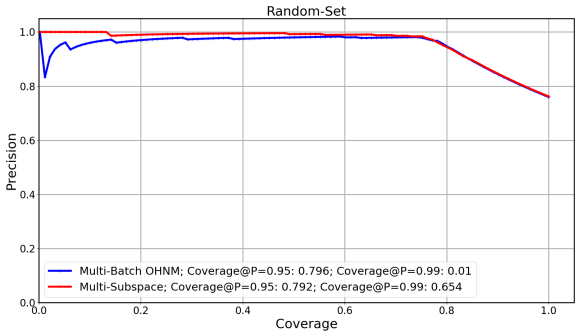

Figure 7. The performance of multi-models on the random set, which belongs to the development set of $Challenge1$ without external data. Best viewed in color.

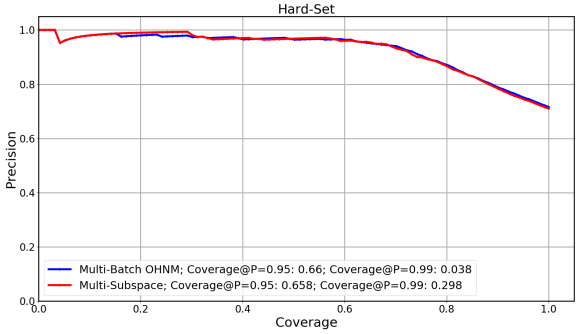

Figure 8. The performance of multi-models on the hard set, which belongs to the development set of $Challenge1$ without external data. Best viewed in color.

It can be observed that *Multi-Subspace* has improved a lot over the single models in Fig.5, *e.g.*, the coverage increases from 0.602 to 0.654 in the random set and from 0.236 to 0.298 in the hard set under the precision of 0.99. This demonstrates that the inter-class and intra-class information learned in the classification and triplet network respectively can be complementary to enhance the recognition ability. However, we do not see the same improvement for *Multi-Batch OHNM*.

Compared to the single models in Fig.5, the performance of *Multi-Batch OHNM* has decreased a lot, *e.g.*, the coverage decreases from the baseline 0.414 to 0.01 under the precision of 0.99 in the random set, while the coverage in the hard set remains basically the same. In Fig.5, the coverage of *Softmax* has dropped a lot as the confidence score increases, which implies that the classification model is not confident to differentiate similar identities, while it is more confident to recognize semi-similar identities as precision is high when the coverage ranges from $0.2 \sim 0.6$. This comes from the fact that *Softmax* focuses more on the inter-class information, but misses details in similar identities. Compared to *Softmax*, *Triple-Batch OHNM+Softmax* performs much more confident as high precision can be achieved under a high confidence score, but the precision is lower than *Softmax* when the converge ranges from $0.2 \sim 0.6$, and this may be caused by the fact that the triplet network with batch OHNM focuses more on the hard triplets. As a result, the average of the two models increases the precision in low coverage and decreases the precision in mid-level coverage, which results in the large drop of performance under the precision of 0.99.

TeamName	Data	Cov@P=0.95	TeamName	Data	Cov@P=0.95
Orion	Aligned	**0.75**	**Orion**	Aligned	**0.606**
DRNfLSR	Aligned	0.734	CIGIT_NLPR	Aligned	0.534
ITRC-SARI	Aligned	0.707	DRNfLSR	Aligned	0.486
CIGIT_NLPR	Aligned	0.684	faceman	Aligned	0.33
ms3rz	Aligned	0.646	ms3rz	Aligned	0.26
1510	Aligned	0.57	FaceAll	Aligned	0.254
FaceAll	Aligned	0.554	BUPT_PRIS	Aligned	0.21
faceman	Aligned	0.461	IMMRSB3RZ	Aligned	0.042
BUPT_PRIS	Aligned	0.421	BUPT_MCPRL	Cropped	0.04
IMMRSB3RZ	Aligned	0.171	CIIDIP	Aligned	0.02
CIIDIP	Aligned	0.154	ITRC-SARI	Aligned	0.004
BUPT_MCPRL	Cropped	0.064	DS_NFS	Aligned	0.001
NII-UIT-KAORI	Aligned	0.001	1510	Aligned	0.001
DS_NFS	Aligned	-	Paparazzi	Aligned	-
Paparazzi	Aligned	-	NII-UIT-KAORI	Aligned	-

Table 2. The final results on the random set(left) and hard set(right) in $Challenge1$ without using external data. The performance is given by the coverage under the precision of 0.95.

For the final evaluation, we adopt the *Multi-Subspace* for result submission. Table.5.3 shows the final results of our method(Orion) and other teams on the random set(left) and hard set(right) in $Challenge1$ without using external data, and the performance is given by the coverage under the precision of 0.95. It can be clearly observed that our method has achieved the state-of-the-art performance in both the random set and hard set, and the coverage in the hard set has improved a lot over other teams, *e.g.*, from 0.534 by *CIGIT_NLPR* to our 0.606. Particularly, in the random set,

we have obtained the coverage of 0.75, which is the upper limit in $Challenge1$ without using external data as only 75% training identities are included in the test set. This result is surprising because we have obtained 100% recall on the training identities under the precision of 0.95. Table.5.3 also shows the final results on the random and hard sets using the external data. Even though no external data is used in our method, we can achieve competitive performance to some teams, *e.g.*, the *SmileLab*.

