Supplemental Materials of CRT

In this supplemental material, we provide (1) more detailed information on the benchmark datasets, implementation details, and algorithm summary, (2) theoretical understanding of our method, and (3) more ablation studies.

1 More Details on the Benchmark Datasets and Algorithm Implementation

In this section, we provide more details on the benchmark datasets used in our experiments, further implementation details, and algorithm summary.

1.1 Benchmarks Datasets

The following datasets are used in our experiments for performance evaluations. (1) The **CUB-200-2011** [1] consists of 11,788 images from 200 bird categories. We use the first 100 classes (5,864 images) for training and the remaining 100 classes (5,924 images) for testing. (2) The **Cars-196** [2] dataset contains 16,185 images of 196 cars classes. We use the first 98 classes (8,054 images) for training and the remaining 98 classes (8,131 images) for testing. (3) The **Stanford Online Product (SOP)** [3] dataset consists of 120,053 images with 22,634 classes crawled from Ebay. Classes are hierarchically grouped into 12 coarse categories (e.g. cup, bicycle, etc.). Following the existing protocol, we split the first 11,318 classes with 59,551 images for training, and the remaining 11,316 classes with 60,502 images for retrieval. In the test set, each image is also used as the query image. (4) The **In-Shop Clothes Retrieval (In-Shop)** [4] dataset consists of 52,712 images with 7,986 clothing classes. We use the predefined 25,882 training images of 3,997 classes for training. The remaining 3985 classes are partitioned into a query set (14,218 images) and a gallery set (12,612 images).

1.2 Implementation Details

In Table 3 of the main paper, ResNet-50 with frozen Batch-Normalization is used, just as the method by Roth *et al.* [5]. Training runs on the CUB and Cars196 datasets are performed over 70 epochs and 60 epochs for the SOP and In-Shop datasets, respectively. The initial learning rate is $3e^{-5}$, and decays 10 times after 1000 iterations for the CUB dataset, 3000 iterations for the Cars dataset, and 9000 iterations for the SOP and In-Shop datasets. The batch size is set to 80 on the CUB and Cars datasets, 150 for the MiT-B2 backbone on the SOP and In-Shop datasets, and 180 for other banckbones on the SOP and In-Shop datsets. The batch size and learning rate decay values are determined based on the performance on the validation set. We implement our experiments in PyTorch. Experiments are performed on GPU servers with a GeForce RTX 3090 GPU and 24GB memory. Each reported result is averaged over five seeds.

1.3 The MS Loss Function

We provide a more detailed explanation of the MS loss function [6] used in our paper, which is defined as follows,

$$\begin{aligned} \mathrm{L}_{ms} \quad &= \tfrac{1}{N}\sum_{I_i\in\mathcal{I}}\left[\tfrac{1}{\alpha}\log[1+\textstyle\sum_{j\in\mathcal{P}_i}\exp(-\alpha(\hat{d}_c(i,j)-\lambda))]\right] \\ &\quad + \textstyle\sum_{I_i\in\mathcal{I}}\left[\tfrac{1}{\beta}\log[1+\sum_{k\in\mathcal{N}_i}\exp(\beta(\hat{d}_c(i,j)-\lambda))]\right] \end{aligned}, \tag{1}$$

where $\mathcal{P}_i$ and $\mathcal{N}_i$ are the sets of positive pairs and negative pairs for image I_i in the mini-batch, respectively. α, β, and λ are scale parameters, we follow [6] and set them to be 2, 40, and 0.5, respectively. $\hat{d}_c(i,j)$ denotes the cosine similarities between embedding $\mathbf{f}_i$ and $\mathbf{f}_j$, which is defined as follows,

$$\hat{d}_c(i,j) = \begin{cases} d_c(\mathbf{f}_i,\mathbf{f}_j) & d_c(\mathbf{f}_i,\mathbf{f}_j) > \min_{j\in\mathbf{P}_i} d_c(\mathbf{f}_i,\mathbf{f}_j) - \epsilon \\ d_c(\mathbf{f}_i,\mathbf{f}_j) & d_c(\mathbf{f}_i,\mathbf{f}_j) < \max_{k\in\mathbf{N}_i} d_c(\mathbf{f}_i,\mathbf{f}_k) + \epsilon \\ 0 & \text{otherwise} \end{cases}. \tag{2}$$

Following [6], ϵ is set to be 0.1.

