Relationship-Aware Hard Negative Generation in Deep Metric Learning

Jiaqi Huang[1,2], Yong Feng[1,2](✉), Mingliang Zhou[2,3], and Baohua Qiang[4,5]

[1] College of Computer Science, Chongqing University, Chongqing 400030, China
fengyong@cqu.edu.cn
[2] Key Laboratory of Dependable Service Computing in Cyber Physical Society, Ministry of Education, Chongqing University, Chongqing 400030, China
[3] State Key Lab of Internet of Things for Smart City, University of Macau, Taipa 999078, Macau, China
[4] Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China
[5] Guangxi Key Laboratory of Optoelectronic Information Processing, Guilin University of Electronic Technology, Guilin 541004, China

Abstract. Data relationships and the impact of synthetic loss have not been concerned by previous sample generation methods, which lead to bias in model training. To address above problem, in this paper, we propose a relationship-aware hard negative generation (RHNG) method. First, we build a global minimum spanning tree for all categories to measure the data distribution, which is used to constrain hard sample generation. Second, we construct a dynamic weight parameter which reflects the convergence of the model to guide the synthetic loss to train the model. Experimental results show that the proposed method outperforms the state-of-the-art methods in terms of retrieval and clustering tasks.

Keywords: Deep metric learning · Sample generation · Distribution quantification · Minimum spanning tree · Relationship preserving

1 Introduction

Deep metric learning (DML) aims at training a deep learning model to learn effective metrics, which measure the similarities between data points accurately and robustly. Deep metric learning has been successfully applied to a variety of tasks, including recommendation [22,26], image retrieval [1,4], person re-identification [10,21], and many others.

Many recent deep metric learning approaches are built on similarity or distance between samples, such as Triplet loss [19], N-pair Loss [15] and Angular Loss [18], etc. Sampling strategy intends to search for samples that profit training most to achieve faster convergence and higher performance, such as Semi-hard Triplet [14] and lifted Structured [11], etc. Recently, sample generation methods

G. Li et al. (Eds.): KSEM 2020, LNAI 12275, pp. 388–400, 2020.
https://doi.org/10.1007/978-3-030-55393-7_35

have been proposed to optimize the network globally by synthetic hard samples. Duan *et al.* [2] utilized a generator with adversarial loss to synthesize potential hard negatives. Zheng *et al.* [25] proposed a hardness-aware method to synthesized hard samples.

However, shortcomings still remain in those sample generation methods. First, they randomly sample different categories of samples to form sample pairs as the basis for synthesizing hard samples, without considering the global geometric distribution of data. Synthetic samples are constrained to be close to the anchor point and remain in different categories, while the distribution of these two categories are of great discrepancy. Although such synthetic sample can generate large gradients for training, it does not conform to the distribution characteristics. Compare to the standard hard sample, it more like an outlier in the data than a hard sample. Training with synthetic outlier samples will make the model learn the wrong features. Second, they did not take the interactions between the training progress and loss of hard synthetic into account. In the early stages of training, the model can easily extract informative samples for training, thus premature training with synthetic samples will prevent the model from learning the characteristics of original data well.

To address the above problems, we propose the Relationship-Aware Hard Negative Generation (RHNG) method. One idea in RHNG is that we construct a continuously updated global minimum spanning tree, which acts as a sampler to screen suitable sample pairs as the basis for synthetic hard samples. Furthermore, we design an adaptive dynamic weight to control the effect of synthetic loss on model training as the training progresses. In this way, the synthetic loss will continuously participate in training to avoid the early impact on training, which will further promote model performance.

In general, the main innovations and contributions of this paper can be summarized as follows:

- We utilize graph structure to learn the data relationship, which is used to maintain the distribution of synthetic samples. As such, synthetic hard negatives are more conducive to promote model training.
- An adaptive dynamic weight is elaborately designed to guide the synthetic loss on model training, which encourages the model fully to learn the original samples and improve the performance by synthetic loss.
- Extensive experimental results demonstrate that the proposed method outperforms the state-of-the-art methods in terms of retrieval and clustering tasks.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of the state-of-the-art related works. Section 3 details our proposed RHNG method. The experimental results and analysis are provided in Sect. 4 and Sect. 5 concludes our work.

