Hard Example Mining with Auxiliary Embeddings

Evgeny Smirnov
Speech Technology Center
smirnov-e@speechpro.com

Aleksandr Melnikov
ITMO University
melnikov_a@corp.ifmo.ru

Andrei Oleinik
ITMO University
aoleinik@corp.ifmo.ru

Elizaveta Ivanova
Speech Technology Center
ivanova-e@speechpro.com

Ilya Kalinovskiy
Speech Technology Center
kalinovskiy@speechpro.com

Eugene Luckyanets
ITMO University
152974@niuitmo.ru

Abstract

Hard example mining is an important part of the deep embedding learning. Most methods perform it at the mini-batch level. However, in the large-scale settings there is only a small chance that proper examples will appear in the same mini-batch and will be coupled into the hard example pairs or triplets. Doppelganger mining was previously proposed to increase this chance by means of class-wise similarity. This method ensures that examples of similar classes are sampled into the same mini-batch together. One of the drawbacks of this method is that it operates only at the class level, while there also might be a way to select appropriate examples within class in a more elaborated way than randomly. In this paper, we propose to use auxiliary embeddings for hard example mining. These embeddings are constructed in such way that similar examples have close embeddings in the cosine similarity sense. With the help of these embeddings it is possible to select new examples for the mini-batch based on their similarity with the already selected examples. We propose several ways to create auxiliary embeddings and use them to increase the number of potentially hard positive and negative examples in each mini-batch. Our experiments on the challenging Disguised Faces in the Wild (DFW) dataset show that hard example mining with auxiliary embeddings improves the discriminative power of learned representations.

1. Introduction

Deep embedding learning is a powerful set of techniques to learn representations for various problems like face verification and recognition [59, 51, 55], speaker recognition [40], anti-spoofing for speaker recognition systems [24], image retrieval [69, 37, 60], person [70] and vehicle [3] re-identification, fine-grained recognition [13]. To be useful, deep embeddings are trained to obtain some required properties. Embeddings corresponding to the objects from the same class should be close to each other (according to some metric), while different classes should be separated. Embeddings with this property are called discriminative. To train embeddings in such manner one should consider using appropriate loss functions, deep neural network architecture and example mining method.

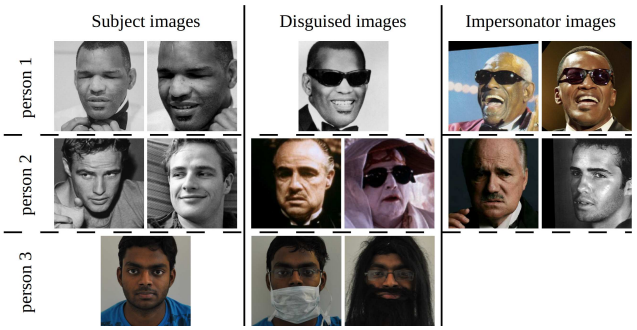

Figure 1. Examples of subject, disguised and impersonator images from DFW dataset. Some people do not have impersonator images (e.g. person 3 on the figure).

There are several ways to train deep embeddings. One of them is to use example-to-example distances in the embedding space. Instead of using single examples, this approach considers pairs [8], N-pairs [56], triplets [51, 50], quadruplets [19, 7] or some other structures [41], constructed from the groups of examples. Random sampling is not appropriate in this case because the major part of these structures will consist of easy examples and result in poor gradients. Thus, examples should be sampled in a more intelligent fashion. Some kind of hard example mining is usually used for this purpose. There are two main types of hard example mining found in literature: *mini-batch level hard example mining* [51, 69] and *hard class mining* [56, 55]. Mini-batch level hard example mining operates inside the constructed mini-batch and selects appropriate examples to

create good pairs, triplets or other structures. Hard class mining operates on the class level and is used to construct mini-batches by sampling examples using "hard class pairs" from the dataset.

In this paper, we propose hard example mining with auxiliary embeddings — a novel method, which is used to construct mini-batches, containing a large numbers of hard examples.

2. Related work

Appropriate loss functions and hard example mining are essential for learning useful deep embeddings. In this section we review some of them.

2.1. Loss functions

There are two main types of loss functions suitable for embedding learning: *prototype-based* and *exemplar-based* loss functions. The first one uses prototype objects of some kind to represent classes in the dataset. Training with this type of loss functions usually performed through the classification task (though there are other ways of prototype usage). Each training example is compared with class prototypes and assigned to some class. Classification error is backpropagated through the network to adjust the weights. This kind of loss function could be seen as "global" because it uses global embedding space. The models with this type of supervision are usually very fast to train, do not require complicated example mining schemes and provide good embeddings, especially when there are a lot of examples for each class [17]. However, a significant amount of pairwise example distance information, which is presented in the dataset, is not used. Also, loss functions of this kind are usually computationally expensive when the number of classes is large and not very useful for the datasets with small number of examples per class. The most popular loss function of this kind is Softmax loss. However, better results in deep embedding learning tasks are achieved with its variants. For example, L-Softmax [29], which includes margin into softmax, encourages intra-class compactness. L_2-Softmax [46] introduces feature normalized version of the Softmax for the same purpose. NormFace [63] and COCO Loss [31, 32] normalize both features and classifier weights, providing a way to optimize for cosine similarity directly. SphereFace [28], Soft-Margin Softmax [27], CosFace [64] and ArcFace [10] introduce different kinds of soft, angular and cosine margins to Softmax Loss, improving discriminative properties of the resulting embeddings. Another approaches include Selective Softmax [77], Noisy Softmax [6], L-GM Loss [61], Correlation Loss [11] and Orthogonal Low-rank Embedding [25].