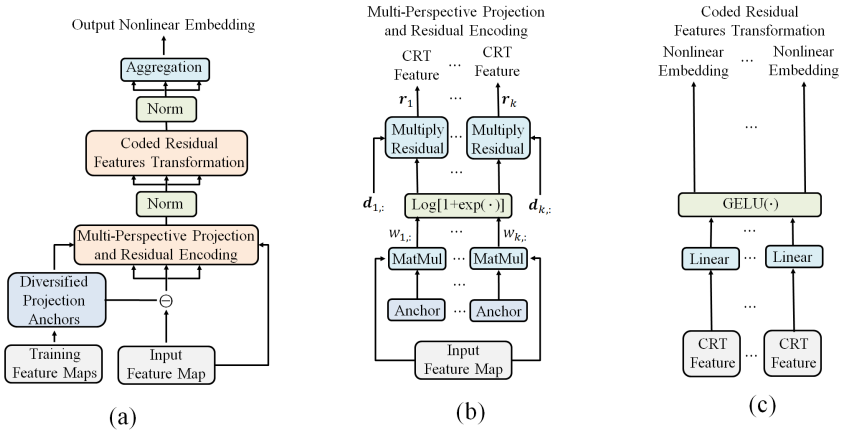


Figure 1: (a) Coded Residual Transform. (b) Multi-Perspective Projection and Residual Encoding. (c) Coded Residual Features Transformation.

1.4 Evaluation Metric

The evaluation metric used throughout this work is the recall@K, especially Recall@1, as it offers strong insights into the retrieval performance of the learned embedding. Given a set of feature embeddings $\mathbf{f}_i \in \mathcal{F}$ with $\mathbf{f}_i = \mathbf{\Gamma}(\Phi(I_i))$, and $I_i \in \mathcal{I}$. We define the k nearest neighbours of a sample $\mathbf{f}_a$ following [5] as follows,

$$\mathcal{G}_a^k = \min \text{sort}_{d(\mathbf{f}_a,\cdot)} \arg \min_{\mathcal{G}\in\mathcal{I},|\mathcal{G}|=k} \sum_{I_b\in\mathcal{G}} d(\mathbf{f}_a, \mathbf{f}_b). \tag{3}$$

The, the Recall@K is defined as:

$$\text{Recall@K} = \frac{1}{|\mathcal{I}|} \sum_{I_i\in I} \begin{cases} 1, & \exists I_k \in \mathcal{G}_i^k, \text{ s.t., } y_k = y_i \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

which evaluates how likely the positive pairs will occur in the set of k nearest neighbours.

1.5 Algorithm Summary

As shown in Figure 1 (a), we first learn anchor network to generate a set of diversified anchor features $\mathcal{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_K\}, \boldsymbol{c}_k \in \mathbb{R}^L$. For each anchor $\boldsymbol{c}_k$, we project the whole feature map $\boldsymbol{\mathcal{X}}$ onto this anchor. Specifically, for each feature $\boldsymbol{x}_j \in \boldsymbol{\mathcal{X}}$ inside the feature map, we find the relative difference $\boldsymbol{d}_{kj} = \boldsymbol{x}_j - \boldsymbol{c}_k$ between the feature $\boldsymbol{x}_j$ and the anchor $\boldsymbol{c}_k$. Let $w_{kj} = \boldsymbol{x}_j \cdot \boldsymbol{c}_k^T$ be their correlation coefficient. Then, with anchor $\boldsymbol{c}_k$, as shown in Figure 1 (b), the whole map is transformed into the following feature vector using weighted summation of $\boldsymbol{d}_{kj}$:

$$\boldsymbol{r}_k = \sum_{j=1}^{H\times W} \log[1 + \exp(\boldsymbol{x}_j^T \boldsymbol{c}_k)](\boldsymbol{x}_j - \boldsymbol{c}_k), \tag{5}$$

where the nonlinear function $\log[1+\exp(w_{kj})]$ converts the correlation coefficient w_{kj} into a positive number. In this way, each anchor $\boldsymbol{c}_k$ will generate a separate coded residual representation $\boldsymbol{r}_k$ of the feature map for the input image. Finally, as shown in Figure 1 (c), $\boldsymbol{r}_k$ is further transformed by a nonlinear embedding network $\mathbf{\Gamma}_k$ with multiple fully connected layers and one GELU nonlinear activation layer [7]. By aggregating those nonlinear embeddings, the final embedded feature $\boldsymbol{f}$ is then given by

$$\boldsymbol{f} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{\Gamma}_k(\mathbf{r}_k). \tag{6}$$

We summarize the training algorithm in **Algorithm** 1.