2 Related Work

Metric Learning. Metric learning aims at learning effective metrics to measure the similarity of the input data. Many existing methods pay more attention on building a proper loss function to illustrate relation between samples. Contrastive loss [5] is proposed to train on pairwise samples, which gives penalty to negative pairs with distances smaller than a margin. Triplet loss [19] is proposed to construct triplet input samples to ensure the relative distance order between positives and negatives. Furthermore, more classes are involved in N-pair Loss [15]. Instead of comparing samples in euclidean space, distance with angle constraint was measured in Angular Loss [18], which captures additional local structure. Xu *et al.* [20] proposed asymmetric loss for deep metric learning.

Hard Sample Mining. Hard sample mining [6,23] plays an essential role in speeding up convergence and boosting performance of DML model, which continuously mines hard samples that give more information to promote model training. Semi-hard mining [14] is proposed to further utilize the relation among samples by constructing semi-hard triplets within a batch. To take full advantage of the relative relationship, lifted Structured Loss [11] emphasized incorporating all negative samples within a batch. Wang *et al.* [2] focused on applying weights on samples to reflect their importance on optimization. Ge *et al.* [3] proposed Hierarchical Triplet Loss (HTL) that builds a hierarchical tree of all classes.

Hard Negative Generation. Hard negatives usually account for a small part in the training set. Therefore, synthesis methods are proposed to generate potential hard negatives, these synthetics contain more information to fully train the model. Duan *et al.* [2] utilized a generator with adversarial loss to synthesize potential hard negatives that exploit a large number of easy negatives on training. Similarly, Zhao *et al.* [24] proposed an adversarial approach to generate training examples by an adversarial network, which is used to improve triplet-based training. Zheng *et al.* [25] exploited an auto-encoder architecture to synthesize label-preserving hard samples by exploiting existing samples with adaptive hardness-aware augment.

3 Relationship-Aware Hard Negative Generation in Deep Metric Learning

3.1 Overview

Figure 1 shows the overall network architecture of our proposed method, which is comprised of four key components: encoder, graph-based sampler, generator and classifier. The solid arrows represent the data flow and the dotted arrows represent the loss being calculated. The encoder consists of two parts, one is a typical convolutional neural network and the other are several continuous fully-connected layers. Convolutional network acts a feature extractor to

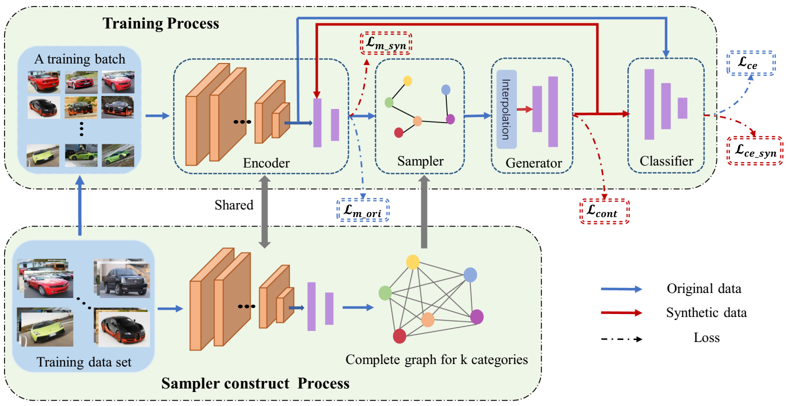

Fig. 1. Overview of the proposed RHNG framework.

extract meaningful image representations, and the fully-connected layers act as the embedding projector following the feature extractor to obtain the embedding. Graph-based sampler is a sample mining process that composes sample pairs based on a minimum spanning tree. The generator takes the sample pairs as input to synthesize hard negatives then maps them back to the feature space which will be exploited for training. However, a generator alone is insufficient to generate reliable synthetic samples, so we use a classifier with cross entropy loss to constrain the generator.

We employ the widely used triplet on our proposed method. Let $\mathcal{X}$ be the training data set, $\mathbf{X} = \{x_i\}_{i=1}^{N}$ is the mini-batch set from $\mathcal{X}$ and $L = \{l_i\}_{i=1}^{N}$ is the corresponding labels where $l_i \in \{1 \ldots k\}$. A triplet $<x_i, x_i^+, x_j^->$ is composed by an anchor point x_i, a positive point x_i^+ and a negative point x_j^- with its label $l_j^- \neq l_i$. We denote the feature extractor by F, whose output can be expressed as $\mathbf{Y}(y_i) = F(\mathbf{X})$ where $y_i \in \mathbb{R}^D$ represents a D-dimensional feature. Similarly, $\mathbf{Z}(z_i) = E(\mathbf{Y})$ and $z_i \in \mathbb{R}^M$ express that the embedding projector E projects the features into metric space.

