Doppelganger Mining for Face Representation Learning

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

Aleksandr Melnikov
ITMO University
melnikov-a@speechpro.com

Sergey Novoselov
ITMO University
novoselov@speechpro.com

Eugene Luckyanets
Speech Technology Center
lukyanets@speechpro.com

Galina Lavrentyeva
ITMO University
lavrentyeva@speechpro.com

Abstract

In this paper we present Doppelganger mining - a method to learn better face representations. 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. It is especially useful for methods, based on exemplar-based supervision. Usually hard example mining comes with a price of necessity to use large mini-batches or substantial extra computation and memory cost, particularly for datasets with large numbers of identities. Our method needs only a negligible extra computation and memory. In our experiments on a benchmark dataset with 21,000 persons we show that Doppelganger mining, being inserted in the face representation learning process with joint prototype-based and exemplar-based supervision, significantly improves the discriminative power of learned face representations.

1. Introduction

Deep Convolutional Neural Networks have achieved great results in face verification and recognition [38, 11, 1, 47]. The main component of this success is their ability to benefit from large-scale training datasets [10, 25] and learn powerful discriminative face representations [4, 43, 30]. Convolutional Neural Networks usually represent faces in the form of embedding vectors [33], such that Euclidean or cosine distance between two different face embeddings of the same person is less than distances between the face embeddings of two different persons. Training of these networks is based on the proper choice of network architecture, loss function and sampling method. While there is a very active ongoing research in the field of neural network architectures [45, 28, 13, 3, 46, 41, 52, 37, 50] and loss functions [33, 43, 34, 24, 40, 27, 30, 42, 14, 18, 29, 20, 51, 35, 31, 7, 2], sampling method receives less attention [44, 16].

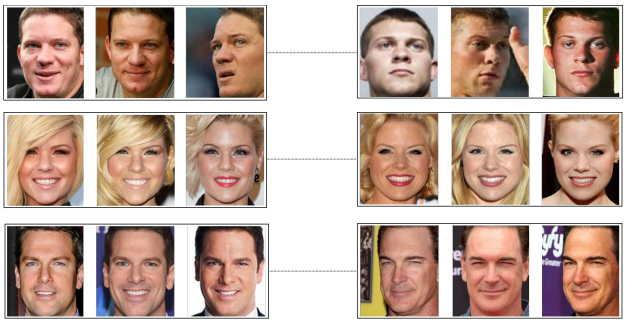

Figure 1. Examples of the doppelganger identities from the Low-shot face recogntion challenge dataset (base set) [9]. Identities at the left and their corresponding doppelgangers at the right.

In this paper we propose *Doppelganger mining* - a new simple sampling method, which improves the training of Deep Convolutional Neural Networks for face recognition. The essence of this method is to keep track of a *doppelgangers* - identities in the training dataset, which are different, but look similar and are close in the embedding space. For this purpose we create a list, which contains IDs of the most similar identities for each identity in the dataset. This list is not fixed, it changes with the progress of training, keeping doppelgangers up-to-date with the current state of the neural network. This acquired pairwise similarity information is used at the mini-batch generation stage. Part of the identities in the mini-batch are sampled randomly, and the other part consists of the first part's doppelgangers. Mini-batches, created with this strategy, tend to contain more hard examples, which are essential for the training of face representations.

Doppelganger mining is particularly suitable for a joint supervision setting, when the network is trained with a combination of prototype-based loss (*Softmax* [38], L_2-*Softmax* [30], *NormFace* [42], *A-Softmax* [18], *Center loss* [43]) and

exemplar-based loss (*Contrastive loss* [4], *Triplet loss* [33], *N-pair loss* [34]). With this kind of supervision, Doppelganger mining can obtain identitiy similarity information with the help of prototype-based loss, use it to find doppelgangers, and then benefit exemplar-based loss with better mini-batches, containing more hard examples.