External	Team Name	Data	Cov@P=0.95_Random	Cov@P=0.95_Hard
Yes	Panasonic-NUS	Aligned	0.875	0.791
Yes	Turtle	Aligned	0.862	0.73
Yes	SmileLab	Aligned	0.792	0.61
No	Orion	Aligned	0.75	0.606
Yes	D**	Cropped	0.745	0.454
Yes	BMTV	Cropped	0.724	0.409
Yes	FaceSecret	Aligned	0.641	0.002

Table 3. The final results on the random set and hard set in $Challenge1$ with external data. The performance is given by the coverage under the precision of 0.95.

5.4. Results of Cleaning

In this part, we give an evaluation of the noise removing method. Table.5.4 shows the LFW accuracy of three different noisy removing methods on MS-Celeb-1M, and a classification model with softmax loss is used to remove noise. *Original Data* uses all the data in training; while *Fixed Ratio* uses other clean datasets to train a model as feature extractor, which keeps a fixed ratio of images in each identity, and this is used in last year's competition. It can be observed that our removing method obtains the LFW accuracy of 99.36%, which is much higher than other methods, and Fig.3 shows an example. Besides, we see that directly using all the data in training is even better than *Fixed Ratio*, and this indicates CNN can be robust to a small number of noises, as demonstrated in [8]. Based on our observation, different identities have different number of noises. As a result, *Fixed Ratio* will remove many clean samples for clean identities and keep many noisy samples in noisy identities, thus it may fail.

Method	Original Data	Fixed Ratio	Ours
LFW Acc(%)	98.96	98.85	99.36

Table 4. The LFW accuracy based on three different noisy removing methods on the training set of MS-Celeb-1M.

5.5. Results of Retrieval

In this part, we give an evaluation of the retrieval efficiency. In testing, we have 99891 identities with about $5.04M$ images in the retrieval set, and the objective is to determine the identity for a given test image. Table.5.5 gives the time cost(s) of retrieving one image with different retrieval methods. *All* directly calculates the Euclidean distance to all samples, $2-Hierarchy$ denotes our two-layer hierarchical retrieval trick, and CPU/GPU are the different hardware implementations. It can be observed that with the CPU implementation, our trick can largely accelerate the retrieval by $69\times$. This is as expected as All needs $5.04M$ multiplications, while $2-Hierarchy$ only needs $99891+50$ multiplications, which can save a lot of time. To further accelerate the retrieval, we implement matrix multiplication by GPU and the time is reduced to only $0.052s$ for each image, which is very efficient. In the final evaluation, we adopt $2-Hierarchy$ with GPU to finish the testing with $50K$ images in about 1 hour(including feature extraction and retrieval).

Figure 9. Results of noise removing in two identities. The images in the green dotted rectangles are noise faces.

Method	CPU Time(s)	GPU Time(s)
All	32.09	4.646
2-Hierarchy	0.464	0.052

Table 5. The time cost(s) of retrieving one image with different retrieval methods and CPU/GPU implementations.

6. Conclusion

In this paper, we take face recognition as a breaking point and study how to train triplet networks with large-scale data. Firstly, to find similar identities for more effective hard triplets, we have proposed the triplet with subspace learning, and this is our major contribution. Then, to handle noisy faces and large-scale retrieval, we propose two tricks that use a three-step noise removing trick and a two-layer hierarchical retrieval trick. Combined with the subspace learning, we have achieved the state-of-the-art performance on the MS-Celeb-1M $Challenge1$ without external data. In our future work, we will study triplet in a larger scale, *e.g.*, $1M$ or $10M$ identities.

Acknowledgement

We would like to thank Huajiang Xu and Yangang Zhang for their discussions and supports on data processing and computing architectures.

References

[1] W. Chen, X. Chen, J. Zhang, and K. Huang. A multi-task deep network for person re-identification. In *AAAI*, 2017.

[2] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *ECCV*, 2016.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[4] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification. In *arXiv:1703.07737*, 2017.

[5] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *ICLR*, 2015.

[6] E. Learned-Miller, G. B. Huang, A. RoyChowdhury, and G. Hua. Labeled faces in the wild: A survey. In *Advances in Face Detection and Facial Image Analysis*, 2016.

[7] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *BMVC*, 2015.

[8] M. P. L. B. R. F. S. Sukhbaatar, J. Bruna. Training convolutional networks with noisy labels. In *ICLR Workshop*, 2015.

[9] S. Sankaranarayanan, A. Alavi, C. Castillo, and R. Chellappa. Triplet probabilistic embedding for face verification and clustering. In *ICLR*, 2015.

[10] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[11] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[12] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, 2016.

[13] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *CVPR*, 2014.

[14] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*, 2015.

[15] B. Zhuang, G. Lin, C. Shen, and I. Reid. Fast training of triplet-based deep binary embedding networks. In *CVPR*, 2016.