3.2 Relationship-Aware Hard Negative Generation

We employ minimum-spanning tree algorithm to construct the connected subgraphs of all categories in the training set. Categories with edges connected means they are close in the metric space and more reasonable to have hard negatives.

We calculate cluster center for each category to reduce computational complexity. Concretely, we use the encoder to get the encoding for all instances which are denoted as $\mathbf{U} = [\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^k]$ where $\mathbf{u}^i$ represents all instances in i-th category. Denote $C^i = [d_0, d_1, \ldots, d_m]$ as the cluster center of i-th category and d_m is m-th element, we utilize element-wise average for all instances in the category to calculate the cluster center. For p-th element of C^i, we have:

$$C^i(d_p) = \frac{1}{n_i} \sum_{z \in \mathbf{u}^i} z(d_p) \tag{1}$$

where n_i is the number of training samples in the i-th category and z are instances. Then we calculate a distance matrix for all categories in the training set. The distance between the i-th and the j-th category is computed as:

$$d(C^i, C^j) = \sqrt{\sum_{p=1}^{m} (C^i(d_p) - C^j(d_p))^2} \tag{2}$$

Finally, we follow the work in [12] to calculate minimum spanning tree, which acts the sampler to form sample pairs according to the edge connection. The time complexity of the algorithm is $O(n^2)$, in our work n is the number of categories in the dataset. The sampler updates interactively at the certain iterations over the training.

Inspired by the work in [25], we utilize linear interpolation to obtain the hardness of the synthetic samples. The generator takes the embedding of mini-batch $\mathbf{X}$ and the sampled triplet sets $\mathcal{T}$ as input. For samples in $\mathbf{X}$, we perform no transformation, i.e. $\widetilde{z} = z$, while construct hard embedding $\hat{z}^-$ by linear interpolation based on triplets in $\mathcal{T}$:

$$\hat{z}^- = z^- + \lambda(z - z^-), \quad \lambda = \begin{cases} \frac{d^- - d^+}{d^-} & , d^+ < d^- \\ 1 & , d^+ \geqslant d^- \end{cases} \tag{3}$$

where $d^+ = \|z - z^+\|_2$ and $d^- = \|z - z^-\|_2$ denote the distance between positive pair and negative pair, respectively. The sample embedding obtained by simple interpolation may not meet the sample characteristics, so we apply a network g to map the original embedding $\widetilde{z}$ and the synthesize $\hat{z}^-$ to the feature space respectively:

$$\mathbf{Y}(y_i) = g(z_i), \quad (\mathbf{Y}, z_i) \in \{(\widetilde{\mathbf{Y}}, \widetilde{z}_i), (\hat{\mathbf{Y}}, \hat{z}_i)\} \tag{4}$$

where $\hat{\mathbf{Y}}$ is the synthetic hard negative that will be re-input to the encoder to calculate the synthetic loss, and $\widetilde{\mathbf{Y}}$ is used to calculate the reconstruction loss.

3.3 Loss Function

The basic objective function loss used to train the encoder is defined as follows:

$$L_{tri}(<x, x^+, x^->) = [d(x, x^+) - d(x, x^-) + m]_+ \tag{5}$$

where $d(x, x^+) = \|x - x^+\|_2^2$ is the squared Euclidean distance between two embedding vectors, $[\cdot]_+$ represents the loss function only takes the positive component as input and m is the margin. For each batch, we first get its feature $\mathbf{Y}$ and embedding $\mathbf{Z}$ through the encoder. We use the feature set $\mathbf{Y}$ to train the classifier, then use the trained classifier to constrain generator. We train the classifier by minimizing the following cross-entropy loss:

$$\min_{\theta_c} J_{cla} = \mathcal{L}_{ce}(\mathbf{Y}, L) = -log \frac{\exp(y^{l_j})}{\sum_{i=1}^{k} \exp(y^{l_i})} \tag{6}$$

where y^{l_j} represents a feature vector of j-th category and θ_c is the parameter of the classifier.