2. Related work

There are two different types of loss functions usually used for learning discriminative face representations. One kind is prototype-based loss functions, which create prototype objects for each class and then use them to train the network to discriminate between classes (identities). These prototype objects can be in the form of class centroids [43, 29], classifier weights [28, 30, 42, 20, 18] or proxies [24]. They evolve with the neural network and contain global, dataset-wise information about classes in the embedding space. The most popular loss function of this kind is *Softmax loss* [28, 38]. However, while it encourages inter-class separability, it does not ensure intra-class compactness, and this property is important for face representations, as we need face embeddings of one identity to be close in embedding space. *Large-Margin Softmax loss* [19], which introduces an adjustable margin between classes, was proposed for this purpose. It encourages both intra-class compactness and inter-class separability. *Center loss* [43] is another loss function, created to make classes more compact in the embedding space. It is an auxiliary loss function applied together with Softmax loss and using trainable class centroids as the prototype objects. Center loss stimulates face embeddings to be close to their corresponding class centroid embeddings. *Contrastive-Center loss* [29] is a modification of Center loss, which also stimulates face embeddings to be far from their non-corresponding class centers. L_2-*Softmax loss* [30] is a modification of Softmax loss with L_2-normalized embeddings and adjustable scale parameter. *Congenerous Cosine loss* [20] and *NormFace* [42] L_2-normalize both the face embeddings and the classifier weights. *A-Softmax loss* [18] is a variant of Large-Margin Softmax loss with L_2-normalized classifier weights. All these loss functions use example embeddings with their distances to corresponding and non-corresponding prototypes to train the network. While global dataset-wise information, containing in prototypes, is very important, a lot of information is hidden in the pairwise example-to-example relations.

The other type is exemplar-based loss functions. They use example-to-example distance information to train the networks. *Contrastive loss* [4] is one of the most famous functions in this kind. It uses pairwise distances between examples in embedding space and encourages the same class examples to be close to each other, and the different class examples to be far. The next and the most popular in face recognition networks is *Triplet loss* [33] which uses triplets of examples, containing anchor, positive and negative examples. Triplet loss stimulates the network to train in such way that the distance between anchor and positive example in embedding space became less than distance between anchor and negative example by some notable margin. *Lifted Structured Feature Embedding* [27] is another loss function, which also encourages examples of the same class be closer than examples of different classes, while taking into account the matrix of all pairwise distances in the mini-batch to select better positives and negatives. This way the training is much faster. Other examples of exemplar-based loss functions are *N-pair loss* [34], *Margin based loss* [44], *PDDM* [14], *Histogram loss* [40] and *Range loss* [51]. To combine the advantages of prototype-based and exemplar-based losses, it is possible to use joint supervision [36, 30], to fine-tune the trained neural network with different loss function [28] or just to train an additional embedding layer on top of the trained neural network [32].

Exemplar-based loss functions benefit heavily from better example pairs, used in training. The most valuable pairs are hard negatives and hard positives, i.e. examples of the different classes that are very close in the embedding space, and examples of the same class, that are far in the embedding space. There are several different methods to acquire such example pairs in the training. One way is *online hard example mining* when the most hard example pairs, triplets or quadruplets [14] within a mini-batch are selected. In some cases too hard examples can prevent the network to train properly, so there is *online semi-hard example mining* [33, 28, 27], when example pairs are selected randomly from the "hard enough" pairs in the minibatch (pairs that violate some distance margin). The network can get useful information not only from hard example pairs, but also from not-so-hard example pairs [49], so there are also methods to select a distance-weighted mix of hard and not too hard samples [44]. All above methods are performed inside a mini-batch, so they are dependent on the mini-batch size (larger size means more examples to choose from) and mini-batch generation method (i.e. how to select what examples to include in the mini-batch in the first place). Large mini-batches demand more memory and computation, limiting the size of possible neural networks. Also when the training dataset consists of the large number of distinct classes, randomly chosen examples in mini-batch may contain not enough hard example pairs. There are methods to select hard examples globally [21, 6], but they work with individual examples, not example pairs. The problem of sampling better mini-batches for pairwise exemplar-based learning was studied in [34]. They proposed *hard class mining*: first, some number of random classes are selected, then a couple of examples from each class is used to find classes that are close to each other. Finally, more examples of these

classes are sampled in the mini-batch. While benefiting the performance, this method is computationally expensive and not very good for the datasets with large number of classes. Another way to sample better mini-batches is *Smart Mining with Fast Approximate Nearest Neighbour Graph* [16] which is also computationally expensive and requires the construction of a nearest neighbour index.