Other variants of prototype-based losses use class centroids or some kind of clustering to obtain prototypes and to encourage examples to be close to corresponding prototypes in the embedding space. Losses of this kind include Center Loss [67] and Contrastive-Center Loss [45], which use class centroids as prototypes. Another one is Magnet Loss [48], which introduces prototypes in the form of several cluster centroids for each class. "No Fuss Metric Learning" [37] uses "proxies" in the same fashion. Other techniques include [20, 68, 57, 5, 15, 13, 3, 36]. Some of them use triplets on the exemplar-prototype or prototype-prototype level and can be seen as hybrid of prototype-based and exemplar-based loss functions.

The exemplar-based loss functions use example-to-example distances to train deep embeddings. Usually this is achieved by using some kind of example structures like pairs [8] or triplets [51], introducing pairwise distances between examples. The network with such loss functions is trained to produce discriminative embeddings. The loss functions of this kind include Contrastive Loss [8], operating with pairs of examples, Triplet Loss [51, 42, 50, 62], operating with triplets of examples, and several variants of quadruplet losses [19, 60]. Other approaches include margin-based losses [69, 55, 70] and angular variants of triplets [65]. Some losses operate with more complex example structures like N-pair Loss [56], (N+M)-tuplet cluster loss [30] and Lifted Structured Embedding [41].

Exemplar-based loss functions are especially useful for training on datasets with large number of classes and small number of examples per class. Even if a class has only one example, it still can be paired with all other examples in the dataset, creating a large number of training pairs and even more triplets. Moreover, exemplar-based losses use the information hidden in the pairwise distances between examples (which is lost in the case of prototype-based losses). However, the large number of possible pairs or triplets can slow down training with exemplar-based loss because not all of them are useful for training, especially at the later stages, when the network is already well trained. Thus, to fully use the potential of exemplar-based training some kind of hard example mining should be applied. Also it is possible to use joint supervision with prototype-based loss [58, 55], benefiting from both prototype and exemplar-based training.

2.2. Hard example mining

One way to find hard examples for exemplar-based training is to use online hard example mining [19, 70] and select hardest pairs, triplets or quadruplets on the mini-batch level. However, in practice it is better to use online semi-hard example mining [51, 42, 41], when example pairs are chosen at random from the "hard enough" pairs in the mini-batch i.e. the pairs with the distance exceeding some margin. Also, not only hard pairs contain useful information [78], so there are several ways to find another useful pairs. One way to do this is to use distance-weighted sampling

on the mini-batch level, so the hard examples are sampled along with "regular" ones [69]. Utilization of different levels of "hardness" has also proved to be beneficial [74]. All these methods improve exemplar-based training, however, they all heavily rely on the presence of the hard examples in the mini-batch. If the dataset is small and the mini-batch is large, it is likely to happen. However, if the dataset is large in terms of the number of classes and the number of images, and the potential mini-batch size is small e.g. due to GPU memory limitations, this assumption is no longer valid. Thus, there is a need to use some global, dataset-level hard example mining approach to create mini-batches, which contain enough hard example pairs.

The problem of hard example mining on the dataset level is well studied, but usually solutions are developed for individual examples, and not for the pairs [14, 34]. There are several recent solutions to this problem for the example pairs case. One way, presented in [56], is to use "hard class mining" to find pairs of classes, which are "hard" with respect to each other, and generate mini-batches, containing examples for both of them. Although this method improves training efficiency, it is computationally intensive and does not scale well to large datasets. The other method is Doppelganger Mining [55]. The main idea of this method is to maintain a list with the most similar identities for each identity in the training set. This list is used to generate better mini-batches by sampling pairs of similar-looking identities ("doppelgangers") together. This method works only on the class level, while the selection of examples, which will be included into the mini-batch, remains random. However, one identity can have both useful and unuseful examples in the dataset, and it would be better to include the former ones into the mini-batch more often. Other methods also work only on class level [77] or require significant computational resources in the case of very large dataset [22].

3. Auxiliary embeddings

In this paper, we propose to use auxiliary embeddings, assigned to each training example in the dataset. They are used to fill a mini-batch with appropriate examples based on previously added to the mini-batch. Auxiliary embeddings are vectors, which possess following properties:

- Each training example has pre-computed auxiliary embedding in the dataset;
- Auxiliary embeddings of two "hard positive" examples are far from each other according to the cosine similarity metric;
- Auxiliary embeddings of two "hard negative" examples are close to each other according to the cosine similarity metric;
- Auxiliary embeddings have relatively low dimensionality (for fast cosine similarity calculation).

These embeddings are used at the mini-batch generation stage (see Figure 2 for the illustration).