For the purpose of maintaining the semantic characteristics of the synthesized samples, we formulate the objective function of the generator as follows:

$$\begin{aligned} \min_{\theta_g} J_{gen} &= \mathcal{L}_{cont}(\mathbf{Y}, \widetilde{\mathbf{Y}}) + \mathcal{L}_{ce\_syn}(\widehat{\mathbf{Y}}, L) \\ &= \sum_{y \in \mathbf{Y}, \widetilde{y} \in \widetilde{\mathbf{Y}}} \|y - \widetilde{y}\|^2 + \sum_{\hat{y} \in \hat{\mathbf{Y}}} \mathcal{L}_{ce}(\hat{y}, L) \end{aligned} \tag{7}$$

where θ_g is the parameter of the generator, $\mathcal{L}_{cont}$ and $\mathcal{L}_{ce\_syn}$ are the content loss and classification loss, respectively.

Finally, we obtain the embedding of synthetic samples $\check{z}$ by $\check{\mathbf{Z}} = E(\widehat{\mathbf{Y}})$. Based on Eq. (5), combining two metric losses calculated from the original and synthetic samples, the objective function to train the encoder can be formulated as:

$$\begin{aligned} \min_{\theta_f} J_{metric} &= \mathcal{L}_{m\_ori} + \beta \mathcal{L}_{m\_syn} \\ &= \sum_{i=1}^{N} [d(z_i, z_i^+) - d(z_i, z_j^-) + m]_+ \\ &+ \beta \sum_{i=1}^{N} [d(z_i, z_i^+) - d(z_i, \check{z}_j^-) + m]_+ \end{aligned} \tag{8}$$

where θ_f is the parameter of the encoder, and β is a trade-off parameter.

We construct the adaptive dynamic β according to the convergence of $\mathcal{L}_{m\_ori}$. In order to eliminate the training fluctuation, we get a smooth convergence curve by using *Exponential Moving Average* (EMA) [7] for $\mathcal{L}_{m\_ori}$:

$$v_t = \gamma v_{t-1} + (1 - \gamma) \mathcal{L}_{m\_ori}^t \tag{9}$$

where v_t represents the value in t-th iteration and γ is a constant that set to 0.98, which means v_t is calculated from the latest 50 iterations according to the characteristics of the EMA. Then we use the hyperbolic tangent function to map the β between 0 and 1 as:

$$\beta = 1 - tanh(\eta \cdot v_t) \tag{10}$$

where η is a trade-off hyper-parameter.

The necessity of dynamic β lies in two aspects. On the one hand, in the early stages of training, the model can't measure the distance metric of samples well, only using the random triples can generate enough gradients to promote the model convergence. With the training process, the original samples can not support the optimization of the model and synthesize hard negatives as complements to the original ones that helps the model to be further optimized. On the other, we synthesize hard negatives based on the global distribution and the distance of sample pairs, so synthetic hard negatives in the early stage of training process may meaningless that easily damage the metric space structure, resulting the model train in the wrong direction from the beginning.

3.4 Summary of the Proposed Method

The details of the training process of proposed approach are described in Algorithm 1. It is worth noting that even though the methods in [2,24,25] and our method focus on hard sample generation, moreover, some procedures (such as the linear interpolation and autoencoder generation architecture, etc.) are the same as the method in [25]. However, the emphasis is quite different from the following aspects: First, RHNG takes advantage of the graph structure to obtain more precise synthetic hard samples for deep metric learning. Second, previous generation methods lack of considering the impact of synthetic loss on training. However, our method design adaptive dynamic weight to guide the synthetic loss on model training.

Algorithm 1. Training process of proposed RHNG

```
Input: Training set $\mathcal{X}$; hyper-parameters: $\eta$, update epoch and the margin $m$;
    iteration numbers $T$
Output: Parameters of the encoder $\theta_f$, the generator $\theta_g$ and the classifier $\theta_c$
 1: Initialize the graph-based sampler according to Eq. (1) and Eq. (2) by feed-
    forwarding $\mathcal{X}$ into the encoder
 2: for $iter = 1$ to $T$ do
 3:   Sample triplet set $\mathcal{T}$ by the sampler
 4:   Generate synthetic hard negatives per Eq. (3) and Eq. (4)
 5:   Calculate $\mathcal{L}_{cont}$, $\mathcal{L}_{ce\_syn}$ and $\mathcal{L}_{m\_syn}$ per Eq. (7) and Eq. (8)
 6:   Update $\theta_c$ by minimising $J_{cla}$ in Eq. (6)
 7:   Update $\theta_g$ by minimising $J_{gen}$ in Eq. (7)
 8:   Update $\theta_f$ by minimising $J_{metric}$ in Eq. (8)
 9:   Update the minimum spanning tree with current model
10: end for
11: return $\theta_c$, $\theta_g$, $\theta_f$
```

4 Experiments

4.1 Experimental Settings

Evaluation Metrics and Dataset. We evaluated the proposed method and existing methods on both retrieval and clustering tasks.