3. Proposed Method

We propose a new method for mini-batch generation called *Doppelganger mining*. The main motivation behind this method is that to create better mini-batches for exemplar-based face representation training, we must carefully select which classes to include in the mini-batch. When the number of classes is not very large, there is a good chance that hard class pairs will appear in the mini-batch and create good hard example pairs for training. But when the number of classes is large, and each of them has only a few hard class pairs in the dataset, we must not rely on a random sampling, but sample classes more intelligently.

We propose to maintain a list of *doppelgangers* - paired hard negative classes, one doppelganger for each identity in the training dataset. This list is created, updated and used through the course of neural network training with the help of joint prototype-based and exemplar-based supervision. In this paper we use L_2-Softmax loss as a prototype-based loss and a combination of Lifted Feature Embedding and Margin based loss as an exemplar-based loss, but the method is general and can be used with different losses.

3.1. Mini-batch generation

To generate a mini-batch of size M first we must choose how many classes N_c and how many examples per class it will contain. To find N_c we sequentially randomly sample the number of examples for each prospective class in the mini-batch from some predefined range (for example, from 2 to 8) until total number of examples reaches M. As the result, we determine that there will be N_c different classes in the current mini-batch. Now we must select each of these N_c classes from N_{total} (total number of classes in the dataset). N_r (it is a hyperparameter) of these classes are selected randomly from the total number of classes in the training dataset. Another N_d = N_c-N_r classes are selected with C_i = Doppelganger($C_{i\text{ - }N_r}$). Doppelganger(x) returns a doppelganger class (from the doppelganger list) for the class x, or the random class from N_{total} if there is no doppelganger yet in the list for the class x, or this doppelganger class is already sampled in the current mini-batch (to prevent cyclic doppelganger class sampling). So, basically, first N_r classes in the mini-batch are sampled randomly, and all other classes are their doppelgangers or doppelgangers of their doppelgangers or random classes (if there are no corresponding doppelganger classes, or have already been sampled in the current mini-batch).

3.2. Doppelganger mining

At the start of the training the doppelganger list is empty. To fill it with doppelgangers we use activation scores, computed by the prototype-based loss function (in this paper we use L_2-Softmax loss). The neural network outputs the face embedding vector, which is used for exemplar-based and prototype-based supervision. The latter takes the form of a classifier with N_{total} classes. Each training example, passed through the network, outputs the scores for each class, and the class with maximum score is used as a predicted class for this example. These scores are used by the prototype-based loss to provide the net with supervision. To mine doppelgangers, we just use the same already calculated scores and find the highest scored class (excluding the correct class) for each of the N_c selected classes in the current mini-batch. These highest scored incorrect classes are saved in the doppelganger list as the doppelgangers for the corresponding N_c classes.

The list of doppelgangers isn't fixed to be one "most similar person in dataset per identity". It keeps continuously updating according to the current state of the neural network. Despite that it was introduced to make benefits mostly for exemplar-based loss training in joint supervision settings, Doppelganger mining also helps prototype-based loss training: if a class becomes a doppelganger to many classes - it means that this class is hard and can easily be confused with, so there is a need to provide more attention to this class. Because it is a doppelganger to a lot of other classes, Doppelganger mining will sample it in mini-batches more often, and it will help the network to train better. But the main advantages of Doppelganger mining still come from using exemplar-based loss functions in the joint supervision setting together with prototype loss, especially when some batch-wise hard example mining method also used.