Algorithm 1 Summary of Training Procedures

Input: Random sampling a mini-batch images. Set the number of anchors K_1 and K_2 in the first embedding branch and the second embedding branch to be 49 and 64. Set the embedding dimensions d and D in these two embedding branches to be 128 and 1024, respectively.
Output: Linear embeddings in the first embedding branch.
Compute feature maps of the input images based on the backbone network $\mathbf{\Phi}$.
Encode these feature maps in the first and the second embedding branches based on the CRT method (Figure 1 (a) and (b)).
Transform the encoded features into nonlinear embeddings in each embedding branch using network $\mathbf{\Gamma}$ (Figure 1 (c)).
Transform the nonlinear embeddings in the first embedding branch into linear embeddings.
Compute the diversity loss of anchors $L_{DIV}^{(1)}$ and $L_{DIV}^{(2)}$ for these two embedding branches, respectively.
Compute the multi-similarity loss $\mathrm{L}_{MS}^{(1)}$ and $\mathrm{L}_{MS}^{(2)}$ based on the output embeddings in these two embedding branches, respectively.
Compute local correlation matrices $\mathbb{S}^{(1)}$ and $\mathbb{S}^{(2)}$ for linear embeddings and nonlinear embeddings in the first embedding branch and the second embedding branch, respectively.
Compute the consistency loss L_{CON} based on $\mathbb{S}^{(1)}$ and $\mathbb{S}^{(2)}$.
Optimize the parameters of the whole model in an end-to-end manner. ▷ The consistency loss does not require backpropagation in the second embedding branch during optimization.

2 Further Understanding of the Proposed CRT Algorithm

This section aims to provide further understanding of our CRT method. In our method, we first learn a set of diversified anchor features, project the feature map onto each anchor, and then encode its features using their projection residuals weighted by their correlation coefficients with each anchor. Then, we transform each encoded CRT feature $\mathbf{r}_k$ into an independent embedding space using an embedding network $\mathbf{\Gamma}_k$. Finally, we average all the embeddings to obtain the final nonlinear embeddings. Here, we show that the embedding and average process aims to realize dimension reduction for the concatenated CRT features.

Suppose that the concatenated CRT features of images I_i and I_j are $\mathbf{R}_i \in \mathbb{R}^{LK\times 1}$ and $\mathbf{R}_j \in \mathbb{R}^{LK\times 1}$, where L is the channel dimension of the feature map, K is the number of anchors. The Mahalanobis distance between $\mathbf{R}_i$ and $\mathbf{R}_j$ is defined as,

$$d(I_i, I_j) = (\mathbf{R}_i - \mathbf{R}_j)^T \mathbf{M} (\mathbf{R}_i - \mathbf{R}_j), \tag{7}$$

In (7), $\mathbf{M}$ is a learnable matrix that can be decomposed as $\mathbf{G}^T\mathbf{G}$, and $\mathbf{G} \in \mathbb{R}^{D\times LK}$, $D < LK$, then $d(I_i, I_j)$ can be re-written as,

$$\begin{aligned} & (\mathbf{R}_i - \mathbf{R}_j)^T \mathbf{M} (\mathbf{R}_i - \mathbf{R}_2 j) \\ = \; & (\mathbf{R}_i - \mathbf{R}_j)^T \mathbf{G}^T \mathbf{G} (\mathbf{R}_i - \mathbf{R}_j) \\ = \; & (\mathbf{G}\mathbf{R}_i - \mathbf{G}\mathbf{R}_j)^T (\mathbf{G}\mathbf{R}_i - \mathbf{G}\mathbf{R}_j) \\ = \; & ||\mathbf{G}\mathbf{R}_i - \mathbf{G}\mathbf{R}_j||^2, \end{aligned} \tag{8}$$

where $\mathbf{G}$ is a learnable matrix which can be implemented using a fully connected network $\mathbf{\Gamma}$. The inputs of $\mathbf{\Gamma}$ are the CRT features $\mathbf{R}_i$ and $\mathbf{R}_j$. The outputs are the reduced dimension of the embeddings $\mathbf{G}\mathbf{R}_i$ and $\mathbf{G}\mathbf{R}_j$. The parameters of the network $\mathbf{\Gamma}$ are $\mathbf{G}$. According to the block-wise matrix multiplication rules, $\mathbf{G}\mathbf{R}_i$ and $\mathbf{G}\mathbf{R}_j$ can be re-written as:

$$\mathbf{G}\mathbf{R}_i = \textstyle\sum_{k=1}^{K} \mathbf{G}_k \mathbf{r}_{i,k}, \quad \mathbf{G}\mathbf{R}_j = \sum_{k=1}^{K} \mathbf{G}_k \mathbf{r}_{j,k}, \tag{9}$$

where $\mathbf{r}_{i,k} \in \mathbb{R}^{L\times 1}$ and $\mathbf{r}_{j,k} \in \mathbb{R}^{L\times 1}$ are slice vectors in $\mathbf{R}_i$ and $\mathbf{R}_j$, respectively. $\mathbf{G}_k \in \mathbb{R}^{D\times L}$ is the k-th matrix block in $\mathbf{G}_k$. Thus, we have

$$\tfrac{1}{K}\mathbf{G}\mathbf{R}_i = \tfrac{1}{K}\textstyle\sum_{k=1}^{K} \mathbf{G}_k \mathbf{r}_{i,k}, \quad \tfrac{1}{K}\mathbf{G}\mathbf{R}_j = \tfrac{1}{K}\sum_{k=1}^{K} \mathbf{G}_k \mathbf{r}_{j,k}. \tag{10}$$