The first stage of the mini-batch generation process assumes selection of those classes (identities) which will appear in the mini-batch (Figure 2 (a)). Doppelganger Mining [55] or some other class selection method could be used for this purpose. In the case of Doppelganger Mining several classes are selected randomly, and other classes are selected using doppelganger list. Each class is intended to have some number of examples in the mini-batch (the number is chosen by hyperparameter). Each new example of class x is randomly selected according to one of three strategies. It could be:

- Selected randomly from all examples of class x (Figure 2 (b));
- Selected as the "hard positive" for an example X of class x, which has been earlier added to the mini-batch. (Figure 2 (c)). For this purpose the auxiliary embedding of X is compared using cosine similarity with a random subset of auxiliary embeddings of examples in class x. The example, which has the smallest cosine similarity value, is added to the mini-batch;
- Selected as the "hard negative" for an example Y of class y, which has been earlier added to the mini-batch and is different from the current class x. (Figure 2 (d)). For this purpose the auxiliary embedding of Y is compared using cosine similarity with a random subset of auxiliary embeddings of examples in class x. The example, which has the largest cosine similarity value, is added to the mini-batch. When a certain hard class mining method is used, the best way to find hard negatives is to look at the examples of hard paired classes (doppelgangers in case of Doppelganger Mining). If no such method is applied, then the class y could be selected randomly from the classes, which already have examples in the current mini-batch.

Probabilities of each choice are determined by hyperparameters. After the mini-batch is filled (Figure 2 (e)), it contains a set of good hard examples.

This approach to mini-batch generation benefits both from class-wise (by Doppelganger Mining or some other method of this kind) and example-wise (by auxiliary embeddings) hard positive and negative mining. Further improvement can be achieved with some mini-batch level hard example mining method like in [55].

There are many possible ways to get auxiliary embeddings.

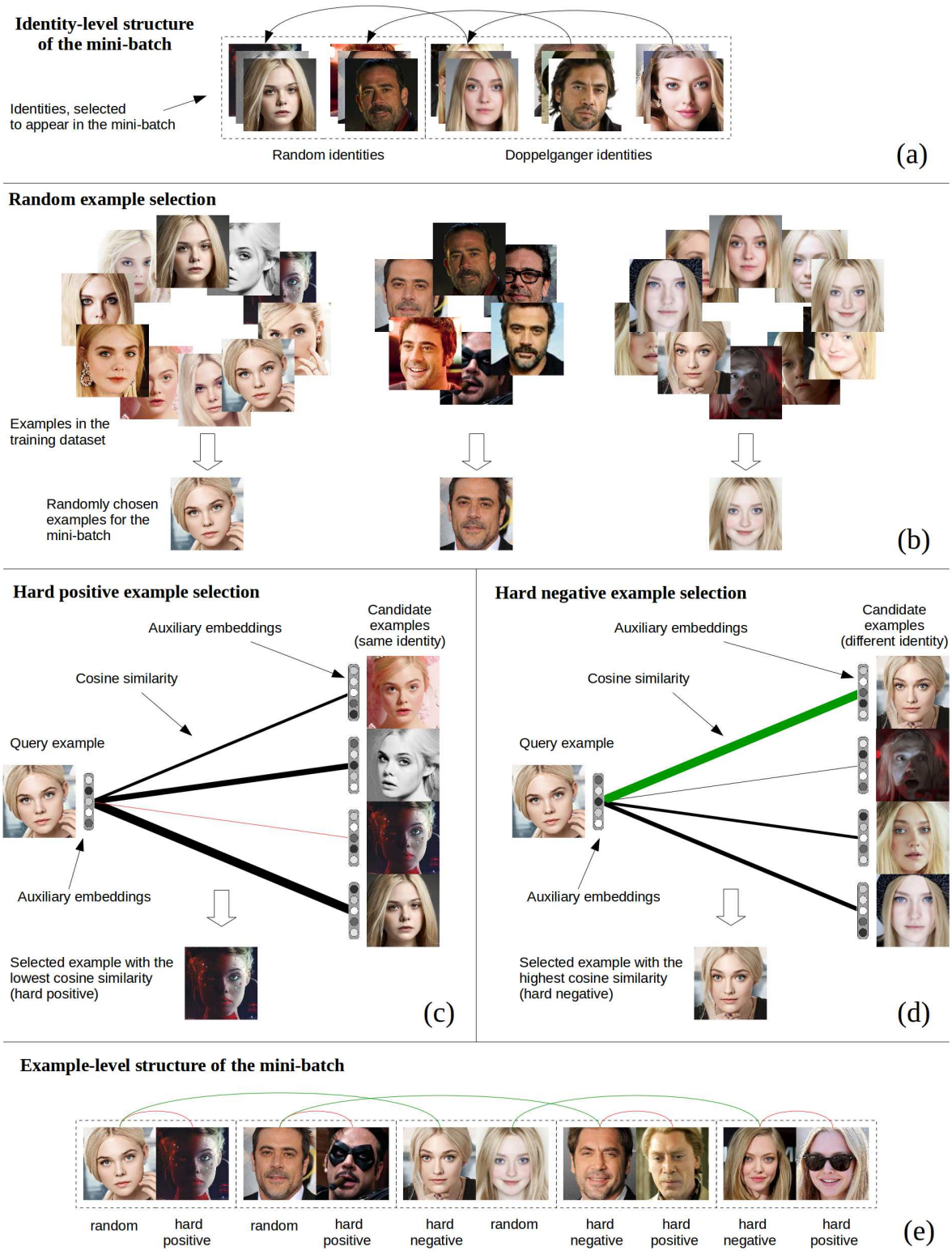

Figure 2. Hard example mining with auxiliary embeddings. a) Identity-level structure of the mini-batch. b) Random example selection. c) Hard positive example selection. d) Hard negative example selection. e) Example-level structure of the mini-batch.