The amount of doppelgangers in the mini-batch can be controlled with N_r hyperparameter, which defines the number of random classes in the mini-batch. When N_r = N_c, there are no doppelgangers in the mini-batch, it is just a random sampling. However, if the doppelganger mining is on, the doppelganger list is getting updates and collects doppelgangers, just don't use them for mini-batch generation. If $N_r = \frac{N_c}{2}$, then for each random class in the mini-batch there is one doppelganger class in the same mini-batch. When $N_r = \frac{N_c}{3}$, then for each random class in the mini-batch there is a doppelganger class in the current mini-batch, and this doppelganger class also has its own doppelganger class in the same mini-batch. When N_r = 1, then there is just one random class, and all other classes in the mini-batch are related to it in the doppelganger-of-

doppelganger sense.

Doppelganger mining is easy to implement and has only a negligible extra computation and memory cost. When used as a part of joint supervision, it uses the result of prototype-based loss computation, and there is only a need to calculate the highest non-target values amongst the activation scores. Extra amount of memory, which is needed for Doppelganger mining, is only one number per class in the training dataset.

4. Experiments

In this section, we will evaluate Doppelganger mining on the One-shot Face Recognition [9] benchmark dataset.

4.1. Dataset and Evaluation metric

One-shot Face Recognition [9] benchmark dataset consists of 20,000 persons for face feature learning (base set) and 1,000 persons for one-shot learning (novel set). Each person in the base set has 50-100 face images for training and 5 images for testing. Each person in the novel set has only one face image for training and 20 images for testing. There is also a development part of the dataset, which has 20,000 images of persons from the base set and 5,000 images of persons from the novel set. Only base and novel set of the training dataset can be used for training.

The task is to develop an algorithm to recognize the persons in both the datasets - base set and novel set. In particular, the main focus is on the recognition accuracy for persons in the novel set, while maintaining good recognition accuracy for those in the base set. Coverage rates at Precision 99% and 99.9% are used as the evaluation metrics.

4.2. Implementation details

For this task we have trained several neural networks with different loss functions and hyperparameters, with and without Doppelganger mining. In the next section we will describe our solution.

4.2.1 Architecture

We used neural network architecture, inspired by FaceResNet [43]. We used larger image size, more filters in the layers, larger filter sizes in the first layers, a couple of dilated convolution layers [48] to provide the network with more context information, and also maxout [8] layer to make it more robust to noisy labels [45]. Final face embedding is a L_2-normalized vector of dimension 512. The architecture is summarized in the Table 1.

Output	Description
259×259×3	259×259×3
128×128×64	5×5, 64 conv, stride 2
124×124×64	5×5, 64 conv
62×62×64	2×2 maxpool, stride 2
62×62×128	3×3, 128 resblock (x1)
60×60×256	3×3, 128+64+64 conv, d=1,2,3
30×30×256	2×2 maxpool, stride 2
30×30×256	3×3, 256 resblock (x2)
28×28×512	3×3, 256+256 conv, d=1,2
14×14×512	2×2 maxpool, stride 2
14×14×512	3×3, 512 resblock (x5)
12×12×1024	3×3, 1024 conv
6×6×1024	2×2 maxpool, stride 2
6×6×1024	3×3, 1024 resblock (x3)
512	fc + maxout, group = 2
512	L_2-normalization

Table 1. Neural network architecture

4.2.2 Activation function

In this experiments we used a novel Adaptive Rational Fraction Activation (ARFA):

$$F(x) = \begin{cases} kx & , x > 0 \\ \frac{kx}{(a-x)} & , x \leq 0 \end{cases} \quad (1)$$

It is a variant of Hyperbolic Linear Unit (HLU) [17] with channel-wise trainable parameter *a*, inspired by the PReLU [12], and layer-wise parameter *k*, which is not trainable, but initialized with LSUV [23] to make the variance of the function's outputs closer to 1. ARFA is similar to ELU [5], PELU [39] and SELU [15] (see Figure 2.), while being computationally cheaper because it is not using exponentiation operation. We use ARFA in all experiments in this paper.

