Improving Deep Metric Learning With Virtual Classes And Examples Mining

Pierre Jacob, David Picard, Aymeric Histace

IMPROVING DEEP METRIC LEARNING WITH VIRTUAL CLASSES AND EXAMPLES MINING

Pierre Jacob[1], *David Picard*[2], *Aymeric Histace*[1]

[1]ETIS UMR 8051, CY Cergy-Paris University, ENSEA, CNRS, F-95000, Cergy, France
[2]LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France

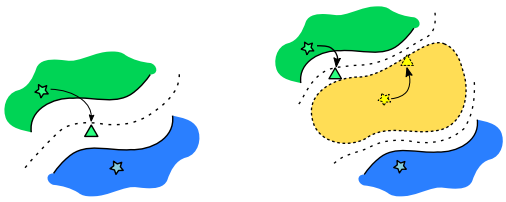
ABSTRACT

In deep metric learning, the training procedure relies on sampling informative tuples. However, as the training procedure progresses, it becomes nearly impossible to sample relevant hard negative examples without proper mining strategies or generation-based methods. Recent work on hard negative generation have shown great promises to solve the mining problem. However, this generation process is difficult to tune and often leads to incorrectly labeled examples. To tackle this issue, we introduce *MIRAGE*, a generation-based method that relies on virtual classes entirely composed of generated examples that act as buffer areas between the training classes. We empirically show that virtual classes significantly improve the results on popular datasets (Cub-200-2011 and Cars-196) compared to other generation methods. ***Index Terms*—** image retrieval; metric learning; example mining; virtual classes; example generation

1. INTRODUCTION

Deep metric learning (DML) is an important, yet challenging task in the Computer Vision community, with numerous applications such as multi-modal retrieval [1], face verification [19] or person re-identification [11]. DML methods intend to learn an embedding space, where visually-related images (*e.g.*, two different birds from the same breed) have similar representations, while unrelated images (*e.g.*, two different breeds of crows from North America and Europe) have dissimilar representations. To learn this embedding space, recent contributions focus on three main points: (1) loss functions to improve generalization [26], (2) ensemble methods to tackle the embedding space diversity [15] and (3) hard example mining strategies to resume the training when randomly sampling informative tuples becomes nearly impossible [27].

Example generation has recently been proposed as a hard negative mining strategy. In this case, a generator and the metric learning network are trained together to provide informative tuples using either VAEs [10] or GANs [4, 31, 32]. In the case of VAEs, a large amount of examples is generated by sampling with respect to the training sample distribution estimated from the data. Usually, this leads to sampling inside the class manifolds and rarely produces hard negative examples. Such variational approaches are interesting in the case of few training samples per class but they are not well suited for mining informative examples at later training stages. On the opposite, GAN-based approaches generate discriminative examples. However, adversarial generators are difficult to tune due to the contrary objectives of the DML network and the adversarial learning of the generator. On the one hand, if the adversarial loss is much lower than the DML loss, the generated examples tend to be at the center of the class manifold and the method faces the same problems as VAE generators. On the other hand, if the adversarial loss is much higher than the DML loss, some examples can be generated beyond the boundary of the class manifolds and lead to label ambiguity as illustrated on Figure 1a. The mining strategy then produces examples with incorrect labels with respect to the training classes.



(a) Standard hard negative generation

(b) Virtual class hard negative generation

Fig. 1: Hard negative generation. The standard hard negative generation on Figure 1a can lead to incorrect label if the generated example is sampled beyond the boundary of the class manifold. By adding a virtual class between the training classes on Figure 1b, hard negative examples generated beyond the boundary of the class manifold are still within the correct classes with respect to training classes.

As the main contribution of this paper, we propose *MIRAGE*, a method that leverages virtual classes composed solely of generated examples to tackle the problem of label ambiguity arising from hard negative generation. Virtual classes play the role of buffer areas as shown on Figure 1b. Hard negative examples that lie between the training class manifolds are generated inside these buffer areas, without any label ambiguity, by sampling the virtual classes. In addition to solving the problem of label ambiguity, virtual classes example generation leads to better generalization capabilities: The metric learning network has better results on unseen classes than other adversarial approaches [4, 31, 32].