$\mathbf{G}_k\mathbf{r}_{i,k}$ and $\mathbf{G}_k\mathbf{r}_{j,k}$ can be implemented using a fully connected network $\mathbf{\Gamma}_k$, where $\mathbf{\Gamma}_k$ is a sub-network in $\mathbf{\Gamma}$. The input of the network $\mathbf{\Gamma}_k$ are CRT features $\mathbf{r}_{i,k}$ and $\mathbf{r}_{j,k}$. The outputs are the

embeddings with reduced dimensions $\mathbf{G}_k\mathbf{R}_{i,k}$ and $\mathbf{G}_k\mathbf{R}_{j,k}$. The parameters of $\mathbf{\Gamma}_k$ are $\mathbf{G}_k$. Then, we have

$$\boldsymbol{f} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{\Gamma}_k(\mathbf{r}_k). \tag{11}$$

The above derivation process is based on the linear transformation condition, which is shown in Figure 2(a). In our implementation, we used the nonlinear activation function GELU$(\cdot)$ for each sub-matrix $\mathbf{G}_k$. In this case, the left and right sides of (9) and (10) are not strictly equal. According to the above analysis, our CRT method can be considered as a form of nonlinear metric learning to realize dimensionality reduction. Specifically, as shown in Figure 2(b), $\forall k$, if we set $\mathbf{G}_k = \mathbf{G}_0$ or $\mathbf{\Gamma}_k = \mathbf{\Gamma}_0$, the transformation of CRT features is the single-perspective transformation. The experimental results for these two cases are shown in Table 7 in the main paper.

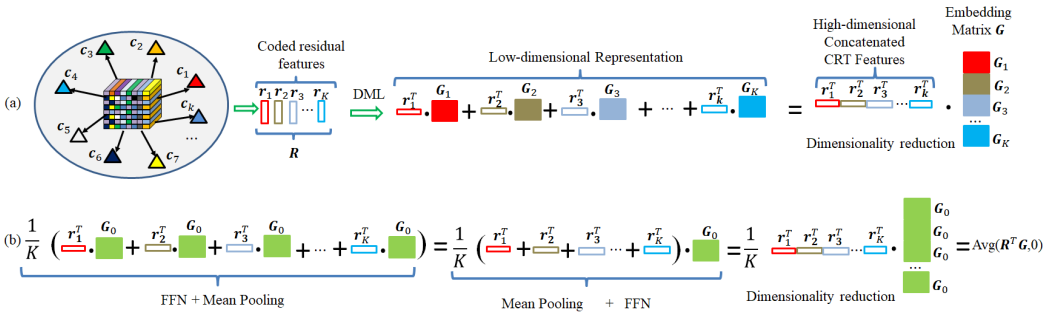


Figure 2: Dimension reduction of deep metric learning (DML) on the coded residual features.

3 More Ablation Studies

This section provides more experimental results and further ablation studies to understand our CRT method and demonstrate its performance.

3.1 Performance Comparisons with Different Configurations of the Backbone Network

Different configurations of the backbone network may have a major impact on the final embedding performance. In Section 4.3 of the main paper, we provide performance comparisons with different backbone networks. Here, we experiment with different parameter sizes of the model and complexity of the backbone network. Table 1 summarizes the parameter sizes and complexity levels (measured by FLOPs) of different backbone networks. We can see that the parameters and FLOPs of ResNet-50, DeiT-S and MiT-B2 are approximately the same. The parameter sizes and FLOPs of MiT-B1 and BN-Inception are also approximately equal. In the following, we provide the experimental results with the MiT-B0, MiT-B1 and MiT-B2 backbone networks based on the CRT method. Results on the CUB and Cars datasets are shown in Table 2. Table 3 shows the experimental results on the SOP and In-Shop datasets with the MiT backbone networks.

We can see that the performance increases consistently with the increasing of model complexity.

Table 1: Parameters count and FLOPs at resolution 227×227.

Complexity Index	GoogLeNet	BN-Inception	ResNet-50	DeiT-S	MiT-B0	MiT-B1	MiT-B2
Params(M) ↓	5.60	10.27	23.51	21.96	3.31	13.14	24.17
FLOPs(G) ↓	1.52	2.06	4.65	4.24	0.47	1.83	3.55

Table 2: Experimental results on the CUB and Cars datasets for different MiT backbone networks.