4.2.3 Data preprocessing and augmentation

We preprocess all face images by rescaling them to be 259×259 pixels, subtracting value 127.5 from each color channel, and then dividing each value by 128. At the training time we augment face images with random horizontal mirroring, random color change (add or subtract to each color channel a random value in the range from -10 to 10), random grayscaling (each image has 10% probability to lose the color). At the test time we used a "Mirror trick": neural network received a face image and its horizontally mirrored copy. Two embeddings, which it produced, were averaged (before L_2-normalization) to generate one final embedding. This trick got us slightly better results.

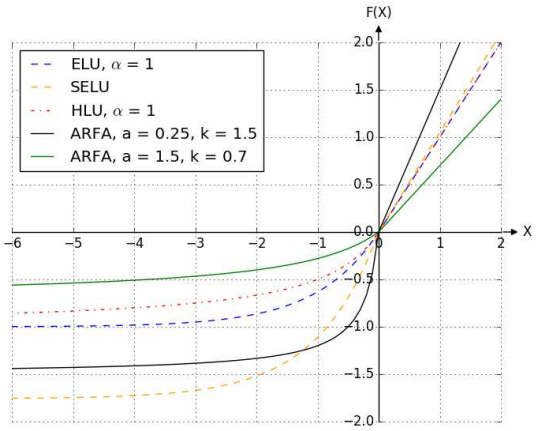

Figure 2. Shapes of different activation functions

4.2.4 Loss functions

We used L_2-Softmax loss [30] as the prototype-based loss part for the joint supervision. For the exemplar-based loss part we used Lifted Structured Feature Embedding [27] and a modified version of Margin based loss with distance weighted sampling [44]. We modify the Margin based loss to work with cosine similarity measure instead of the euclidean distance:

$$\ell^{\text{margin}}(i,j) := (\alpha - y_{\text{ij}}(S_{\text{ij}} - \beta))_+ \tag{2}$$

where β is a trainable variable that determines the boundary between positive and negative pairs, α controls the margin of separation, y_{ij} = 1 for positive pair and -1 for negative pair, and S_{ij} is cosine similarity for this pair.

We also modified distance-weighted sampling procedure [44]. For each example in the mini-batch we select one or none positive pairs and one or none negative pairs. The probability of the pair to be selected is determined by how strongly it violates the margin. The violation value v for positive pairs:

$$v_{\text{pos}} = ((\beta + \alpha) - S_{\text{ij}})_+ \tag{3}$$

and for negative pairs:

$$v_{\text{neg}} = (S_{\text{ij}} - (\beta - \alpha))_+ \tag{4}$$

Probability of the positive pair t to be selected from all K possible pairs is:

$$p_t = \frac{v_t}{\sum_{k=1}^{K} v_k} \tag{5}$$

the same goes for the negative pairs.

Method	C@99%	C@99.9%
L_2S + LE	71.06%	45.70%
L_2S + LE + MB	72.60%	49.62%

Table 2. Results of the initial experiment on the novel set (from the development set) with L_2-Softmax (L_2S), Lifted Feature Embedding (LE) and Margin Based Loss (MB), without Doppelganger mining.

Method	C@99%
SmileLab	92.64%
UP Term [9]	77.48%
SIS (our model)	73.86%
CNC240	63.17%
KATE	61.21%

Table 3. Results on the novel set from the test set

4.2.5 Initial experiment

For the initial experiment we used four networks: two with a combination of L_2-Softmax loss (loss weight 1.0, α is trainable, initial value 16) and Lifted Feature Embedding (loss weight 0.1, with trainable scale α like in [30], with initial value 12) and two with a combination of L_2-Softmax loss (loss weight 1.0, α is trainable, initial value 16), Lifted Feature Embedding (loss weight 0.1, α is trainable, initial value 12) and Margin based loss (loss weight 1.0, α = 0.1, β is trainable, initial value 0.5). We trained the networks on the base set [9] for 140,000 iterations with mini-batch size of 81, using 3 images per class (this means a total number of classes in mini-batch is 27). We used SGDR [22] learning rate schedule without restarts, starting from learning rate 0.01 and finishing at 0.00001. We used Nesterov Accelerated Gradient [26] with momentum of 0.9 and weight decay of 0.0005.