Table 1. Experimental results (%) on CUB-200-2011 dataset

Method	R@1	R@2	R@4	R@8	NMI	F_1
Triplet	43.61	55.73	68.93	80.08	55.79	21.54
Semi-hard	45.29	58.25	70.99	80.71	56.37	22.88
N-pair	46.28	59.85	72.21	81.83	58.08	24.48
Lifted	49.72	62.10	74.35	83.86	57.58	26.01
Angular	49.39	61.30	74.48	83.40	58.96	27.58
DAML (Triplet)	37.60	49.30	61.30	74.40	51.30	17.60
HDML (Triplet)	48.51	61.37	74.02	83.65	59.30	26.55
RHNG (Ours)	**50.47**	**63.30**	**75.10**	**84.21**	**59.58**	**27.34**

Table 2. Experimental results (%) on Cars196 dataset

Method	R@1	R@2	R@4	R@8	NMI	F_1
Triplet	55.70	68.20	78.68	86.31	53.82	22.60
Semi-hard	57.72	70.47	80.05	87.84	54.94	23.66
N-pair	60.53	73.28	83.42	89.70	56.82	24.14
Lifted	64.84	76.54	85.49	90.00	57.69	25.10
Angular	68.84	80.23	87.35	92.57	61.31	28.80
DAML (Triplet)	60.60	72.50	82.50	89.90	56.50	22.90
HDML (Triplet)	66.65	78.10	86.17	91.82	59.91	26.61
RHNG (Ours)	**68.68**	**79.28**	**87.45**	**92.89**	**61.64**	**28.74**

For retrieval task, we calculated the percentage of retrieved samples with the same label to the query images in K nearest neighbors as performance metrics, where $K \in \{1, 2, 4, 8\}$, marked as R@K. For clustering task, we employed standard K-means algorithm in test set, which evaluated with normalized mutual information (NMI) and F_1 score. NMI consists of the ratio of mutual information divided by the average entropy of clusters and the average entropy of labels. F_1 score computes the harmonic mean of precision and recalls on determining whether sample attribution to a specific cluster.

We conduct the experiment on widely-used CUB-200-2011 [17] dataset and Cars196 [9] dataset.

- CUB-200-2011 [17] dataset contains 11,788 bird images in 200 bird categories. We exploited the first 100 categories with 5,684 images as training set and the rest 5,924 images in 100 categories for testing.
- Cars196 [9] dataset includes 16,185 car images with 196 categories. We used the first 98 categories with 8,054 images for training and remaining rest 100 categories with 8,131 images for test set.

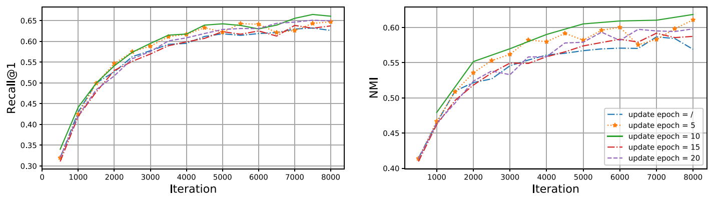

Fig. 2. Comparison of using different update epoch in the clustering (right) and retrieval (left) on Cars196 dataset with $\eta = 0.001$.

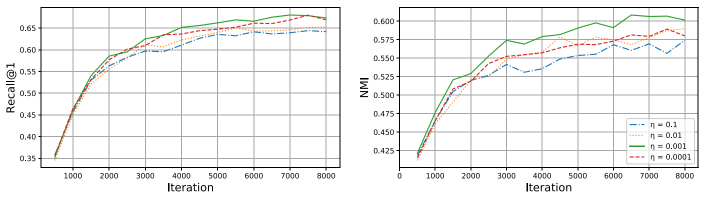

Fig. 3. Comparison of using different values of η in the clustering (right) and retrieval (left) task on Cars196 dataset with update epoch = 10.

Implementation Details. We used GoogLeNet [16] architecture as the feature extractor to extract 1024 dimensional features, and embedding projector consists of three full connection layers whose output dimension are 512, 256 and 128 respectively. Meanwhile, we implemented the generator with two fully connected layers whose output dimension are 512 and 1,024 respectively. The classifier is also consisted by three fully-connected layers.