The paper is organized as follow: in section 2, we present the related work in deep metric learning, the recent contributions in example generation and motivate the need for our method. In section 3, we expose the core aspects of *MIRAGE* and its simple implementation. In section section 4, we experimentally show that *MIRAGE* indeed produces buffer areas between the training classes and we perform an ablation study of the different aspects of the method. Finally, in section 5, we show that our method improves over other sample generation methods on two DML datasets (Cub-200-2011 and Cars-196), and obtains results comparable to the state-of-the-art.

2. RELATED WORK

In DML, we train a deep network to provide representations and a corresponding metric to measure similarities. Training procedure

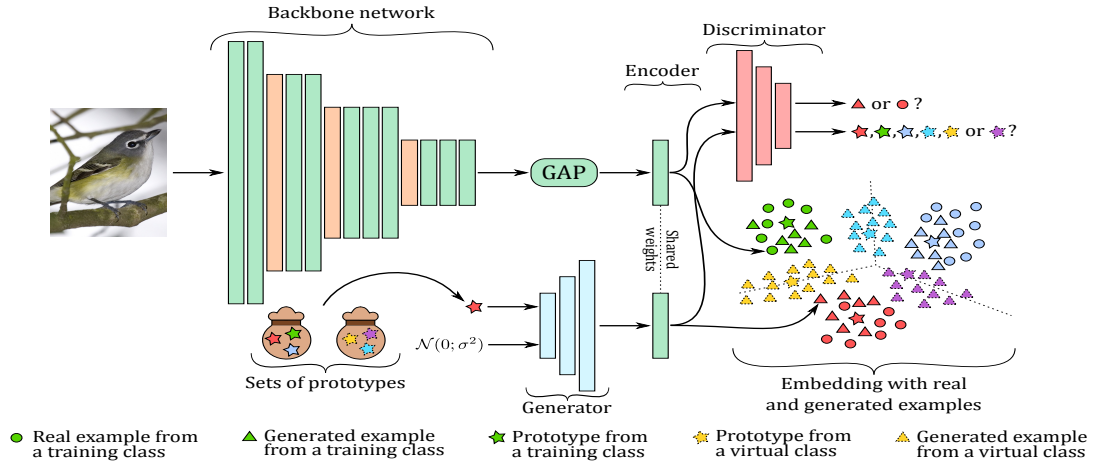


Fig. 2: Architecture overview. Feature vector is extracted from an image using the backbone network and a global average pooling (GAP). Then, it is projected into an embedding space where a metric is learned. The framework relies on sets of prototypes (stars) for the training classes (plain lines) or the virtual classes (dashed lines). A generator produces a new feature vector from a sampled prototype and a Gaussian noise. It is then projected into the same embedding space as the training images. To train this generator, we use a conditional discriminator to determine whether the sample is real or generated and the class to which it belongs.

relies on a loss function, a sampling strategy and optionally, an ensemble method. For loss function, original methods consider pairs [3] or triplets [19] of examples. These approaches have been extended to larger tuples [2, 14, 20, 22] or by improving loss function properties [18, 24, 25, 29]. Sampling strategies based on scalable mining strategies [5, 19] or proxy-based approximations [12, 13, 18] can be used to accelerate training and improve generalization. Finally, ensemble methods are a popular way to increase performances [8, 15, 16, 28, 30]. Our proposed *MIRAGE* is a complementary approach to loss functions and ensemble methods.