After the training on the base set was finished, we used trained networks to calculate average embeddings for each person in base set (based on all the person's images) and for each person in novel set (based on one image per person). We used these average embeddings as a prototypes. We classified each image in development and test sets into 21,000 classes according to the largest cosine similarity value between image's embedding and prototype's embedding. The results for the networks are in Tables 2 and 3: For the test results we used averaged cosine similarity scores for four our networks.

4.2.6 Experiments with Doppelganger mining

For the next experiments we used the same network architecture and training schedule, but trained for 300,000 iterations instead of 140,000. Here we compare the training with random sampling and with Doppelganger mining for

Method	Random	DM
L_2S	77.76%	**78.80%**
L_2S + LE	79.24%	**80.22%**
L_2S + MB	78.34%	**87.74%**
L_2S + LE + MB	78.20%	**87.50%**

Table 4. Results of the Doppelganger mining experiments on the novel set from the development set. L_2S for L_2-Softmax, LE for Lifted Feature Embedding, MB for Margin based loss. DM for Doppelganger mining. Evaluation metric is coverage rate at precision 99%

different combinations of loss functions, used with joint supervision. All following results are achieved on the development set. In these experiments we used Doppelganger mining with N_r = 9 and N_c = 27. In other words, we sampled 9 random identities in each mini-batch and used their 9 doppelgangers as the next 9 identities, and then for this next identities we also used their 9 doppelgangers to get final 9 identities for the mini-batch. To perform classification of the development set we once again used cosine similarity measure between face embeddings and identity prototypes in the form of average embeddings for each of the 21,000 possible classes. The results are presented in the Table 4.

With this experiment we show that Doppelganger mining improved the results for all configurations. L_2-Softmax loss doesn't have an exemplar-based loss part, but it still gains a couple of extra coverage percents with the usage of the Doppelganger mining. Exemplar-based losses, especially equipped with a good minibatch-level hard example mining methods, like Margin based loss with distance-weighted sampling, are benefitting from Doppelganger mining the most.

4.2.7 Experiments with the random class ratio

In this section we present experiments with a network, trained by joint supervision (L_2-Softmax + Margin based loss) and Doppelganger mining with different values of N_r (or, more precisely, with different random class ratios $\frac{N_r}{N_c}$). The results are in Table 5. As we can see, even a small fraction of doppelgangers in the mini-batch considerably improves the results.

4.2.8 Experiments with classifier learning

In the previous experiments we used simple cosine similarity measure to classify images in test and development sets. Only the base set images were used to train the network. For the next experiments we also use novel set images (one training image per class) to train 21,000-way classifiers on top of the trained networks. For each network we remove the 20,000-way L_2-Softmax layer, which was used to train

Random class ratio	C@99%	C@99.9%
1 / 27	88.16%	64.68%
2 / 27	87.84%	**75.68%**
3 / 27	87.66%	72.38%
6 / 27	**88.32%**	65.06%
9 / 27	87.74%	70.98%
15 / 27	87.84%	68.50%
21 / 27	85.06%	53.68%
27 / 27	78.34%	44.00%

Table 5. Results of the experiments with different random class ratios. The network is trained with L_2-Softmax + Margin based loss. Evaluation metric is coverage rate at precision 99% and at precision 99.9%

Method (random ratio)	C@99%	C@99.9%
L_2S + MB (1 / 27)	98.9%	94.1%
L_2S + MB (2 / 27)	98.88%	94.6%
L_2S + MB (3 / 27)	98.76%	94.18%
L_2S + MB (6 / 27)	98.74%	93.2%
L_2S + MB (9 / 27)	99%	**94.92%**
L_2S + MB (15 / 27)	98.76%	93.02%
L_2S + MB (21 / 27)	**99.2%**	94.74%
L_2S + MB (27 / 27)	97.8%	91.8%
L_2S + LE + MB (9 / 27)	98.72%	93.76%
L_2S + LE + MB (27 / 27)	98.5%	90.42%
Ensemble (only nets with DM)	99.78%	**96.7%**
Ensemble (all nets)	**99.84%**	95.76%