We initialized the GoogLeNet with weights pretrained on ImageNet ILSVRC dataset [13] and all other fully-connected layers with random weights. We set hyper-parameters $\eta = 10^{-3}$, *update epoch* = 10 and the margin $m = 1$. We employed the ADAM [8] optimizer with a learning rate of $8e-4$ on training.

4.2 Comparisons with the State-of-Art Methods

We compared our method with some famous deep metric learning approaches, including the triplet loss [19] and the Semi-hard triplet loss [14], the classical sample mining method N-pair loss [15], the state-of-the-art lifted structure [11] and Angular loss [18], the hard negative generation method DAML [2] and HDML [25]. We followed the settings mentioned in these original papers through the comparison experiments.

Table 1 and Table 2 show the result compared to other baseline methods in two benchmark datasets respectively. We observe our proposed method can achieve competitive performance in terms of two datasets on retrieval and clustering tasks. The comparison with the state-of-the-art DAML and HDML shows the effectiveness of our proposed method in sample generation approaches. For the state-of-the-art Angular loss, we reached a higher performance on the smaller

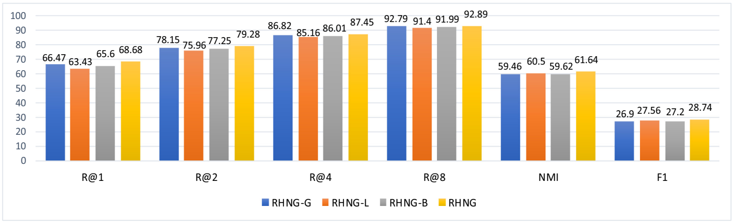

Fig. 4. Results of RHNG and its three sub-models.

CUB-200-2011 dataset, but failed to achieve their performance on the larger Cars196 with a small margin in some metrics. We analyse that the difference comes from the size of the training set, as the increase of samples in each class, we fail to collect sufficient information within a fixed number of samples through simple sampling.

4.3 Hyper-parameters Sensitivity

There are two hyper-parameters affecting our method, which are the "update epoch" and trade-off parameter η, respectively. We discuss the effect of various parameters for the clustering and retrieval tasks in terms of Cars196 dataset.

Figure 2 shows the effect of different update frequencies. Larger "update epoch" means slower update frequency and "update epoch= /" means that we only use the initialized minimum spanning tree and no longer update it. Theoretically, the higher the frequency of updating the minimum spanning tree, the timelier that data distribution can be reflected in different training periods of the model. However, the performance of updating every 10 epochs is better than updating every 5 epochs. We speculate that it is because the model needs sufficient training to learn at different periods, replacing hard negatives too frequently will make the training insufficient.

The impact of η is shown in Fig. 3. We can observe that the performance increases with the increase of η. The proposed method can achieve the best performance when $\eta = 0.001$, and then the performance gradually decreases. We speculate that it is because the constructed dynamic parameter β can better reflect the learning state of the model when $\eta = 0.001$.

4.4 Validation for Single Modules

Due to the deep network architecture, the performance improvements can be caused by many factors. To investigate the impact of different factors on the performance of the proposed method, we conduct a series of validation experiments on the Cars196 dataset. We investigate three RHNG variants: (1) "RHNG-G" is a RHNG variant without graph-based sampler, that is, randomly selecting sample pairs to synthesize hard negatives as the previous generation method. (2) "RHNG-L" is a RHNG variant without linear interpolation, which means

$\hat{z}^- = z^-$. (3) “RHNG-B” is a RHNG variant without dynamic weight β by a constant $\beta = 1$.

Figure 4 shows the experimental results of RHNG and its three variants. From the results, we can see that the full RHNG performs best on all evaluation metrics, which indicates that all of the three components in the model contribute to the final performs. The performance degradation of RHNG-G shows the significance of the graph-based sampler, which enhances the generalization ability of the model by maintaining the data distribution. Furthermore, the performance of RHNG outperforms RHNG-L, which demonstrates that the synthetic samples without enough hard levels cannot promote further training of the model. At the same time, the result of RHNG-B also proves that the strategy of using dynamic parameter to control the weight of synthesis loss is correct.