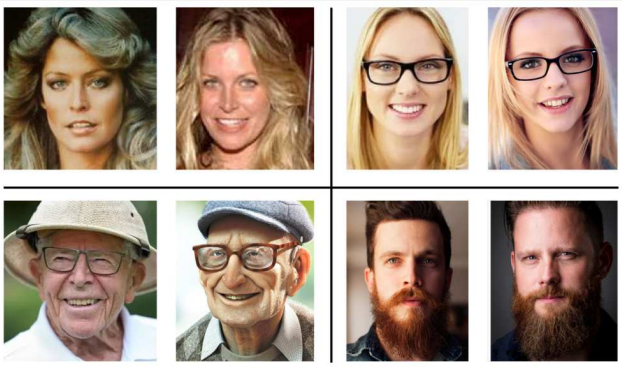

Figure 3. Examples of the images with similar appearance embeddings.

3.1. Appearance embeddings

The first approach is to use some appearance information about training examples. The basic assumption here is that hardness of the examples is related to appearance of images. Images from different classes with similar appearance and images from the same class with different appearance are likely to form hard examples. Adding them to the same mini-batch should help the neural network to ignore some misleading appearance features. In the context of face recognition, it can be achieved by means of a pre-trained attribute classification neural network. For example, this network may evaluate pose, age and ethnicity or some aggregated embedding, containing this information. Figure 3 displays examples, which are close in the appearance embeddings space. The advantage of this type of auxiliary embeddings is that they can be computed independently from the trained deep embedding network and reused for different models.

3.2. Embeddings from previous training stages

The other way to get auxiliary embeddings is to use multistage training schedule [35]. In this case, auxiliary embeddings are not used at the first learning stage. After the first learning stage is finished, the resulting neural network is used to compute the embeddings for the entire training set. This process is repeated after each training stage. Thus, the network improves after each training stage along with the auxiliary embeddings. This approach is more "natural" because the embeddings are optimized precisely to perform well on the target task. However, appearance embeddings can be useful in case of one-stage training (or on the first stage). Moreover, appearance embeddings and the embeddings from previous training stages can be combined.

4. Experiments

In this section, we evaluate proposed hard example mining with auxiliary embeddings on the challenging task of disguised face recognition [23]. Caffe [21] and PyTorch [43] frameworks were used in the experiments.

4.1. Disguised face recognition

In the past three decades, researchers have significantly improved the performance of face recognition methods in various scenarios. Historically, the first (and probably the simplest) practically significant scenario is face recognition in controlled environment. In this case, the person to be recognized collaborates with the system by maintaining neutral face expression and looking directly into the camera. Moreover, the location and orientation of the camera as well as ambient illumination can be adjusted manually to maximize the performance of the recognition system.

The majority of the state-of-the-art approaches deal with another scenario, namely face recognition "in the wild". This scenario assumes that the recognized person does not collaborate with the system and may be unaware of its presence. This leads to the problems of Pose, Illumination and Expression (referred as PIE [52]). Other difficulties may include poor quality of the face images (low resolution, blurring, non-linear distortions) as well as aging of the recognized person.

The Disguised Faces in the Wild (DFW) [23] challenge raises the issue of the disguised faces recognition [44, 12, 54, 71, 47, 73, 75, 26, 66]. Disguise includes significant changes of the face appearance, such as heavy makeup, masks, sunglasses, beard (fake or natural), etc. In the worst case, these changes are made intentionally to hide one's identity or imitate the appearance of another person. The DFW scenario is much harder than regular face recognition "in the wild". Thus, the existing approaches should be adapted and modified to provide acceptable quality in such conditions.

The DFW dataset contains face images for 1000 persons. For each person, there are images of the following types:

- "*Subject*" is a normal face image of the person;
- "*Disguised*" is a disguised face image of the same person;
- "*Impersonator*" is a face image of another person that bears a significant resemblance to the "*subject*" image.

Figure 1 shows examples of images from DFW dataset. The training set includes 400 people and $3,385$ face images; the test set includes 600 people and $7,771$ images. In total, there are $14,131$ genuine and $9,218,712$ imposter pairs. The evaluation metric is genuine accept rate (GAR) at false accept rates (FAR) of 1% and 0.1%.