Table 6. Results with classifier training. L_2S for L_2-Softmax loss, MB for Margin based loss, LE for Lifted Feature Embedding.

the network on the base set, and switch it to the new 21,000-way L_2-Softmax layer. Classifier weights are initialized with average embeddings for each person. In the next experiments we train only this last classifier layer. The training is performed for 500 iterations with initial learning rate of 0.001 and final learning rate of 0.0001. For these experiments we don't use Doppelganger mining and exemplar-based losses. Also we sample one image per class to create mini-batches with the largest possible number of different classes (N_r = N_c = 81 for a mini-batch of size 81). The results are presented in the Table 6. On the development base set all these networks achieve more than 99.88% classification accuracy, so the fine-tuning with the novel set doesn't hurt performance on the base set.

As we can see from the result tables, Doppelganger mining significantly improves learned face representations. In our experiments we have achieved 99.2% coverage at 99% precision with a single network and 99.84% coverage at 99% precision with an ensemble of several networks on the one-shot face recognition benchmark with 21,000 persons.

5. Conclusions

In this paper, we have presented a novel method called Doppelganger mining, which improves the training of face representations with neural networks by the generation of more useful mini-batches. This method provide the most benefits in the joint supervision settings with the help of prototype-based and exemplar-based losses. The core idea of this method is to maintain a list of hard class pairs for each identity in the dataset. Prototype-based losses (e.g. L_2-Softmax loss) produce scores, which can be interpreted as a similarity measure between current example and all identities, presented in the dataset. We can use these scores to find hardest identity pairs in the dataset almost without extra computational cost. These paired identities are later sampled together in the mini-batch, providing it with better candidates for hard example mining methods. It helps exemplar-based losses to train better face representations. Doppelganger mining needs only a small amount of extra memory and is easy to implement. Experiments on the one-shot face recognition benchmark dataset with 21,000 persons show that Doppelganger mining significantly improves face recognition results.

Acknowledgement

This work was partially financially supported by the Ministry of Education and Science of the Russian Federation, Contract 14.578.21.0189 (ID RFMEFI57816X0189), and by the Government of the Russian Federation, Grant 074-U01

References

[1] A. Bansal, C. Castillo, R. Ranjan, and R. Chellappa. The do's and don'ts for cnn-based face verification. *arXiv preprint arXiv:1705.07426*, 2017.

[2] B. Chen, W. Deng, and J. Du. Noisy softmax: Improving the generalization ability of dcnn via postponing the early softmax saturation. *arXiv preprint arXiv:1708.03769*, 2017.

[3] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks. *arXiv preprint arXiv:1707.01629*, 2017.

[4] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.

[5] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). 2015.

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

[7] A. Ghosh, H. Kumar, and P. Sastry. Robust loss functions under label noise for deep neural networks. In *AAAI*, pages 1919–1925, 2017.

[8] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *International Conference on Machine Learning*, pages 1319–1327, 2013.

[9] Y. Guo and L. Zhang. One-shot face recognition by promoting underrepresented classes. *arXiv preprint arXiv:1707.05574*, 2017.

[10] 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*, pages 87–102. Springer, 2016.

[11] A. Hasnat, J. Bohné, S. Gentric, and L. Chen. Deepvisage: Making face recognition simple yet with powerful generalization skills. *arXiv preprint arXiv:1703.08388*, 2017.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] C. Huang, C. C. Loy, and X. Tang. Local similarity-aware deep feature embedding. In *Advances in Neural Information Processing Systems*, pages 1262–1270, 2016.

[15] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. *arXiv preprint arXiv:1706.02515*, 2017.