Method	Backbones	**CUB**				**Cars**			
		R@1	R@2	R@4	R@8	R@1	R@2	R@4	R@8
CRT	MiT-B0	69.19	79.66	86.83	92.23	85.34	91.28	94.75	97.05
	MiT-B1	75.95	84.47	90.26	94.51	89.60	94.20	96.47	97.79
	MiT-B2	78.98	86.68	91.61	95.04	91.16	94.92	96.79	98.03

Table 3: Experimental results on the SOP and In-Shop datasets for different MiT backbone networks.

Method	Backbones	**SOP**				**In-Shop**			
		R@1	R@10	R@100	R@1000	R@1	R@10	R@20	R@30
CRT	MiT-B0	80.16	91.60	96.50	98.90	92.90	99.06	99.42	99.51
	MiT-B1	82.32	93.02	97.19	99.20	93.51	99.11	99.50	99.63
	MiT-B2	83.41	93.86	97.66	99.31	94.48	99.37	99.68	99.75

3.2 Impact of Hyper-parameters

In this experiment, we analyze the impact of metric learning parameters, including the loss weight λ_1, cross-CRT consistency loss weight λ_2, different embedding sizes and different batch sizes. The value of λ_1 in the first embedding branch (the low-dimensional embedding branch) is always set to be 1.0 in our experiments. In the second embedding branch (the high-dimensional embedding branch), we set $\lambda_1 = 1.0 - \lambda_2$, the retrieval performance with different values of λ_2 are shown in Table 4. We can see that the best recall@1 is obtained at $\lambda_2 = 0.9$.

Table 4: The Recall@K (%) for different λ_2.

λ_2	**CUB**			
	R@1	R@2	R@4	R@8
0	73.09	82.38	89.18	93.91
0.1	73.87	83.12	89.92	93.99
0.3	74.93	83.96	90.46	94.16
0.5	74.22	83.52	90.16	93.94
0.7	75.14	84.10	90.48	94.46
0.9	**75.95**	84.47	90.26	**94.51**
1.0	75.07	**84.54**	**90.77**	94.16

Table 5 shows the results for different dimensions of embeddings on the CUB dataset. We can see that, when the feature dimension of the second embedding branch is 1024, the performance is less sensitive to embeddings with different dimensions obtained by the first embedding branch. Previous studies indicated that the batch size has a significant impact on the performance of feature embedding [6, 8]. In this experiment, we conduct ablation studies on the dimension of feature embeddings. Results are shown in Table 6. We can see that the batch size has a major impact on the embedding performance when it is smaller than 64. When the batch size is larger than 80, the performance changes very little.

Moreover, we have compared our method with the MS method based on the feature dimension of 128 in Table 5 of the main paper. Here, we conduct experiment to compare with the MS method based on the BN-Inception backbone network and the feature dimension of 512. This experiment is conducted on the CUB dataset. The top-1 recall rate is 66.5% for the proposed CRT method, which is 0.8% higher than the MS method (65.7% as reported in the original paper). This experiment shows the robustness and generalization ability of our CRT method for different dimensions of feature embeddings.

Table 5: The Recall@K (%) for different dimensions of embedding.

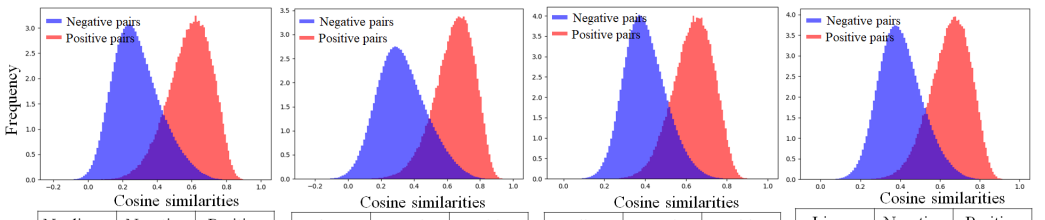
The First Embedding Branch					The Second Embedding Branch				
Dim d	R@1	R@2	R@4	R@8	Dim D	R@1	R@2	R@4	R@8
64	72.57	81.70	89.10	93.62	1024	75.74	84.47	90.38	94.41
128	75.95	84.47	90.26	94.51	1024	76.52	85.33	90.94	94.65
256	76.16	84.28	90.63	94.60	1024	76.82	84.89	90.78	94.60
512	76.18	85.23	90.99	94.68	1024	75.66	85.08	90.75	94.48
1024	77.09	85.42	91.19	95.05	2048	76.40	85.43	91.46	94.99

Table 6: The Recall@K (%) for different batch sizes.