Description	Output
input image	$255 \times 255 \times 3$
$5 \times 5 \times 32$ Conv, stride 2	$126 \times 126 \times 32$
$3 \times 3 \times 64$ Conv	$124 \times 124 \times 64$
2×2 MaxPool, stride 2	$62 \times 62 \times 64$
$3 \times 3 \times 64$ ResBlock	$62 \times 62 \times 64$
$3 \times 3 \times (96 + 64 + 32)$ Conv, d=1,2,3	$60 \times 60 \times 192$
2×2 MaxPool, stride 2	$30 \times 30 \times 192$
$(3 \times 3 \times 192$ SE-ResBlock, r=16)$\times 2$	$30 \times 30 \times 192$
$3 \times 3 \times (336 + 112)$ Conv, d=1,2	$28 \times 28 \times 448$
2×2 MaxPool, stride 2	$14 \times 14 \times 448$
$(3 \times 3 \times 448$ SE-ResBlock, r=16)$\times 5$	$14 \times 14 \times 448$
$3 \times 3 \times 1024$ Conv	$12 \times 12 \times 1024$
2×2 MaxPool, stride 2	$6 \times 6 \times 1024$
$(3 \times 3 \times 1024$ SE-ResBlock, r=16)$\times 5$	$6 \times 6 \times 1024$
$512 + 512$, fc + maxout, group = 2	512
L_2-normalization	512

Table 1. ARFANet architecture.

4.2. Implementation details

4.2.1 Neural network architecture

In order to carry out the experiments we used neural network architecture, inspired by [55] with such modifications, as Squeeze-and-Excitation blocks [18], different numbers of convolutional filters, using more layers and smaller input image size. We used ARFA [55] as the activation function.

Final face embedding at the training stage is the L_2-normalized vector of dimension 512. At the testing stage, vectors for the original image and its horizontally flipped version are used. These vectors are concatenated and L_2-normalized, resulting in the final vector of dimension 1024. This architecture is called ARFANet and summarized in Table 1.

4.2.2 Preprocessing

The most common face alignment algorithm is based on similarity transformation. It is performed using the eyes position: the image is warped so that the eyes are on desired position on the 224×224 crop pattern. This approach fails when face pose is far from frontal. Therefore, if the estimated face deviation from frontal pose is high, the square bounding box from the face detector is used to create a crop.

In our work MTCNN face detector [76] is adopted for the alignment of the datasets. If a detection from MTCNN is consistent with the markup from dataset, then eyes' coordinates, supplied by the detector, are used for further alignment. If the detector fails to find the proper face, then the bounding box from markup file is used and FAN landmark detector [4] is applied to the corresponding image region. To compensate the noisiness of FAN detector we fit PDM model [9] with 20 DoF from Menpo library [2] to obtained landmarks. This PDM is trained on LS3D-W dataset [4] and allow wide variations of face position.

4.2.3 Dataset

Training is performed on the combined dataset, which consists of public face datasets: MS-Celeb-1M [16], CASIA-WebFace [72], VGG-Face [42], Megaface [38] and DFW training set [23]. The resulting dataset includes $391,187$ identities and $8,365,919$ face images. All DFW test set identities have been removed from the dataset. The DFW training set was added to the combined dataset according to the following algorithm:

- compute a mean embedding for each identity from DFW training set;
- compute a distance from DFW embeddings to all mean identities' embeddings from combined dataset;
- for the current DFW identity:
  - if there is no distance smaller than δ threshold, then this identity is added to the combined dataset as a new one;
  - if there is a distance smaller than ϵ threshold with some identity from combined dataset, then these two identities are merged with each other;
- all DFW identities that have distances between ϵ and δ are manually verified and added to the combined dataset.

Thresholds ϵ and δ are selected experimentally according to accuracy on the corresponding internal test set.

4.2.4 Data augmentation and training

We used the same data augmentation technique as in [55] for all models. The loss function is a weighted combination of Margin-based loss [55] and Working Memory Prototype Loss [1]. We used batch size of 80 examples, 4 example per class. For models with Doppelganger mining (ARFANet-DM), the random class number is set to 2. To perform the hard example mining for models with auxiliary embeddings we used following hyperparameters:

- Probability of random example selection: 0.2;
- Probability of hard positive example selection: 0.4;
- Probability of hard negative example selection: 0.4;
- Maximum number of hard example candidates for the process of hard example selection: $10,000$.

We used multi-stage training schedule with SGDR [35]. The first stage includes $100,000$ iterations, with learning rate decreasing from 0.01 to 0.00001, the second stage includes $200,000$ iterations, with learning rate decreasing from 0.001 to 0.000001, the third stage includes $300,000$ iterations, with learning rate decreasing from 0.001 to 0.000001. We used Nesterov Accelerated Gradient [39] with momentum set to 0.9 and weight decay set to 0.0005. We trained eight models:

- ARFANet: The model without Doppelganger mining and auxiliary embeddings;
- ARFANet + FL: The model without Doppelganger mining, but with auxiliary embeddings, taken from the final (deep embedding) layer of the network;
- ARFANet-DM: The model with Doppelganger mining, but without auxiliary embeddings;
- ARFANet-DM + MOON: The model with Doppelganger mining and appearance embeddings;
- ARFANet-DM + GAP: The model with Doppelganger mining and auxiliary embeddings, taken from the global averagely pooled outputs of the trained network's last convolutional layer;
- ARFANet-DM + FL: The model with Doppelganger mining and auxiliary embeddings, taken from the final (deep embedding) layer of the network;
- ARFANet-DM + FL-S: The model with Doppelganger mining and auxiliary embeddings, taken from the trained "ARFANet-DM + FL" model, fixed and used from the start of training;
- ARFANet-DM + FL-1M: The model "ARFANet-DM + FL", trained for one more stage ($400,000$ iterations, $1,000,000$ in total);

Auxiliary embeddings for "ARFANet + FL", "ARFANet-DM + FL" and "ARFANet-DM + FL-1M" models were constructed (starting from the stage 2) in the following way: for the image and it's mirrored version outputs of the final layer of the neural network are concatenated and the resulting 1024-size vector is L_2-normalized.