[16] V. B. Kumar, B. Harwood, G. Carneiro, I. Reid, and T. Drummond. Smart mining for deep metric learning. *arXiv preprint arXiv:1704.01285*, 2017.

[17] J. Li, H. Xu, J. Deng, and X. Sun. Hyperbolic linear units for deep convolutional neural networks. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 353–359. IEEE, 2016.

[18] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. *arXiv preprint arXiv:1704.08063*, 2017.

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

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

[21] I. Loshchilov and F. Hutter. Online batch selection for faster training of neural networks. In *International Conference on Learning Representations (ICLR 2016). Workshop Track.*, 2015.

[22] I. Loshchilov and F. Hutter. Sgdr: stochastic gradient descent with restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[23] D. Mishkin and J. Matas. All you need is a good init. 2015.

[24] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh. No fuss distance metric learning using proxies. *arXiv preprint arXiv:1703.07464*, 2017.

[25] A. Nech and I. Kemelmacher-Shlizerman. Level playing field for million scale face recognition. *arXiv preprint arXiv:1705.00393*, 2017.

[26] 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.

[27] 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*, pages 4004–4012, 2016.

[28] O. M. Parkhi, A. Vedaldi, A. Zisserman, et al. Deep face recognition. In *BMVC*, volume 1, page 6, 2015.

[29] C. Qi and F. Su. Contrastive-center loss for deep neural networks. *arXiv preprint arXiv:1707.07391*, 2017.

[30] R. Ranjan, C. D. Castillo, and R. Chellappa. L2-constrained softmax loss for discriminative face verification. *arXiv preprint arXiv:1703.09507*, 2017.

[31] O. Rippel, M. Paluri, P. Dollar, and L. Bourdev. Metric learning with adaptive density discrimination. *arXiv preprint arXiv:1511.05939*, 2015.

[32] S. Sankaranarayanan, A. Alavi, C. D. Castillo, and R. Chellappa. Triplet probabilistic embedding for face verification and clustering. In *Biometrics Theory, Applications and Systems (BTAS), 2016 IEEE 8th International Conference on*, pages 1–8. IEEE, 2016.

[33] 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*, pages 815–823, 2015.

[34] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, 2016.

[35] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy. Deep metric learning via facility location. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[36] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems*, pages 1988–1996, 2014.

[37] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017.

[38] 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*, pages 1701–1708, 2014.

[39] L. Trottier, P. Giguère, and B. Chaib-draa. Parametric exponential linear unit for deep convolutional neural networks. *arXiv preprint arXiv:1605.09332*, 2016.

[40] E. Ustinova and V. Lempitsky. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems*, pages 4170–4178, 2016.

[41] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang. Residual attention network for image classification. *arXiv preprint arXiv:1704.06904*, 2017.

[42] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille. Normface: l_2 hypersphere embedding for face verification. *arXiv preprint arXiv:1704.06369*, 2017.

[43] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer, 2016.

[44] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Sampling matters in deep embedding learning. *arXiv preprint arXiv:1706.07567*, 2017.

[45] 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.

[46] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.

[47] L. Xiong, J. Karlekar, J. Zhao, J. Feng, S. Pranata, and S. Shen. A good practice towards top performance of face recognition: Transferred deep feature fusion. *arXiv preprint arXiv:1704.00438*, 2017.

[48] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[49] Y. Yuan, K. Yang, and C. Zhang. Hard-aware deeply cascaded embedding. *arXiv preprint arXiv:1611.05720*, 2016.

[50] C. Yunpeng, J. Xiaojie, K. Bingyi, F. Jiashi, and Y. Shuicheng. Sharing residual units through collective tensor factorization in deep neural networks. *arXiv preprint arXiv:1703.02180*, 2017.

[51] X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao. Range loss for deep face recognition with long-tail. *arXiv preprint arXiv:1611.08976*, 2016.

[52] X. Zhang, Z. Li, C. C. Loy, and D. Lin. Polynet: A pursuit of structural diversity in very deep networks. *arXiv preprint arXiv:1611.05725*, 2016.