MIRAGE differs from other hard negative example generation methods, such as DAML [4], HTG [31], HDML [32] and DVML [10]. Both DAML and HTG generate hard examples through adversarial training but they suffer from label ambiguity illustrated in Figure 1. HDML [32] tries to alleviate this effect by generating first an intermediate example that may be outside its class manifold; then a generator projects this example into the class manifold. However, they tend to discard the hardest examples which limits its generation capability. DVML gets rid of the triplet constraints for the generator by leveraging variational approach to generate examples. Because there is no adversarial training, examples are mostly sampled at the center of the distribution and slightly contribute to the DML loss.

To solve the problem of label ambiguity while generating hard examples, we propose to insert virtual classes as buffer areas between the training classes. Sampling examples from the virtual classes allows us to use a generative sampling process similar to DVML [10] which is simpler than the triplet based adversarial methods. At the same time, it also removes the need to take into account the possible incorrectness of the labels since these generated examples do not correspond to existing classes.

3. PROPOSED METHOD

3.1. Method Overview

MIRAGE improves DML by using the following core aspects:

DML training. Like any other DML method, *MIRAGE* uses a deep network to embed feature vectors into a latent representation space where visually-related images have similar representations and where unrelated images have dissimilar representations. We use standard metric learning approach which extracts deep local features using a backbone network, computes a feature vector and projects it into an embedding space where the metric is learned.

Training class sample generation. *MIRAGE* generates artificial examples from training classes similarly to variational approaches. By doing so, class manifolds are filled with synthetic examples. These examples are added to the mini-batch along with real examples to have larger batches with informative tuples. These additional tuples are then used to train the DML model.

Virtual class hard negative sample generation. *MIRAGE* also generates hard examples using adversarial learning. To tackle label ambiguity illustrated in Figure 1a, we add virtual classes between training classes that play the role of buffer areas (see Figure 1b). Hard negative examples are consequently generated inside these buffer areas by sampling within these virtual class manifolds. Similarly to training class generation, these examples are added to the mini-batch. We experimentally show that it leads to better performances than other generation-based methods.

3.2. Deep metric learning

The first part is to extract local features and to aggregate them into a global feature $\boldsymbol{f}_t$. Then, we want to learn a Mahalanobis distance $d_{\boldsymbol{M}}$ so that the distance between two feature vectors $\boldsymbol{f}_{t_i}$ and $\boldsymbol{f}_{t_j}$ is:

$$d^2_{\boldsymbol{M}}(\boldsymbol{f}_{t_i}, \boldsymbol{f}_{t_j}) = \|\boldsymbol{W}^\top \boldsymbol{f}_{t_i} - \boldsymbol{W}^\top \boldsymbol{f}_{t_j}\|_2^2 \quad (1)$$

with $\boldsymbol{M} = \boldsymbol{W}\boldsymbol{W}^\top$ a low-rank approximation. The feature vectors are projected into the embedding space with $\boldsymbol{W}$ where their corresponding examples are denoted $\boldsymbol{x}_t$. In practice, all examples $\boldsymbol{x}_t$ are also ℓ_2-normalized to ease the optimization. We note E the function which transforms a feature vector $\boldsymbol{f}_t$ into an example $\boldsymbol{x}_t$ The network is trained using a metric learning loss $\mathcal{L}_{\text{DML}}$. Following [12], classes are represented by prototypes $\boldsymbol{p}_t$ to accelerate the training. E and $\boldsymbol{p}_t$ are trained together using $\mathcal{L}_{\text{DML}}$.

(a) Cub-200-2011 (b) MNIST

Fig. 3: Visualization of the training and virtual prototypes on MNIST and Cub-200-2011 datasets. For the Cub-200-2011 dataset (left), a t-SNE on the prototypes is run to visualize the high-dimensional embedding space. Virtual classes are in gray while training classes are in green. For the MNIST dataset (right), a 2D embedding space is learned without the ℓ_2-norm. Training examples are represented by the numbers, and the prototypes are represented by the crosses. Virtual examples are in pale color.