Appearance embeddings extraction for "ARFANet-DM + MOON" model was performed by CNN with VGG-16 architecture [53]. Training and loss function configurations are taken from MOON network [49]. Training was performed on CelebA dataset [33]. Output of the penultimate layer has size of 128. It is L_2-normalized and used as appearance embedding with cosine similarity.

Method	GAR@1%	GAR@0.1%
Baseline [23]	33.76%	17.73%
ARFANet	70.42%	47.53%
ARFANet + FL	71.77%	49.11%
ARFANet-DM	79.58%	64.03%
ARFANet-DM + MOON	79.77%	64.14%
ARFANet-DM + GAP	79.84%	64.56%
ARFANet-DM + FL	**80.51%**	**64.84%**
ARFANet-DM + FL-S	61.22%	35.57%
ARFANet-DM + FL-1M	**81.93%**	**67.59%**

Table 2. The results on the Disguised Faces in the Wild dataset. The evaluation metrics are genuine accept rate (GAR) at false accept rates (FAR) of 1% and 0.1%.

Auxiliary embedding for "ARFANet-DM + GAP" model was constructed (starting from the stage 2) in the following way: for the image and it's mirrored version outputs of the last convolutional layer were global averagely pooled, concatenated and the resulting 2048-size vector is L_2-normalized.

Auxiliary embeddings for "ARFANet + FL", "ARFANet-DM + GAP", "ARFANet-DM + FL" and "ARFANet-DM + FL-1M" models were updated at the end of each training stage.

Auxiliary embeddings for "ARFANet-DM + FL-S" model were computed with "ARFANet-DM + FL" model, trained for three stages ($600,000$ iterations in total), and used for the whole training process from start without updating.

4.3. Results

The results are presented in Table 2 and Figure 4. All proposed models achieved improvement over baseline. Due to the large training dataset size, the models with Doppelganger mining performed better than the models without it. Models, using hard example mining with auxiliary embeddings, achieved better results than their counterparts without auxiliary embeddings. The best result of 80.51% GAR@1%FAR and 64.84% GAR@0.1%FAR was achieved when the auxiliary embeddings from the last layer of the network were used. Training for one more stage improved this result to 81.93% GAR@1%FAR and 67.59% GAR@0.1%FAR. Although the "ARFANet-DM + MOON" model did not provide the best results, it performed better than the model "ARFANet-DM" without auxiliary embeddings. "ARFANet-DM + FL-S" model performed worse than other ARFANet-based models. One possible reason for this is that the training examples were too hard from the beginning instead of increasing the hardness gradually, so the network didn't learn good features for the simple cases.

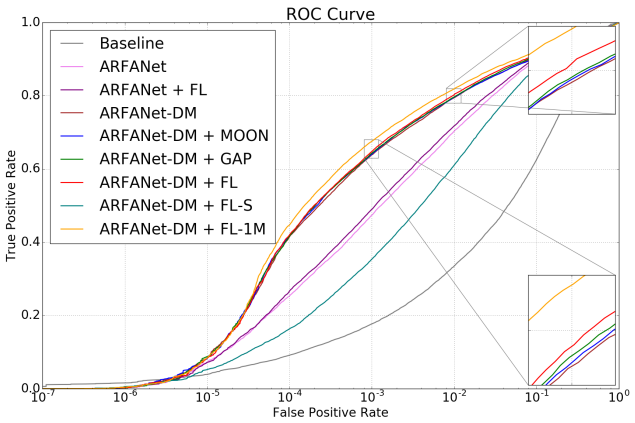

Figure 4. The ROC curve for the Disguised Faces in the Wild (DFW) dataset.

5. Conclusions

In this paper, we have presented a novel method of hard example mining with auxiliary embeddings to improve the training of deep embedding models by the generation of more informative mini-batches.

The core idea of this method is to compute auxiliary embeddings for each training example in the dataset and use them at the mini-batch generation stage to select better examples to be included in the mini-batch. These auxiliary embeddings are constructed to have large cosine similarity value in the case of hard negative example pairs and small cosine similarity value in the case of hard positive example pairs. This method performs best in combination with Doppelganger Mining or some other hard class mining algorithm.

We have proposed several ways to produce these auxiliary embeddings and performed experiments on the challenging Disguised Faces in the Wild dataset. The experimental results have confirmed the fact that using hard example mining with auxiliary embeddings improves deep embedding models.

Acknowledgement

This work was financially supported by the Ministry of Education and Science of the Russian Federation, Contract 14.578.21.0189 (ID RFMEFI57816X0189).

References

[1] Working memory prototype loss for deep representation learning. *Manuscript in preparation*.

[2] J. Alabort-i Medina, E. Antonakos, J. Booth, P. Snape, and S. Zafeiriou. Menpo: A comprehensive platform for parametric image alignment and visual deformable models. In *ACM Multimedia*, pages 679–682. ACM, 2014.

[3] Y. Bai, Y. Lou, F. Gao, S. Wang, Y. Wu, and L. Duan. Group sensitive triplet embedding for vehicle re-identification. *IEEE Transactions on Multimedia*, 2018.