5 Conclusion

In this paper, we proposed a relationship-aware hard negative generation (RHNG) method in deep metric learning. We utilize global minimum spanning tree to constrain the generation of synthesized hard negatives. Besides, we construct a dynamic weight parameter to guide the synthetic loss to train the model, which prevents synthetic loss from misleading model. Experimental results demonstrate that our RHNG is effective and outperforms some state-of-art methods. In future work, we will focus on precise relationship constraint and efficient synthesis strategy to improve our proposed method.

Acknowledgments. Supported by National Key R&D Program of China (No. 2017YFB1402400), National Nature Science Foundation of China (No. 61762025), Guangxi Key Laboratory of Trusted Software (No. kx202006), Guangxi Key Laboratory of Optoelectroric Information Processing (No. GD18202), and Natural Science Foundation of Guangxi Province, China (No. 2019GXNSFDA185007).

References

1. Deng, C., Yang, E., Liu, T., Li, J., Liu, W., Tao, D.: Unsupervised semantic-preserving adversarial hashing for image search. IEEE Trans. Image Process. **28**(8), 4032–4044 (2019)
2. Duan, Y., Zheng, W., Lin, X., Lu, J., Zhou, J.: Deep adversarial metric learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2780–2789 (2018)
3. Ge, W., Huang, W., Dong, D., Scott, M.R.: Deep metric learning with hierarchical triplet loss. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11210, pp. 272–288. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01231-1_17
4. Grabner, A., Roth, P.M., Lepetit, V.: 3D pose estimation and 3D model retrieval for objects in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3022–3031 (2018)

5. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), vol. 2, pp. 1735–1742. IEEE (2006)
6. Huang, C., Loy, C.C., Tang, X.: Local similarity-aware deep feature embedding. In: Advances in Neural Information Processing Systems, pp. 1262–1270 (2016)
7. Hunter, J.S.: The exponentially weighted moving average. J. Qual. Technol. **18**(4), 203–210 (1986)
8. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (Poster) (2015)
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 2013), Sydney, Australia (2013)
10. Liu, Z., Wang, D., Lu, H.: Stepwise metric promotion for unsupervised video person re-identification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2429–2438 (2017)
11. Oh Song, H., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4004–4012 (2016)
12. Prim, R.C.: Shortest connection networks and some generalizations. Bell Syst. Tech. J. **36**(6), 1389–1401 (1957)
13. Russakovsky, O., et al.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015)
14. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: a unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 815–823 (2015)
15. Sohn, K.: Improved deep metric learning with multi-class n-pair loss objective. In: Advances in Neural Information Processing Systems, pp. 1857–1865 (2016)
16. Szegedy, C., et al.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
17. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-UCSD birds-200-2011 dataset. Technical report CNS-TR-2011-001, California Institute of Technology (2011)
18. Wang, J., Zhou, F., Wen, S., Liu, X., Lin, Y.: Deep metric learning with angular loss. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2593–2601 (2017)
19. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. J. Mach. Learn. Res. **10**(Feb), 207–244 (2009)
20. Xu, X., Yang, Y., Deng, C., Zheng, F.: Deep asymmetric metric learning via rich relationship mining. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4076–4085 (2019)
21. Yu, H.X., Wu, A., Zheng, W.S.: Cross-view asymmetric metric learning for unsupervised person re-identification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 994–1002 (2017)
22. Yu, J., Gao, M., Song, Y., Zhao, Z., Rong, W., Xiong, Q.: Connecting factorization and distance metric learning for social recommendations. In: Li, G., Ge, Y., Zhang, Z., Jin, Z., Blumenstein, M. (eds.) KSEM 2017. LNCS (LNAI), vol. 10412, pp. 389–396. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63558-3_33
23. Yu, R., Dou, Z., Bai, S., Zhang, Z., Xu, Y., Bai, X.: Hard-aware point-to-set deep metric for person re-identification. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11220, pp. 196–212. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01270-0_12

24. Zhao, Y., Jin, Z., Qi, G., Lu, H., Hua, X.: An adversarial approach to hard triplet generation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11213, pp. 508–524. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01240-3_31
25. Zheng, W., Chen, Z., Lu, J., Zhou, J.: Hardness-aware deep metric learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 72–81 (2019)
26. Zuo, X., Wei, X., Yang, B.: Trust-distrust aware recommendation by integrating metric learning with matrix factorization. In: Liu, W., Giunchiglia, F., Yang, B. (eds.) KSEM 2018. LNCS (LNAI), vol. 11062, pp. 361–370. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99247-1_32