N	CUB			
	R@1	R@2	R@4	R@8
32	73.38	83.20	90.23	94.29
64	75.19	84.47	90.45	94.43
80	75.95	84.47	90.26	94.51
120	75.79	84.39	90.38	94.50
150	75.78	84.72	90.75	94.68
180	75.32	84.59	90.34	94.13

4 More Supporting Results for the Proposed CRT Method

In this section, we study the distributions of the learned features on the CUB, Cars, SOP and In-Shop datasets. Figure 3-6 show the similarity distributions of positive pairs and negative pairs [9] for the baseline method and our CRT method. We can see that the similarity distributions of the embeddings generated by the first embedding branch are close to that of the embeddings generated by the second embedding branch based on our proposed method. However, these two distributions for the baseline method are significantly different. Figure 7-10 show the t-SNE [10] plots of the learned embedding for these two branches. We can see that the t-SNE plots of the low-dimensional embeddings are similar to those of the high-dimensional embeddings in on our method. The above results demonstrate that our method exhibits better generalization capabilities [5].



Nonlinear	Negative	Positive
Mean	0.2841	0.5489
Var	0.0185	0.0244

(a) High-dimensional embedding without CC

Linear	Negative	Positive
Mean	0.3280	0.6030
Var	0.0220	0.0241

(b) Low-dimensional embedding without CC

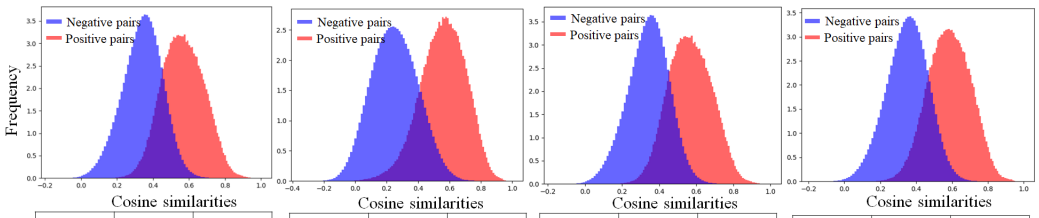
Nonlinear	Negative	Positive
Mean	0.4002	0.6035
Var	0.0106	0.0148

(c) High-dimensional embedding with CC

Linear	Negative	Positive
Mean	0.4036	0.6186
Var	0.0120	0.0161

(d) Low-dimensional embedding with CC

Figure 3: The distributions of cosine similarities with and without cross-CRT correlation consistency (CC) for high-dimensional embeddings and low-dimensional embeddings on the CUB dataset.



Nonlinear	Negative	Positive
Mean	0.2365	0.5018
Var	0.0168	0.0203

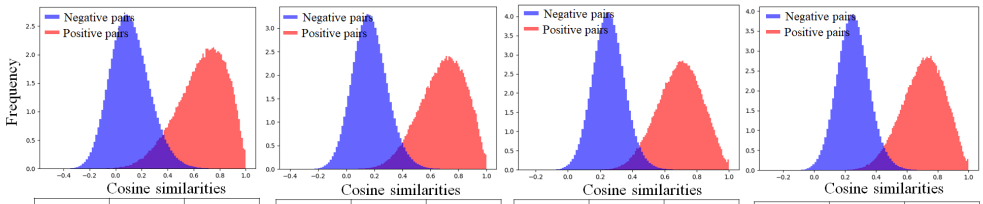
Linear	Negative	Positive
Mean	0.2508	0.5581
Var	0.0217	0.0210

Linear	Negative	Positive
Mean	0.3407	0.5673
Var	0.0134	0.0123

Nonlinear	Negative	Positive
Mean	0.3495	0.5850
Var	0.0135	0.0140

(a) High-dimensional embedding without CC (b) Low-dimensional embedding without CC (c) High-dimensional embedding with CC (d) Low-dimensional embedding with CC

Figure 4: The distributions of cosine similarities with and without cross-CRT correlation consistency (CC) for high-dimensional embeddings and low-dimensional embeddings on the Cars dataset.



Nonlinear	Negative	Positive
Mean	0.1106	0.6614
Var	0.0224	0.0349

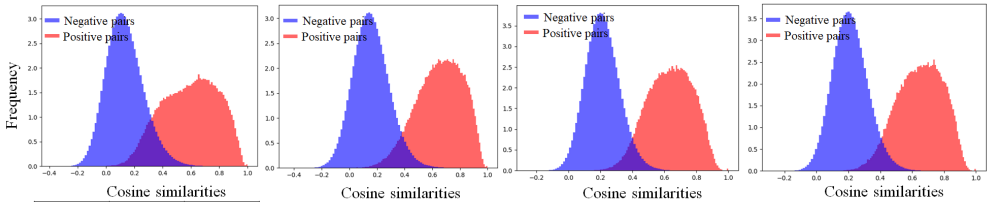
Linear	Negative	Positive
Mean	0.1686	0.6852
Var	0.0152	0.0256

Linear	Negative	Positive
Mean	0.2535	0.6928
Var	0.0098	0.0186

Nonlinear	Negative	Positive
Mean	0.2533	0.7057
Var	0.0107	0.0186

(a) High-dimensional embedding without CC (b) Low-dimensional embedding without CC (c) High-dimensional embedding with CC (d) Low-dimensional embedding with CC

Figure 5: The distributions of cosine similarities with and without cross-CRT correlation consistency (CC) for high-dimensional embeddings and low-dimensional embeddings on the SOP dataset.