[4] A. Bulat and G. Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In *ICCV*, 2017.

[5] J. Cai, Z. Meng, A. S. Khan, Z. Li, and Y. Tong. Island loss for learning discriminative features in facial expression recognition. *arXiv preprint arXiv:1710.03144*, 2017.

[6] B. Chen, W. Deng, and J. Du. Noisy softmax: Improving the generalization ability of dcnn via postponing the early softmax saturation. In *CVPR*, 2017.

[7] W. Chen, X. Chen, J. Zhang, and K. Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. In *CVPR*, volume 2, 2017.

[8] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, volume 1, pages 539–546. IEEE, 2005.

[9] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models-their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.

[10] J. Deng, J. Guo, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *arXiv preprint arXiv:1801.07698*, 2018.

[11] W. Deng, B. Chen, Y. Fang, and J. Hu. Deep correlation feature learning for face verification in the wild. *IEEE Signal Processing Letters*, 24(12):1877–1881, 2017.

[12] T. I. Dhamecha, R. Singh, M. Vatsa, and A. Kumar. Recognizing disguised faces: Human and machine evaluation. *PloS one*, 9(7):e99212, 2014.

[13] Y. Em, F. Gag, Y. Lou, S. Wang, T. Huang, and L.-Y. Duan. Incorporating intra-class variance to fine-grained visual recognition. In *ICME*, pages 1452–1457. IEEE, 2017.

[14] Y. Fan, F. Tian, T. Qin, J. Bian, and T.-Y. Liu. Learning what data to learn. *arXiv preprint arXiv:1702.08635*, 2017.

[15] B. Gecer, V. Balntas, and T.-K. Kim. Learning deep convolutional embeddings for face representation using joint sample- and set-based supervision. In *ICCV Workshops*, 2017.

[16] 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*, pages 87–102. Springer, 2016.

[17] S. Horiguchi, D. Ikami, and K. Aizawa. Significance of softmax-based features in comparison to distance metric learning-based features. *arXiv preprint arXiv:1712.10151*, 2017.

[18] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017.

[19] C. Huang, C. C. Loy, and X. Tang. Local similarity-aware deep feature embedding. In *NIPS*, pages 1262–1270, 2016.

[20] R. Huang, X. Xie, Z. Feng, and J. Lai. Face recognition by landmark pooling-based cnn with concentrate loss. In *ICIP*, pages 1582–1586. IEEE, 2017.

[21] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[22] V. B. Kumar, B. Harwood, G. Carneiro, I. Reid, and T. Drummond. Smart mining for deep metric learning. In *ICCV*, 2017.

[23] V. Kushwaha, M. Singh, R. Singh, M. Vatsa, N. Ratha, and R. Chellappa. Disguised Faces in the Wild. Technical report, IIIT Delhi, March 2018.

[24] G. Lavrentyeva, S. Novoselov, E. Malykh, A. Kozlov, O. Kudashev, and V. Shchemelinin. Audio replay attack detection with deep learning frameworks. In *Interspeech*, pages 82–86, 2017.

[25] J. Lezama, Q. Qiu, P. Musé, and G. Sapiro. OlÉ: Orthogonal low-rank embedding, a plug and play geometric loss for deep learning. *arXiv preprint arXiv:1712.01727*, 2017.

[26] J. Li, B. Li, Y. Xu, K. Lu, K. Yan, and L. Fei. Disguised face detection and recognition under the complex background. In *CIBIM*, pages 87–93. IEEE, 2014.

[27] X. Liang, X. Wang, Z. Lei, S. Liao, and S. Z. Li. Soft-margin softmax for deep classification. In *ICONIP*, pages 413–421. Springer, 2017.

[28] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*, pages 6738–6746, July 2017.

[29] W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, pages 507–516, 2016.

[30] X. Liu, B. Kumar, J. You, and P. Jia. Adaptive deep metric learning for identity-aware facial expression recognition. In *CVPR Workshops*, pages 522–531, 2017.

[31] Y. Liu, H. Li, and X. Wang. Learning deep features via congenerous cosine loss for person recognition. *arXiv preprint arXiv:1702.06890*, 2017.

[32] Y. Liu, H. Li, and X. Wang. Rethinking feature discrimination and polymerization for large-scale recognition. In *NIPS*, 2017.

[33] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[34] I. Loshchilov and F. Hutter. Online batch selection for faster training of neural networks. In *ICLR 2016. Workshop Track*, 2016.

[35] I. Loshchilov and F. Hutter. Sgdr: stochastic gradient descent with restarts. In *ICLR*, 2017.

[36] Z. Ming, J. Chazalon, M. Muzzamil Luqman, M. Visani, and J.-C. Burie. Simple triplet loss based on intra/inter-class metric learning for face verification. In *ICCV Workshops*, pages 1656–1664, 2017.

[37] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh. No fuss distance metric learning using proxies. 2017.

[38] A. Nech and I. Kemelmacher-Shlizerman. Level playing field for million scale face recognition. In *CVPR*, pages 3406–3415. IEEE, 2017.