3.3. Training class example generation

A conditional generator G produce examples from the training classes $\widetilde{\boldsymbol{x}}_t$ from the prototype $\boldsymbol{p}_t$ of class t and a Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}; \boldsymbol{\Sigma})$, as follows:

$$\widetilde{\boldsymbol{x}}_t = E\left(G\left(\frac{\boldsymbol{p}_t + \boldsymbol{\epsilon}}{\|\boldsymbol{p}_t + \boldsymbol{\epsilon}\|_2}\right)\right) \tag{2}$$

$\widetilde{\boldsymbol{x}}_t$ is then used as a training example to optimize the loss function described in the previous section. To train G, we use a reconstruction loss by computing the ElasticNet loss between a feature vector $\boldsymbol{f}_t$ extracted from a real image and a feature vector generated by feeding G with $E(\boldsymbol{f}_t)$, as follows:

$$\mathcal{L}_{\text{rec}} = \|\boldsymbol{f}_t - G(E(\boldsymbol{f}_t))\|_1 + \|\boldsymbol{f}_t - G(E(\boldsymbol{f}_t))\|_2^2 \tag{3}$$

3.4. Virtual class example generation

To generate examples from virtual classes, each virtual class v is associated with a prototype $\boldsymbol{p}_v$. Examples are generated from these prototypes exactly like if they where training classes. To produce hard samples, we encourage prototypes and the generator to output realistic samples between the training classes. To that end, we use a discriminator D. D is trained using binary cross-entropy to distinguish between real and generated samples (with output D_g). D is also trained using categorical cross-entropy to predict classes (with output D_c). The combined loss $\mathcal{L}_{\text{adv}}$ for training D is a two head classification loss based on cross-entropy:

$$\mathcal{L}_{\text{adv}} = \underbrace{-\log D_g(E(\boldsymbol{f}_t)) - \log\left(1 - D_g \cdot E \cdot G\left(\frac{\boldsymbol{p}_v + \boldsymbol{\epsilon}}{\|\boldsymbol{p}_v + \boldsymbol{\epsilon}\|_2}\right)\right)}_{\text{binary cross-entropy on real/generated samples}}$$
$$\underbrace{-y_t \log D_c(E(\boldsymbol{f}_t)) - y_v \log\left(D_c \cdot E \cdot G\left(\frac{\boldsymbol{p}_v + \boldsymbol{\epsilon}}{\|\boldsymbol{p}_v + \boldsymbol{\epsilon}\|_2}\right)\right)}_{\text{categorical cross-entropy on the classes}}$$

where y_t and y_v are the class labels of the prototypes $\boldsymbol{p}_t$ and $\boldsymbol{p}_v$ respectively. To encourage the generator to output realistic samples that are between the training classes, G is trained to maximize $\mathcal{L}_{\text{adv}}$, with D being fixed. By optimizing over D_g, the generator is encouraged to output generated samples that are indistinguishable from real samples. By optimizing over D_c, the generator is encouraged to output samples at the boundaries of the classes (*i.e.*, in the buffer area described in Figure 1b). Just like the training class sample generation, virtual class sample generation is used to populate the mini-batches used for training using $\mathcal{L}_{\text{DML}}$.

3.5. *MIRAGE* architecture

MIRAGE architecture follows Figure 2. A set of deep local features is first extracted from the image using a backbone network such as GoogleNet [21] or BN-Inception [6]. These local features are then aggregated into global feature vectors using average pooling. They are followed by the encoder E which is composed of a single fully-connected layer without bias followed by a ℓ_2 normalization. The generator G is composed of two fully-connected layers with ReLU activation. The discriminator D is composed of a fully-connected layer with ReLU activation which is followed by two fully-connected layers: One with sigmoid activation for the binary classification of real or virtual feature vectors and one with softmax activation for the class prediction.