Nonlinear	Negative	Positive
Mean	0.1580	0.6526
Var	0.0170	0.0271

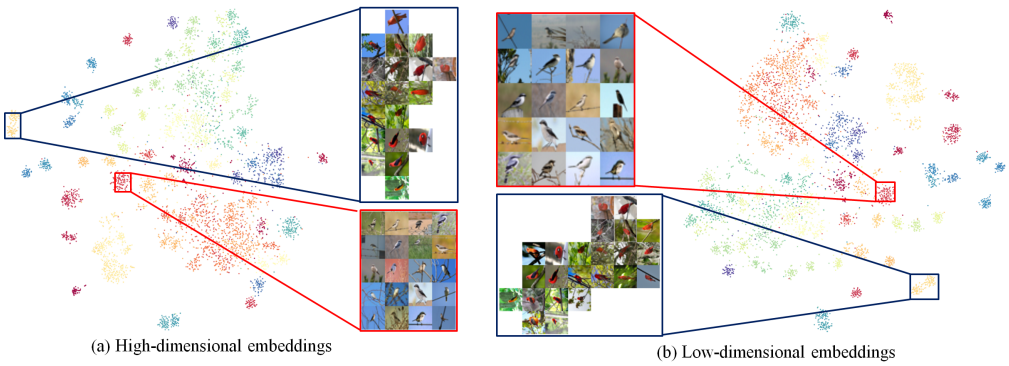
Linear	Negative	Positive
Mean	0.1343	0.5931
Var	0.0173	0.0375

Linear	Negative	Positive
Mean	0.2134	0.6351
Var	0.0113	0.0195

Nonlinear	Negative	Positive
Mean	0.2121	0.6493
Var	0.0123	0.0202

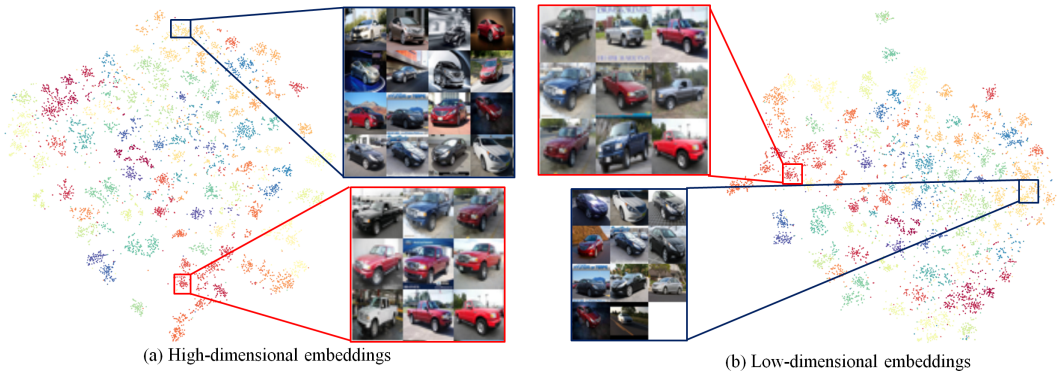
(a) High-dimensional embedding without CC (b) Low-dimensional embedding without CC (c) High-dimensional embedding with CC (d) Low-dimensional embedding with CC

Figure 6: The distributions of cosine similarities with and without cross-CRT correlation consistency (CC) for high-dimensional embeddings and low-dimensional embeddings on the In-Shop dataset.



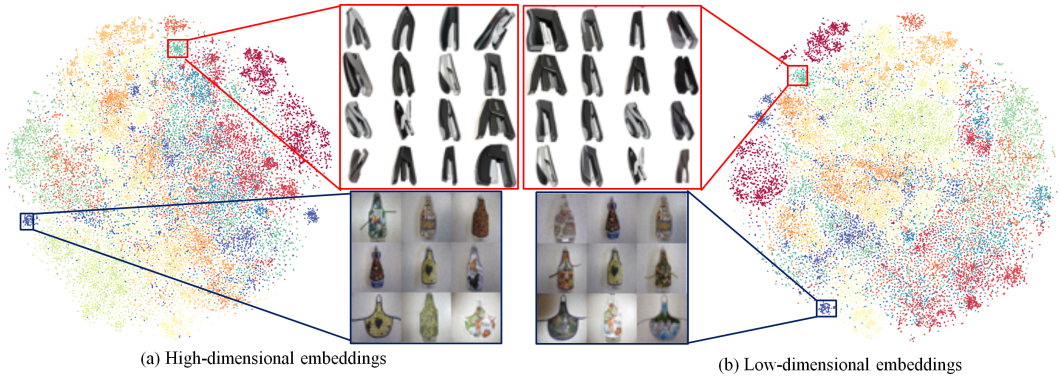
(a) High-dimensional embeddings (b) Low-dimensional embeddings

Figure 7: The t-SNE visualizations of high-dimensional embeddings and low-dimensional embeddings on the CUB dataset.