[39] Y. Nesterov. A method of solving a convex programming problem with convergence rate o(1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[40] S. Novoselov, O. Kudashev, V. Schemelinin, I. Kremnev, and G. Lavrentyeva. Deep cnn based feature extractor for text-prompted speaker recognition. *arXiv preprint arXiv:1803.05307*, 2018.

[41] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, pages 4004–4012, 2016.

[42] O. M. Parkhi, A. Vedaldi, A. Zisserman, et al. Deep face recognition. In *BMVC*, 2015.

[43] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[44] K. Patterson and A. Baddeley. When face recognition fails. *Journal of Experimental Psychology: Human Learning and Memory*, 3(4):406, 1977.

[45] C. Qi and F. Su. Contrastive-center loss for deep neural networks. In *ICIP*, pages 2851–2855, 2017.

[46] R. Ranjan, C. D. Castillo, and R. Chellappa. l_2-constrained softmax loss for discriminative face verification. *arXiv preprint arXiv:1703.09507*, 2017.

[47] G. Righi, J. J. Peissig, and M. J. Tarr. Recognizing disguised faces. *Visual Cognition*, 20(2):143–169, 2012.

[48] O. Rippel, M. Paluri, P. Dollar, and L. Bourdev. Metric learning with adaptive density discrimination. In *ICLR*, 2016.

[49] E. M. Rudd, M. Günther, and T. E. Boult. Moon: A mixed objective optimization network for the recognition of facial attributes. In *European Conference on Computer Vision*, pages 19–35. Springer, 2016.

[50] S. Sankaranarayanan, A. Alavi, C. D. Castillo, and R. Chellappa. Triplet probabilistic embedding for face verification and clustering. In *BTAS*, pages 1–8. IEEE, 2016.

[51] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015.

[52] T. Sim, S. Baker, and M. Bsat. The cmu pose, illumination, and expression (PIE) database. In *FG*, pages 53–58. IEEE, 2002.

[53] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[54] A. Singh, D. Patil, G. M. Reddy, and S. Omkar. Disguised face identification (dfi) with facial keypoints using spatial fusion convolutional network. In *ICCV Workshops*, 2017.

[55] E. Smirnov, A. Melnikov, S. Novoselov, E. Luckyanets, and G. Lavrentyeva. Doppelganger mining for face representation learning. In *ICCV Workshops*, pages 1916–1923, Oct 2017.

[56] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NIPS*, pages 1857–1865, 2016.

[57] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy. Deep metric learning via facility location. In *CVPR*, 2017.

[58] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *NIPS*, pages 1988–1996, 2014.

[59] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, pages 1701–1708, 2014.

[60] E. Ustinova and V. Lempitsky. Learning deep embeddings with histogram loss. In *NIPS*, pages 4170–4178, 2016.

[61] W. Wan, Y. Zhong, T. Li, and J. Chen. Rethinking feature distribution for loss functions in image classification. In *CVPR*, 2018.

[62] C. Wang, X. Zhang, and X. Lan. How to train triplet networks with 100k identities? In *ICCV Workshops*, 2017.

[63] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille. Normface: l_2 hypersphere embedding for face verification. In *ACM Multimedia*, pages 1041–1049. ACM, 2017.

[64] H. Wang, Y. Wang, Z. Zhou, X. Ji, Z. Li, D. Gong, J. Zhou, and W. Liu. Cosface: Large margin cosine loss for deep face recognition. *arXiv preprint arXiv:1801.09414*, 2018.

[65] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin. Deep metric learning with angular loss. In *ICCV*, 2017.

[66] T. Y. Wang and A. Kumar. Recognizing human faces under disguise and makeup. In *ISBA*, pages 1–7. IEEE, 2016.

[67] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *ECCV*, pages 499–515. Springer, 2016.

[68] B. Wu, Z. Chen, J. Wang, and H. Wu. Exponential discriminative metric embedding in deep learning. *Neurocomputing*, 2018.

[69] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Sampling matters in deep embedding learning. In *ICCV*, 2017.

[70] Q. Xiao, H. Luo, and C. Zhang. Margin sample mining loss: A deep learning based method for person re-identification. *arXiv preprint arXiv:1710.00478*, 2017.

[71] Y. Xu, Y. Zhai, J. Gan, and J. Zeng. Disguised face recognition based on local feature fusion and biomimetic pattern recognition. In Z. Sun, S. Shan, H. Sang, J. Zhou, Y. Wang, and W. Yuan, editors, *CCBR*, 2014.

[72] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.

[73] M. Yoshino, K. Noguchi, M. Atsuchi, S. Kubota, K. Imaizumi, C. D. L. Thomas, and J. G. Clement. Individual identification of disguised faces by morphometrical matching. *Forensic science international*, 127(1-2):97–103, 2002.

[74] Y. Yuan, K. Yang, and C. Zhang. Hard-aware deeply cascaded embedding. In *ICCV*, 2017.

[75] W.-h. Yun, D. Kim, H.-S. Yoon, and J. Lee. Disguised-face discriminator for embedded systems. *ETRI journal*, 32(5):761–765, 2010.

[76] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, Oct 2016.

[77] X. Zhang, L. Yang, J. Yan, and D. Lin. Accelerated training for massive classification via dynamic class selection. In *AAAI*, 2018.

[78] J.-X. Zhong, G. Li, and N. Li. Deep metric learning with false positive probability. In *ICONIP*, 2017.