To train *MIRAGE*, we generate mini-batches composed of training examples $\boldsymbol{x}_t$, generated examples from the training class $\widetilde{\boldsymbol{x}}_t$ and generated examples from virtual classes $\widetilde{\boldsymbol{x}}_v$. The ratio of training examples and generated examples in the mini-batch corresponds to how much each aspect of *MIRAGE* is used. This ratio is investigated in the ablation studies. The backbone network, the encoder E and the prototypes are trained together using the entire mini-batch minimizing $\mathcal{L}_{\text{DML}}$ from subsection 3.2. The generator G is trained on $\widetilde{\boldsymbol{x}}_t$ minimizing $\mathcal{L}_{\text{rec}}$ from subsection 3.3 and on $\widetilde{\boldsymbol{x}}_v$ maximizing $\mathcal{L}_{\text{adv}}$ from subsection 3.4. Finally, the discriminator D is trained on the entire batch minimizing $\mathcal{L}_{\text{adv}}$ from subsection 3.4.

4. ABLATION STUDY

We provide different ablation studies that include (1) a visualization of the learned embedding and the virtual classes as well as some relevant statistics, (2) the impact of the number of generated example in the mini-batches, and (3) the impact of the virtual class generation. In this section, we train a GoogleNet with a 512 dimensional embedding using contrastive loss on the Cub-200-2011 dataset using a fixed batch size of real examples $|\mathcal{B}| = 40$ for all experiments (referred as the *Baseline*).

	Training class examples ratio						Virtual class ratio					
r (%)	0	10	50	100	200	400	0	10	50	100	200	400
R@1	57.0	57.8	**58.8**	58.5	58.7	58.7	57.0	58.2	59.0	**59.3**	58.6	58.9

Table 1: Impact of the number of training class generated examples and of the number of virtual class in the mini-batches.

4.1. Prototype visualization

The first ablation consists in an empirical analysis of the learned embedding space with the virtual classes. The objective is to verify that our architecture encourages the virtual classes to settle between training classes. We perform a t-SNE visualization (Figure 3a) of the training and virtual prototypes of the model trained on Cub-200-2011. Virtual prototypes (in gray) are indeed in the middle of the training class prototypes. Quantitatively, we found that more than 80% of the training class prototypes have a virtual class prototype as nearest neighbor in the 512 dimensional latent space. This numerically shows that our architecture is able to produce virtual class as buffer areas between training classes.

To avoid the bias introduced by the 2D embedding performed by t-SNE, we also train a model on the popular MNIST dataset with a latent space of dimension 2 and show the resulting prototypes as well as real and generated examples in Figure 3b. As we can see, the virtual prototypes (denoted F and in pale colors) are indeed acting as buffer between the training classes, even with the high constraints of having such a low dimensional latent space.

4.2. Sample generation

First, we evaluate the impact of the number of generated training class examples. For that purpose, we do not use virtual class prototypes. We vary the size of the generated example set $\widetilde{\mathcal{B}}$ with respect to a ratio r of the real example set $\mathcal{B}$, such that: $|\widetilde{\mathcal{B}}| = r\,|\mathcal{B}|$. We report Recall@1 on the Cub-200-2011 dataset in Table 1 for $r \in \{0, 10\%, 50\%, 100\%, 200\%, 400\%\}$.

The reported value for $r = 0\%$ means that no examples have been generated and obtains a strong *Baseline* of 57.0% Recall@1. One can note that even a small amount of generated example greatly increases the performances by around 1%, which confirms the benefit of a generation-based mining strategy to improve DML. With even more generated examples, performances are improved up to 58.8% Recall@1, a significant increase of nearly 2%.

Next, we evaluate the impact of the number of virtual classes. We fix both batch $|\widetilde{\mathcal{B}}|$ and $|\mathcal{B}|$ to 40. Then, we vary the number of the virtual class prototype N_v as a ratio of the number of the training class N_t, such that $N_v = r\,N_t$. We only generate examples from these virtual classes and not from the training classes. We report Recall@1 on the Cub-200-2011 dataset in Table 1 for $r \in \{0, 10\%, 50\%, 100\%, 200\%, 400\%\}$. Interestingly, even a small number of additional classes already improves the *Baseline* by a significant increase of more than 1.0% in Recall@1. Increasing the number of virtual classes improves even more the performances, and leads to the best results for Recall@1 with 59.3% - a significant increase of more than 2% over the *Baseline*.