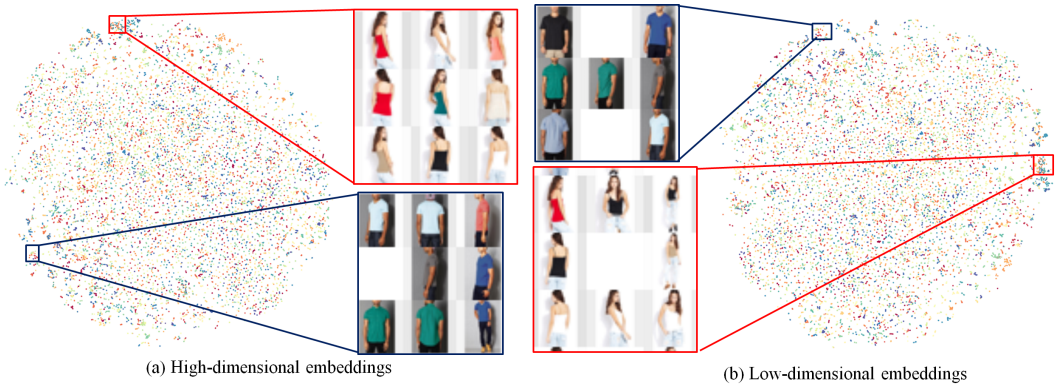
(a) High-dimensional embeddings (b) Low-dimensional embeddings

Figure 8: The t-SNE visualizations of high-dimensional embeddings and low-dimensional embeddings on the Cars dataset.



(a) High-dimensional embeddings (b) Low-dimensional embeddings

Figure 9: The t-SNE visualizations of high-dimensional embeddings and low-dimensional embeddings on the SOP dataset.



(a) High-dimensional embeddings (b) Low-dimensional embeddings

Figure 10: The t-SNE visualizations of high-dimensional embeddings and low-dimensional embeddings on the In-Shop dataset.

Figure 11 shows the similarity matrix between anchors learned on the CUB dataset. We can see that most of the similarities between anchors are very low, only very few similarities computed between anchors are relatively large. This result shows that most of the anchors learned by our method are independent.

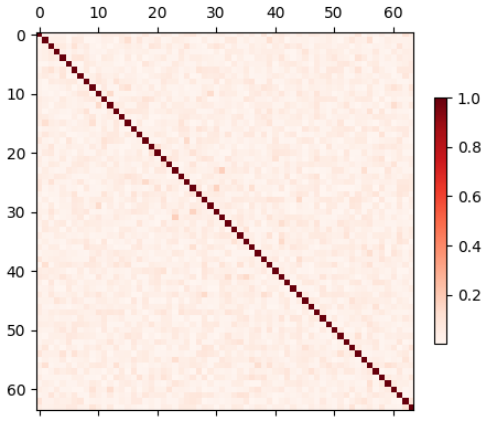


Figure 11: The similarity matrix visualization for anchors learned on the CUB dataset.

References

[1] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. *California Institute of Technology*, pages 1–8, 2011.

[2] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *2013 IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2013, Sydney, Australia, December 1-8, 2013*, pages 554–561, 2013.

[3] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4004–4012, 2016.

[4] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 1096–1104, 2016.

[5] Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Björn Ommer, and Joseph Paul Cohen. Revisiting training strategies and generalization performance in deep metric learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 8242–8252, 2020.

[6] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R. Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 5022–5030, 2019.

[7] Hendrycks Dan and Gimpel Kevin. Gaussian error linear units (gelus). *arXiv preprint*, abs/1606.08415, 2016.

[8] Aleksandr Ermolov, Leyla Mirvakhabova, Valentin Khrulkov, Nicu Sebe, and Ivan V. Oseledets. Hyperbolic vision transformers: Combining improvements in metric learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, Louisiana, USA, June 21-24, 2022*, 2022.

[9] Shichao Kan, Yigang Cen, YangLi, Mladenovic Vladimir, and Zhihai He. Local semantic correlation modeling over graph neural networks for deep feature embedding and image retrieval. *IEEE Trans. Image Processing*, 31:2988–3003, 2022.

[10] Laurens van der Maaten. Accelerating t-sne using tree-based algorithms. *J. Mach. Learn. Res.*, 15(1):3221–3245, 2014.