5. COMPARISON TO THE STATE-OF-THE-ART

In this section, we present the benefits of *MIRAGE* on two deep metric learning datasets named Cub-200-2011 [23] and Cars-196 [9].

Backbone	Method	CUB-200-2011			Cars-196		
		R@1	R@2	R@4	R@1	R@2	R@4
GoogleNet	DAMLRMM [27]	55.1	66.5	76.8	73.5	82.6	89.1
	DAML [4]	52.7	65.4	75.5	75.1	83.8	89.7
	DVML [10]	52.7	65.1	75.5	82.0	88.4	93.3
	HDML [32]	53.7	65.7	76.7	79.1	87.1	92.1
	***MIRAGE* (Ours)**	**59.7**	**71.1**	**80.4**	**82.1**	**89.1**	**93.6**
BN-Inception	MS loss [26]	65.7	77.0	**86.3**	84.1	90.4	94.0
	SoftTriplet [17]	65.4	76.4	84.5	84.5	90.7	94.5
	HORDE [7]	**66.8**	77.4	85.1	**86.2**	**91.9**	**95.1**
	***MIRAGE* (Ours)**	66.4	**78.9**	84.6	83.9	90.3	94.4

Table 2: Comparison to the state-of-the-art on Cub-200-2011 and Cars-196 datasets. Results are reported using GoogleNet as backbone network for fair comparison with generation-based methods. Results are also reported with BN-Inception backbone for comparison with other recent methods.

We follow the standard splits from [16] and Recall@K are reported for each dataset in Table 2.

We first compare our architecture with recent sample generation approaches from the literature using the now standard GoogleNet Backbone to ensure all results are fairly comparable. As we can see, *MIRAGE* obtains very strong results on Cub-200-2011 and Cars-196. This shows the importance of combining in class sample generation, like in [10] with hard sample generation like [32], which *MIRAGE* achieves with a simple architecture.

In order to compare *MIRAGE* with recent methods, we also report Recall@K using BN-Inception [6] with the same hyper-parameters as the ones used for GoogleNet. *MIRAGE* obtains strong performances when compared to very recent state-of-the-art methods. On Cub-200-2011, we obtain second best performances, being only 0.4% behind HORDE [7]. On Cars-196, we obtain performances comparable to that of Multi-Similarity loss [26] and SoftTriplet [17]. We want to emphasize that the reported results were obtained using the constrastive loss function, and yet bring improvements to the baseline comparable to that of using a much more advance loss function such as [26], [17] or [7]. We believe this demonstrates the soundness of our approach.

6. CONCLUSION

In this paper, we introduce *MIRAGE*, a generation-based strategy that naturally solves the generation of hard examples. *MIRAGE* naturally solves the problem of generating incorrectly labeled hard negative examples by relying on a set of virtual class prototypes solely composed of generated examples. Even when the generator produces examples beyond their class manifolds, the presence of virtual classes ensures that the examples are still generated with the correct labels regarding the training classes. We empirically show that *MIRAGE* outperforms the state-of-the-art mining strategies and leads to competitive results when compared to complementary approaches. This is validated on two deep metric learning datasets named Cub-200-2011 and Cars-196.

7. REFERENCES

[1] Carvalho, M., Cadène, R., Picard, D., Soulier, L., Thome, N., Cord, M.: Cross-modal retrieval in the cooking context: Learning semantic text-image embeddings. In: SIGIR Conference on Research & Development in Information Retrieval (2018) 1

[2] Chen, W., Chen, X., Zhang, J., Huang, K.: Beyond triplet loss: A deep quadruplet network for person re-identification. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017) 2

[3] Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2005) 2

[4] Duan, Y., Zheng, W., Lin, X., Lu, J., Zhou, J.: Deep adversarial metric learning. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018) 1, 2, 4

[5] Harwood, B., Kumar B G, V., Carneiro, G., Reid, I., Drummond, T.: Smart mining for deep metric learning. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2017) 2

[6] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning (Jul 2015) 3, 4

[7] Jacob, P., Picard, D., Histace, A., Klein, E.: Metric learning with horde: High-order regularizer for deep embeddings. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019) 4

[8] Kim, W., Goyal, B., Chawla, K., Lee, J., Kwon, K.: Attention-based ensemble for deep metric learning. In: The European Conference on Computer Vision (ECCV) (September 2018) 2

[9] Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13) (Dec 2013) 4

[10] Lin, X., Duan, Y., Dong, Q., Lu, J., Zhou, J.: Deep variational metric learning. In: The European Conference on Computer Vision (ECCV) (September 2018) 1, 2, 4

[11] Liu, H., Tian, Y., Yang, Y., Pang, L., Huang, T.: Deep relative distance learning: Tell the difference between similar vehicles. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016) 1

[12] Movshovitz-Attias, Y., Toshev, A., Leung, T.K., Ioffe, S., Singh, S.: No fuss distance metric learning using proxies. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2017) 2

[13] Oh Song, H., Jegelka, S., Rathod, V., Murphy, K.: Deep metric learning via facility location. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017) 2

[14] Oh Song, H., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016) 2

[15] Opitz, M., Waltner, G., Possegger, H., Bischof, H.: Bier - boosting independent embeddings robustly. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2017) 1, 2

[16] Opitz, M., Waltner, G., Possegger, H., Bischof, H.: Deep metric learning with BIER: boosting independent embeddings robustly. IEEE transactions on pattern analysis and machine intelligence (2018) 2, 4

[17] Qian, Q., Shang, L., Sun, B., Hu, J., Li, H., Jin, R.: Softtriple loss: Deep metric learning without triplet sampling. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019) 4

[18] Rippel, O., Paluri, M., Dollar, P., Bourdev, L.: Metric learning with adaptive density discrimination. International Conference on Learning Representations (ICLR) (May 2016) 2

[19] Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2015) 1, 2

[20] Sohn, K.: Improved deep metric learning with multi-class n-pair loss objective. In: Advances in Neural Information Processing Systems 29 (Dec 2016) 2

[21] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2015) 3

[22] Ustinova, E., Lempitsky, V.: Learning deep embeddings with histogram loss. In: Advances in Neural Information Processing Systems 29 (Dec 2016) 2

[23] Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011) 4

[24] Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z., Liu, W.: Cosface: Large margin cosine loss for deep face recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018) 2

[25] Wang, J., Zhou, F., Wen, S., Liu, X., Lin, Y.: Deep metric learning with angular loss. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2017) 2

[26] Wang, X., Han, X., Huang, W., Dong, D., Scott, M.R.: Multi-similarity loss with general pair weighting for deep metric learning. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019) 1, 4

[27] Xu, X., Yang, Y., Deng, C., Zheng, F.: Deep asymmetric metric learning via rich relationship mining. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4076 – 4085 (June 2019) 1, 4

[28] Xuan, H., Souvenir, R., Pless, R.: Deep randomized ensembles for metric learning. In: The European Conference on Computer Vision (ECCV) (September 2018) 2

[29] Yu, B., Liu, T., Gong, M., Ding, C., Tao, D.: Correcting the triplet selection bias for triplet loss. In: The European Conference on Computer Vision (ECCV) (September 2018) 2

[30] Yuan, Y., Yang, K., Zhang, C.: Hard-aware deeply cascaded embedding. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2017) 2

[31] Zhao, Y., Jin, Z., Qi, G.j., Lu, H., Hua, X.s.: An adversarial approach to hard triplet generation. In: The European Conference on Computer Vision (ECCV) (September 2018) 1, 2

[32] Zheng, W., Chen, Z., Lu, J., Zhou, J.: Hardness-aware deep metric learning. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019) 1, 2, 4